Table des matières

Table des tableaux

Table des figures

Chapitre I

Analyse d'algorithmes et de structures de données.

1. Introduction

L'analyse des performance d'un algorithme fait partie intégrante de l'ensemble de la démarche algorithmique qui vise à définir des méthodes de calcul systématiques pour la résolution d'un problème. Pour le concepteur d'algorithme, l'efficacité de la méthode mise en œuvre est en effet, après sa validité, le critère principal de qualité de l'algorithme. L'évaluation des coûts d'utilisation (temps d'exécution, place mémoire, nombres d'entrées-sorties,...) détermine ainsi le réalisme et la "faisabilité" d'une méthode algorithmique; elle donne aussi des critères de choix entre différentes solutions algorithmiques d'un même problème.

Le domaine de l'analyse d'algorithme a pris son essor dans les années 1970, à la suite de l'ouvrage de base de D.E. Knuth "The Art of Computer Programming" qui fournit une classification précise des différents algorithmes de base de l'informatique. L'analyse en moyenne d'algorithme a depuis considérablement évoluée, d'une part par l'introduction de nouveaux domaines d'études en algorithmique, mais surtout par l'utilisation de méthodes mathématiques plus puissantes. Des développements récents en analyse combinatoire et l'utilisation de méthodes asymptotiques reposant sur l'analyse complexe permettent d'étendre considérablement le champ des algorithmes ou des schémas algorithmiques que l'on sait traiter.

A la suite des travaux de J-M. Steyaert et P. Flajolet ([Ste84], [FSt87],[Fla87],[Fla85b],...), il s'est mis en place une méthodologie permettant de systématiser l'analyse exacte ou asymptotique pour différentes grandes classes de programmes. Le premier exemple de cette démarche concerne la classe des programmes récursifs dont le schéma repose sur un parcours descendant de structures arborescentes. Cette approche, que nous reprenons dans ce travail, consiste dans un premier temps à traduire la structure combinatoire de l'algorithme sous forme de "descripteurs de complexité" qui prennent la forme de séries génératrices de coûts moyens d'exécution. Pour des structures récursives, cette traduction fait alors apparaître des équations implicites sur les descripteurs de complexité. Dans une seconde phase, l'analyse réelle et surtout l'analyse complexe permettent de passer de ces descripteurs de complexité aux valeurs des coûts moyens de manière exacte ou plus généralement sous forme d'équivalents et de développements asymptotiques.

Cette démarche combine ainsi des visions récentes de l'analyse combinatoire (cf. Rota, Joyal, Foata, Schutzenberger,...) avec des méthodes d'analyse asymptotique liées à la formule de Cauchy et à ses conséquences en analyse complexe (analyse de singularités, méthodes de col, ...). Les séries génératrices constituent alors le noyau de base autour duquel va s'articuler l'ensemble de l'analyse en moyenne. L'aspect systématique de la méthodologie et les possibilités offertes par le développement des systèmes de calcul formel permettent d'envisager, dans certaines limites, l'automatisation du processus d'analyse dans ces deux phases principales : traduction et détermination asymptotique des coûts moyens. Ceci est déjà réalisé par le système $\Lambda\Upsilon\Omega$[FSZ89a] pour l'analyse d'une large classe d'algorithmes opérant sur des structures de données "décomposables" à partir de constructions ensemblistes de base (union, produit cartésien, suite de, ensemble de, multi-ensemble de, ...).

L'objectif de notre travail est de montrer l'applicabilité de cette méthodologie d'analyse en moyenne pour des schémas algorithmiques très classiques issus des structures d'arbres de recherche ou, de manière équivalente, de l'algorithme de tri rapide Quicksort. On cherche ainsi à calculer ou à caractériser, dans le cadre le plus général possible, les valeurs moyennes et les distributions de probabilité liées aux coûts des algorithmes de tri rapide ou de recherche dans les arbres de recherche. On s'attache de plus, grâce à l'analyse de différentes variantes de l'algorithme, à décortiquer des schémas généraux d'analyse applicables et automatisables sur une classe plus large de problèmes.

L'algorithme Quicksort, introduit par Hoare en 1962 [Hoa62], a été étudié par Knuth [Knu73] et de façon plus détaillée par R. Sedgewick dans sa thèse [Sed80]. Les résultats obtenus par des méthodes classiques de résolution d'équations de récurrence portent sur les valeurs moyennes des coûts pour différentes implantations et différentes extensions de l'algorithme (tri par insertion sur les petites listes, médiane de 3,...). La variance du nombre de comparaisons nécessaires au tri est aussi déterminée dans le cas de l'algorithme standard sans médiane ni traitement particulier des petites listes. Compte tenu de ce que l'on connaît maintenant de l'algorithme, il apparaît toutefois difficile d'obtenir plus de résultats avec les techniques utilisées.

L'utilisation plus systématique de séries génératrices pour résoudre le problème apparaît dans la thèse de D.H. Greene [Gre83]. L'analyse porte sur les arbres "réduits" de recherche qui combinatoirement correspondent aux structures sous-tendues par le tri rapide dans sa variante médiane de k. On note que le calcul des coûts moyens pour la recherche et pour le tri rapide sont deux problèmes équivalents (cf. chapitre II), il n'en est pas de même des distributions de probabilité. La démarche utilisée par D.H. Greene s'appuie sur un formalisme de grammaire étiquetée et n'est pas très éloignée de celle que l'on développe ici pour décrire la structure combinatoire du problème et la traduire automatiquement par des équations sur les séries génératrices des coûts moyens. La résolution qui est faîte n'utilise toutefois que très peu les outils asymptotiques et ne fournit ainsi des résultats que sur les valeurs moyennes des coûts.

On se propose ici de montrer comment l'introduction de méthodes combinatoires et asymptotiques axées sur l'utilisation de séries génératrices permet de synthétiser et de systématiser l'analyse des coûts moyens et des distributions de probabilité pour l'algorithme Quicksort et ses principales variantes.

Pour cela, on introduit d'abord un algorithme général qui prend en compte simultanement les différentes variantes considérées : traitement particulier des petites listes, médiane de k et multipartitionnement. Pour correspondre à un schéma le plus général possible, cet algorithme cherche à s'affranchir aux maximum de contraintes liées à l'implantation. Il ne va dépendre que de trois paramètres associés aux paramètres de chacune des variantes.

On montre ensuite comment la structure combinatoire de cet algorithme peut se décrire synthétiquement à l'aide de constructeurs ensemblistes simples opérants l'ensemble de données de l'algorithme, c'est-à-dire ici les permutations. Le passage aux séries génératrices de coût moyen s'effectue grâce à des règles de traduction ad-hoc et aboutit à une caractérisation de l'algorithme général par une équation différentielle sur les séries génératrices.

Pour ce type d'équation différentielle, on introduit un schéma de résolution qui prend en compte de manière systématique les aspects asymptotiques de la résolution et permet de synthétiser la détermination des coûts moyens en fonction des paramètres caractéristiques de l'algorithme général. Au cours de l'ensemble du processus, on peut ainsi suivre facilement l'influence des différentes variantes de l'algorithme et décortiquer les comportements qu'elles induisent aussi bien dans le schéma d'analyse qu'au niveau des résultats sur les coûts moyens.

L'intérêt de ce schéma de résolution est qu'il s'applique de la même façon à la détermination des variances et des moments d'ordre supérieure des distributions de probabilité liées aux algorithmes de tri ou de recherche. Il est de plus assez simple pour être facilement automatisable grâce à l'utilisation d'un système de calcul formel. On est alors capable de calculer facilement les premiers moments des distributions de probabilité et ainsi d'étendre considérablement, tant d'un point de vue pratique que théorique, les résultats obtenus jusqu'a présent.

Dans la suite de ce chapitre, on introduit rapidement les outils formels qui servent de base à la description de la structure combinatoire des algorithmes ou des structures de données. On donnera de plus quelques exemples très simples d'utilisation.

Le chapitre II décrit différentes structures d'arbres de recherche et l'algorithme Quicksort avec ses variantes : médiane de k, multipartitionnement, traitement particulier des petites listes. On montre en particulier les équivalences combinatoires qui existent en ces deux types de schémas algorithmiques. On définit de plus l'algorithme général qui globalise les différentes variantes et qui va servir de base à notre analyse.

Dans le chapitre III, on s'intéresse à la détermination des valeurs moyennes des coûts d'exécution pour l'algorithme général. On se place pour cela dans l'hypothèse où les données sont des listes constituées de clés distinctes avec une distribution uniforme de l'ordre relatif des clés. On montre comment l'opérateur de produit partitionnel permet de traduire de manière synthétique la structure combinatoire de l'algorithme. On définit de plus le schéma de résolution qui va permettre un calcul aisé des valeurs moyennes des coûts d'exécution et par la suite des moments des distributions de probabilité.

Dans le chapitre IV, on cherche à caractériser les distributions de probabilité pour les coûts de la recherche d'une part et du tri rapide d'autre part. On montre l'existence de lois limites gaussiennes pour les coûts de la recherche fructueuse ou infructueuse dans les arbres généraux que l'on a défini. On démontre aussi l'existence d'une loi limite pour le nombre de comparaisons dans l'algorithme de tri. On essaye enfin de caractériser le plus finement possible cette distribution limite.

Dans le dernier chapitre, on reprend l'étude de coûts moyens en supposant l'existence de clés identiques dans les listes à trier. On montre en particulier comment évoluent les valeurs moyennes des coûts en fonction du taux moyen de répétition des clés.

2. Représentation combinatoire et séries génératrices

Cette section est une présentation rapide de la méthodologie et des outils algébriques mis en œuvre pour la description combinatoire et l'analyse en moyenne d'algorithmes ou de structures de données. Pour une description plus complète de la démarche, on peut se reporter aux articles de synthèse de P. Flajolet [Fla85a] et [Fla87].

Définition I.1 *On appelle classe de structures combinatoires un ensemble* $\mathbf{C}$*, fini ou dénombrable, muni d'une fonction taille* $|.|_{\mathbf{C}} : C \mapsto \mathbf{N}$ *telle que* $\forall n \in \mathbf{N}$ *l'ensemble* $\mathbf{C}_n = \{x \in \mathbf{C} / |x|_{\mathbf{C}} = n\}$ *est fini.*

Une classe représente l'ensemble des objets combinatoires que l'on étudie comme par exemple les permutations ou une famille d'arbre. Le problème de dénombrement sur une classe $\mathbf{C}$ consiste alors à déterminer la suite des $c_n = card(\mathbf{C}_n)$. Il est pour cela intéressant d'introduire des séries génératrices qui vont représenter de manière globale l'ensemble de la suite des c_n.

Définition I.2 *A toute classe* $\mathbf{C}$ *de structures combinatoires on associe formellement les séries génératrices ordinaires et exponentielles définies respectivement par :*

$$C(z) = \sum_{x \in \mathbf{C}} z^{|x|_{\mathbf{C}}} = \sum_{n=0}^{\infty} c_n z^n \, ,$$

$$\hat{C}(z) = \sum_{x \in \mathbf{C}} \frac{z^{|x|_{\mathbf{C}}}}{(|x|_{\mathbf{C}})!} = \sum_{n=0}^{\infty} c_n \frac{z^n}{n!} \, ,$$

avec $c_n = card(\mathbf{C}_n)$.

Le caractère formel de ces séries correspond au fait que l'on ne se préoccupe pas à priori de leur convergence ou de leur rayon de convergence dans le plan complexe.

On considère maintenant des constructions combinatoires sur les classes. Ces constructions correspondent à des constructeurs ensemblistes classiques. Il faut toutefois préciser comment la fonction taille dans l'ensemble ainsi produit est obtenue à partir des tailles dans les ensembles de départ. Les constructions usuelles d'union et de produit cartésien sont ainsi définies par :

Définition I.3 *L'union de deux classes disjointes* $\mathbf{A}$ *et* $\mathbf{B}$ *est la classe constituée des éléments de l'union des deux ensembles avec la fonction taille définie par :*

$$|x|_{\mathbf{A} \cup \mathbf{B}} = \begin{cases} |x|_{\mathbf{A}} & si \ x \in \mathbf{A} \\ |x|_{\mathbf{B}} & si \ x \in \mathbf{B} \end{cases} .$$

Le produit cartésien de deux classes de structures combinatoires $\mathbf{A}$ *et* $\mathbf{B}$ *est la classe constituée des éléments du produit cartésien des deux ensembles avec la fonction taille définie par :*

$$|(a, b)|_{\mathbf{A} \times \mathbf{B}} = |a|_{\mathbf{A}} + |b|_{\mathbf{B}}.$$

Il est alors facile de traduire ces deux descriptions combinatoires par des relations sur les séries génératrices associées aux différentes classes de structures combinatoires.

Lemme I.1 *Si* $\mathbf{P} = \mathbf{A} \times \mathbf{B}$*, les séries génératrices ordinaires vérifient :*

$$P(z) = A(z)B(z).$$

Si $\mathbf{P} = \mathbf{A} \cup \mathbf{B}$*, on a :*

$$P(z) = A(z) + B(z).$$

Une construction combinatoire sera dite admissible si la suite de dénombrement $c_n = card(\mathbf{C}_n)$ du résultat est totalement déterminée par les suites de dénombrement des arguments. Une construction admissible sur des classes de structures combinatoires, $\mathbf{P} = \Phi[\mathbf{A}, \mathbf{B}]$, définit ainsi des opérateurs ϕ et $\hat{\phi}$ sur les séries formelles associées

$$P(z) = \phi[A(z), B(z)] \qquad \text{et} \qquad \hat{P}(z) = \hat{\phi}[\hat{A}(z), \hat{B}(z)].$$

On trouvera dans [VF89] et dans [GJ83] un ensemble des constructions admissibles les plus intéressantes en combinatoire.

Nous donnons à titre d'exemple quelques unes de ces constructions.

Pour les séries génératrices ordinaires, on a les opérateurs :

Union disjointe	$\mathbf{C} = \mathbf{A} \cup \mathbf{B}$	$C(z) = A(z) + B(z)$
Produit Cartésien	$\mathbf{C} = \mathbf{A} \times \mathbf{B}$	$C(z) = A(z)B(z)$
Suite finie de	$\mathbf{C} = \mathbf{A}^*$	$C(z) = (1 - A(z))^{-1}$
Ensemble fini de	$\mathbf{C} = 2^{\mathbf{A}}$	$C(z) = exp(A(z) - \frac{A(z^2)}{2} + \frac{A(z^3)}{3} + \cdots)$
Substitution	$\mathbf{C} = \mathbf{A}[\mathbf{B}]$	$C(z) = A(B(z))$

Dans ces différentes constructions la fonction taille est définie comme la somme des tailles des composantes.

Pour les séries génératrices exponentielles, les principaux opérateurs sont :

Union disjointe	$\mathbf{C} = \mathbf{A} \cup \mathbf{B}$	$\hat{C}(z) = \hat{A}(z) + \hat{B}(z)$
Produit partitionnel	$\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$	$\hat{C}(z) = \hat{A}(z)\hat{B}(z)$
P-Suite (Complexe partitionnel)	$\mathbf{C} = \mathbf{A}^{<*>}$	$\hat{C}(z) = (1 - \hat{A}(z))^{-1}$
P-ensemble (Complexe partitionnel abélien)	$\mathbf{C} = \mathbf{A}^{[*]}$	$\hat{C}(z) = exp(\hat{A}(z))$

Ces dernières constructions introduites par Foata [Foa74] sont plus particulièrement adaptées à la description de structures étiquetées pour lesquelles on distingue entre tous les renumérotages possibles. L'opérateur de produit partitionnel sera à la base de la description combinatoire de l'algorithme de tri rapide et des structures d'arbres de recherche dans les chapitres suivants.

On note que le cas particulier d'un isomorphisme entre deux classes correspond aussi à un constructeur admissible.

Définition I.4 *Deux classes de structures combinatoires sont isomorphes, si l'on a isomorphisme des ensembles associés et égalité des tailles entre structures correspondantes.*

Lemme I.2 *Les séries génératrices ordinaires (resp. exponentielles) de deux classes isomorphes sont égales.*

Le problème de dénombrement des structures d'une classe **C** se ramène donc à essayer de décrire, à isomorphismes près, cette classe à partir de constructions admissibles sur des objets plus simples dont on connaît la suite de dénombrement, ou récursivement sur la classe **C** elle-même. Cette représentation formelle de la structure combinatoire des objets analysés se traduit alors directement par une équation fonctionnelle explicite ou implicite sur les séries génératrices. On dispose ensuite des outils classiques de l'analyse pour déterminer, quand cela est possible, la série génératrice ou le terme général c_n sous une forme close. L'analyse complexe, et en particulier la théorie des fonctions analytiques, permet de plus d'effectuer à partir de la série génératrice une analyse asymptotique directe de la suite des dénombrements dans le cas où l'on n'a pas d'expression simple pour les c_n.

Par rapport aux techniques combinatoires classiques de traitement des suites et des récurrences, les séries génératrices apportent donc un caractère synthétique à la résolution par l'utilisation des outils de l'analyse réelle. Elles permettent surtout, grâce aux techniques asymptotiques, d'étendre considérablement l'ensemble des problèmes que l'on sait résoudre.

On considère maintenant une fonction f définie sur les éléments d'une classe de structures combinatoires **C**. Si **C** décrit l'ensemble des données possibles d'un algorithme, f est par exemple le coût d'exécution de l'algorithme pour chaque instance des données. Pour analyser la fonction f sur l'ensemble des données de la classe **C**, on introduit une classe pondérée définie formellement par :

Définition I.5 *A toute fonction f définie sur les éléments d'une classe* **C** *de structures combinatoires, on associe la classe de coût :*

$$\mathbf{f}(\mathbf{C}) = \sum_{x \in \mathbf{C}} f(x).x \qquad \textit{avec} \qquad |f(x).x|_{\mathbf{f}(\mathbf{C})} = |x|_{\mathbf{C}}.$$

La série génératrice ordinaire associée est :

$$f_{\mathbf{C}}(z) = \sum_{x \in \mathbf{C}} f(x).z^{|x|_{\mathbf{C}}} = \sum_{n=0}^{\infty} f_n z^n ,$$

où f_n est la somme des coûts f sur toutes les structures de taille n de **C**.

La quantité f_n/c_n est alors égale à la valeur moyenne de la fonction f sur l'ensemble des structures supposées équiprobables de la classe **C**. L'analyse en moyenne d'un algorithme consiste donc à décrire au moyen de constructeurs admissibles la classe **C** des données possibles et la classe $\mathbf{f}(\mathbf{C})$ des coûts d'exécution sur ces données. On en déduit alors des équations fonctionnelles sur les séries génératrices, et donc une définition implicite du coût moyen.

3. Exemples

Nous donnons ici quelques exemples de base pour montrer dans des cas simples l'utilisation des classes de structures combinatoires et des constructions admissibles associées. Pour un éventail plus complet d'exemples d'utilisation de la méthodologie mise en œuvre pour l'analyse automatique d'algorithmes ou de structures de données, on peut se reporter au livre de recettes $\Lambda\Upsilon\Omega$[FSZ89b].

3.1. Dénombrement d'arbres.

Arbres binaires.

De façon classique, les arbres binaires sont définis récursivement à partir de deux objets : la feuille et le nœud interne. Un arbre binaire est en effet une feuille ou un nœud interne relié à deux arbres binaires. En introduisant deux classes triviales $\mathbf{F}$ et $\mathbf{N}$ constituées d'un seul élément, la classe des arbres binaires est donc décrite formellement par :

$$\mathbf{B} = \mathbf{F} + \mathbf{N} \times \mathbf{B} \times \mathbf{B}. \tag{I.1}$$

En définissant la taille d'un arbre binaire comme le nombre de nœuds internes, on a alors $F(z) = z^0 = 1$, $N(z) = z^1 = z$ et

$$B(z) = 1 + zB(z)^2. \tag{I.2}$$

La série de dénombrement des arbres binaires est ainsi caractérisée par une équation fonctionnelle implicite que l'on résout classiquement sous la forme :

$$B(z) = \frac{1 - \sqrt{1-4z}}{2z}.$$

On note que l'autre solution de l'équation du second degré n'est pas admissible car elle ne donne pas $B(0) = B_0 = 1$, c'est-à-dire le nombre d'arbres binaires de taille nulle.

Les propriétés des coefficients binomiaux et la formule de Stirling donnent alors l'expression suivante pour le nombre d'arbres binaires ayant n nœuds internes :

$$B_n = \frac{1}{n+1}\binom{2n}{n} \sim \frac{4^n}{\sqrt{\pi} n^{\frac{3}{2}}} \qquad \text{pour } n \to \infty. \tag{I.3}$$

On retrouve ainsi le résultat avec les nombres de Catalan qui peut aussi être obtenu de manière classique avec des relations de récurrence.

Arbres généraux.

En reprenant les notations utilisées pour les arbres binaires, on définit formellement la classe $\mathbf{G}$ des arbres généraux par :

$$\mathbf{G} = \mathbf{F} + \mathbf{N} \times \mathbf{G} + \mathbf{N} \times \mathbf{G} \times \mathbf{G} + \mathbf{N} \times \mathbf{G} \times \mathbf{G} \times \mathbf{G} + \cdots$$

Si la taille est égale au nombre total de sommets, les séries de dénombrement vérifient alors : $F(z) = N(z) = z$ et

$$G(z) = z(1 + G(z) + G(z)^2 + G(z)^3 + \cdots) = \frac{z}{1 - G(z)}$$

On a donc une équation du second degré qui se résout sous la forme :

$$G(z) = \frac{1 - \sqrt{1-4z}}{2} = zB(z).$$

Ce résultat est plus difficile et plus long à obtenir par les techniques classiques de récurrence, sauf bien sûr si l'on remarque l'isomorphisme qui existe entre les arbres binaires à $n-1$ sommets internes et les arbres généraux de taille n (cf. [Knu73]).

Arbres s-aires.

Les arbres s-aires sont définis comme ayant des noeuds d'arité 0 ou s. Ils sont donc décrits par :

$$\mathbf{A} = \mathbf{F} + \mathbf{N} \times \underbrace{\mathbf{A} \times \cdots \times \mathbf{A}}_{s}$$

En comptant comme taille le nombre total de sommets, on a donc :

$$A(z) = z(1 + A(z)^s).$$

La série $A(z)$ n'a ainsi pas de forme close en général. Le dénombrement exact peut toutefois être obtenu dans ce cas en utilisant le théorème d'inversion de Lagrange (cf. [Ste84]).

De façon plus générale, on peut étudier, grâce au théorème d'inversion de Lagrange, les équations implicites obtenues pour des familles simples d'arbres [Ste84] et obtenir par l'analyse asymptotique un nombre d'arbres de la forme :

$$T_n \sim c\rho^{-n} n^{-\frac{3}{2}},$$

où c et ρ dépendent de la famille considérée.

3.2. Longueurs de cheminement sur les arbres binaires.

On désire calculer la valeur moyenne, sur les arbres binaires supposés équiprobables, de la longueur de cheminement interne. Elle est définie comme étant la somme des profondeurs de chaque sommet de l'arbre. On peut donc la décomposer récursivement en suivant le schéma de description des arbres binaires :

$$lci(b) = \begin{cases} O & \text{si } b = f \in F \\ t(b_1) + t(b_2) + lci(b_1) + lci(b_2) & \text{si } b = (n, b_1, b_2) \in N \times B \times B \end{cases}.$$

où t compte le nombre total de sommets d'un arbre. t vaut donc $2n + 1$ pour un arbre binaire à n nœuds internes.

Cette définition récursive de la longueur de cheminement interne permet de décrire maintenant la classe de coût $lci(\mathbf{B})$ sous la forme :

$$\begin{aligned} lci(\mathbf{B}) &= lci(f).f + \sum_{(b_1,b_2)\in B^2} (g(b_1) + g(b_2)).(n, b_1, b_2) \\ &= lci(\mathbf{F}) + \sum_{(b_1,b_2)\in B^2} (n, g(b_1), b_2) + (n, b_1, g(b_2).b2) \\ &= lci(\mathbf{F}) + \mathbf{N} \times g(\mathbf{B}) \times \mathbf{B} + \mathbf{N} \times \mathbf{B} \times g(\mathbf{B}), \end{aligned} \tag{I.4}$$

avec la fonction g définie par $g(x) = t(x) + lci(x)$.

Ceci se traduit alors simplement sur les séries génératrices par :

$$lci(z) = 2zB(z)(t(z) + lci(z)).$$

En reprenant la série $B(z)$ définie précédemment, la série $t(z)$ est alors obtenue par $t_n = (2n+1)B_n$, d'où $t(z) = 2zB'(z) + B(z)$.

On en déduit donc une forme close pour la longueur de cheminement interne.

$$lci(z) = \frac{2zB(z)t(z)}{1 - 2zB(z)} = \frac{(1 - 2z - \sqrt{1-4z})}{z(1-4z)}$$

On est maintenant facile d'en extraire une expression exacte, ou asymptotique par la formule de Stirling, pour lci_n On en déduit alors la valeur moyenne de la longueur de cheminement interne $l_n = lci_n/B_n$ dans les arbres à n sommets internes. On obtient ainsi :

$$l_n = 2\sqrt{\pi} n^{-\frac{3}{2}}.$$

Chapitre II

Quicksort et arbres de recherche.

Le but de ce chapitre est de présenter les algorithmes ou les structures de données que l'on va analyser dans ce travail. On décrira successivement l'algorithme Quicksort [Hoa62, Sed80] et ses variantes : médiane de $k = 2t + 1$ clés, multipartitionnement et traitement différent pour les listes de petite taille. Nous montrerons l'équivalence combinatoire entre ces algorithmes et différentes structures d'arbre binaire de recherche. On verra en particulier les correspondances entre la médiane et les arbres réduits de recherche [Gre83], entre le multipartitionnement et les arbres s-aires (ou multiway) [Knu73], et entre l'élagage des arbres de recherche et le fait de tronquer le tri Quicksort pour les petites listes. L'ensemble de ces variantes sera combiné sous la forme d'un algorithme général qui servira de base à l' analyse en moyenne. Nous n'entrerons volontairement pas dans les détails d'implantations particulières qui ne sont pas utiles pour l'analyse assez générale que l'on veut en faire ; nous renvoyons, sur ce point, le lecteur aux références bibliographiques (en particulier [Knu73, vol 3],[AHU74], et [Sed80]).

1. Equivalence combinatoire.

1.1. Arbres binaires de recherche.

Définition II.1 *Un arbre binaire de recherche est un arbre binaire étiqueté contenant une clé sur chaque nœud interne, avec la propriété : la clé d'un nœud est plus grande que toutes les clés du sous-arbre gauche et est plus petite que toutes les clés du sous-arbre droit. L'ensemble des clés est muni d'une relation d'ordre total.*

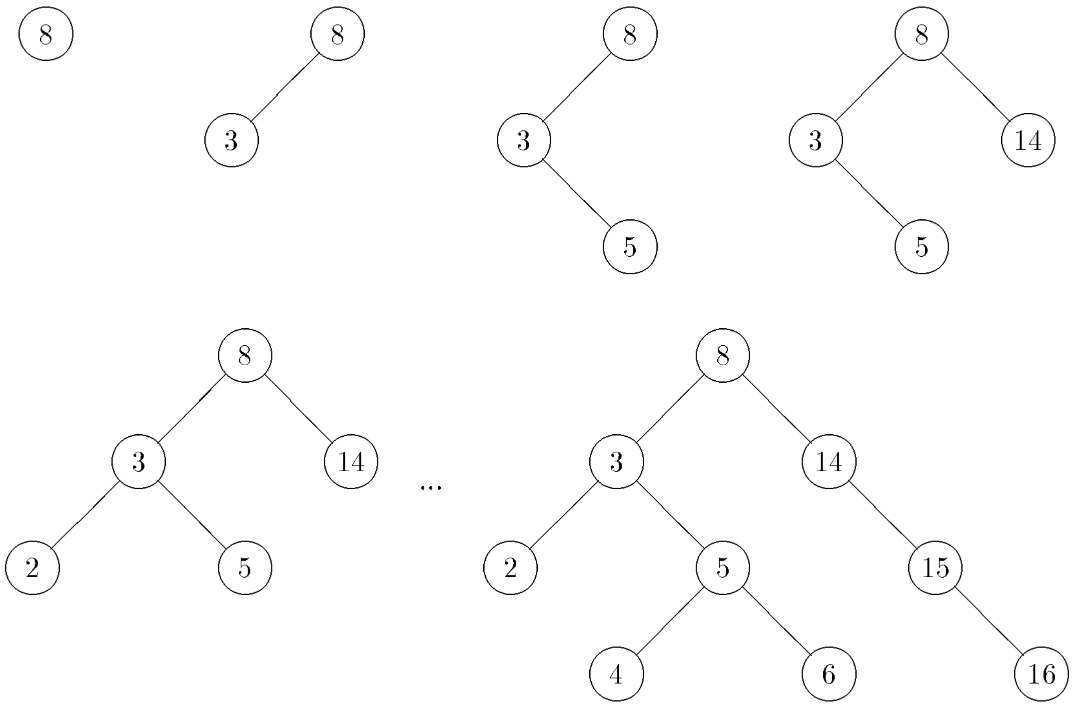
La recherche d'un élément dans un arbre binaire de recherche s'effectue en descendant récursivement dans l'arbre à partir de la racine. La direction à prendre à chaque nœud, fils droit ou fils gauche, est alors déterminée par comparaison entre l'élément et la valeur de la clé en ce nœud.

Si la clé recherchée est déjà dans l'arbre, on parle de recherche avec succès. Le coût en nombre de comparaisons est alors égal à la profondeur du nœud contenant la clé plus 1, en comptant la racine à une profondeur 0. Le coût moyen de la recherche d'un élément quelconque de l'arbre est donc déterminé par la longueur de cheminement interne lci de l'arbre. On a plus précisément : Coût moyen $= 1 + lci/n$ pour un arbre ayant n nœuds internes.

Dans le cas où la clé recherchée n'est pas dans l'arbre, on parle de recherche infructueuse, la procédure de recherche finit alors obligatoirement sur une feuille. C'est donc la longueur de cheminement externe lce de l'arbre qui donne le coût moyen d'une recherche avec : Coût moyen = $lce/(n+1)$ pour un arbre ayant $n+1$ feuilles, c'est-à-dire n nœuds internes.

La construction d'un arbre binaire de recherche s'effectue de manière classique par insertions successives de ses clés. On associe ainsi à chaque liste de clés un arbre de recherche. L'algorithme d'insertion consiste à faire une recherche infructueuse de la clé à insérer et à remplacer la feuille où l'on aboutit par un nouveau nœud interne contenant la valeur de la clé. Le coût moyen de l'insertion d'un nouvel élément quelconque dans un arbre donné est donc directement lié au coût de la recherche infructueuse dans cet arbre.

Exemple II.1 : Pour la liste 8 3 5 14 2 6 4 15 16 l'insertion successive des clés conduit à la suite d'arbres suivante :



Remarque : Le parcours récursif d'un arbre binaire de recherche en ordre symétrique ou infixe, c'est-à-dire visite du sous-arbre gauche puis de la racine puis du sous-arbre droit, donne la liste ordonnée des clés de l'arbre.

Considérons maintenant le coût de construction de l'arbre binaire de recherche compté en nombre total de comparaisons à effectuer. Pour chaque clé, le nombre de comparaisons utilisées pour placer la clé dans l'arbre, c'est-à-dire pour créer le nœud correspondant, est égal à la profondeur de ce nœud. La clé est en effet comparée seulement avec les valeurs des nœuds situés sur le chemin entre la racine et le nœud associé à cette clé. Pour une liste donnée, le coût de la construction par insertions successives de l'arbre binaire de recherche associé est donc égal à la longueur de cheminement interne de l'arbre ainsi créé.

Par exemple, pour l'arbre précédent, le coût de construction est de 17 comparaisons et le coût moyen de la recherche d'une clé de l'arbre est de $1 + 17/9 = 26/9$ comparaisons.

1.2. Algorithme de tri rapide.

L'algorithme de tri Quicksort a été introduit par A. R. Hoare en 1962 [Hoa62]. Il consiste à diviser récursivement la liste des clés à trier en deux sous-listes contenant respectivement les éléments inférieurs et supérieurs à une clé de partitionnement donnée. On peut décrire schématiquement l'algorithme par la procédure suivante :

```
Procedure Tri_rapide(L : liste);
    si taille(L) ≤ 1
      alors  retourner(L);
      sinon  Choisir un élément v de L;
             Construire les listes L1 et L2 contenant les clés de L
                respectivement plus petites et plus grandes que v;
             retourner(Tri_rapide(L1).v.Tri_rapide(L2));
    fin si ;
fin ;
```

Le tri d'une liste de taille n se compose donc d'une étape linéaire ($\approx n$ comparaisons) de partitionnement et du tri de deux sous-listes de tailles p et q avec $p + q = n - 1$. Dans le cas le plus défavorable, c'est-à-dire si l'une des tailles p ou q est proche de zéro à chaque étape, le coût en nombre de comparaisons est quadratique. Le cas optimum consiste lui à avoir à chaque partitionnement la même taille pour les deux sous-listes : $p \approx q \approx n/2$. Le coût du tri d'une liste de taille n est alors voisin de $n \log_2 n$, c'est-à-dire de l'ordre de grandeur de la borne minimum que l'on peut espérer pour les tris opérant par comparaisons (cf. [Knu73, 5.3.1]).

On pourrait envisager de choisir comme clé de partitionnement v l'élément médian de la liste L à partitionner. Cette opération de sélection peut être faite en temps linéaire (cf. [AHU74, section 3.6]), on obtiendrait ainsi un algorithme de tri en $O(n \ln n)$ dans tous les cas. Cette approche n'est toutefois pas intéressante dans la pratique car la recherche de l'élément médian est coûteuse et rend l'algorithme de tri très peu efficace par rapport, par exemple, à Heapsort.

Le chapitre suivant permettra de montrer que pour un choix aléatoire de la clé v le comportement moyen de l'algorithme de tri est en $O(n \ln n)$. On montrera de plus une façon simple et peu coûteuse d'améliorer le choix de v qui permet de réduire sensiblement la constante de temps en facteur du terme $n \ln n$.

Dans le cas où les listes sont représentées par des tableaux, on dispose d'une procédure simple et quasi-optimale en nombre de clés déplacées pour effectuer le partitionnement. Cette procédure est de plus économique en mémoire et n'utilise qu'un nombre constant de mémoires auxiliaires.

L'algorithme consiste à prendre deux pointeurs placés initialement en début et en fin de liste. Chaque pointeur va parcourir la liste en sens inverse à la recherche d'une clé mal placée : c'est-à-dire d'une clé supérieure à v en début de liste ou inférieure à v en fin de liste. Les clés mal placées détectées par les deux pointeurs sont alors échangées et on réitère le processus jusqu'a ce que les pointeurs se croisent.

On trouve différentes variantes de cet algorithme dans [Sed80] ; on n'en donne ici qu'une illustration par la procédure suivante qui partitionne une liste autour de son premier élément :

Procedure partition(L:liste);
  i:=1; j:=taille(L)+1;
  v:=L[1]; cond:=TRUE;
  tant que cond **faire**
    répéter j:=j-1 **jusqu'a** L[j]$\leq v$;
    Si i$\geq$j **alors** j:=i; cond:=FALSE; **fin si**;
    L[i]:=L[j];
    répéter i:=i+1 **jusqu'a** L[i]$\geq v$;
    Si i$\geq$j **alors** cond:=FALSE; **fin si**
    L[j]:=L[i];
  fin tant que;
  L[j]:=v
fin

La suite du tri s'effectue alors sur les sous-listes données par les indices 1 à j-1 et j+1 à taille(L).

Exemple II.2 : Appliquons la procédure précédente sur la liste :

8 3 5 14 2 6 4 15 16

On libère d'abord la position de la clé 8 de partitionnement (les | indiquent la position des pointeurs i et j) :

. | 3 5 14 2 6 4 15 16 |

La première clé mal placée en fin de liste est 4 ; on la met dans la place libre en début de liste :

4 | 3 5 14 2 6 | . 15 16

On va ensuite déplacer la clé 14 du début vers la fin de la liste :

4 3 5 . | 2 6 | 14 15 16

Puis la clé 6 :

4 3 5 6 | 2 | . 14 15 16

Les pointeurs se croisent enfin et on met la clé de partitionnement 8 en place :

4 3 5 6 2 || 8 14 15 16

Dans la suite, on aura besoin de considérer un algorithme de référence pour Quicksort, ceci afin d'expliciter simplement les équivalences et les descriptions combinatoires liés à la structure de l'algorithme. La procédure de référence consiste à toujours choisir le premier élément de la liste comme clé de partitionnement et à conserver l'ordre relatif des clés en partitionnant. L'ordre des clés dans les deux sous-listes est ainsi le même que dans la liste initiale.

```
Procedure Partition Référence(L:liste);
    i:=1; j1:=0; j2:=0;
    v:=L[1];
    tant que i < taille(L) faire
        i:=i+1;
        Si L[i]≤v;
            alors j1:=j1+1; L1[j1]:=L[i];
            sinon j2:=j2+1; L2[j2]:=L[i];
        fin si;
    fin tant que;
fin
```

L'algorithme Quicksort opère ensuite le tri sur les deux sous-listes L1 et L2 de tailles j1 et j2.

Dans la pratique, cet algorithme apparaît naturellement si on travaille sur des listes chaînées. Il permet toutefois de décrire, sous des hypothèses d'uniformité des distributions des données, un ensemble très général d'implantations comme on le verra au début du prochain chapitre. L'algorithme de référence correspond de plus à l'isomorphisme le plus naturel entre le tri Quicksort et les arbres de recherche construits par insertions successives des clés.

Exemple II.3 : Le tri de la liste 8 3 5 9 13 2 10 11 6 4 12 7 1 14 par l'algorithme de référence donne les partitions successives (autour des clés en gras) :

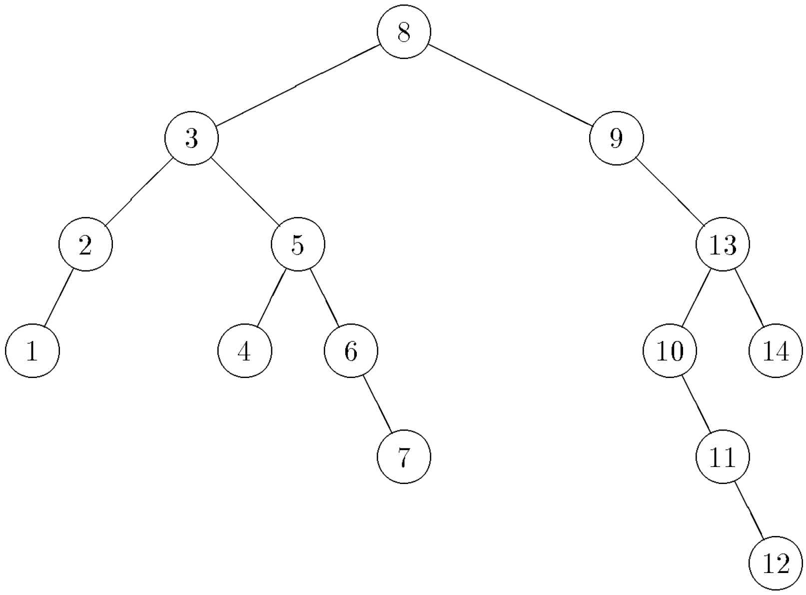
8	3	5	9	13	2	10	11	6	4	12	7	1	14
3	5	2	6	4	7	1	**8**	9	13	10	11	12	14
2	1	**3**	5	6	4	7							
1	**2**												
			4	**5**	6	7							
					6	7							
								9	13	10	11	12	14
									10	11	12	**13**	14
									10	11	12		
										11	12		

1.3. Equivalences.

Les algorithmes que l'on considère ne travaillent pas directement sur les valeurs des clés mais effectuent seulement des comparaisons sur ces clés. Seul l'ordre relatif des clés intervient, on peut donc remplacer leur valeur par leur numéro d'ordre dans la liste triée de toutes les clés. Les listes de taille n deviennent alors des permutations de $[1, \ldots, n]$.

A une permutation σ donnée, on associe un arbre binaire décrivant l'exécution du tri de cette permutation par l'algorithme de référence (cf. [Sed80]). Chaque partitionnement correspond à un nœud interne de l'arbre étiqueté par la valeur de la clé utilisée pour partitionner. Les deux sous-listes créées par le partitionnement sont alors associées à chacun des deux sous-arbres. On fait de plus correspondre les listes de taille 1 à des nœuds internes sans descendants.

Exemple II.4 : En reprenant la permutation $\sigma = 8\ 3\ 5\ 9\ 13\ 2\ 10\ 11\ 6\ 4\ 12\ 7\ 1\ 14$ de l'exemple précédent (II.3), on obtient ainsi l'arbre associé :



Proposition II.1 *L'arbre associé à l'exécution de l'algorithme Quicksort de référence sur une permutation donnée est exactement l'arbre binaire de recherche obtenu par insertions successives des clés à partir de cette permutation.*

On peut voir simplement ce résultat en considérant l'algorithme d'insertions successives de façon globale sur l'ensemble de la permutation. Au lieu de traiter les clés une par une, on décide de traiter l'ensemble des clés restantes à chaque fois que l'on crée un nouveau nœud dans l'arbre. A la création d'un nœud, on sait en effet pour les clés restant à insérer dans quel sous-arbre elles doivent se placer, et dans quel ordre elles s'y insèrent. On peut donc décider, sans modifier les résultats de l'algorithme, qu'a la création d'un nœud on anticipe une partie de l'insertion des clés suivantes en effectuant la traversée du nœud pour toutes ces clés. Le nouvel algorithme consiste donc à créer la racine avec la première clé, puis à insérer temporairement les clés suivantes dans deux listes qui correspondent aux deux sous-arbres à créer ; l'insertion se faisant en fin de liste. On réitère ensuite récursivement sur les deux sous-arbres. On obtient ainsi quelque chose de totalement équivalent à l'algorithme de tri de référence dans le mesure où l'anticipation de l'insertion des clés suivantes correspond directement au partitionnement des listes.

On a alors comme conséquence simple la proposition suivante :

Proposition II.2 ([Knu73]) *Le nombre de comparaisons nécessaires pour trier une permutation avec l'algorithme de référence est égal au nombre de comparaisons utilisées pour construire l'arbre binaire de recherche associé ; il est aussi égal à la longueur de cheminement interne de l'arbre.*

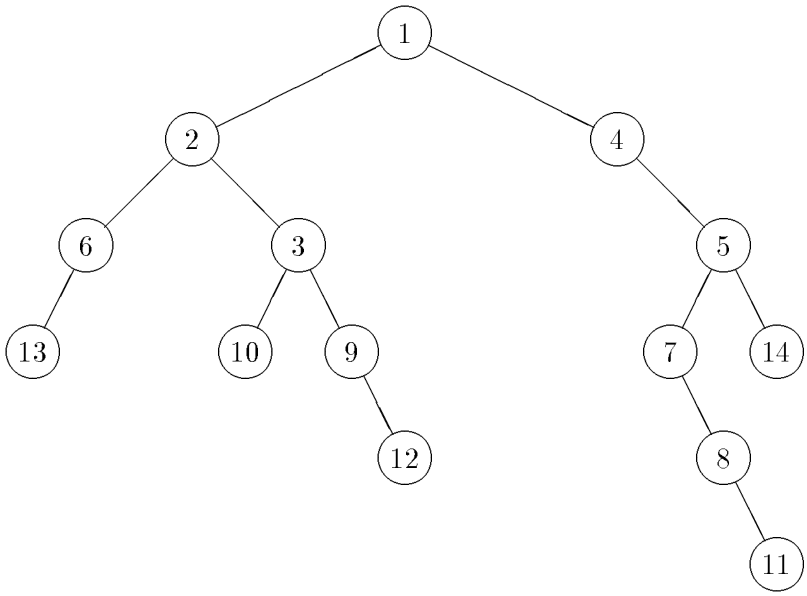
L'analyse en moyenne du tri Quicksort est donc équivalente à l'analyse en moyenne de la construction des arbres binaires de recherche ou de la recherche dans de tels arbres.

Signalons enfin l'isomorphisme entre les arbres binaires de recherche et les tournois binaires (heap) [Fra76]. C'est en particulier cet isomorphisme qu'utilise D. H. Greene [Gre83] pour décrire et analyser les arbres réduits de recherche que nous présentons dans la section suivante.

Un tournoi binaire est un arbre dont les nœuds internes sont étiquetés avec la propriété : la valeur d'un nœud est supérieure aux valeurs dans chacun de ses sous-arbres.

On associe à un arbre binaire de recherche, construit à partir d'une permutation σ donnée, un arbre tournoi de même forme en remplaçant chaque étiquette par son rang dans σ. L'arbre tournoi représente l'ordre de création des différents nœuds de l'arbre binaire de recherche associé.

Exemple II.5 : L'arbre tournoi associé à l'arbre de recherche de l'exemple précédent, c'est-à-dire à $\sigma = 8\ 3\ 5\ 9\ 13\ 2\ 10\ 11\ 6\ 4\ 12\ 7\ 1\ 14$, est :



Réciproquement, si on se donne un tournoi binaire, on vérifie facilement que le parcours en ordre symétrique ou infixe de l'arbre donne exactement la permutation inverse σ^{-1}. On peut donc reconstruire l'unique arbre binaire de recherche associé au tournoi. Dans notre exemple, on obtient ainsi :

$$\sigma^{-1} = \ 13\ 6\ 2\ 10\ 3\ 9\ 12\ 1\ 4\ 7\ 8\ 11\ 5\ 14.$$

2. Médiane et rééquilibrage d'arbres.

2.1. Partition avec médiane.

La première amélioration de l'algorithme de tri que l'on considère concerne le choix de la clé de partitionnement. Cette variante apparaît dans le volume 3 de D. Knuth [Knu73] et dans la thèse de R. Sedgewick [Sed80] qui donne une analyse des coûts moyens dans le cas médiane de 3.

Au lieu de prendre une clé complètement au hasard comme dans le cas standard, on choisit comme clé de partition l'élément médian, dans l'ordre du tri, de $k = 2t + 1$ clés quelconques. On espère ainsi limiter l'importance des cas les plus défavorables en interdisant à l'algorithme de produire des sous-listes de taille trop petite ; les sous-listes ont au moins t éléments. On montrera dans le chapitre suivant que cette méthode permet de gagner sur la constante du terme en $n \ln n$ des coûts moyens, et ceci d'autant plus que t est grand. Cette variante n'est toutefois intéressante d'un point de vue pratique que dans le cas médiane de 3 ou de 5. La détermination de l'élément médian introduit en effet une contribution linéaire dans le coût moyen ; ce surcoût devient rapidement prépondérant si t croît, et annule le gain asymptotique pour des listes de taille raisonnable ($n < 50000$).

L'utilisation de la sélection avec médiane de $2t + 1$ éléments donne pour la clé de partitionnement une distribution d'autant plus resserrée vers le milieu de la liste que t est grand. La probabilité de partitionner en listes de taille p et q une liste de taille $n = p + q + 1$ est en effet :

$$P_{p,q} = \frac{\binom{p}{t}\binom{q}{t}}{\binom{p+q+1}{2t+1}} \tag{II.1}$$

La figure II.1 montre cette distribution des tailles des sous-listes obtenues par partitionnement d'une liste de taille n pour t allant de 0 à 10 ; on considère le cas continu, c'est-à-dire n grand.

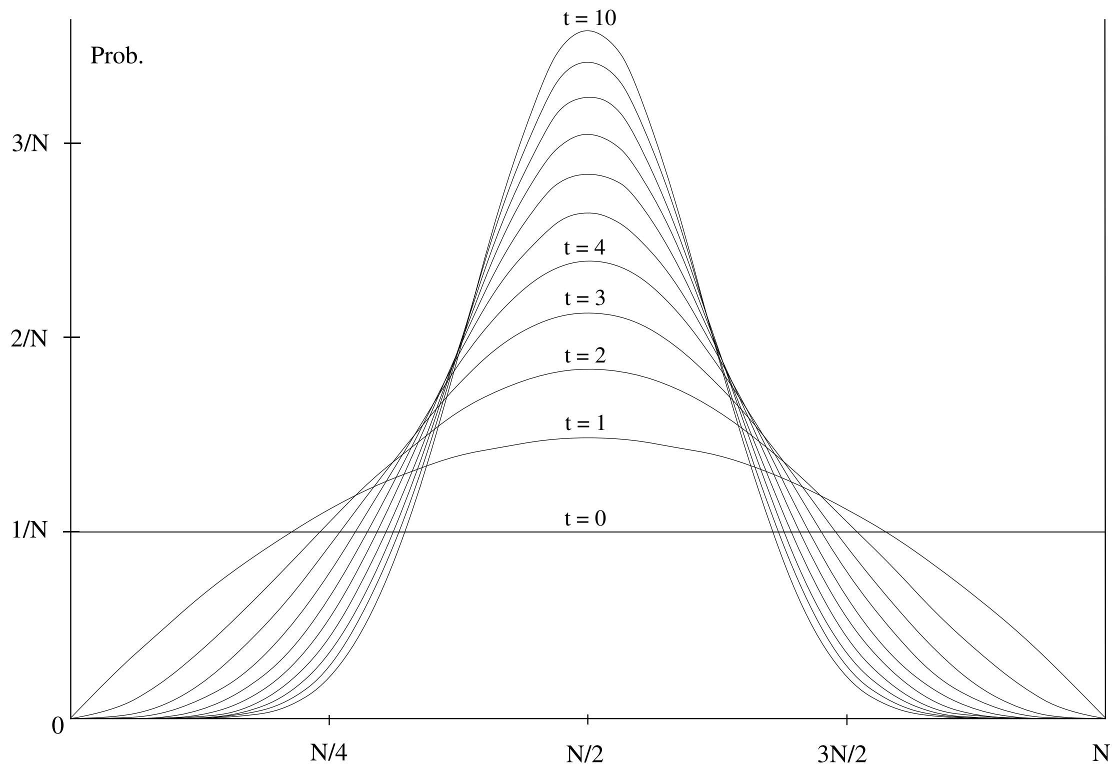


Figure II.1 :Distribution de probabilité pour un partitionnement avec médiane de $2t + 1$ clés.

Pour définir complètement l'algorithme Quicksort avec médiane de k clés, il est nécessaire de préciser comment trier les listes de taille inférieure à k pour lesquelles on ne peut pas appliquer la sélection avec médiane de k. Pour notre analyse, cette question n'est toutefois pas très importante et sera traitée de façon plus générale dans la section 4. de ce chapitre. On considérera, pour le moment, que les listes de taille 0 à $k-1$ ne sont pas traitées par l'algorithme récursif et seront triées ultérieurement.

L'algorithme de référence utilisé dans le cas standard se généralise simplement à la méthode avec médiane de k. On considère que les k clés choisies pour déterminer la clé de partition sont les k premiers éléments de la liste ; le partitionnement s'effectue ensuite normalement en conservant l'ordre des clés. De plus, les petites listes restent temporairement non triées.

Exemple II.6 : L'algorithme de référence, avec médiane de 3, donne ainsi les partitionnements :

8	3	5	9	13	2	10	11	6	4	12	7	1	14
3	2	4	1	**5**	8	9	13	10	11	6	12	7	14
2	1	**3**	4										
					8	6	7	**9**	13	10	11	12	14
					6	**7**	8						
									10	**11**	13	12	14
											12	**13**	14

De même, avec médiane de 5, on obtient :

8	3	5	9	13	2	10	11	6	4	12	7	1	14
3	5	2	6	4	7	1	**8**	9	13	10	11	12	14
3	2	1	**4**	5	6	7							
								9	10	**11**	13	12	14

2.2. Arbres réduits.

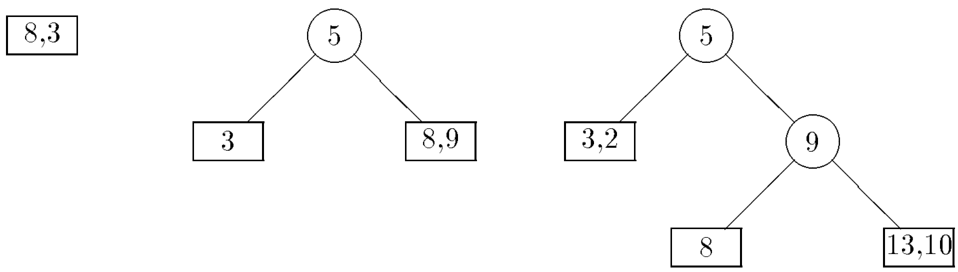
Les arbres réduits correspondent exactement à la transposition de la sélection avec médiane aux arbres binaires de recherche. Ces arbres apparaissent dans les travaux de Walker et Wood [WW76] et de Itai et Rodeh [IR80]. Ils sont analysés de façon plus systématique dans la thèse de D. H. Greene [Gre83] qui détermine, en fonction du paramètre t, les coûts moyens liés aux arbres réduits. Cette analyse utilise une description des arbres tournois associés au moyen de grammaires étiquetées, et débouche sur une résolution analytique proche de celle que l'on présente ici.

L'idée pour les arbres réduits est de retarder la création d'un nœud interne de l'arbre de recherche de façon à mieux choisir la clé associée à ce nœud. On va pour cela observer les valeurs de plusieurs clés qui doivent venir s'insérer sur le sous-arbre correspondant, et créer un sous-arbre équilibré en fonction de cet échantillon de valeurs.

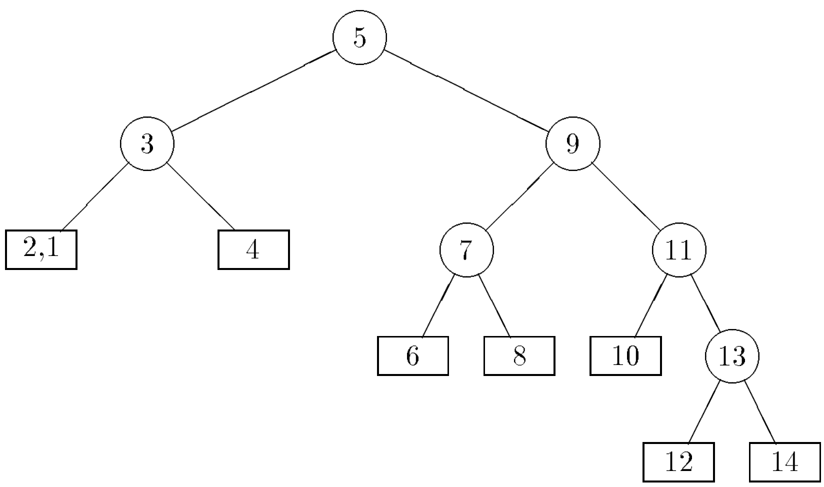
Les arbres réduits sont construits comme les arbres binaires de recherche par insertions successives d'une liste de clés. La différence réside dans la nature des feuilles qui pour les arbres réduits de paramètre $k = 2t + 1$ sont des listes de clés de taille inférieure à k. L'insertion dans une feuille consiste à rajouter la clé dans la liste correspondante. Si ainsi on obtient une feuille pleine, c'est-à-dire contenant k clés, on crée un nouveau nœud interne étiqueté par la valeur de la clé médiane des k éléments. Les autres clés sont alors réparties sur deux feuilles de taille t. Les feuilles ont donc toujours une taille entre t et $2t = k - 1$ sauf initialement pour la racine.

Comme pour le tri avec médiane, les arbres réduits sont complètement définis à condition de préciser la manière de gérer les listes de clés dans chaque feuille : listes triées, arbres de recherche, etc. Le choix entre ces implantations est motivé par les opérations à effectuer sur les feuilles, c'est-à-dire l'insertion, la recherche de l'élément médian et le partitionnement autour de cet élément. On considère pour l'instant que ces listes ne sont pas triées et contiennent les clés dans l'ordre où elles sont insérées.

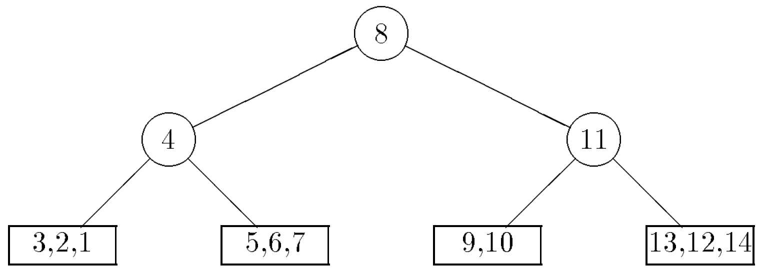
Exemple II.7 : En reprenant la permutation $\sigma = 8\ 3\ 5\ 9\ 13\ 2\ 10\ 11\ 6\ 4\ 12\ 7\ 1\ 14$, l'algorithme d'insertions successives dans un arbre réduit avec médiane de 3 donne les arbres suivants :



et finalement :



En considérant la médiane de 5, on obtient dans ce cas l'arbre réduit :



On voit maintenant facilement l'équivalence combinatoire entre les arbres réduits de paramètre k et l'algorithme Quicksort de référence avec médiane de k. Il suffit pour cela de reprendre l'algorithme d'insertions successives globalement sur l'ensemble de la liste comme précédemment. Chaque nœud créé contient bien l'élément médian des k premières clés arrivant sur ce nœud ; les sous-arbres correspondent alors à chaque sous-liste du partitionnement.

Proposition II.3 (Equivalence) *L'arbre associé au tri d'une permutation par l'algorithme de référence avec médiane de k est l'arbre réduit de paramètre k obtenu par insertions successives des clés de cette permutation.*

Corollaire II.1 *La distribution de la taille des sous-arbres dans les arbres réduits est exactement celle de la taille des sous-listes dans le tri avec médiane (cf. formule II.1 et figure II.1).*

Les coûts pour le tri avec médiane sont donc directement liés aux coûts des arbres réduits.

Une manière équivalente de voir les arbres réduits est de les considérer comme un rééquilibrage local d'arbres binaires de recherche [MP85]. Par exemple, pour médiane de 3, on effectue

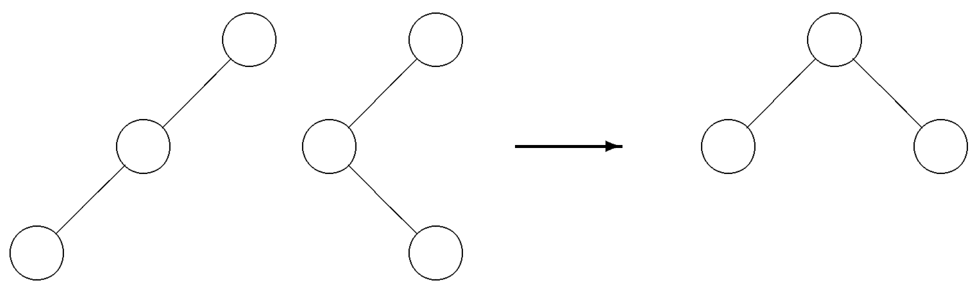


Figure II.2 : Rééquilibrage des arbres réduits (médiane de 3)

une rotation sur les sous-arbres de taille 3 pour obtenir un sous-arbre équilibré comme le montre la figure II.2. Cette opération locale de rééquilibrage doit être effectuée à chaque insertion lors de la construction de l'arbre de recherche pour obtenir l'effet global désiré.

3. Multipartitionnement et arbres s-aires.

3.1. Partitionnements multiples.

Au lieu de partitionner seulement en deux sous-listes, on peut considérer une variante où l'on choisit à chaque étape $s-1$ clés et où l'on partitionne en s sous-listes autour de ces clés. Le cas standard correspond à $s = 2$.

Le cas $s = 3$ est étudié par Sedgewick dans sa thèse [Sed80]. Il apparaît en particulier que pour le tri d'un tableau de valeurs, la méthode est assez inefficace du fait du grand nombre de clés à déplacer pour chaque partitionnement.

Comme dans le cas de la médiane, il faut définir le traitement des listes de taille inférieure strictement à s. On considère pour le moment qu'elles restent non triées. La deuxième chose plus importante à préciser est la façon d'effectuer le multipartitionnement autour de $s-1$ valeurs. Il faut en effet déterminer comment, et avec quel coût, on compare une clé donnée avec les $s-1$ valeurs de partitionnement pour savoir dans laquelle des s sous-listes elle doit aller. Une manière naturelle de voir est de considérer un arbre binaire de recherche de taille fixée construit à partir des $s-1$ valeurs de partitionnement. Cette arbre que l'on appelle arbre de décision donne la suite des comparaisons à effectuer pour déterminer à quel intervalle des $s-1$ valeurs appartient une clé. Le multipartitionnement consiste à faire pour chaque nouvelle clé une recherche infructueuse dans l'arbre de décision et à associer les s feuilles de l'arbre avec les s sous-listes à produire. Cet arbre de décision apparaît obligatoirement, de façon implicite ou explicite, dans la mesure où l'on veut considérer un algorithme de tri opérant uniquement par comparaisons.

Le nombre de comparaisons nécessaires à chaque multipartitionnement est linéaire mais avec un coefficient qui dépend de la forme de l'arbre de décision utilisé. Si $a(s)$ est le coût moyen de parcours de l'arbre, $a(s) = lce/s$, le nombre moyen de comparaisons nécessaires pour partitionner une liste de taille n autour des $s-1$ valeurs quelconques est alors $a(s)\,n + O(1)$. La forme de l'arbre de décision peut à priori être différente pour chaque multipartitionnement ce qui donne un caractère aléatoire à la valeur de $a(s)$. Pour l'analyse en moyenne, on suppose que la forme des arbres de décision est totalement indépendante de la taille n de la liste à partitionner. On peut alors confondre $a(s)$ avec sa valeur moyenne $\bar{a}(s)$ sur la distribution des arbres de décision utilisés.

Dans le meilleur des cas, c'est-à-dire si on construit un arbre de décision optimum ayant ses s feuilles sur 2 niveaux consécutifs, on a :

$$\bar{a}(s) = h + 2\frac{s - 2^h}{s} \quad \text{avec} \quad h = \lceil \log_2 s \rceil \,. \tag{II.2}$$

Les premières valeurs de $\bar{a}(s)$ sont données par le tableau D.1 p. 137.

L'analyse va montrer le faible intérêt pratique du multipartitionnement dans le cas général où $\bar{a}(s)$ est défini par l'équation précédente. Le gain asymptotique, donné par le coefficient du terme en $n \ln n$ pour le nombre moyen de comparaisons, est en effet plus important pour la médiane de 3 que pour le multipartitionnement avec $s = 8$ (cf. table D.3 p. 138). Ce gain est de plus totalement nul dans le cas ternaire $s = 3$. Si on assouplissait la contrainte qui est de faire un tri avec uniquement par comparaisons, on pourrait toutefois imaginer des implantations pour lesquelles le coût moyen d'un partitionnement serait inférieur à $\bar{a}(s)$. On peut en effet se permettre un prétraitement des $s-1$ clés concernées qui n'introduit qu'une contribution linéaire dans les coûts moyens, mais donne un gain significatif sur la terme dominant $O(n \ln n)$.

Comme précédemment, on définit l'algorithme Quicksort de référence avec multipartitionnement en prenant pour éléments de partitionnement les $s-1$ premières clés de la liste et en conservant l'ordre des clés dans les s sous-listes.

Exemple II.8 : algorithme de partitionnement pour $s = 3$ donne donc :

8	3	5	9	13	2	10	11	6	4	12	7	1	14
2	1	**3**	5	6	4	7	**8**	9	13	10	11	12	14
			4	**5**	**6**	7							
								9	10	11	12	**13**	14
									10	**11**	12		

De même pour $s = 4$, on obtient :

8	3	5	9	13	2	10	11	6	4	12	7	1	14
2	1	**3**	4	**5**	6	7	**8**	9	13	10	11	12	14
								9	**10**	11	12	**13**	14

3.2. Arbres s-aires de recherche.

Le multipartitionnement d'ordre s dans le tri Quicksort correspond clairement aux arbres s-aires de recherche définis par :

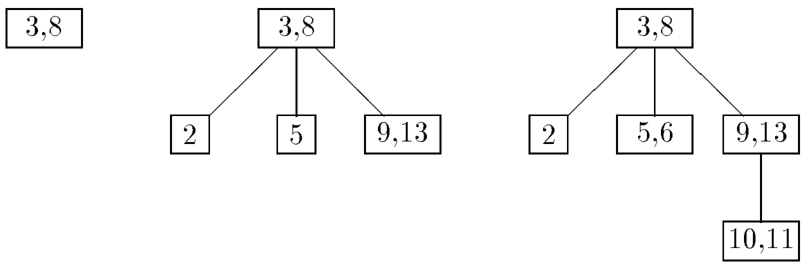
Définition II.2 *Un arbre s-aire de recherche est un arbre dont les nœuds internes contiennent $s-1$ étiquettes et ont s fils. La $i^{\grave{e}me}$ clé d'un nœud doit être supérieure aux clés du $i^{\grave{e}me}$ sous-arbre et inférieure aux clés du $i+1^{\grave{e}me}$ sous-arbre.*

Les arbres $s-aires$ de recherche apparaissent dans Knuth [Knu73] et sont analysés de manière plus détaillée dans [Mah86], [MP89], et [BY87]. H. M. Mahoud et B. Pittel obtiennent en particulier des résultats de distribution limite qui sont complémentaires de l'analyse faite au chapitre IV.

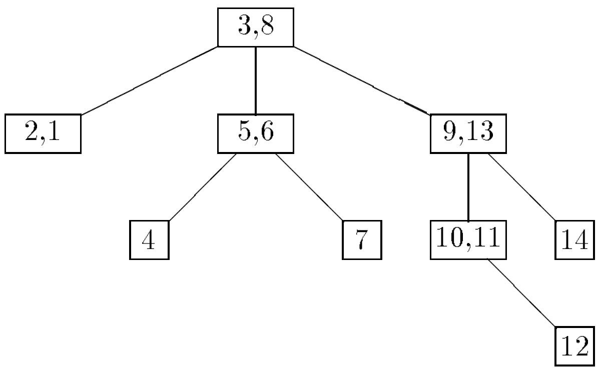
Les algorithmes de construction ou de recherche dans un arbre s-aire sont tout à fait similaires au cas binaire ; seul le parcours des nœuds internes est modifié. Comme pour le multipartitionnement, les nœuds de l'arbre sont, dans le cas le plus général, des arbres binaires de recherche. On peut donc voir l'ensemble de l'arbre s-aires comme un arbre binaire de recherche. Le rééquilibrage apparaît alors comme une optimisation des nœuds s-aires, c'est-à-dire de parties connexes de taille $s-1$ de l'arbre binaire de recherche. Ceci montre en particulier l'équivalence entre le cas ternaire $s = 3$ et le cas standard $s = 2$, car il n'y a rien à optimiser sur un arbre binaire à deux éléments.

L'utilisation des arbres s-aire n'est toutefois intéressante dans la pratique que si on arrive à gagner sur la constante $\bar{a}(s)$, c'est-à-dire sur le coût moyen de traversée d'un nœud interne de l'arbre.

Exemple II.9 : En reprenant la permutation $\sigma = 8\ 3\ 5\ 9\ 13\ 2\ 10\ 11\ 6\ 4\ 12\ 7\ 1\ 14$, l'insertion successive dans un arbre ternaire de recherche ($s = 3$) donne les arbres :



et finalement:



Cet arbre est à rapprocher de l'arbre binaire de recherche obtenu dans le cas standard (exemple II.4 p. 16). On a simplement regroupé des paires de nœuds sans gain sur la longueur de cheminement réelle, c'est-à-dire en tenant compte du coût de traversée d'un nœud ternaire ($a(3) = 5/3$).

4. Elagage ou tri partiel.

Si on étudie l'algorithme Quicksort sur les listes de petite taille, on s'aperçoit qu'il est plus coûteux que d'autres méthodes classiques de tri. Par exemple, le tri par insertion est plus rapide, en nombre moyen de comparaisons, que Quicksort pour des listes de taille inférieure à 14 (cf. tableau II.1).

N	2	3	4	5	6	12	13	14	15
Quicksort : $2(n+1)(H_{n+1} - \frac{4}{3})$	3	6	9.5	13.4	17.6	48	53.7	59.5	65.5
Insertion : $(n-1)(n/4+1)$	1.5	3.5	6	9	12.5	44	51	58.5	66.5

Tableau II.1 : Nombre moyen de comparaisons pour trier avec Quicksort ou par insertion.

L'idée pour améliorer Quicksort est donc de transformer légèrement l'algorithme de manière à utiliser une méthode de tri plus rapide pour les listes de taille inférieure ou égale à un paramètre M donné. Une analyse précise du coût réel de l'algorithme permet alors de déterminer la valeur optimale de M pour une implantation donnée (cf. [Sed80]).

L'utilisation d'une méthode de tri différente pour des petites listes est aussi motivée par les variantes du multipartitionnement ou de la médiane qui ne savent pas trier les listes de taille inférieure à s ou $k = 2t + 1$.

D'un point de vue algorithmique, cette modification ne pose aucun problème. Il suffit juste de changer le test d'arrêt des appels récursifs dans la procédure Quicksort. En reprenant la procédure décrite au début de ce chapitre, on a donc :

Procedure Tri_rapide(L : liste);
    Si taille(L) $\leq M$
        alors retourner(Autre_Tri(L));
        sinon Choisir un élément v de L;
                Construire les listes L1 et L2 contenant les clés de L
                    respectivement plus petites et plus grandes que v;
                Retourner(Tri_rapide(L1).v.Tri_rapide(L2));
    fin si ;
fin ;

Remarque : Si on utilise le tri par insertion pour les petites listes, il n'est pas nécessaire d'effectuer ce tri pour chaque sous-liste de taille $\leq M$ rencontrée. On peut en effet, avec un coût équivalent, appliquer Quicksort sans trier les petites listes et n'effectuer le tri par insertion que sur l'ensemble de la liste partiellement triée ainsi obtenue (cf. [Sed80]). On a de cette manière un gain sur le nombre d'appels de procédures.

Au niveau des arbres binaires de recherche associés, le tri différent des petites listes revient à regrouper chaque sous-arbre de taille $\leq M$ sous la forme d'une feuille contenant la liste triée des étiquettes. Cette opération d'élagage ou de compactage des sous-arbres ne s'effectue de façon naturelle qu'une fois l'arbre de recherche complètement construit. Elle n'apparaît donc pas très utile d'un point de vue pratique, sauf pour les variantes des arbres réduits ou des arbres s-aires où l'on pose $M = 2t$ ou $M = s - 2$.

5. Algorithme général.

On se propose maintenant de combiner l'ensemble des variantes décrites précédemment sous la forme d'un seul algorithme paramétré par les trois valeurs s, k, et M qui décrivent respectivement le multipartitionnement, la sélection avec médiane, et tri partiel des listes. L'une des originalités de notre travail est de pouvoir analyser directement cet algorithme général pour obtenir des expressions asymptotiques des coûts moyens ou des moments d'ordre supérieur en fonction de s, k et M.

Pour trier une liste, on choisit un échantillon de $k = s(t + 1) - 1$ éléments quelconques de la liste et on détermine les $s - 1$ clés équiréparties parmi ces k valeurs. C'est-à-dire que l'on prend la $t+1^{\text{ème}}$, la $2(t+1)^{\text{ème}}$ jusqu'a la $(s-1)(t+1)^{\text{ème}}$ clés dans la liste triée des k éléments. Les $s-1$ clés ainsi choisies découpent donc l'échantillon de taille k en s intervalles de taille t.

On partitionne ensuite l'ensemble de la liste en s sous-listes autour de ces $s-1$ valeurs et on réitère récursivement l'algorithme sur chaque sous-liste de taille supérieure strictement à M. On a donc à priori $M \geq k-1$.

L'algorithme de référence consiste dans ce cas à prendre comme échantillon les k premiers éléments de la liste, et à partitionner en conservant l'ordre dans chaque sous-liste.

L'équivalent de cet algorithme pour les arbres de recherche revient donc à travailler avec des arbres s-aires réduits. Les feuilles sont alors des listes de taille comprise entre t et $k=s(t+1)-1$. L'éclatement d'une feuille consiste à déterminer les $s-1$ clés du nouveau nœud interne qui séparent la liste des k valeurs en s nouvelles feuilles de même taille t.

Une généralisation simple de la formule II.1 page 18, donne la distribution de la taille des sous-listes ou des sous-arbres dans le cas de l'algorithme général. La probabilité de partitionner une liste de taille n en s listes de tailles $n_1, n_2, \ldots, n_s$ est en effet :

$$P_{n_1,n_2,\ldots,n_s} = \frac{\binom{n_1}{t}\binom{n_2}{t}\cdots\binom{n_s}{t}}{\binom{n}{st+s-1}} \qquad \text{avec} \quad \sum_{i=1}^{s} n_i = n-(s-1)\,. \tag{II.3}$$

Dans la section 4. du chapitre suivant, nous considérerons l'analyse en moyenne d'un algorithme encore un peu plus général. On suppose que les $s-1$ clés de partitionnement ne découpent pas obligatoirement les k clés de l'échantillon de manière équitable. Dans la pratique, il est en effet plus simple de faire un partage à peu près équitable en moyenne que de découper en intervalles de taille exactement t.

Plus précisément, étant donnés k et s, on considère que les $s-1$ clés de partitionnement sont déterminées de telle sorte qu'elles découpent les k valeurs de l'échantillon en s intervalles de tailles $t_1, t_2, \ldots, t_s$, et ceci avec une probabilité $P(t_1, t_2, \ldots, t_s)$. L'algorithme est donc défini à partir d'une distribution de probabilité $P(t_1, t_2, \ldots, t_s)$ pour $\sum t_i = k-(s-1)$. L'analyse montrera que le coût moyen est asymptotiquement optimum dans le cas équiréparti, c'est-à-dire si $P(t_1, t_2, \ldots, t_s) = 0$ dès que deux t_i diffèrent de plus d'une clé.

La formule II.3 se généralise simplement et donne l'expression suivante pour la probabilité de partitionner en sous-listes de tailles $n_1, n_2, \ldots, n_s$ avec $\sum_{i=1}^{s} n_i = n-(s-1)$:

$$P_{n_1,n_2,\ldots,n_s} = \sum_{t_1+\cdots+t_s=k-s+1} P(t_1, t_2, \ldots, t_s)\, \frac{\binom{n_1}{t_1}\binom{n_2}{t_2}\cdots\binom{n_s}{t_s}}{\binom{n}{k}}\,. \tag{II.4}$$

L'algorithme général qui sert de base à notre analyse en moyenne peut être décrit par la procédure suivante qui dépend de trois constantes M, s et k avec $M+1 \geq k \geq s-1$ et d'une procédure Autre-Tri pour les petites listes.

Procedure Tri_rapide(L : liste);
  si taille(L) $\leq M$

alors retourner(Autre_Tri(L));
sinon Choisir un échantillon de k éléments quelconques de L;
Sélectionner $s-1$ clés $v_1, \ldots, v_{s-1}$ parmi cet échantillon de taille k ;
Construire un arbre de décision $A(v_1, \ldots, v_{s-1})$ à partir des clés $v_1, \ldots, v_{s-1}$;
Construire avec la procédure partition les listes $L_1, \ldots, L_s$ telles que :
clés $(L_i) < v_i <$ clés (L_{i+1}) pour $i \in [1..s-1]$;
Retourner(Tri_rapide(L_1).v_1.Tri_rapide(L_2).$\cdots$.v_{s-1}.Tri_rapide(L_s));
fin si ;
fin ;

La procédure de partitionnement est alors décrite par :

Procedure partition(L : liste ; $v_1, \ldots, v_{s-1}$:clé);
Pour chaque $x \in L/\{v_1, \ldots, v_{s-1}\}$ faire
Rechercher x dans l'arbre binaire de recherche $A(v_1, \ldots, v_{s-1})$;
Insérer x dans la liste L_i où i est l'indice de la feuille où aboutit la recherche de x ;
fin Pour chaque ;
fin ;

Pour notre analyse en moyenne, il n'est pas nécessaire de préciser l'algorithme de construction des arbres de décision ou la procédure de tri des petites listes. Il suffit simplement de connaître la valeur $\bar{a}(s)$ du coût moyen de la recherche infructueuse dans les arbres de décision.

De plus, à l'exception de la section III.4., la sélection des $s-1$ clés de partitionnement se fait de manière équirépartie. C'est-à-dire que k est de la forme $k = st + s - 1$ et que les $s-1$ clés sélectionnées partitionnent les k éléments de l'échantillon en s listes de taille t.

Pour obtenir l'algorithme de référence qui sert de base à la description de la structure combinatoire du tri rapide, il suffit d'imposer les conditions supplémentaires :

1. l'échantillon de k clés est constitué des k premiers élément de la liste L,
2. la procédure de partition parcourt dans l'ordre les éléments x de la liste L (boucle **Pour chaque**), et insère toujours en fin des listes L_i.

Sous des hypothèses de répartition uniforme des données, ces conditions ne modifient toutefois pas la structure combinatoire de l'algorithme et donc les résultats de l'analyse en moyenne.

Chapitre III

Analyse en moyenne du tri.

Nous allons analyser les coûts moyens liés à l'exécution de l'algorithme Quicksort général avec médiane, multipartitionnement, et tri distinct des petites listes. Compte tenu du principe d'équivalence discuté au chapitre précédent, cette analyse s'applique aussi à l'étude des arbres réduits s-aires de recherche. Le tri présente toutefois un caractère un peu plus général car il correspond à un champ plus large d'implantations possibles en pratique.

La moyenne est définie à partir des données possibles pour l'algorithme. On suppose ici que les données sont des listes constituées de clés distinctes avec une distribution uniforme sur l'ordre relatif des clés. Dans le chapitre V, on étudiera le cas où les données peuvent contenir des clés identiques.

L'analyse va consister dans un premier temps à décrire formellement la structure récursive de l'algorithme par une équation sur les classes de structures combinatoires associées à l'ensemble des données possibles. Pour cela, l'outil principal est l'opérateur de produit partitionnel introduit par D. Foata [Foa74] qui s'apparente à un produit cartésien sur des structures étiquetées et va permettre de traduire l'opération de partitionnement proprement dite. Nous introduirons aussi un opérateur d'enracinement sur les classes de structures combinatoires pour prendre en compte les mécanismes liés au choix des clés de partition. On obtient ainsi une description formelle récursive des classes de coût associées à l'algorithme général (cf. théorème III.1 p. 40).

Cette description se traduit immédiatement par une équation différentielle sur les séries génératrices associées aux coûts moyens de l'algorithme. Nous donnons ici une méthode d'intégration assez systématique pour le type d'équation obtenue qui permet une résolution synthétique et donne des solutions paramétrées par les valeurs caractéristiques de l'algorithme s, t, et M. L'utilisation de lemmes de transfert (cf. P. Flajolet et A. M. Odlyzko [FO90]) permet en outre de n'effectuer qu'une résolution partielle pour l'obtention de développements asymptotiques des coûts moyens. On donne ainsi la forme générale des coûts moyens et un moyen rapide de les calculer de façon exacte pour s et t fixés ou asymptotique avec un terme d'erreur en $o(n)$ négligeable dans la pratique, c'est-à-dire pour des valeurs usuelles de s et t. La même méthode s'applique pour l'étude de la distribution des coûts dans le chapitre suivant et permet un calcul effectif des moments.

Cette partie analytique est à rapprocher du travail de D. H. Greene [Gre83] sur le coût moyen des arbres réduits de recherche dans le cas binaire $s = 2$. L'originalité de notre travail réside d'une part dans la généralisation de l'algorithme considéré, intégration des paramètres s et M, et d'autre part dans le traitement plus systématique des règles d'intégration et des équivalences asymptotiques sur les séries et sur les coûts moyens. Cette approche qui facilite l'analyse des coûts moyens devient primordiale pour la détermination des moments d'ordre supérieur.

Les formules donnant les coûts moyens pour les quantités caractéristiques introduites par R. Sedgewick [Sed80] sont regroupées dans les sections 1. et 2. de l'annexe C.

1. Description combinatoire.

1.1. Hypothèses probabilistes.

On suppose que les algorithmes de tri ou de recherche opèrent sur des listes constituées de clés toutes distinctes. Ces algorithmes travaillent uniquement par comparaison de clés, ils ne font donc intervenir que l'ordre relatif des éléments et non leur valeur. On assimile alors l'ensemble des données à l'ensemble S_n des permutations sur $[1..n]$. Le passage d'une liste à la permutation associée s'effectue en remplaçant les clés par leur numéro d'ordre dans la liste triée.

Si une liste L, associée à la permutation σ, se partitionne en $L_1, L_2, \ldots, L_s$ autour de $s-1$ clés données, on note $\sigma_1, \sigma_2, \ldots, \sigma_s$ les permutations associées à chaque sous-liste et σ_0 la permutation obtenue à partir de σ en supprimant les $s-1$ clés de partition. De plus, pour simplifier les notations, on identifie les couples de permutations $(\sigma_1, \sigma_2) \in S_p \times S_q$ aux couples (σ_1, σ_2') où σ_2' est la permutation sur les éléments de $[p+2..p+q+1]$ obtenue à partir de σ_2 en décalant les étiquettes de $p+1$; on a ainsi $(13425, 231) \Leftrightarrow (13425, 897)$. De la même façon, on identifie les s-uplets de $S_{p_1} \times S_{p_2} \times \cdots \times S_{p_s}$ aux s-uplets de $S(1) \times S(2) \times \cdots \times S(s)$ où $S(i)$ désigne les permutations sur les entiers de $p_1 + \ldots + p_{i-1} + i$ à $p_1 + \ldots + p_{i-1} + p_i + i - 1$.

Hypothèse H.I (sur les données) *Les données de tailles n sont considérées comme des permutations de S_n avec une distribution de probabilité uniforme ; la probabilité de chaque permutation valant donc $1/n!$.*

Cette hypothèse donne le modèle probabiliste utilisé pour définir la valeur moyenne et la distribution de probabilité des coûts que l'on analyse. La condition H.I est standard pour l'analyse en moyenne [Knu73], [Sed80] et correspond en particulier au cas où les clés sont tirées indépendamment suivant une même loi de distribution continue (I.I.D.). Le tirage des clés suivant une loi de distribution discrète décrit lui des données réparties uniformément sur l'ensemble des mots de taille n et sera traité dans le chapitre V sur le tri avec clés répétées.

Hypothèse H.II (sur l'algorithme) *L'algorithme de partitionnement ou de multipartitionnement conserve l'uniformité des distributions sur chacune des permutations produites. C'est-à-dire que pour des clés de partitionnement fixées et donc des tailles de sous-listes $p_1, \ldots, p_s$ fixées, chaque s-uplet $(\sigma_1, \ldots, \sigma_s)$ de $S_{p_1} \times \cdots \times S_{p_s}$ provient le même nombre de fois du partitionnement des permutations $\sigma_0 \in S_{p_1+\cdots+p_s}$.*

Cette condition est cruciale pour l'analyse car elle permet d'exprimer la structure récursive de l'algorithme par une relation de récurrence sur les coûts totaux, c'est-à-dire la somme des coûts sur l'ensemble des permutations. En pratique, il n'est pas toujours simple de montrer qu'un algorithme vérifie l'hypothèse H.II. On dispose toutefois d'un critère simple et relativement général qui implique l'uniformité de l'algorithme de partitionnement.

Critère H.III *L'algorithme de partitionnement n'effectue des comparaisons qu'avec les $s-1$ clés de partitionnement.*

Ce critère est raisonnable dans la mesure où il correspond à des implantations naturelles et efficaces du tri ou de la recherche. Les comparaisons entre deux clés à partitionner apparaissent en effet soit superflues soit équivalentes à des comparaisons directes avec les clés de partition. L'idée sous-jacente est que la seule information nécessaire est la position de chaque clé par rapport aux $s-1$ clés de partition ; toute information supplémentaire est donc inutile ou risque de détruire l'uniformité des sous-listes. Ce critère permet de plus une compréhension simple de la structure combinatoire de l'algorithme.

Proposition III.1 *Le critère H.III implique l'hypothèse H.II.*

Preuve : Plaçons-nous dans le cas standard s=2.

Soit p et q fixés, une méthode de partitionnement autour de la clé $v = p+1$ est caractérisée par la donnée pour chaque permutation σ_0 de taille $p+q$, de la permutation $\Sigma(\sigma_0)$ qui fait passer de σ_0 à $\sigma_1.\sigma_2$. On appelle marquage de σ_0 relativement à p, $M(\sigma_0)$, l'ensemble des positions dans σ_0 des p plus petites clés. On peut, par exemple, représenter $M(\sigma_0)$ par un vecteur sur $\{0,1\}^{p+q}$.

Le critère H.III implique alors que $\Sigma(\sigma_0)$ dépend uniquement du marquage $M(\sigma_0)$. L'algorithme de partitionnement ne peut en effet différencier entre-elles les clés $< v$ (resp. $> v$) ; il a donc un comportement identique pour toutes les permutations de σ_0 qui ont un même marquage.

Ainsi les $p!q!$ permutations de σ_0 qui correspondent à un même marquage M, et donc à une même permutation $\Sigma(\sigma_0)$, produisent par partition une fois et une seule chaque paire (σ_1, σ_2) de permutations des p plus petites et des q plus grandes clés de σ_0. D'où la condition H.II. □

La preuve s'étend facilement pour s quelconque en considérant des vecteurs de $[1..s]^n$ pour les marquages.

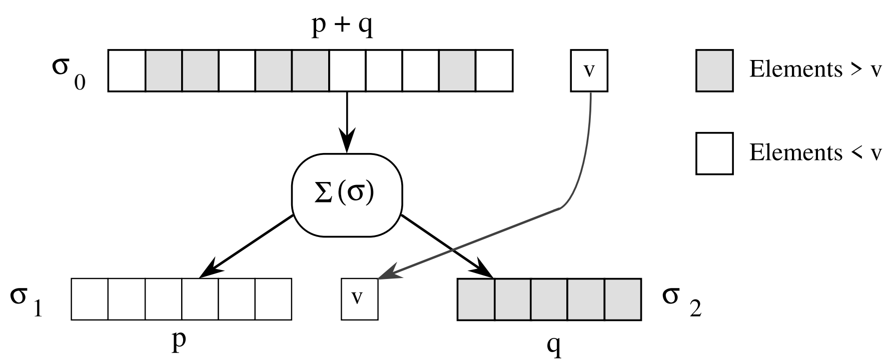


Figure III.1 : Schéma du partitionnement dans le cas binaire $s = 2$.

Globalement pour des tailles de sous-listes $p_1, \ldots, p_s$ fixées avec $p_1 + \cdots + p_s = n$, chaque s-uplet $(\sigma_1, \ldots, \sigma_s) \in S_{p_1} \times \cdots \times S_{p_s}$ provient donc du partitionnement de $n!/p_1!p_2!\cdots p_s!$ permutations $\sigma_0 \in S_n$ à raison d'une par marquage.

Si on considère l'algorithme de référence, on peut voir le passage d'un s-uplet $(\sigma_1, \ldots, \sigma_s)$ à l'une de ses permutations σ_0 antécédentes comme une opération de mélange ou *shuffle*. Un mélange (battage de cartes à l'américaine) consiste à intercaler les éléments des différentes permutations en conservant l'ordre initial dans chaque σ_i. Chaque manière d'intercaler les clés (et donc chaque mélange) est ainsi clairement associée à un marquage M et produit la permutation antécédente σ_0 correspondant à ce marquage M.

Dans le cas plus général d'un algorithme vérifiant le critère H.III, l'opération de shuffle est légèrement différente (Q-shuffle). Chaque algorithme vérifiant H.III détermine sa propre bijection entre les marquages possibles M et les permutations de partitionnement Σ. Pour un marquage donné, on à donc une permutation fixe qui définit la modification de l'ordre des clés entre une sous-liste et la liste antécédente. On peut donc voir l'opération de Q-shuffle en trois étapes. On détermine d'abord le marquage M de la liste antécédente que l'on veut produire. On effectue ensuite sur chacune des listes σ_i une permutation $\Pi_i(M)$ déterminée par le marquage et par l'algorithme. On opère enfin le mélange proprement dit de la même manière que pour l'algorithme de référence.

Remarque : Dans notre analyse, nous supposons que les algorithmes sont déterministes ; en particulier les clés de partitions et $\Sigma(\sigma)$ sont bien déterminés pour chaque permutation. Toutefois, des implantations non déterministes qui respectent les hypothèses H.I et H.II conduisent exactement aux mêmes résultats car elles correspondent à chaque étape aux mêmes distributions de probabilité des clés et des tailles de listes. Le fait de choisir les clés de partitionnement à des positions aléatoires est par exemple équivalent au choix des clés à des positions fixes. Une approche possible de l'analyse (cf. [Rös89]) peut même consister à vouloir trier une liste déjà triée en introduisant le caractère aléatoire seulement au niveau du choix uniforme des clés de partitionnement.

1.2. Description formelle et produit partitionnel.

On considère la classe des permutations $\mathbf{S} = \cup \mathbf{S_n}$ et on veut étudier une fonction de coût $C(\sigma)$ définie sur les permutations. C peut par exemple compter le nombre de comparaisons utilisées pour trier une permutation. On introduit alors la classe pondérée $\mathbf{C}(\mathbf{S})$ constituée formellement des éléments $C(\sigma).\sigma$ ayant pour taille $|\sigma|$.

On associe aux classes de structures combinatoires $\mathbf{S}$ et $\mathbf{C}(\mathbf{S})$, les séries génératrices exponentielles définies par :

$$S(z) = \sum_{\sigma \in S} \frac{z^{|\sigma|}}{|\sigma|\,!} = \sum_n n! \frac{z^n}{n!} = \frac{1}{1-z} \quad \text{et} \quad C(z) = \sum_{\sigma \in S} C(\sigma) \frac{z^{|\sigma|}}{|\sigma|\,!} = \sum_n C_n \frac{z^n}{n\,!} \tag{III.1}$$

où C_n désigne le coût total sur l'ensemble des permutations de S_n. D'après l'hypothèse d'uniformité des données H.I, $C_n/n!$ est donc le coût moyen sur les données de taille n et $C(z)$ correspond aussi à la série génératrice ordinaire de ce coût moyen.

Pour une permutation σ qui se décompose en $\sigma_1, \ldots, \sigma_s$ après un partitionnement, la structure récursive de l'algorithme permet de décrire le coût sous la forme :

$$C(\sigma) = CP(\sigma) + C(\sigma_1) + C(\sigma_2) + \cdots + C(\sigma_s) \tag{III.2}$$

où la fonction $CP(\sigma)$ compte le coût d'un appel de la procédure de tri sur σ. Pour le nombre de comparaisons, on a par exemple $CP(\sigma) \approx |\sigma|$. On introduit donc la classe pondérée $\mathbf{CP(S)}$ et la série génératrice exponentielle $CP(z)$ associées à cette fonction de coût. Cette série est bien sûr la série génératrice ordinaire du coût moyen d'un partitionnement.

Reprenant la démarche classique présentée dans le premier chapitre, nous allons définir récursivement la classe $\mathbf{S}$ des permutations en utilisant la décomposition $\sigma \longrightarrow (\sigma_1, \ldots, \sigma_s)$ induite par l'algorithme de partitionnement. Pour cela, on utilise le produit partitionnel $\otimes$ introduit par D. Foata [Foa74]. Le produit partitionnel est une extension du produit cartésien pour des structures étiquetées et correspond intuitivement à l'opération de mélange ou de shuffle décrite précédemment. Les k-partitions décrivent alors les marquages associés à chaque mélange de k structures.

Définition III.1 *Une k-partition de l'intervalle $[1..n]$ est un k-uplet $(I_1, \ldots, I_k)$ tel que :*

$$\bigcup_{i=1}^{k} I_i = [1..n] \qquad \textit{et} \qquad I_i \cap I_j = \emptyset \quad \textit{pour} \quad i \neq j.$$

Définition III.2

- *Le produit partitionnel de deux structures s_1 et s_2 de taille p et q est l'ensemble $s_1 \otimes s_2$ des paires $[(s_1, I_1), (s_2, I_2)]$ où (I_1, I_2) est une bipartition de $[1..p+q]$ avec $card(I_1) = p$ et $card(I_2) = q$. La taille d'un élément de $s_1 \otimes s_2$ est alors $p+q$, la cardinalité de $s_1 \otimes s_2$ est $(p+q)!/p!q!$.*
- *Le produit partitionnel $\mathbf{C_1} \otimes \mathbf{C_2}$ de deux classes de structures combinatoires $\mathbf{C_1}$ et $\mathbf{C_2}$ est la réunion des ensembles $s_1 \otimes s_2$ pour $(s_1, s_2) \in \mathbf{C_1} \times \mathbf{C_2}$.*

Ces définitions s'étendent facilement au produit partitionnel de k classes de structures combinatoires en considérant les k-partitions.

Définition III.3 *Le produit partitionnel de k structures s_i, de taille p_i avec $\sum p_i = n$ pour i allant de 1 à k, est l'ensemble des $n!/p_1! \cdots p_k!$ k-uplets $[(s_1, I_1), \ldots, (s_k, I_k)]$ où $(I_1, \ldots, I_k)$ est une k-partition de $[1..n]$ avec $card(I_i) = p_i$.*

Remarque : Le produit partitionnel est un opérateur associatif au même titre que le produit cartésien, c'est-à-dire à un isomorphisme naturel près. Le produit partitionnel de k classes donné par la définition III.3 est donc équivalent à ce que l'on obtient en faisant les produits partitionnels deux à deux. On retrouve là la décomposition canonique des coefficients multinomiaux.

Pour traduire le rôle joué par les clés de partitionnement, nous devons aussi introduire un opérateur d'enracinement qui décrit le passage entre σ et σ_0. On peut voir cette opération d'enracinement comme une concaténation par une permutation de taille donnée qui représente les $s-1$ clés de partitionnement.

Définition III.4

- *On appelle enracinement d'une classe* $\mathbf{C}$*, la classe* $\circ(\mathbf{C})$ *contenant les mêmes éléments que* $\mathbf{C}$ *avec une taille incrémentée de 1 :* $|x|_{\circ(\mathbf{C})} = |x|_{\mathbf{C}} + 1$.

- *On appelle enracinement multiple d'ordre i, la classe* $\circ_i(\mathbf{C}) = \mathbf{S_i} \times \mathbf{C}$ *avec la fonction taille induite par le produit cartésien :* $|(\sigma, x)|_{\circ_i(\mathbf{C})} = |x|_{\mathbf{C}} + i$.

Considérons maintenant l'algorithme Quicksort standard, c'est-à-dire avec partitionnement binaire, sans médiane et sans traitement particulier des petites listes ($s = 2, t = 0, M = 0$). Pour décrire combinatoirement la structure du partitionnement, on sépare la décomposition caractéristique $\sigma \longrightarrow (\sigma_1, \sigma_2)$ en deux étapes :

1. Choix de la clé v de partition : $\sigma \longrightarrow (v, \sigma_0)$ où σ_0 est obtenue à partir de σ en supprimant la clé de partitionnement. On suppose que la clé v est prise à une position fixe indépendante de σ, par exemple la première clé de la liste dans le cas l'algorithme de référence. L'hypothèse H.I rend toutefois équivalents les différents choix possibles.

2. Partitionnement de σ_0 autour de la clé v : $(v, \sigma_0) \longrightarrow (\sigma_1, \sigma_2)$ avec $|\sigma_1| = v - 1$.

En introduisant la classe $\mathbf{U} = \{(j, \sigma)/\sigma \in \mathbf{S} \text{ et } 0 \leq j - 1 \leq |\sigma|\}$ avec $|(j, \sigma)|_{\mathbf{U}} = |\sigma|$, on a alors les propriétés suivantes :

Proposition III.2 *1. L'application* $\sigma \longrightarrow (v, \sigma_0)$ *est un isomorphisme de* $\mathbf{S} - \mathbf{S_0}$ *dans* $\circ(\mathbf{U})$.

2. L'application $(v, \sigma_0) \longrightarrow (\sigma_1, \sigma_2)$ *est un isomorphisme de* $\mathbf{U}$ *dans* $\mathbf{S} \otimes \mathbf{S}$.

Preuves :

1. Etant donné un couple $(v, \sigma_0) \in \circ(\mathbf{U})$, on obtient l'unique antécédent $\sigma \in \mathbf{S}$ comme suit : On augmente de un la valeurs des étiquettes $\geq v$ dans σ_0, et on insère la clé v à la position fixée par la stratégie de choix de l'élément de partition. La correspondance des tailles des structures combinatoires provient alors directement de l'utilisation de l'opérateur d'enracinement $|\sigma| = |\sigma_0| + 1$. On note de plus que $\circ(\mathbf{U})$ ne contient pas d'élément de taille 0 ce qui explique que l'on considère la classe $\mathbf{S} - \mathbf{S_0}$.

2. On a une correspondance naturelle entre les bipartitions de $[1..n]$ et les marquages de permutations de taille n. Pour un algorithme vérifiant le critère H.III, on associe donc bijectivement les éléments du produit partitionnel $\sigma_1 \otimes \sigma_2$ aux antécédents σ_0 de (σ_1, σ_2). On effectue pour cela l'opération de Q-shuffle associée à chaque bipartition du produit partitionnel. La valeur de v est alors donné par $v = |\sigma_1| + 1$.

   Dans le cas général défini par l'hypothèse H.II, l'isomorphisme n'est pas naturel mais est donné par un argument de dénombrement. L'équiprobabilité du partitionnement implique en effet qu'un couple de permutations (σ_1, σ_2) de taille p et q provient exactement du partitionnement autour de $v = p + 1$ de $(p + q)!/p!q!$ permutations σ_0 de taille $p + q$. L'égalité des cardinalités implique donc l'existence de l'isomorphisme recherché.

$\square$

Corollaire III.1 *Il existe un isomorphisme entre les classes* $\mathbf{S}$ *et* $\circ(\mathbf{S} \otimes \mathbf{S}) + \mathbf{S_0}$ *qui respecte la décomposition en sous-listes induite par l'algorithme Quicksort standard* ($s = 2$, $t = 0$, $M = 0$).

Ce corollaire nous permet donc d'identifier les deux classes et de décomposer selon la relation de récurrence III.2 page 32, la classe $\mathbf{C}(\mathbf{S})$ du coût de l'algorithme. On obtient ainsi la description formelle suivante :

$$\begin{aligned} \mathbf{S} &= \circ(\mathbf{S} \otimes \mathbf{S}) + \mathbf{S_0} \\ \mathbf{C}(\mathbf{S}) &= \mathbf{CP}(\mathbf{S}) + \circ(\mathbf{C}(\mathbf{S}) \otimes \mathbf{S}) + \circ(\mathbf{S} \otimes \mathbf{C}(\mathbf{S})) \end{aligned} \tag{III.3}$$

où $\mathbf{CP}(\mathbf{S})$ est la classe pondérée associée au coût d'une étape de l'algorithme. La composante $\mathbf{C}(\mathbf{S_0})$ du coût pour les listes de taille nulle est comprise dans $\mathbf{CP}(\mathbf{S})$.

On peut maintenant traduire directement cette description formelle de la structure récurrente de l'algorithme par des équations sur les séries génératrices $S(z)$ et $C(z)$. On utilise pour cela les lemmes suivants qui donnent la correspondance entre les opérateurs formels sur les classes et des opérateurs simples sur les séries génératrices (cf. [VF89]).

Lemme III.1 *Pour les séries génératrices exponentielles* $A(z)$, $B(z)$, $C(z)$ *associées à des classes de structures combinatoires* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, *on a :*

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \Longrightarrow C(z) = A(z)B(z).$$

$$\mathbf{C} = \circ(\mathbf{A}) \Longrightarrow C'(z) = A(z).$$

Preuves :

- Si $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, on a le dénombrement suivant sur les objets de taille n :

$$C_n = \sum_{p+q=n} \frac{(p+q)!}{p!\,q!} A_p B_q$$

  d'où

$$\frac{C_n}{n!} = \sum_{p+q=n} \frac{A_p}{p!} \frac{B_q}{q!} \qquad \text{et} \qquad C(z) = A(z)B(z).$$

- Si $\mathbf{C} = \circ(\mathbf{A})$, on a alors $C_{n+1} = A_n$, d'où :

$$C'(z) = \sum_{n\geq 1} nC_n \frac{z^{n-1}}{n!} = \sum_{n\geq 1} A_{n-1} \frac{z^{n-1}}{(n-1)!} = A(z).$$

□

La description formelle III.3 se traduit donc par des équations différentielles sur les séries génératrices qui s'écrivent sous la forme :

$$\begin{aligned} S(z) &= 1 + \int_0^z S^2(x)\,dx \\ C(z) &= CP(z) + 2\int_0^z C(x)\,S(x)\,dx \end{aligned} \tag{III.4}$$

On vérifie facilement que la première équation redonne bien $S(z) = 1/(1-z)$. la deuxième équation s'intègre alors sous la forme :

$$C(z) = \frac{\int (1-x)^2\,CP(x)\,dx}{(1-z)^2} \tag{III.5}$$

1.3. Description de l'algorithme général.

L'objet de cette section est de montrer comment se généralise la description formelle III.3 pour les variantes du tri partiel des petites listes, du multipartitionnement et de la médiane. La résolution des équations caractéristiques obtenues sur les séries génératrices se fera dans la section suivante dans le cas de l'algorithme général décrit au chapitre II.

Considérons d'abord le cas du traitement particulier effectué sur les petites listes, décrit dans la section II.4. p. 24. On suppose donc que les listes de taille inférieure ou égale à M sont triées par un algorithme différent. La fonction de coût étudiée sur le nouvel algorithme pour les petites listes sera alors notée $CF(\sigma)$. Pour fixer les idées, $CF(\sigma)$ peut par exemple compter le nombre de comparaisons nécessaires pour un tri par insertion sur σ avec $|\sigma| \leq M$. On associe alors à ce coût la classe pondérée $\mathbf{CF}(\mathbf{S})$ et la série génératrice exponentielle correspondante $CF(z)$ qui est en réalité un polynôme car la classe $\mathbf{CF}(\mathbf{S})$ ne contient que des éléments de taille inférieure ou égale à M.

Le traitement différent des petites listes se traduit simplement sur les classes de structures combinatoires grâce à un opérateur de troncature qui consiste à enlever les éléments de la classe qui ont une taille inférieure à un paramètre donné.

Définition III.5 *L'opérateur de troncature d'ordre M, T_M, sur les classes de structures combinatoires est défini par :*

$$T_M(\mathbf{C}) = \{s \in \mathbf{C} / |s| > M\}.$$

On note alors R_M l'opérateur qui donne la partie polynomiale d'une classe :

$$R_M(\mathbf{C}) = \{s \in \mathbf{C} / |s| \leq M\} = \mathbf{C} - T_M(\mathbf{C}).$$

Considérons une classe $\mathbf{C}$ définie de la façon suivante : les éléments de $\mathbf{C}$ de taille $> M$ sont les éléments d'une classe $\mathbf{A}$ donnée, et les éléments de taille $\leq M$ sont les éléments d'une classe $\mathbf{B}$. La classe $\mathbf{C}$ est alors construite à partir de $\mathbf{A}$ et $\mathbf{B}$ par :

$$\mathbf{C} = T_M(\mathbf{A}) + R_M(\mathbf{B}). \tag{III.6}$$

Le fait de tronquer à l'ordre M le tri Quicksort standard change donc la description formelle III.3 p. 35 en :

$$\begin{aligned} \mathbf{S} &= T_M[\circ(\mathbf{S}\otimes\mathbf{S})] + R_M(\mathbf{S}) \\ \mathbf{C}(\mathbf{S}) &= T_M[\mathbf{CP}(\mathbf{S}) + \circ(\mathbf{S}\otimes\mathbf{C}(\mathbf{S})) + \circ(\mathbf{C}(\mathbf{S})\otimes\mathbf{S})] + \mathbf{CF}(\mathbf{S}). \end{aligned} \tag{III.7}$$

La troncature sur les classes T_M se traduit trivialement par une opération de troncature, aussi notée T_M, sur les séries génératrices associées ; d'où le lemme suivant.

Lemme III.2 *Pour deux classes de structures combinatoires* $\mathbf{A}$ *et* $\mathbf{C}$*, on a :*

$$\mathbf{C} = T_M(\mathbf{A}) \Longrightarrow C(z) = T_M(A(z)) = \sum_{n>M} A_n \frac{z^n}{n!} = A(z) - \sum_{n\le M} A_n \frac{z^n}{n!}.$$

La seconde variante que nous allons voir maintenant concerne le rôle du multipartitionnement (cf. section II.3.). On utilise pour cela les opérateurs d'enracinement multiple d'ordre $s-1$ et de produit partitionnel de s structures (cf. les définitions III.3 et III.4).

Comme dans le cas binaire, on divise la décomposition caractéristique $\sigma \longrightarrow (\sigma_1,\ldots,\sigma_s)$ en deux étapes :

1. Choix des $s-1$ clés v_i de partition : $\sigma \longrightarrow ([v_1,\ldots,v_{s-1}],\sigma_0)$ où σ_0 est obtenue à partir de σ en supprimant les clés v_i. On suppose que les v_i sont pris à des positions fixes indépendantes de σ, par exemple les $s-1$ premières clés de la liste dans le cas de l'algorithme de référence. Les v_i sont de plus indexés dans l'ordre où ils apparaissent dans σ.

2. Multipartitionnement de σ_0 autour des $s-1$ clés v_i : $([v_1,\ldots,v_{s-1}],\sigma_0) \longrightarrow (\sigma_1,\ldots,\sigma_s)$ avec $|\sigma_i| = v_i - v_{i-1} - 1$ si on pose $v_0 = 0$ et $v_s = |\sigma_0| + s = |\sigma| + 1$.

Proposition III.3 *L'application* $\sigma \longrightarrow (\sigma_1,\ldots,\sigma_s)$*, induite par l'algorithme de multipartitionnement, est un isomorphisme entre* $\mathbf{S}$ *et* $\circ_{s-1}(\underbrace{\mathbf{S}\otimes\mathbf{S}\otimes\cdots\otimes\mathbf{S}}_{s}) + R_{s-2}(\mathbf{S})$.

Preuve : La preuve généralise tout à fait le cas binaire $s = 2$. Soit $(\sigma_1,\ldots,\sigma_s) \in \mathbf{S}^s$, on a une correspondance bijective entre le produit partitionnel de ces s permutations et l'ensemble des permutations antécédentes σ_0 par le partitionnement. Dans le cas du critère H.III, cette correspondance est donnée par l'opération de mélange ou de Q-shuffle des s sous-listes. Pour un algorithme vérifiant seulement l'hypothèse H.II, la bijection est donnée par un argument de dénombrement.

Le s-uplet $(\sigma_1,\ldots,\sigma_s) \in \mathbf{S}^s$ détermine de plus la liste ordonnée des valeurs v'_i des clés de partitionnement par : $v'_i = |\sigma_1| + |\sigma_2| + \ldots + |\sigma_i| + i$. La liste des v_i, c'est-à-dire l'ordre des clés de partitionnement dans σ, est alors obtenue en considérant toutes les permutations possibles des clés v'_i.

Pour obtenir les permutations σ de taille $\geq s-1$ qui se partitionnent en $(\sigma_1,\ldots,\sigma_s)$, on décale donc de $i-1$ les clés de chaque σ_i et on insère ensuite toutes les permutations des clés v'_i sur chaque élément σ_0 du produit partitionnel des σ_i. Dans le cas de l'algorithme de référence, l'insertion correspond effectivement à une concaténation de $s-1$ clés ; d'où l'opérateur d'enracinement $\circ_{s-1}$. De façon plus générale, l'insertion consiste à placer les clés aux positions fixes données par l'algorithme d'échantillonnage ; ceci reste toutefois combinatoirement équivalent à l'opération de concaténation. □

L'algorithme Quicksort avec multipartitionnement d'ordre s et tri partiel de paramètre $M \geq s-2$ est alors décrit formellement par :

$$\mathbf{S} = T_M\Big[\circ_{s-1}(\underbrace{\mathbf{S}\otimes\mathbf{S}\otimes\cdots\otimes\mathbf{S}}_{s})\Big] + R_M(\mathbf{S})$$

$$\mathbf{C}(\mathbf{S}) = T_M\Big[\underbrace{\circ_{s-1}(\,\mathbf{S}\otimes\cdots\otimes\mathbf{C}(\mathbf{S})\,)+\cdots+\circ_{s-1}(\,\mathbf{C}(\mathbf{S})\otimes\cdots\otimes\mathbf{S}\,)}_{s}\Big] \qquad \text{(III.8)}$$
$$+\,T_M\left[\mathbf{CP}(\mathbf{S})\right] + \mathbf{CF}(\mathbf{S})\,.$$

Considérons enfin la variante de Quicksort avec médiane de $k = s(t+1)-1$ clés décrites dans les sections II.2. et II.5. On suppose que l'algorithme qui sélectionne de manière équirépartie les $s-1$ clés de partitionnement, effectue dans le même temps le partitionnement uniforme des autres éléments de l'échantillon de k clés. Ceci permet de prendre en compte, dans la description combinatoire du partitionnement, le rôle particulier joué par les k éléments de l'échantillon par rapport au reste de la liste. L'uniformité des sous-listes de taille t produites lors de la sélection, est là pour assurer l'uniformité globale du partitionnement (cf. condition H.II p. 30).

Dans la pratique, il est assez naturel d'effectuer le partitionnement de l'échantillon en même temps que la sélection, dans la mesure où celle-ci donne déjà les informations nécessaires pour le partitionnement du reste de l'échantillon. L'hypothèse d'uniformité n'est toutefois pas toujours naturelle car la détermination de la médiane ou des $s-1$ éléments "médians" donne généralement plus informations sur l'ordre des clés de l'échantillon que le strict nécessaire. L'algorithme de sélection ne vérifie alors pas le critère d'uniformité H.III. Un algorithme qui utiliserait l'information issue de la phase de sélection et aurait tendance à ordonner les s sous-listes de taille t pourrait donc être légèrement plus rapide que celui que nous analysons.

Dans le cas particulier de l'algorithme de référence et donc des arbres de recherche, la condition d'uniformité sur le partitionnement de l'échantillon n'est pas cruciale et n'affecte que l'analyse de tri sur les petites listes. A chaque étape, les sous-listes de taille t produites lors de la sélection sont en effet intégralement reprises dans les échantillons de taille k de l'étape suivante ; l'ordre des clés y est alors indifférent et ne change pas le processus de partitionnement. Si les sous-listes de l'échantillon de k clés ont une certaine distribution non uniforme, c'est donc seulement au moment de trier les petites listes qu'il faut en tenir compte et analyser la méthode de tri des petites listes (détermination de $CF(z)$) suivant cette nouvelle distribution non uniforme des listes de taille $\leq M$.

Plaçons nous dans le cas ternaire $s = 3$ comme le montre la figure III.2. Si une permutation σ se partitionne en σ_1, σ_2 et σ_3, on note $\sigma = \tau.\sigma'$, $\sigma_i = \tau_i.\sigma_i'$ pour $i = 1\ldots 3$, où τ et τ_i représentent respectivement les k premières et les t premières clés des permutations σ et σ_i, et où les σ' correspondent aux clés restantes.

Les clés v_1 et v_2 de partitionnement sont réparties d'autant plus équitablement par rapport à σ que le paramètre t est grand, mais elles sont distribuées totalement uniformément dans σ'. Le produit partitionnel $\sigma_1'\otimes\sigma_2'\otimes\sigma_3'$ décrit donc exactement l'ensemble des permutations antécédentes σ'. D'un autre coté, les permutations τ sont générées comme dans le cas sans médiane par la construction $\circ(\tau_1\otimes\tau_2\otimes\tau_3)$.

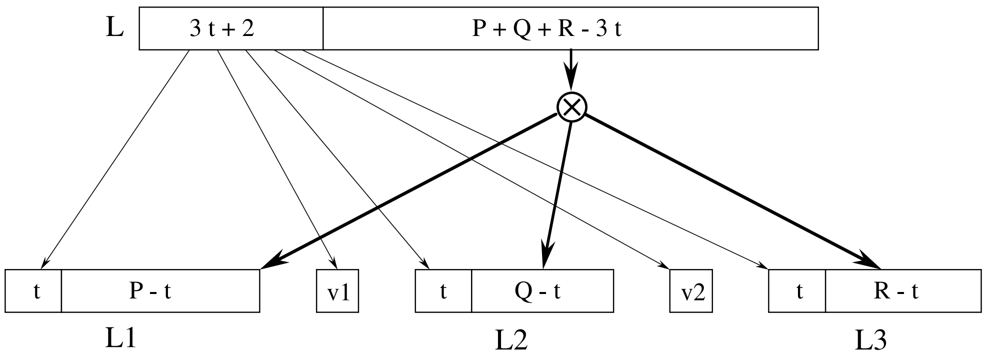


Figure III.2 : Schéma du partitionnement avec médiane dans le cas binaire $s = 3$.

Les comportements des permutations σ', σ'_1, σ'_2, σ'_3 et τ, τ_1, τ_2, τ_3 ne sont liés que par le rôle des deux clés v_1 et v_2. Ainsi pour σ'_1, σ'_2 et σ_3 fixés, considérons les $(t!)^3$ triplets $(\sigma_1, \sigma_2, \sigma_3)$ obtenues en effectuant toutes les permutations possibles de τ_1, τ_2, et τ_3. Ces triplets ont exactement pour antécédent l'ensemble des permutations obtenues en concaténant les éléments σ' de $\sigma'_1 \otimes \sigma'_2 \otimes \sigma_3$ avec les $k!$ permutations des clés de $\{v1, v2\} \cup \tau_1 \cup \tau_2 \cup \tau_3$. Le passage aux permutations antécédentes de trois sous-listes données consiste donc à enlever les t premiers éléments de chaque liste, à effectuer le mélange, puis à concaténer sur la permutation obtenue un shuffle des $k = 3t + 2$ clés de l'échantillon.

Pour décrire, sur les classes de structures combinatoires, le passage d'une permutation σ à la permutation σ' associée, on introduit un opérateur de suppression qui correspond à l'opérateur inverse de l'enracinement multiple $\circ_i$:

Définition III.6 *On note Δ_i l'opérateur qui supprime les i premières composantes d'une structure. Si $s = \sigma_i \times T$ avec $|\sigma_i| = i$, on a $\Delta_i(s) = T$ avec la taille définie par $|\Delta_i(s)| = |T| = |s| - i$.*

La traduction de ces opérateurs sur les classes par des opérations sur les séries génératrices exponentielles est alors donnée par le lemme suivant.

Lemme III.3 *Pour deux classes* $\mathbf{A}$ *et* $\mathbf{B}$ *de structures combinatoires, on a :*

$$\mathbf{B} = o_i(\mathbf{A}) \Longrightarrow B^{(i)}(z) = i! A(z)\,,$$

$$ou \quad \mathbf{A} = \Delta_i(\mathbf{B}) \Longrightarrow A(z) = \frac{B^{(i)}(z)}{i!}\,,$$

où $^{(i)}$ désigne la dérivée d'ordre i.

Preuve : Si $\mathbf{C} = \circ_i(\mathbf{A})$, on a alors $C_{n+i} = i!\, A_n$, d'où :

$$\frac{C^{(i)}(z)}{i!} = \sum_{n \geq i} \binom{n}{i} C_n \frac{z^{n-i}}{n!} = \sum_{n \geq i} A_{n-i} \frac{z^{n-i}}{(n-i)!} = A(z).$$

□

On peut donc décrire formellement la décomposition combinatoire induite par l'algorithme avec médiane de $k = 3t + 2$ ($s = 3$) par la relation :

$$\mathbf{S} = T_M[\, \circ_{3t+2}(\, \Delta_t(\mathbf{S}) \otimes \Delta_t(\mathbf{S}) \otimes \Delta_t(\mathbf{S})\,)\,] + R_M(\mathbf{S})\,. \qquad \text{(III.9)}$$

Nous pouvons maintenant, en récapitulant ce qui précède, établir le théorème suivant qui donne la description formelle associée à l'algorithme général décrit dans la section II.5.

Théorème III.1 *L'algorithme Quicksort avec multipartitionnement d'ordre s, médiane de k clés avec $k = s(t+1) - 1$, et tri distinct pour les listes de taille inférieure ou égale à M avec $M \geq k-1$, décompose les permutations suivant le schéma :*

$$\mathbf{S} = T_M[\circ_k\Big(\underbrace{\Delta_t(\mathbf{S}) \otimes \cdots \otimes \Delta_t(\mathbf{S})}_{s}\Big)] + R_M(\mathbf{S}),$$

$$\mathbf{C}(\mathbf{S}) = T_M\Big[\underbrace{\circ_k\Big(\overbrace{\Delta_t(\mathbf{S}) \otimes \cdots \otimes \Delta_t(\mathbf{C}(\mathbf{S}))}^{s}\Big) + \cdots + \circ_k\Big(\overbrace{\Delta_t(\mathbf{C}(\mathbf{S})) \otimes \cdots \otimes \Delta_t(\mathbf{S})}^{s}\Big)}_{s}\Big]$$
$$+ T_M[\mathbf{CP}(\mathbf{S})] + \mathbf{CF}(\mathbf{S}).$$

où CF compte le coût moyen sur les petites listes, et CP compte le coût moyen d'un partitionnement.

Moyennant la règle de dérivation : $T_M[\,a(z)\,]^{(k)} = T_{M-k}[\,a^{(k)}(z)\,]$, on en déduit les équations sur les séries génératrices $S(z)$ et $C(z)$.

Corollaire III.2

- *La série génératrice $S(z)$ du nombre de permutations vérifie :*

$$\frac{S^{(k)}(z)}{k!} = T_{M-k}[\,\left(\frac{S^{(t)}(z)}{t!}\right)^s\,] + R_{M-k}(\frac{1}{(1-z)^{k+1}}).$$

- *La série génératrice d'un coût C pour l'algorithme général vérifie l'équation :*

$$C^{(k)}(z) = T_{M-k}[\,CP^{(k)}(z) + s\,\frac{k!}{t!}\,C^{(t)}(z)\,\left(\frac{S^{(t)}(z)}{t!}\right)^{s-1}\,] + CF^{(k)}(z). \qquad \text{(III.10)}$$

On peut vérifier en considérant le terme général de la série $C(z)$ que l'équation précédente est totalement équivalente à la récurrence caractéristique sur les coûts moyens $\bar{c}_n$:

$$\bar{c}_n = \begin{cases} \bar{cp}_n + \displaystyle\sum_{n_1+n_2+\cdots+n_s=n-s+1} P_{n_1,n_2,\ldots,n_s}(\bar{c}_{n_1} + \bar{c}_{n_2} + \cdots + \bar{c}_{n_s}) & \text{si } n > M \\ \bar{cf}_n & \text{si } n \leq M \end{cases} \qquad \text{(III.11)}$$

où $\bar{cp}_n$ est le coût moyen d'un partitionnement, $\bar{cf}_n$ est le coût moyen du tri d'une petite liste, et $P_{n_1,n_2,\ldots,n_s}$ est la probabilité, donné par l'équation II.3 p. 26, de partitionner en sous-listes de taille $n_1, n_2, \ldots, n_s$.

2. Résolution.

On remplace $S(z)$ par sa valeur $S(z) = 1/(1-z)$, l'équation caractéristique III.10 sur la série de coût $C(z)$ se réécrit alors sous la forme :

$$\frac{(1-z)^k C^{(k)}(z)}{k!} = \frac{(1-z)^k CP^{(k)}(z)}{k!} + s\frac{(1-z)^t C^{(t)}(z)}{t!} + (1-z)^k Q_{M-k}(z) \tag{III.12}$$

où $Q_{M-k}(z)$ est un polynôme de degré $M-k$. Ce polynôme $Q_{M-k}(z)$ dépend uniquement du polynôme $CF(z)$ et de la partie polynomiale d'ordre M de $CP(z)$. Il n'est toutefois pas utile de déterminer explicitement $Q_{M-k}(z)$ pour le moment.

La série de coût $C(z)$ est donc caractérisée par une équation différentielle d'ordre k. Cette équation présente toutefois une certaine homogénéité qui permet une résolution assez générale en fonction des paramètres s, t, M et $k = s(t+1)-1$. Cette résolution est assez semblable à la méthode utilisée par D. H. Greene [Gre83] pour l'analyse des arbres réduits binaires ($s = 2$). Nous généralisons ici les règles d'intégrations pour prendre en compte une classe plus large de fonctions $CP(z)$ (cf. proposition III.4). Ceci permet alors la détermination de la variance et des moments d'ordre supérieur en suivant exactement le même schéma de résolution que pour la moyenne. Nous généralisons de plus, grâce à des lemmes de transfert (proposition III.5), l'utilisation d'équivalents et de développements asymptotiques sur les séries génératrices et sur leur termes généraux. On a alors la possibilité de faire une résolution sélective dans le mesure où l'on sait négliger dès le début et tout au long de la résolution les termes qui produiront des composantes négligeables dans les coûts moyens.

2.1. Méthode.

Pour simplifier l'écriture, on effectue le changement de variable $X = 1-z$ et on définit les fonctions génératrices $\tilde{C}(X) = C(1-X) = C(z)$, $\tilde{CP}(X) = CP(1-X) = CP(z)\ldots$. On introduit d'autre part un opérateur différentiel Θ qui permet de décrire simplement l'homogénéité de l'équation différentielle caractéristique III.12.

Définition III.7 *L'opérateur différentiel Θ est défini par :*

$$\Theta.F(X) = X\,F'(X)\,.$$

On vérifie alors simplement par récurrence que :

$$\binom{\Theta}{i}.F(X) = \frac{\Theta(\Theta-1)\cdot(\Theta-i+1)}{i!}.F(X) = \frac{X^i F^{(i)}(X)}{i!}. \tag{III.13}$$

Ainsi, en utilisant l'opérateur différentiel Θ, on transforme l'équation caractéristique III.12 en :

$$\begin{aligned} &PQ_{s,t}(\Theta).\tilde{C}(X) = (-1)^k\binom{\Theta}{k}.\tilde{CP}(X) + X^k\tilde{Q}_{M-k}(X) \\ \text{avec}\quad &PQ_{s,t}(x) = (-1)^k\binom{x}{k} - s(-1)^t\binom{x}{t} \qquad \text{et}\quad k = s(t+1)-1. \end{aligned} \tag{III.14}$$

Pour faire le lien avec les relations de récurrence sur le coût moyen c_n, on regarde comment Θ opère au niveau des termes généraux des séries. La définition de Θ donne ainsi la relation :

$$(\frac{-z}{1-z})^i \binom{\Theta}{i} \sum_n c_n z^n = \sum_n \binom{n}{i} c_n z^n. \tag{III.15}$$

Pour les récurrences qui interviennent dans l'analyse du tri ou des arbres de recherche, on en déduit alors une règle de traduction qui permet d'obtenir directement les équations différentielles sur séries génératrices à partir des relations de récurrence.

Lemme III.4 (Règle de traduction) *Si la suite C_n est définie par la récurrence :*

$$\binom{n}{a+b+1} C_n = \sum_{p+q=n-1} \binom{p}{a} A_p \binom{q}{b} B_q,$$

on a

$$-\binom{\Theta}{a+b+1} \tilde{C}(X) = X \binom{\Theta}{a} \tilde{A}(X) \binom{\Theta}{b} \tilde{B}(X),$$

avec les séries génératrices définies par $\tilde{C}(X) = C(1-X) = \sum_n C_n(1-X)^n$.

Remarque : Dans le cas particulier où $B_q = 1$ pour tout q, la règle de traduction devient :

$$\binom{n}{a+b+1} C_n = \sum_{p+q=n-1} \binom{p}{a} \binom{q}{b} A_p \Longrightarrow (-1)^{b+1} \binom{\Theta}{a+b+1} \tilde{C}(X) = \binom{\Theta}{a} \tilde{A}(X). \tag{III.16}$$

La résolution d'équations différentielles du type de l'équation III.14, c'est-à-dire caractérisées par un polynôme en Θ, est donnée par la proposition suivante :

Proposition III.4 *Soit $R(x)$ un polynôme de la forme $R(x) = (x-\beta)^m Q(x)$ avec $m \geq 0$ et $Q(x)$ ayant pour racines α_1, α_2, $\ldots, \alpha_n$ avec les multiplicités m_1, m_2, $\ldots, m_n$. La solution générale de l'équation différentielle $R(\Theta).F(X) = X^\beta \ln^p X$ est :*

$$F(X) = X^\beta \sum_{i=0}^{p} \frac{p!}{(p-i)!} \left(\frac{1}{Q(x)}\right)^{(p-i)}_{x=\beta} \frac{\ln^{i+m} X}{(i+m)!} + \sum_{j=0}^{n} \sum_{i=0}^{m_j-1} \lambda_{i,j} X^{\alpha_j} \ln^i X,$$

en posant $\alpha_0 = \beta$ et $m_0 = m$ et où les $\lambda_{i,j}$ sont les constantes d'intégration à valeurs dans le même domaine (réel ou complexe) que les α_j.

Preuve : La preuve se fait par récurrence sur le degré du polynôme $R(x)$ à partir de la formule de base pour $R(x) = (x-\alpha)$.

On a en effet :

$$(\Theta - \alpha).F(X) = X^\beta \ln^p X \Longrightarrow F(X) = X^\alpha \int x^{\beta-\alpha-1} \ln^p x \, dx \tag{III.17}$$

et donc par récurrence sur p :

$$F(X) = \begin{cases} X^\beta \dfrac{\ln^{p+1} X}{p+1} + \lambda X^\beta & \text{si } \alpha = \beta \\ X^\beta \displaystyle\sum_{i=0}^{p} (-1)^{p-i} \frac{p!}{i!} \frac{\ln^i X}{(\beta-\alpha)^{p-i+1}} + \lambda X^\alpha & \text{si } \alpha \neq \beta \end{cases}. \tag{III.18}$$

On suppose maintenant la formule de la proposition vraie pour le polynôme $R(x)$. L'équation précédente permet alors de passer à la solution de l'équation différentielle avec le polynôme $R_1(x) = (x-\alpha)\,R(x)$, en distinguant les deux cas $\alpha = \beta$ et $\alpha \neq \beta$. On retrouve ainsi l'expression désirée pour la solution de $R_1(\Theta).F(X) = X^\beta \ln^p X$, en tenant compte de la relation suivante sur les coefficients :

$$\left(\frac{1}{R_1(x)}\right)^{(k)} = \sum_{i=0}^{k} \frac{k!}{(k-i)!} \left(\frac{1}{R(x)}\right)^{(k-i)} \frac{(-1)^i}{(x-\alpha)^{i+1}}. \tag{III.19}$$

□

Le principe de la preuve et les propriétés de l'intégration vis-à-vis de $O()$, $o()$ et $\sim$ donnent le corollaire suivant :

Corollaire III.3 *La solution générale $G(X)$ de l'équation différentielle $R(\Theta).G(X) = H(X)$ avec $H(X) = O(X^\beta \ln^p X)$ (resp. $o()$, $\sim$) vérifie si elle existe :*

$$G(X) = O(F(X)) + \sum_{j=0}^{n} \sum_{i=0}^{m_j-1} \lambda_{i,j} X^{\alpha_j} \ln^i X$$

où $F(X)$ est une solution particulière de $R(\Theta).F(X) = X^\beta \ln^p X$.

La résolution d'une équation du type $R(\Theta).F(X) = G(X)$ consiste donc à développer $G(X)$ en termes de la forme $X^\beta \ln^p X$. Ces termes se transforment alors par intégration en une somme de termes $X^\beta \ln^i X$ avec $i \leq p+m$ où m est la multiplicité de β comme racine de $R(x)$ ($m = 0$ si $R(\beta) \neq 0$). Les coefficients de chaque terme s'expriment à partir des dérivées du polynôme $R(x)$ au point $x = \beta$ et peuvent donc être calculés indépendamment du degré de $R(x)$ ou de la valeur des racines. Une solution particulière de $R(\Theta).F(X) = G(X)$ est ainsi complètement déterminée par la donnée des valeurs des dérivées du polynôme $R(x)$ aux points $x = \beta$ où β correspond aux exposants de X dans $G(X)$. La valeur et la nature des racines n'interviennent alors qu'au niveau des termes indéterminés solutions de l'équation sans second membre.

De plus, dans le cas qui nous intéresse, c'est-à-dire $X = 1 - z$, les contributions dominantes sur les coûts moyens proviennent des termes généraux des séries $X^\beta \ln^p X$ avec $\Re(\beta)$ minimum et p maximum. On a en effet :

$$[z^n] X^\beta \ln^p X = O(n^{-\beta-1} \ln^p n).$$

De façon plus générale pour ce type de fonction, le passage au terme général de la série est donné par des lemmes de transfert qui permettent de travailler avec des équivalents ou des développements des séries au voisinage de la singularité $z = 1$.

Proposition III.5 (Lemmes de transfert [FO90]) *soit* $c(z) = (1-z)^\beta \ln^p(1-z) = \sum_n c_n z^n$ *et* $f(z)$ *une fonction analytique dans le domaine complexe* $D = \{z / |z| \leq 1 \text{ et } z \neq 1\}$ *on a alors :*

1.

$$c_n = n^{-\beta-1} P(\ln n) + O(n^{-\beta-2} \ln^p n)$$

où P *est un polynôme de degré au plus* p.

2.

$$f(z) \sim c(z) \Longrightarrow f_n \sim c_n$$

3.

$$f(z) = O(c(z)) \Longrightarrow f_n = O(c_n)$$

4.

$$f(z) = o(c(z)) \Longrightarrow f_n = o(c_n)$$

Ces lemmes de transfert se généralise aux fonctions $c(z)$ du type $c(z) = (1-z)^\beta \ln^\alpha(1-z) \ln(\ln^\gamma(1-z))$, cf. [FO90]. La section A.7. donne de plus des récurrences pour le calcul des coefficients du polynôme P.

Pour résoudre l'équation $R(\Theta).F(X) = G(X)$, il suffit donc d'obtenir un développement de la fonction $G(1-z)$ dans l'échelle asymptotique $(1-z)^\beta \ln^p(1-z)$ pour β complexe et p entier. Ce développement de $G(X)$ se traduit alors directement par un développement asymptotique du terme général F_n de la solution $F(X)$ dans l'échelle $n^\alpha \ln^i n$. Le seul calcul à effectuer est la détermination des coefficients du développement qui sont obtenus simplement à partir des coefficients des polynômes P des lemmes de transfert et de valeurs particulières du polynôme R.

On peut ainsi suivre l'importance relative des différentes composantes intervenant dans le coût, et éliminer dès le départ les termes dont la contribution au coût moyen serait négligeable. On évite de plus le calcul de certaines constantes d'intégration, si on a de bonnes minorations sur la partie réelle des racines correspondantes du polynôme $R(x)$. Ces simplifications sont en particulier indispensables pour le calcul de la variance et des moments d'ordre supérieur car on limite ainsi la taille des expressions mises en jeu et le nombre de coefficients à calculer.

Nous aurons besoin dans la suite de la formule suivante qui donne, à l'inverse de la proposition III.4, les règles de dérivation par un polynôme différentiel $R(\Theta)$ pour des termes de la forme $X^\beta \ln^k X$.

Proposition III.6 *Pour un polynôme* $R(x)$ *donné et* $\Theta = X\frac{d}{dX}$, *on a la formule de dérivation :*

$$R(\Theta).[X^\beta \ln^k X] = X^\beta R(\theta + \beta).[\ln^k X] = X^\beta \sum_{i=0}^{k} \binom{k}{i} R^{(i)}(\beta) \ln^{k-i} X.$$

Preuve : Cette formule se vérifie facilement par récurrence sur le degré de $R(x)$. Elle donne de plus une vérification ou une preuve de la proposition III.4. □

2.2. Solution générale.

Les propriétés du polynôme caractéristique $PQ_{s,t}$ sont décrites en annexe B. Le lemme B.6 p. 125 donne en particulier les racines de $PQ_{s,t}$ qui sont : $-2, 0, 1, 2, \ldots, t-1, (s+1)(t+1)-1$ si $(s+1)(t+1)$ est pair, et des couples de racines complexes conjuguées. De plus, toutes les racines sont des racines simples et ont une partie réelle supérieure à -2. C'est la racine -2 qui joue le rôle prépondérant dans la mesure où elle fait apparaître le terme principal en $n \ln n$ dans le coût moyen. Cette racine engendre aussi des contributions linéaires en $O(n)$ avec des coefficients indéterminés par l'équation différentielle. Ces termes linéaires, qui vont dépendre directement du tri des petites listes, seront traités dans la section suivante.

Des propriétés du polynôme $PQ_{s,t}$ et de la proposition III.4, on déduit maintenant le théorème suivant qui donne la forme générale pour la solution de l'équation caractéristique III.14 p. 41.

Théorème III.2 *La série génératrice exponentielle de la fonction de coût C pour l'algorithme Quicksort général, caractérisé par les paramètres s, t et M, a la forme :*

$$\tilde{C}(X) = F(X) + \frac{\lambda_0}{X^2} + \sum_{j=1}^{p}(\lambda_j X^{\alpha_j} + \bar{\lambda}_j X^{\bar{\alpha}_j}) + (\mu \ln X + \lambda) X^{(s+1)(t+1)-1} + R_M(X)$$

où

- *$F(x)$ est une solution particulière de : $PQ_{s,t}(\Theta).F(X) = (-1)^k X^k \tilde{CP}^{(k)}(X)/k!$, òu CP représente le coût moyen d'un partitionnement.*
- *$R_M(X)$ est un polynôme de degré M.*
- *$p = \lfloor (k-t-1)/2 \rfloor$; $k = s(t+1)-1$.*
- *λ, μ, λ_0 sont réels ; λ_j est complexe pour $j = 1, \ldots, p$.*
- *$\lambda = 0$ et $\mu = 0$ si $(s+1)(t+1) = t+k$ est impair ; $\mu = 0$ si $(s+1)(t+1)$ est un entier pair strictement supérieur à M.*

On a alors en passant au terme général de la série :

Théorème III.3 *Sous les conditions du théorème précédent, le coût moyen sur les listes de taille n, avec $n > M$ et $n \geq (s+1)(t+1)$ si $(s+1)(t+1)$ pair, est donné par :*

$$\bar{c}_n = [z^n]F(1-z) + \lambda_0(n+1) - \frac{\mu}{(s+1)(t+1)\binom{n}{(s+1)(t+1)}} + \sum_{j=1}^{p} B_n^j$$

où $B_n^j = 2(-1)^n \Re(\lambda_j \binom{\alpha_j}{n})$ est un terme oscillant d'amplitude $O(n^{-\Re(\alpha_j+1)})$.

Corollaire III.4

$$\bar{c}_n = [z^n]F(1-z) + \lambda_0(n+1) + O(n^{b_{s,t}}) \qquad \text{avec} \qquad b_{s,t} < 1$$

où $b_{s,t} = \max(-\Re(\alpha+1))$ étendu aux racines α de $PQ_{s,t}$ autres que $-2, 0, 1, \ldots, t-1$.

Preuves : Ces résultats découlent directement de la proposition III.4 et du lemme B.6 compte-tenu des remarques suivantes :

- Les termes indéterminés provenant des racines $0,\ 1,\ \ldots, t-1$ sont regroupés dans le polynôme $R_M(X)$.
- Les termes polynomiaux $X^k\tilde{Q}_{M-k}(X)$ de l'équation caractéristique III.14 ne correspondent à aucune racine de $PQ_{s,t}$. Ils restent donc polynomiaux et sont intégrés dans $R_M(X)$. Exception : si $(s+1)(t+1)$ est un entier pair $\leq M$, il apparaît un terme indéterminé correspondant à la constante d'intégration μ.
- $R_M(X)$ introduit une composante nulle dans le coût moyen pour $n > M$.
- Les termes généraux des séries utiles sont données par les formules de la section A.7. p. 120.
- Les termes oscillants sont essentiellement de la forme :

$$B_n^j \sim 2n^{-\Re(\alpha_j)-1}\Re\Big(\frac{\lambda_j e^{\imath\Im(\alpha_j)\ln n}}{\Gamma(-\alpha_j)}\Big). \tag{III.20}$$

□

On remarque donc que le paramètre M et le tri différent des petites listes donné par $CF(z)$ jouent uniquement sur les constantes d'intégrations et en particulier sur le terme linéaire λ_0. Le terme dominant du coût moyen en $n\ln n$, vient alors de $F(X)$ et dépend donc directement du coût moyen d'un partitionnement donné par $CP(z)$ et des valeurs de $PQ_{s,t}(x)$ et de ses dérivées au point $x=-2$. Pour le calcul des dérivées de $PQ_{s,t}$, on se reportera à la section 4. de l'annexe B et à l'annexe A.

On note enfin que le processus de résolution est linéaire et donc que si le coût moyen d'un partitionnement CP est défini comme une combinaison linéaire, il en va de même du coût moyen du tri.

Remarque : La majoration de $b_{s,t}$ est optimale pour $s,t\to\infty$. Dans la pratique pour les premières valeurs de s et t, on a toutefois une majoration bien meilleure qui rend quasiment inutile la détermination des termes indéterminés provenant des racines autres que -2. Par exemple dans le cas binaire $s=2$, $b_{s,t}$ est inférieur à -3.5 pour $t\leq 5$, et reste négatif pour t allant jusqu'à 13 (cf. figure B.1 p. 125 et tableau B.1 p. 126).

Exemple III.1 : Valeurs de $F(X)$.

1. Si on considère le nombre d'appels de la procédure de tri, c'est-à-dire $\bar{\varphi}_n = 1$, on a alors :

$$\tilde{CP}(X) = 1/X. \tag{III.21}$$

D'où l'équation à résoudre :

$$PQ_{s,t}(\Theta).F(X) = 1/X. \tag{III.22}$$

On obtient donc en appliquant la proposition III.4

$$F(1-z) = \frac{1}{PQ_{s,t}(-1)(1-z)} \qquad \text{et} \qquad \bar{c}_n = \lambda_0(n+1) - \frac{1}{s-1} + O(n^{b_{s,t}}). \tag{III.23}$$

Comme on pouvait s'y attendre, le nombre de partitionnements est linéaire et dépend donc directement de M. La constante λ_0 sera déterminée dans la section suivante.

2. Considérons maintenant que le coût moyen pour un partitionnement est de la forme : $\bar{\varphi}_n = a(n+1)+b$. Le cas $a = \bar{a}(s)$, et $b = \bar{a}(s)(2 - s(t+1))$, où $\bar{a}(s)$, défini par l'équation II.2 p. 22, correspond au nombre moyen de comparaisons durant le partitionnement (sans compter la recherche de la médiane). On a alors :

$$\tilde{CP}(X) = a/X^2 + b/X. \tag{III.24}$$

D'où l'équation à résoudre

$$PQ_{s,t}(\Theta).F(X) = a(k+1)/X^2 + b/X. \tag{III.25}$$

On obtient donc en appliquant la proposition III.4

$$F(1-z) = \frac{a(k+1)}{PQ'_{s,t}(-2)} \frac{\ln(1-z)}{(1-z)^2} + \frac{b}{PQ_{s,t}(-1)\,(1-z)} \tag{III.26}$$

ce qui donne pour le coût moyen

$$\bar{c}_n = ak_{s,t}(n+1)H_{n+1} + \lambda_0(n+1) - \frac{b}{s-1} + O(n^{b_{s,t}}). \tag{III.27}$$

avec $k_{s,t} = (H_{s(t+1)} - H_{t+1})^{-1}$. On obtient ainsi le comportement asymptotique du coût moyen en $n \ln n$ avec une constante caractéristique dépendant directement des paramètres s et t. On trouvera en annexe D des tables de valeurs pour $k_{s,t}$ et de $\bar{a}(s)k_{s,t}$.

3. De façon plus générale, si le coût moyen d'un partitionnement est de la forme $\bar{\varphi}_n = an^r$ avec $r \neq -1, -2, \ldots$, le terme général de $F(1-z)$ est de la forme $ac(r)n^r + O(n^{r-1})$ sauf pour $r = 1$ où le terme principal vaut $O(n \ln n)$ et $r = 2$ où l'erreur est en $O(n \ln n)$. On a en effet si $r \neq 1, 2$:

$$\tilde{CP}(X) = ar!X^{-(r+1)} + O(X^{-r}), \tag{III.28}$$

d'où

$$F(1-z) = \frac{ar!\binom{k+r}{k}}{PQ_{s,t}(-r-1)\,(1-z)^{r+1}} + O(\frac{1}{(1-z)^r}). \tag{III.29}$$

Ce qui donne donc pour le coût moyen la contribution en n^r avec pour le coefficient $c(r)$:

$$c(r) = (1 - s\frac{k(k-1)\ldots(t+1)}{(r+k)(r+k-1)\ldots(r+t+1)})^{-1} \quad \text{et} \quad k = s(t+1) - 1\,. \tag{III.30}$$

Remarque : On peut interpréter intuitivement les résultats précédents en les comparant avec ce qui se passe pour le coût du tri dans le meilleur des cas. Supposons que l'algorithme partitionne à chaque étape en s listes de même taille avec un coût égal à n^r. Le coût du tri d'une liste de taille n est alors approximativement :

$$c_n \approx n^r + s(\frac{n}{s})^r + s^2(\frac{n}{s^2})^r \cdots + s^{\log_s n}(\frac{n}{s^{\log_s n}})^r\,. \tag{III.31}$$

On a donc une somme géométrique qui est égale à $n \log_s n$ si $r = 1$, et qui pour $r \neq 1$ vaut :

$$c_n \approx \frac{n^r - n}{1 - s^{1-r}}. \tag{III.32}$$

Le comportement en moyenne du tri s'apparente donc tout à fait au comportement dans le meilleur des cas avec seulement une modification des coefficients des différentes contributions asymptotiques. On vérifie de plus sur ces coefficients que le meilleur des cas correspond à la limite du tri avec médiane quand le paramètre t tend vers l'infini. On voit en effet que la constante $c(r)$ de l'exemple précédent tend vers $(1-s^{1-r})^{-1}$ pour s fixé et t tendant vers l'infini, et que $k_{s,t}$ tend lui vers $1/ln(s)$.

3. Expressions exactes et termes linéaires.

On va maintenant chercher à déterminer les constantes d'intégration λ_i qui dépendent directement du paramètre M et traduisent le traitement particulier des petites listes. Ces constantes réelles ou complexes correspondent à $k-t=(s-1)(t+1)$ valeurs réelles qui sont déterminées, de manière générale par un système linéaire, à partir des coûts moyens du tri sur les listes de taille $M+1$ à $M+k-t$. Ces listes ont en effet la particularité de se partitionner en une étape en sous-listes de taille inférieure ou égale à M ; les coûts moyens s'expriment donc simplement en fonction du coût CF sur les petites listes.

Nous allons montrer qu'il n'est toutefois pas nécessaire d'écrire et de résoudre explicitement le système linéaire d'ordre $k-t$. On peut en effet en travaillant avec les séries génératrices et l'opérateur Θ, calculer séparément chacune des constantes d'intégration en fonction de la valeur de la racine correspondante. De façon générale, si on applique un polynôme différentiel en Θ de degré p sur une série, cela revient à remplacer le terme général en z^n par une combinaison linéaire des termes généraux d'ordre n à $n+p$; ceci provient directement de la définition de Θ : $\Theta.F(1-z)=-(1-z)F'(1-z)$. Considérons une série $G(X)$ dont le terme général en z, pour les listes de taille $M+1$ à $M+k-t$, est égal au coût moyen du tri. Le fait d'appliquer sur $G(X)$ un polynôme en Θ bien choisi, permet de réaliser l'élimination dans le système linéaire pour obtenir avec le terme général d'ordre $M+1$ la valeur d'une des constantes d'intégration.

On obtient ainsi une détermination synthétique et sélective des constantes d'intégration qui permet en particulier de calculer le coefficient λ_0 qui nous intéresse indépendamment des valeurs des racines complexes du polynôme $PQ_{s,t}$.

Le premier problème est donc de calculer à partir de la fonction de coût CF, les coûts moyens du tri $\bar{c}_n$ sur les listes de taille $M+1$ à $M+k-t$.

Considérons la description récursive de la classe $\mathbf{C(s)}$ sous la forme

$$\mathbf{C}(\mathbf{S}) = T_M[constr(\mathbf{CP}(\mathbf{S}), \mathbf{C}(\mathbf{S}), \mathbf{S})] + \mathbf{CF}(\mathbf{S})$$

où le constructeur *constr* est donné par le théorème III.1 p. 40. Les structures de la classe $C(S)$ de taille $M+1$ à $M+k-t$ sont alors décrites en utilisant les opérateurs de troncature et de partie polynomiale par :

$$R_{M+k-t}[T_M[\mathbf{C}(\mathbf{S})]] = R_{M+k-t}[T_M[constr(\mathbf{CP}(\mathbf{S}), \mathbf{C}(\mathbf{S}), \mathbf{S})]].$$

Ces éléments se partitionnant directement en sous-listes de taille inférieure ou égale à M, on peut remplacer la classe $\mathbf{C}(\mathbf{S})$ du second membre, qui représente le coût sur les sous-listes, par la classe $R_M[\mathbf{C}(\mathbf{S})] = \mathbf{CF}(\mathbf{S})$. On a alors la description formelle non récursive :

$$R_{M+k-t}[T_M[\mathbf{C}(\mathbf{S})]] = R_{M+k-t}[T_M[constr(\mathbf{CP}(\mathbf{S}), \mathbf{CF}(\mathbf{S}), \mathbf{S})]]. \tag{III.33}$$

Pour obtenir la caractérisation du coût moyen sur les listes de taille $M+1$ à $M+k-t$, il suffit donc de reprendre l'équation III.12 p. 41 sur les séries génératrices en remplaçant dans le second membre $C(z)$ par $CF(z)$. D'où la proposition :

Proposition III.7 *Si $G(X)$ est une solution particulière de*

$$(-1)^k \binom{\Theta}{k}.[G(X) - \tilde{CP}(X)] = s(-1)^t \binom{\Theta}{t} \tilde{CF}(X).$$

alors pour $n \in [M+1, .., M+k-t]$, le coût moyen du tri des listes de taille n vaut

$$\bar{c}_n = [z^n]\tilde{C}(1-z) = [z^n]G(1-z)\,.$$

On peut de plus remplacer le polynôme $\tilde{CF}(X)$ par n'importe quelles séries ayant les mêmes coefficients en $[z^n]$ pour $n \leq M$.

Remarque : L'équation donnant $G(X)$ se simplifie, en éliminant $\binom{\Theta}{t}$, et donne :

$$\binom{\Theta - t}{k - t}.[G(X) - \tilde{CP}(X)] = (-1)^{k-t} \binom{k+1}{t+1} \tilde{CF}(X). \tag{III.34}$$

On note que cette équation correspond sur le terme général de la série en z à la relation

$$g_n = \bar{cp}_n + \sum_{n_1 + \cdots + n_s = n-s+1} P_{n_1,\ldots,n_s}(\bar{cf}_{n_1} + \cdots + \bar{cf}_{n_s}) \tag{III.35}$$

qui est à rapprocher de la récurrence caractéristique donnée par la formule III.11 p. 40. On retrouve donc bien le fait que $g_n = \bar{c}_n$ pour $M+1 \leq n \leq M+k-t$.

Introduisons maintenant le polynôme $A_{s,t}(x)$ défini par :

$$A_{s,t}(x) = \frac{PQ_{s,t}(x)}{(x+2)\,x(x-1)(x-2)\cdots(x-t+1)}. \tag{III.36}$$

C'est le polynôme de degré $k-t-1$ dont les racines sont les racines complexes de $PQ_{s,t}(x)$ et éventuellement la racine $(s+1)(t+1)-1$. Si on applique le polynôme $A_{s,t}(\Theta)$ à la formule donnée par le théorème III.2 p. 45, on a alors élimination des termes indéterminés générés par les racines de $A_{s,t}$. Il ne reste ainsi que le terme λ_0 qui provient de la racine -2 de $PQ_{s,t}$. Compte tenu de la proposition III.6 p. 44, on a donc :

$$A_{s,t}(\Theta).[\tilde{C}(X)] = A_{s,t}(\Theta).[F(X) + \frac{\lambda_0}{X^2} + R_M(X)]\,, \tag{III.37}$$

d'òu

$$\frac{A_{s,t}(-2)\lambda_0}{X^2} = A_{s,t}(\Theta).[\tilde{C}(X) - F(X) - R_M(X)]\,. \tag{III.38}$$

Le polynôme $A_{s,t}(x)$ étant de degré $k-t-1$, le coefficient du terme z^{M+1} dans la série $A_{s,t}(\Theta).[\tilde{C}(X)-F(X)-R_M(X)]$ ne dépend que des termes généraux d'ordre $M+1$ à $M+k-t$ dans $\tilde{C}(1-z)-F(1-z)-R_M(1-z)$. On peut donc éliminer $R_M(X)$ et remplacer $\tilde{C}(X)$ par $G(X)$ pour obtenir la constante λ_0.

Proposition III.8 *La constante λ_0 du terme linéaire dans le coût moyen est :*

$$\lambda_0 = \frac{[z^{M+1}]A_{s,t}(\Theta).[G(X)-F(X)]}{A_{s,t}(-2)\,(M+2)}$$

où $F(X)$ est défini par le théorème III.2 et où $G(X)$ est une solution particulière de :

$$\binom{\Theta-t}{k-t}.[G(X)-\tilde{CP}(X)] = (-1)^{k-t}\binom{k+1}{t+1}\tilde{CF}(X).$$

Cette proposition s'étend facilement aux autres constantes d'intégration λ_i en modifiant le polynôme $A_{s,t}$. Le calcul effectif de la constante n'est toutefois possible que si on connaît explicitement la valeur de la racine de $PQ_{s,t}$ correspondante, c'est-à-dire pour s et t donnés. Par contre, -2 étant toujours racine, la proposition précédente donne des valeurs générales de λ_0, paramétrées par les quantités caractéristiques s, t et M.

Exemple III.2 :

1. Considérons le nombre de comparaisons durant le partitionnement qui est défini par $\bar{cp}_n = a(n+1)+b$ et $\bar{cf}_n = 0$. Avec ou sans la proposition III.7, on a trivialement $G(X) = \tilde{CP}(X)$. la constante λ_0 est donc donné par

$$\begin{aligned}
\lambda_0(M+2)A_{s,t}(-2) &= [z^{M+1}]A_{s,t}(\Theta).[\tilde{CP}(X)-F(X)] \\
&= [z^{M+1}]A_{s,t}(\Theta).[\frac{a}{X^2}+\frac{b}{X}-ak_{s,t}\frac{-\ln X}{X^2}+\frac{b}{(s-1)X}] \\
&= a\,[z^{M+1}]\frac{A_{s,t}(-2)\,(1+k_{s,t}\ln(1-z))+k_{s,t}A'_{s,t}(-2)}{(1-z)^2} \\
&\quad + b\,[z^{M+1}]\frac{sA_{s,t}(-1)}{(s-1)(1-z)}.
\end{aligned}$$

   D'où

$$\lambda_0 = a\left(1+\frac{k_{s,t}A'_{s,t}(-2)}{A_{s,t}(-2)}-k_{s,t}(H_{M+2}-1)\right)+b\frac{sA_{s,t}(-1)}{(s-1)A_{s,t}(-2)(M+2)}. \qquad \text{(III.39)}$$

   En calculant les valeurs particulières du polynôme $A_{s,t}$ (cf. section B.4.), on obtient, en fonction de s, t, et M, le terme linéaire λ_0 donné dans les formules de la section 2. de l'annexe C.

2. Considérons maintenant un tri par insertion pour les petites listes. Si l'on compte le nombre de clés réellement déplacées durant l'insertion, on a $\bar{cf}_n = n - H_n$ et $\bar{cp}_n = 0$. On prend donc pour $\tilde{CF}(X)$:

$$\tilde{CF}(X) = \frac{1}{X^2}+\frac{\ln(X)-1}{X}. \qquad \text{(III.40)}$$

D'où l'on obtient par intégration

$$G(X) = \frac{1}{X^2} + s\frac{\ln(X) + H_k - H_t - 1}{X}. \tag{III.41}$$

Le coût moyen pour les listes de taille $n \in [M+1, .., M+k-1]$ est alors $\bar{c}_n = n + 1 - sH_n + s(H_k - H_t - 1)$. De plus $F(X) = 0$, donc la constante λ_0 qui donne la composante dominante du coût est obtenue en intégrant $G(X)$ par $A_{s,t}(\Theta)$ (cf. résultats dans la section C.2.).

On donne en annexe des résultats exacts, ou asymptotiques avec une erreur en $o(n)$ pour les valeurs moyennes des quantités caractéristiques définies par Sedgewick [Sed80] pour l'algorithme Quicksort. On étend ainsi les résultats obtenues dans la thèse de D. Greene [Gre83] pour des paramètres liés plus spécifiquement aux arbres réduits binaires.

4. Sélection non uniforme des clés de partitionnement

On considère ici l'algorithme avec la procédure plus générale de sélection de la clé de partitionnement qui est décrite dans la section II.5. p. 25. L'échantillon de k clés, avec k quelconque, est donc découpé en s sous-listes de taille $t_1, t_2, \ldots, t_s$ suivant une distribution de probabilité $P(t_1, t_2, \ldots, t_s)$ avec $\sum t_i = k - s + 1$. Le cas précédent est obtenu en faisant $P(t, t, \ldots, t) = 1$ et $k = st + s - 1$.

4.1. Coût moyen.

La description formelle donnée par le théorème III.1 p. 40 s'adapte simplement au cas non équiréparti. Il suffit pour cela d'introduire une union ou une somme de classes de structures combinatoires pondérée par la distribution de probabilité $P(t_1, t_2, \ldots, t_s)$. On représente donc formellement la décomposition des permutations induite par l'algorithme sous la forme :

$$\mathbf{S} = T_M \Big[\sum_{\sum t_i = k+1-s} P(t_1, t_2, \ldots, t_s) \circ_k \left(\Delta_{t_1}(\mathbf{S}) \otimes \Delta_{t_2}(\mathbf{S}) \otimes \cdots \otimes \Delta_{t_s}(\mathbf{S}) \right) \Big] + R_M(\mathbf{S}). \tag{III.42}$$

Après quelques manipulations similaires au cas équiréparti, on en déduit l'équation différentielle vérifiée par la série génératrice de coût $\tilde{C}(X)$. On constate alors que la généralisation a simplement pour effet de changer le polynôme caractéristique $PQ_{s,t}$. On obtient ainsi l'équation :

$$\begin{aligned} PG_{s,k}(\Theta).\tilde{C}(X) &= (-1)^k \binom{\Theta}{k}.\tilde{CP}(X) + X^k \tilde{Q}_{M-k}(X) \\ \text{avec} \qquad PG_{s,k}(x) &= (-1)^k \binom{x}{k} - \sum_{\sum t_i = k-s+1} P(t_1, t_2, \ldots, t_s) \left[\sum_{i=1}^{s} (-1)^{t_i} \binom{x}{t_i} \right]. \end{aligned} \tag{III.43}$$

Remarque : Le polynôme $PG_{s,k}$ est symétrique en les t_i. Le comportement du tri est donc identique pour toutes les distributions $P(t_1, t_2, \ldots, t_s)$ qui correspondent à la même distribution des tailles indépendamment de l'ordre des t_i. Toute distribution $P(t_1, t_2, \ldots, t_s)$ est alors équivalente à sa distribution symétrique $P'(t_1, \ldots, t_s) = \sum_{\sigma \in S_s} P(t_{\sigma(1)}, \ldots, t_{\sigma(s)})/s!$.

Les propriétés de ce nouveau polynôme $PG_{s,k}$ sont étudiées dans l'annexe B. Le résultat utile est que -2 est toujours la racine de plus petite partie réelle (cf. lemme B.8 p. 127). L'intégration est donc très semblable au cas équiréparti, en particulier si on ne s'intéresse qu'au terme dominant en $n \ln n$ du coût moyen.

Ainsi si $\tilde{CP}(X) = a/X^2 + O(1/X)$, la résolution donne :

$$C(1-z) = \frac{k+1}{PG'_{s,k}(-2)} \frac{\ln(1-z)}{(1-z)^2} + O\left(\frac{1}{(1-z)^2}\right). \tag{III.44}$$

En déterminant la valeur de $PG'_{s,t}(-2)$, on obtient donc la proposition suivante qui donne le comportement asymptotique du coût moyen en fonction de la distribution de probabilité $P(t_1, t_2, \ldots, t_s)$.

Proposition III.9 *Le nombre moyen de comparaisons ($\bar{\varphi}_n = \bar{a}(s)n + O(1)$) pour l'algorithme de tri Quicksort avec médiane généralisée vaut :*

$$\bar{c}_n = \bar{a}(s) g_{s,k} (n+1) H_{n+1} + O(n).$$

où

$$g_{s,k} = \left(H_{k+1} - \sum_{\sum_i t_i = k-s+1} P(t_1, t_2, \ldots, t_s) \left[\sum_{i=1}^{s} \frac{t_i+1}{k+1} H_{t_i+1} \right] \right)^{-1}.$$

4.2. Cas optimal.

On cherche ici quelles sont les distributions de probabilité $P(t_1, t_2, \ldots, t_s)$ qui rendent la constante $g_{s,k}$ minimum, c'est-à-dire qui correspondent asymptotiquement au meilleur coût moyen. On s'attend à ce que le tri soit optimal pour des distributions qui produisent avec une probabilité 1 des t_i égaux ou ne différant que d'une unité. C'est effectivement ce que l'on va montrer en utilisant le concavité de la fonction $f(n) = (n+1)H_{n+1}$.

Pour k et s fixés, on introduit les entiers r et d définis par la division euclidienne :

$$k+1 = (r+1)s + d \qquad \text{et} \qquad 0 \leq d < s. \tag{III.45}$$

On prolonge sur les réels positifs la fonction $f(n) = (n+1)H_{n+1}$ par la fonction affine par morceaux $\bar{f}(x)$:

$$\bar{f}(x) = f(r) + (x-r)(f(r+1) - f(r)) \qquad \text{avec} \qquad r = \lfloor x \rfloor \tag{III.46}$$

La fonction $f(n)$ est convexe et donc la fonction $\bar{f}(x)$ l'est aussi ; d'où pour des entiers quelconques $t_1, t_2, \ldots, t_s$ la relation :

$$\sum_{i=1}^{s} f(t_i) = \sum_{i=1}^{s} \bar{f}(t_i) \geq s\bar{f}\left(\sum_{i=1}^{s} \frac{t_i}{s}\right). \tag{III.47}$$

En reprenant la constante $g_{s,k}$, on a donc

$$\begin{aligned} \frac{1}{g_{s,k}} &\leq H_{k+1} - \sum_{\sum_i t_i = rs+d} P(t_1, t_2, \ldots, t_s) \frac{s}{k+1} \bar{f}\left(\frac{rs+d}{s}\right) \\ &= H_{k+1} - \frac{1}{k+1}\left(s\, f(r) + d\,(f(r+1) - f(r))\right). \end{aligned} \tag{III.48}$$

En notant que $f(r+1) = f(r) + H_{r+1} + 1$, on obtient donc

$$\frac{1}{g_{s,k}} \leq H_{k+1} - H_{r+1} - \frac{d}{k+1}. \tag{III.49}$$

On a ainsi un minorant de la constante $g_{s,k}$ qui est de plus atteint pour certaines distributions des $P(t_1, t_2, \ldots, t_s)$.

Proposition III.10 *Le tri avec médiane généralisée est asymptotiquement optimal pour une distribution $P(t_1, t_2, \ldots, t_s)$ vérifiant :*

$$\forall\, i, j \quad |t_i - t_j| > 1 \Longrightarrow P(t_1, t_2, \ldots, t_s) = 0.$$

La valeur optimale de la constante $g_{s,k}$ est alors :

$$Kopt_{s,k} = \frac{1}{H_{k+1} - H_{r+1} - \frac{d}{k+1}} \qquad \textit{avec} \qquad k+1 = (r+1)s + d.$$

Preuve : Une distribution de probabilité vérifiant la proposition partage l'échantillon de k clés en s listes de taille r ou $r+1$. Plus précisément, on va avoir d sous-listes de taille $r+1$ et $s-d$ de taille r car la somme des tailles doit être $k - s + 1 = rs + d$. On vérifie alors simplement que la constante $g_{s,k}$ est bien égale à $Kopt_{s,k}$. □

Remarque : Si on compare le cas $d = 0$ qui décrit à la médiane classique, et le cas $d = 1$, on s'aperçoit qu'ils correspondent à la même valeur de $Kopt$ ($Kopt_{s,s(r+1)} = k_{s,r+1} = Kopt_{s,s(r+1)+1}$). Dans le cas binaire $s = 2$, cela montre en particulier le faible intérêt de la variante médiane de k clés avec k pair car le gain asymptotique optimal est égal à celui de la médiane de $k-1$ clés.

5. Conclusions.

Les sections 1. et 2. de l'annexe C donnent les formules pour les coûts moyens des quantités caractéristiques de l'algorithme Quicksort introduites par R. Sedgewick [Sed80]. On y détermine les formules exactes pour médiane de 1, 3, 5, et 7 dans le cas binaire $s = 2$ et les développements asymptotiques des coûts dans le cas général du multipartitionnement avec médiane et tri particulier des petites listes. Ces développements asymptotiques contiennent les termes linéaires des coûts moyens et donnent donc une erreur en $O(n^{b_{s,t}}) = o(n)$ qui est négligeable dans la pratique pour des valeurs raisonnables de s et t.

Le coût moyen réel de l'algorithme, c'est-à-dire le temps moyen d'exécution, s'exprime comme combinaison linéaire des quantités caractéristiques et a donc la forme générale :

$$\bar{T}_n = \bar{a}(s)\, k_{s,t} n \ln n + f(M)n + o(n) \tag{III.50}$$

où $k_{s,t} = (H_{s(t+1)} - H_{t+1})^{-1}$ et $\bar{a}(s)$ est le coût moyen de traversée d'un nœud s-aire, c'est-à-dire de recherche infructueuse parmi $s-1$ clés.

Ce temps d'exécution est exprimé dans une unité de temps fixée par la machine utilisée et dépend directement, par la constante $f(M)$, de l'implantation choisie.

L'annexe D donne les premières valeurs de $k_{s,t}$ et de $K_{s,t} = \bar{a}(s)k_{s,t}$ dans le cas le plus général où les nœuds sont des arbres binaires équilibrés (cf. tableaux D.2, D.3 p. 138). On constate de plus facilement que pour de grandes valeurs de s et t, la constante $K_{s,t}$ tend lentement vers la valeur $1/\ln 2 = 1.442\ldots$ qui correspond à la constante optimale que l'on peut espérer pour un tri opérant par comparaison de clés ([Knu73]).

Il apparaît ainsi que l'utilisation de la médiane apporte asymptotiquement un gain significatif sur les coûts moyens en particulier pour les premières valeurs $t = 1$ ou $t = 2$. Le multipartitionnement semble par contre beaucoup moins intéressant dans la pratique sauf si on peut réduire de façon significative la valeur de $\bar{a}(s)$ (par exemple par hachage sur les nœuds). On note de plus que la constante $K_{s,t}$ n'est pas monotone en s ; les cas $s = 3$ ou $s = 5$ sont ainsi moins bons que les multipartitionnements avec $s = 2$ ou $s = 4$.

Les variantes de l'algorithme utiles d'un point de vue pratique sont donc essentiellement la médiane de 3 ou 5 clés dans le cas binaire $s = 2$. En effet, l'utilisation de la médiane d'un plus grand nombre de clés ou du multipartitionnement introduit des surcoûts linéaires qui ne sont compensés par le gain asymptotique que pour des listes de taille très importante (> 10000). En faisant quelques simulations sur des listes de taille raisonnable (< 10000), on constate ainsi un gain réel de l'ordre de 10 à 15% entre le cas standard et la médiane de 3 ou 5 clés.

Nous verrons dans le chapitre suivant, un autre argument important en faveur de l'utilisation de la médiane. La variante de la médiane a en effet pour conséquence de resserrer de façon très significative la distribution des coûts moyens de l'algorithme. Par exemple dans le cas binaire $s = 2$, l'utilisation de la médiane de 3 clés divise l'écart type de la distribution du nombre de comparaison par deux par rapport au cas standard sans médiane. La variante de la médiane réduit ainsi considérablement l'importance et l'influence des cas les plus défavorables ; le cas le pire, $\approx n^2/2(t+1)$ comparaisons, est d'autant moins probable que t est grand.

Le traitement des listes de taille inférieure ou égale à M et les différentes finesses d'implantations agissent directement sur la composante linéaire $f(M)$ du temps moyen d'exécution. Vu la faible croissance de la fonction logarithme, cette composante est loin d'être négligeable d'un point de vue pratique. Il est en particulier très utile de déterminer la valeur du paramètre M qui rend $f(M)$ minimum. Cette valeur optimale de M est indépendante de la taille n des listes et peut donc être calculée simplement pour chaque implantation envisagée par des simulations sur des listes de taille faible ($n \sim 50$). Dans le cas de l'algorithme Quicksort standard, l'utilisation d'une valeur optimale de M, qui est de l'ordre de $M = 10$ à 15, permet ainsi un gain sur le temps moyen d'exécution d'environ 10% pour des listes de taille allant jusqu'à 5000.

Chapitre IV

Distribution des coûts.

Pour affiner la description du comportement moyen de l'algorithme Quicksort ou de la recherche dans les arbres réduits, nous allons nous intéresser dans ce chapitre au calcul de la variance et des moments d'ordre supérieur de la distribution des coûts du tri et de la recherche. Nous allons notamment chercher à démontrer l'existence de lois limites pour ces distributions de probabilité et à caractériser ces lois en donnant un moyen de calculer le plus simplement possible les moments successifs par récurrence. Les distributions de probabilité vont être caractérisées par des équations différentielles sur des séries à 2 variables que l'on traite en utilisant les techniques de calcul décrites au chapitre précédent.

Pour les coûts de la recherche infructueuse et de la recherche avec succès, nous généralisons aux arbres réduits s-aires les résultats classiques de loi limite gaussienne (cf. [Knu73], [BS84], [Lou87], [MP89]). Nous montrons en particulier que les cumulants d'ordre p de la distribution sont tous de la forme $k_p \ln n + O(1)$. Nous verrons de plus comment on peut exprimer la variance et les moments d'ordre supérieur directement ou par récurrence à partir des paramètres probabilistes liés au coût de traversée d'un nœud s-aire de l'arbre ou au coût de la recherche dans une feuille. Nous généralisons ainsi les résultats sur la variance donnés dans [MP85] pour les arbres réduits binaires.

Les distributions de probabilité liées au coût du tri posent un peu plus de problèmes. On n'a notamment pas de méthodes très systématiques qui permettent de déterminer les moments en fonction de la distribution de probabilité du coût d'une étape de partitionnement. Nous verrons toutefois sur l'exemple du nombre de clés déplacées, comment on peut souvent arriver à calculer la variance ou les moments avec les mêmes techniques que pour l'analyse en moyenne.

Nous nous intéresserons plus particulièrement à la distribution du nombre de comparaisons qui constitue, d'un point de vue pratique, la composante principale pour le coût du tri. Pour cette distribution, que l'on caractérise par une équation implicite simple sur des séries génératrices à deux variables, nous allons démontrer l'existence d'une loi limite (non gaussienne). On étend ainsi les preuves de M. Régnier [Rég87] et de U. Rösler [Rös89] au cas du tri avec médiane et multipartitionnement. Nous verrons alors que l'existence de la loi limite implique une forme très simple pour les cumulants limites qui s'expriment rationnellement à partir des valeurs entières de la fonction ζ de Riemann et se calculent rapidement par récurrence. On trouve en annexe les formules générales et des tables de valeurs pour ces cumulants ainsi que les procédures MAPLE pour les déterminer automatiquement.

Cette caractérisation très fine de la distribution limite du nombre de comparaisons est complétée par quelques conjectures concernant entre autres le comportement asymptotique de la suite des cumulants. Le calcul montre en effet un comportement périodique très régulier pour les cumulants qu'il est encore difficile d'interpréter à partir des équations caractéristiques dont on dispose.

1. Moments et séries génératrices à deux variables.

Pour une fonction de coût C, on définit la quantité $C_{n,k}$ comme étant la probabilité sur les objets de taille n que le coût C soit égal à k. Pour des permutations, on a donc :

$$C_{n,k} = \frac{card(\{\sigma \in S_n / C(\sigma) = k\})}{n!}. \tag{IV.1}$$

On associe alors aux paramètres $C_{n,k}$, les séries génératrices $C_n(u)$ et $C(z,u)$ définies de la manière suivante :

Définition IV.1 *La distribution de probabilité de la fonction de coût C est caractérisée par les séries génératrices :*

$$C_n(u) = \sum_{k=0}^{\infty} C_{n,k} u^k \,,$$

$$C(z,u) = \sum_{n=0}^{\infty} C_n(u) z^n = \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} C_{n,k} u^k z^n \,.$$

On vérifie facilement par définition de la distribution de probabilité $C_{n,k}$ que $C_n(1) = 1$ et $C(z,1) = 1/(1-z)$.

La distribution des coûts est complètement caractérisée par le comportement au voisinage de $u = 1$ des fonctions $C_n(u)$ ou $C(z,u)$. Les moments sont notamment donnés par les valeurs des dérivées successives de $C_n(u)$ prises en u=1.

Ainsi le coût moyen sur les permutations de taille n vaut :

$$\bar{c}_n = \sum_k C_{n,k} = C'_n(1) = [z^n] \left. \frac{\partial C(z,u)}{\partial u} \right|_{u=1}. \tag{IV.2}$$

De même pour la variance, on a

$$Var(c_n) = \sum_k k^2 C_{n,k} - \bar{c}_n^2 = C''_n(1) + C'_n(1) - C'_n(1)^2. \tag{IV.3}$$

De façon générale, la $p^{\text{ème}}$ dérivée de $C_n(u)$ en $u = 1$ est égale au moment factoriel d'ordre p de la distribution de probabilité du coût C :

$$\hat{M}_p(n) = \sum_k k(k-1)(k-2)\cdots(k-p+1) C_{n,k} = C_n^{(p)}(1) = [z^n] \left. \frac{\partial^p}{\partial u^p} C(z,u) \right|_{u=1}. \tag{IV.4}$$

Les moments et les moments centrés d'ordre p sont alors obtenus par convolution des moments factoriels avec les nombres de Stirling et les coefficients binomiaux (cf. sections 4. et 8. de l'annexe A).

L'analyse des distributions associées au coût du tri ou de la recherche consiste dans un premier temps à déterminer la fonction $C(z,u)$ à partir des récurrences caractéristiques liant les $C_{n,k}$. Dans le cas des coûts liés à la recherche dans les arbres réduits, l'équation obtenue sur $C(z,u)$ peut être résolue explicitement avec les techniques décrites au chapitre précédent pour le calcul des coûts moyens. On obtient alors rapidement le comportement asymptotique des cumulants de la distribution. Au contraire, dans le cas du tri, on ne peut pas toujours déterminer directement la fonction $C(z,u)$. En dérivant selon la variable u l'équation implicite sur $C(z,u)$, on obtient toutefois en $u=1$ une suite d'équations sur les séries en z que l'on résout de proche en proche pour obtenir les moments factoriels successifs de la distribution.

Dans notre analyse, on décrit plutôt les moments à partir des cumulants ou semi-invariants de la distribution (cf. Knuth vol1 [Knu68]). L'utilisation de ces cumulants est en partie motivée par la forme plus simple des formules obtenues pour les résultats.

Définition IV.2 *Les cumulants de la distribution des coûts pour les permutations de taille n, $\kappa_p(n)$, sont définis à partir du développement en série :*

$$\sum_{p=1}^{\infty} \kappa_p(n) \frac{y^p}{p!} = \ln\left(C_n(e^y)\right).$$

Le cumulant d'ordre p s'exprime ainsi en fonction des dérivées d'ordres 0 à p de $C_n(u)$ en $u=1$ à partir de la formule :

$$\kappa_p(n) = \left.\frac{\partial^p}{\partial y^p} \ln(C_n(e^y))\right|_{t=0}. \tag{IV.5}$$

Le passage entre les cumulants et les moments classiques ou les moments centrés s'effectue donc en utilisant les polynômes de Bell complets et les polynômes logarithmiques (cf. sections 6. et 8. de l'annexe A). On voit alors que le premier cumulant $\kappa_1(n)$ est égal à la moyenne $\bar{c}_n$, $\kappa_2(n)$ est égal à la variance $Var(c_n)$ et $\kappa_3(n)$ est le moment centré d'ordre 3.

Propriétés principales des cumulants :

- *Une loi de distribution est caractérisée par ses cumulants au même titre que par ses moments ; c'est-à-dire moyennant une condition de régularité de type Carleman sur les moments (cf. Feller [Fel71]) qui assure l'unicité de la loi ainsi caractérisée.*
- *Pour une suite de lois, la convergence de chaque cumulant implique la convergence de la suite vers une loi limite caractérisée par les limites des cumulants (si la loi limite est déterminée de façon unique par ces cumulants limites).*
- *Pour une distribution donnée, les cumulants κ'_p associés à la distribution centrée normée ($X' = (X-\bar{X})/\sqrt{Var(X)}$) sont de la forme ($p>1$) :*

  $$\kappa'_p = \frac{\kappa_p}{\sigma^p} \qquad \textit{où} \quad \sigma = \sqrt{\kappa_2} \textit{ est l'écart-type de } X.$$
- *La loi normale est caractérisée par des cumulants nuls : $\kappa_p = 0$ pour $p>2$.*
- *La loi de Poisson est caractérisée par : $\kappa_p = \lambda$ pour $p>0$.*

Les cumulants diffèrent des moments centrés seulement sur la dernière propriété qui donne une caractérisation plus simple de la loi normale avec les cumulants.

2. Coûts de la recherche dans les arbres.

On s'intéresse dans cette section à la distribution des coûts de la recherche avec succès ou de la recherche infructueuse dans les arbres s-aires réduits avec éventuellement traitement particulier des sous-arbres de taille $< M$ (cf. chapitre II). On considère le coût de la recherche d'une clé quelconque, c'est-à-dire avec une valeur distribuée uniformément parmi les n clés de l'arbre (recherche avec succès) ou dans les $n+1$ intervalles formés par ces clés (recherche infructueuse).

On va montrer que les cumulants sont de façon générale de la forme $\kappa_p(n) = k_p \ln n + O(1)$ où les constantes k_p se déterminent simplement par récurrence et ne dépendent pas de M ou du comportement de la recherche dans les feuilles. On prouve donc la convergence vers une distribution gaussienne de la loi centrée et normée associée aux coûts de la recherche infructueuse ou avec succès. On généralise ainsi aux arbres s-aires réduits, le résultat classique sur les arbres binaires de recherche ([Knu73], [BS84], [Lou87]).

2.1. Recherche dans un arbre binaire réduit.

On se place d'abord dans le cas des arbres réduits binaires de paramètre $2t+1$ avec élagage des sous-arbres de taille inférieure à M. Les arbres s-aires sont traités dans le paragraphe suivant et constituent une généralisation simple du cas binaire. On note CS le coût de la recherche avec succès et CI le coût de la recherche infructueuse qui correspond aussi au coût d'insertion d'une nouvelle clé dans l'arbre.

D'après la structure récursive des arbres réduits, les probabilités $CI_{n,k} = Prob\{CI_n = k\}$ sont données par la récurrence suivante, pour n assez grand ($n > M \geq 2t$).

$$CI_{n,k} = \sum_{p+q=n-1} P_{p,q} \left(\frac{p+1}{n+1} CI_{p,k-1} + \frac{q+1}{n+1} CI_{q,k-1} \right) \tag{IV.6}$$

où $P_{p,q}$ représente la distribution de probabilité des tailles des sous-arbres donnée par la formule II.1 p. 18.

En effet, pour un arbre de taille n constitué de deux sous-arbres de taille p et q avec $p+q+1 = n$, le coût de la recherche est égal à un plus le coût de la recherche dans l'un des deux sous-arbres. Les probabilités de poursuivre la recherche dans l'un ou l'autre des sous-arbres sont alors $(p+1)/(n+1)$ et $(q+1)/(n+1)$ car on suppose la clé uniformément répartie parmi les $n+1$ feuilles de l'arbre.

Pour la recherche avec succès, la récurrence sur les $CS_{n,k}$ est assez similaire. La recherche se poursuit dans l'un ou l'autre des sous-arbres avec les probabilités p/n et q/n, mais peut aussi s'arrêter avec une probabilité $1/n$; d'où pour $n > M$:

$$CS_{n,k} = \sum_{p+q=n-1} P_{p,q} \left(\frac{p}{n} CS_{p,k-1} + \frac{q}{n} CS_{q,k-1} + \frac{\delta_{k,1}}{n} \right). \tag{IV.7}$$

En exprimant $P_{p,q}$, la récurrence IV.6, se réécrit :

$$(n+1)\binom{n}{2t+1}CI_{n,k} = 2\sum_{p+q=n-1}(p+1)\binom{p}{t}\binom{q}{t}CI_{p,k-1}. \qquad \text{(IV.8)}$$

Le facteur 2 provient de la symétrie entre p et q. On passe alors facilement à la série génératrice $CI_n(u) = \sum_k CI_{n,k}u^k$, pour laquelle on obtient l'équation :

$$(2t+2)\binom{n+1}{2t+2}(CI_n(u) - CI_n(0)) = 2u\sum_{p+q=n-1}(t+1)\binom{p+1}{t+1}\binom{q}{t}CI_p(u), \qquad \text{(IV.9)}$$

pour $n > M$, et avec $CI_n(0) = CI_{n,0} = \delta_{n,0}$.

En s'inspirant des équations sur les coûts moyens et en particulier de la règle de traduction III.4 p. 42, on obtient alors rapidement une équation caractéristique d'un type connu pour la série génératrice $CI(z,u)$:

$$PI_u(\Theta).(z\,CI(z,u)) = Q_M(z,u) \qquad \text{avec} \qquad PI_u(x) = \binom{x}{2t+2} - u(-1)^{t+1}\binom{x}{t+1} \qquad \text{(IV.10)}$$

où $\Theta = -(1-z)\frac{\partial}{\partial z}$ et $Q_M(z,u)$ est un polynôme de degré M en z qu'il n'est pas utile d'exprimer pour la résolution.

A partir des résultats du chapitre précédent, on peut donc intégrer cette équation caractéristique de la distribution en considérant u comme un paramètre. Les propriétés du polynôme $PI_u(x)$ au voisinage de $u = 1$ permettent de plus d'obtenir rapidement un équivalent asymptotique pour les cumulants, quand la taille n des arbres tend vers l'infini.

Le polynôme $PI_u(x)$ a les racines triviales 0,1, ..,t. Les autres racines notées $\alpha_0(u)$, $\alpha_1(u)$, .., $\alpha_t(u)$ dépendent de u, et sont caractérisées par l'équation :

$$\frac{(2t+1-x)(2t-x)\cdots(t+1-x)}{(2t+2)(2t+1)\cdots(t+2)} = u \qquad \text{(IV.11)}$$

On note d'autre part que le polynôme $PI_u(x)$ pris en $u = 1$ est très voisin du polynôme $PQ_{s,t}(x)$ qui intervient dans l'analyse en moyenne (p. 41). On a en effet

$$PI_{u=1}(x+1) = -\frac{x+1}{2t+2}PQ_{s=2,t}(x), \qquad \text{(IV.12)}$$

et donc en $u = 1$ les racines $\alpha_i(1)$ sont égales aux racines de $PQ_{s,t}(x)$ plus 1. On note alors $\alpha_0(u)$ la racine de $PI_u(x)$ qui vaut -1 en $u = 1$.

Lemme IV.1

- $\Re(\alpha_i(1)) > \alpha_0(1) = -1$ *pour* $i \neq 0$.
- *Pour u dans un voisinage de 1, l'inégalité précédente reste vraie :* $\Re(\alpha_i(u)) > \Re(\alpha_0(u))$.
- *Les racines $\alpha_i(u)$ sont toutes des racines simples dans un voisinage de $u = 1$.*

Preuve : Ces résultats proviennent directement du théorème des fonctions implicites appliqué à l'équation IV.11 et du lemme B.6 p. 125 sur les racines de $PQ_{s,t}(x)$. □

On résout maintenant sans problème l'équation IV.10 pour obtenir la forme explicite :

$$z\,CI(z,u) = \sum_{i=0}^{t} B_i(u)(1-z)^{\alpha_i(u)} + R_M(z,u) \tag{IV.13}$$

où les $B_i(u)$ sont les constantes d'intégration (cf. remarque plus loin) et $R_M(z,u)$ est un polynôme de degré M en z provenant de $Q_M(z,u)$.

On en déduit alors la série $CI_n(u)$ caractéristique de la distribution des coûts pour les listes de taille n avec $n > M$:

$$CI_n(u) = \sum_{i=0}^{t} B_i(u) \binom{\alpha_i(u)}{n+1} (-1)^{n+1} \tag{IV.14}$$

Pour n tendant vers l'infini et α non entier positif, on a l'expression asymptotique :

$$(-1)^n \binom{\alpha}{n} \sim \frac{n^{-\alpha-1}}{\Gamma(-\alpha)} = O(n^{-\Re(\alpha)-1}). \tag{IV.15}$$

Compte tenu du lemme précédent, on peut extraire la composante dominante qui provient de la racine $\alpha_0(u)$ de plus petite partie réelle et obtenir un développement asymptotique de $CI_n(u)$ pour n tendant vers l'infini :

$$\begin{aligned} CI_n(u) &= \frac{B_0(u)}{\Gamma(-\alpha_0(u))} n^{-\alpha_0(u)-1}(1+o(1)) \\ \text{ou} \qquad \ln(CI_n(u)) &= -(\alpha_0(u)+1)\ln n + O(1). \end{aligned} \tag{IV.16}$$

Par définition des cumulants, on a alors la proposition suivante :

Proposition IV.1 *Les cumulants $\kappa_p(n)$ du coût de la recherche infructueuse dans des arbres binaires réduits de taille n avec élagage des petits sous-arbres, sont de la forme :*

$$\kappa_p(n) = k_p \ln n + O(1).$$

Les coefficients k_p sont alors donnés par le développement :

$$\alpha_0(e^y) = -\sum_p k_p \frac{y^p}{p!}.$$

où $\alpha_0(u)$ est la racine de plus petite partie réelle du polynôme PI_u.

Les cumulants de la loi normée associée au coût CI de la recherche infructueuse sont alors de la forme $\kappa'_p(n) = k'_p \ln^{1-p/2} n + O(\ln^{p/2} n)$ et convergent donc vers 0 si $p > 2$; d'où la proposition suivante.

Proposition IV.2 *La distribution normée du coût de la recherche infructueuse converge pour des arbres binaires réduits de grande taille vers une distribution gaussienne.*

Dans le cas des arbres binaires de recherche standards, c'est-à-dire si $t = 0$, ce résultat à été prouvé par G. Brown et B. Shubert [BS84] en utilisant le théorème "central limit". G. Louchard [Lou87] démontre le même résultat pour la recherche avec succès, en employant des techniques d'analyse asymptotique et en particulier la méthode de col. Dans les deux cas, les preuves s'appuient sur une expression explicite de la distribution $C_{n,k}$ ou de la série génératrice $C_n(u)$ pour $t = 0$ (cf. [Knu73],[BS84]). Ces preuves se généralisent donc difficilement aux distributions dans les arbres réduits ou dans les arbres s-aires.

Le développement de la racine $\alpha_0(u)$ au voisinage de $u = 1$ et donc les coefficients k_p se calculent en fonction du paramètre t des arbres réduits en utilisant les polynômes de Bell $B_{p,i}$ et les polynômes logarithmiques L_p (cf. annexe A p. 119). En prenant le logarithme sur l'équation IV.11 en $x = \alpha_0(u)$ et $u = e^y$, on a en effet :

$$\ln(e^y) = y = \sum_{i=t+2}^{2t+2} \ln(\frac{-\alpha_0(e^y) - 1 + i}{i}) = \sum_{p \geq 1} \frac{y^p}{p\,!} \sum_{i=t+2}^{2t+2} L_p(\frac{k_1}{i}, \frac{k_2}{i}, \ldots, \frac{k_p}{i}) \qquad \text{(IV.17)}$$

On retrouve ainsi la constante caractéristique du coût moyen $k_1 = k_{s,t} = (H_{2t+2} - H_{t+1})^{-1}$. Les autres coefficients k_p sont alors déterminés par la proposition :

Proposition IV.3 *Les coefficients κ_p de la proposition IV.1 vérifient la récurrence :*

$$\sum_{i=t+2}^{2t+2} L_p(\frac{k_1}{i}, \frac{k_2}{i}, \ldots, \frac{k_p}{i}) = 0,$$

avec $k_1 = (H_{2t+2} - H_{t+1})^{-1}$.

Cette récurrence est bien définie dans la mesure où $L_p(x_1, \ldots, x_p) = x_p + L_p(x_1, \ldots, x_{p-1}, 0)$; l'équation précédente donne donc k_p en fonction des k_j pour $j < p$. En développant le polynôme logarithmique L_p, on a aussi de façon explicite :

$$k_p = \sum_{j=1}^{p} (-1)^j (j-1)! \frac{H_{2t+2}^{(j)} - H_{t+1}^{(j)}}{H_{2t+2} - H_{t+1}} B_{p,j}(k_1, k_2, \ldots, k_{p-1}, 0). \qquad \text{(IV.18)}$$

Remarque : Dans le cas standard des arbres binaires de recherche (t=0), on montre facilement que les constantes k_p valent toutes 2 : $\forall p \quad \kappa_p(n) = 2 \ln n + O(1)$. On a en effet $\alpha_0(u) = 1 - 2u$.

D'après l'équation de récurrence, on voit donc que les coefficients k_p s'expriment de façon rationnelle à partir des différences harmoniques $(H_{2t+2}^{(i)} - H_{t+1}^{(i)})$. La section C.3. p. 132 donne dans le cas s-aire les expressions générales pour les premières valeurs de k_p (aussi notées $KR_p(s,t)$). Ces constantes k_p sont tabulées suivant les valeurs des paramètres s et t dans l'annexe D p. 138.

Pour la variance du coût de la recherche, on retrouve ainsi l'expression donnée par Poblete et Munro [MP85] :

Corollaire IV.1 *La variance du coût de la recherche infructueuse dans les arbres réduits binaire de paramètre t, est de la forme :*

$$Var(CI_n) = \frac{H_{2t+2}^{(2)} - H_{t+1}^{(2)}}{(H_{2t+2} - H_{t+1})^3} \ln n + O(1).$$

On constate en particulier (cf. tableau D.4 p. 138) le gain très important obtenu sur la variance par l'utilisation des arbres réduits. L'écart-type est ainsi divisé par un facteur 1.5 si on utilise des arbres réduits avec médiane de trois plutôt que les arbres binaires de recherche standards. Ceci montre donc l'intérêt pratique des arbres réduits qui outre le gain de l'ordre de 10% en moyenne, ont pour effet de réduire considérablement l'importance des cas défavorables.

Remarque : Les $t+1$ constantes d'intégrations $B_i(u)$ peuvent être déterminées explicitement par un système linéaire à partir des fonctions $CI_t(u), \ldots, CI_{2t}(u)$ (cf. [MP85]). Ces fonctions traduisent exactement la distribution des coûts de la recherche dans les feuilles de l'arbre réduit et dépendent donc fortement de la manière d'implémenter ces feuilles dans la pratique. Le système linéaire à résoudre est de la forme $C = M.B$ où les vecteurs B et C et la matrice M ont pour composantes

$$C_i = (-1)^{t+i}\frac{(t+i)!}{t!}CI_{t+i}(u), \qquad B_i = \frac{B_i(u)}{\binom{\alpha_i(u)}{t}}, \qquad M_{i,j} = i!\binom{\alpha_j(u)-t}{i}, \qquad i,j \in [0..t],$$

La matrice M est égale au produit de la matrice S_1 des nombres de Stirling de première espèce avec la matrice V de Vandermonde associée aux valeurs $\alpha_j(u) - t$. Les nombres de Stirling donnent en effet la correspondance entre les factoriels descendantes $(M_{i,j})$ et les fonctions puissances (cf. section A.4.). Le système s'inverse donc avec $M^{-1} = V^{-1}S_2$ où $S_2 = S_1^{-1}$ est la matrice des nombres de Stirling de deuxième espèce. On note que V est inversible car les racines $\alpha_j(u)$ sont toutes distinctes. Ce calcul permet donc, si on le désire, de déterminer les expressions exactes ou simplement les termes constants des cumulants pour différentes implantations pratiques des arbres réduits. Poblete et Munro [MP85] utilisent ce calcul pour déterminer les valeurs $B_i(1)$ qui sont nulles pour $i > 0$. Ils obtiennent alors la variance par dérivations successives de l'équation IV.14 évaluée en $u = 1$.

Pour le coût CS de la recherche avec succès, la résolution est très proche de celle que l'on vient de voir et donne les mêmes résultats asymptotiques pour la distribution des coûts. La récurrence caractéristique IV.7 p. 58 se traduit en effet par l'équation differentielle :

$$PS_u(\Theta).(z\frac{\partial}{\partial z}CS(z,u)) = Q_M(z,u) + \frac{u}{1-z} \tag{IV.19}$$

avec

$$PS_u(x) = -\binom{x}{2t+1} - 2u(-1)^t\binom{x}{t} \tag{IV.20}$$

où $Q_M(z,u)$ est un polynôme de degré M en z qui ne va pas intervenir dans l'analyse asymptotique.

Le polynôme $PS_u(x)$ est lié au polynôme $PI_u(x)$ par la relation

$$PI_u(x+1) = -(x+1)PS_u(x)/(2t+2). \tag{IV.21}$$

Les racines non triviales de $PS_u(x)$ ont donc les valeurs $\alpha_i(u) - 1$ avec les $\alpha_i(u)$ définis précédemment.

La résolution donne alors :

$$z\frac{\partial}{\partial z}CS(z,u) = \sum_{i=0}^{t} B_i(u)(1-z)^{\alpha_i(u)-1} + \frac{u}{(1-z)(1-2u)} + R_M(z,u), \tag{IV.22}$$

et en prenant le terme général pour $n > M$:

$$n\,CS_n(u) = \frac{u}{1-2u} + \sum_{i=0}^{t} B_i(u)\binom{\alpha_i(u)-1}{n}. \tag{IV.23}$$

On retrouve ainsi le même comportement asymptotique que pour le coût de la recherche infructueuse $\ln(CS_n(u)) = -(\alpha_0(u)+1)\ln n + O(1)$. Les propositions IV.1 et IV.2 sur la caractérisation de la distribution et sur la convergence vers une loi limite gaussienne, sont donc aussi valables pour la recherche avec succès en prenant les mêmes valeurs des coefficients k_p.

Remarque : Une autre façon d'obtenir ce résultat est de considérer le lien classique entre les longueurs de cheminement internes et externes dans les arbres binaires (cf. Knuth ex. 6.2.1.25 [Knu73]). On obtient ainsi une relation entre CI et CS de la forme :

$$n\,CS_n(u) = \frac{u((n+1)CI_n(u)-1)}{2u-1} \qquad (n > M). \tag{IV.24}$$

2.2. Arbres s-aires réduits avec nœuds fixes ou aléatoires.

On considère maintenant le coût de la recherche dans les arbres s-aires réduits. Ces coûts dépendent fortement du coût de traversée des nœuds s-aires de l'arbre. Pour un nœud donné, on note $a_i(s)$ le coût de traversée nécessaire pour atteindre le $i^{\text{ème}}$ fils de ce nœud avec $i \in [1..s]$. Dans le cas général où les nœuds sont eux-mêmes des arbres (algorithme n'opérant que par comparaison), $a_i(s)$ correspond à la profondeur de la $i^{\text{ème}}$ feuille de ce "nœud-arbre".

Nous supposons dans un premier temps que les nœuds de l'arbre s-aire sont des arbres binaires identiques et complets. C'est-à-dire que si s est de la forme $s = 2^h + s_0$ avec $0 < s_0 \le 2^h$, on a $a_i(s) = h+1$ pour $i \le 2s_0$ et $a_i(s) = h$ sinon. Dans un deuxième temps, nous généraliserons le résultat pour des nœuds quelconques définis par une distribution de probabilité $P(a_1,\dots,a_s)$ sur les coûts de traversée des nœuds s-aire. On obtient ainsi les cumulants pour les coûts de la recherche dans un arbre s-aire réduit directement en fonction des cumulants de cette distribution $P(a_1,\dots,a_s)$ du coût dans les nœuds.

Remarque : Pour avoir toujours des exposants entiers dans les séries génératrices, on suppose que les coûts a_i sont des nombres entiers. Cette contrainte n'est pas gênante et peut bien évidement être réalisée par un bon choix de l'unité de coût.

Nœuds fixes.

On considère que les nœuds sont identiques et ont leurs feuilles aux profondeurs $a_i(s) = h$ ou $h+1$. L'équation de récurrence sur les $CI_{n,k}$ (équation IV.6 p. 58), se généralise simplement sous la forme :

$$CI_{n,k} = \sum_{n_1+\cdots+n_s=n-s+1} P_{n_1,\dots,n_s} \sum_{i=1}^{s} \frac{n_i+1}{n+1} CI_{n_i,k-a_i(s)}. \tag{IV.25}$$

où $P_{n_1,\dots,n_s}$ est la probabilité d'avoir des sous-arbres de tailles $n1,\dots,n_s$ pour un arbre s-aire réduit de taille n (cf. formule II.3 p. 26).

La résolution est alors très similaire au cas binaire et va donner la récurrence :

$$CI_n(u) = \sum_{n_1+\cdots+n_s=n-s+1} P_{n_1,\dots,n_s} \sum_{i=1}^{s} \frac{n_i+1}{n+1} CI_{n_i}(u) u^{a_i(s)}, \tag{IV.26}$$

et l'équation sur la série $CI(z,u)$:

$$PI_{s,u}(\Theta).(zCI(z,u)) = Q_M(z,u)$$

avec

$$PI_{s,u}(x) = (-1)^{s(t+1)}\binom{x}{s(t+1)} + (-1)^t N(u)\binom{x}{t+1} \tag{IV.27}$$

et

$$N(u) = (1 - 2\frac{s_0}{s})u^h + 2\frac{s_0}{s}u^{h+1} = u^h(1 + \frac{2s_0(u-1)}{s})$$

La différence avec le cas binaire $s = 2$, réside uniquement dans le polynôme $PI_{s,u}(x)$ dont les $(s-1)(t+1)$ racines non triviales $\alpha_i(u)$ sont caractérisées par l'équation :

$$\frac{(s(t+1)-1-x)\cdots(t+1-x)}{s(t+1)\cdots(t+2)} = N(u) \tag{IV.28}$$

Pour $u = 1$, on constate que $N(1) = 1$ et que les racines $\alpha_i(1)$ de $PI_{s,u=1}(x)$ sont égales aux racines non triviales du polynôme $PQ_{s,t}(x-1)$. On retrouve donc la racine principale $\alpha_0(u)$ de $PI_{s,u}(x)$ telle que $\alpha_0(1) = -1$. On note de plus, que le polynôme $N(u)$ correspond exactement au polynôme caractéristique associé à la distribution de probabilité des coûts de traversée des nœuds de l'arbre.

Le lemme IV.1 p. 59 s'étend alors facilement aux racines $\alpha_i(u)$ de $PI_{s,u}(x)$ et donne pour $CI(z,u)$ et $CI_n(u)$ les mêmes formules que dans le cas binaire (p. 60), avec des sommes étendues aux $(s-1)(t+1)$ racines de l'équation IV.28. De même, la détermination des constantes d'intégration ne pose pas, d'un point de vue théorique, plus de problèmes que pour $s = 2$.

Les cumulants de la distribution des coûts CI dans le cas des arbres s-aires sont donc données par la proposition :

Proposition IV.4 *Les cumulants $\kappa_p(n)$ du coût la recherche infructueuse dans des arbres s-aires réduits dont les nœuds sont des arbres binaires complets, sont de la forme :*

$$\kappa_p(n) = k_p \ln n + O(1).$$

Les coefficients k_p vérifient la récurrence :

$$\sum_{i=t+2}^{s(t+1)} L_p(\frac{k_1}{i}, \frac{k_2}{i}, \ldots, \frac{k_p}{i}) = L_p(2\frac{s_0}{s}, 2\frac{s_0}{s}, \ldots, 2\frac{s_0}{s}) \qquad \textit{pour} \quad p > 1\,,$$

$$\textit{et} \quad k_1 = \frac{h + 2\frac{s_0}{s}}{H_{s(t+1)} - H_{t+1}} = \frac{\bar{a}(s)}{H_{s(t+1)} - H_{t+1}}$$

où $\bar{a}(s) = h + 2s_0/s$ est le coût moyen de traversée d'un nœud.

La récurrence se déduit simplement de l'équation IV.28 et du développement de la fonction $\ln(N(e^y))$ qui donne les termes $L_p(2s_0/s,\ldots)$. On retrouve pour le coût moyen la constante k_1 attendue qui dépend directement du coût moyen $\bar{a}(s)$ sur chaque nœud. On trouve dans les sections C.3. et D.2., les formules pour les premiers coefficients k_p et des tables de valeurs de ces coefficients suivant les paramètres s et t.

Pour la variance, on obtient ainsi l'expression :

Corollaire IV.2

$$Var(CI_n) = (\frac{\bar{a}(s)(H^{(2)}_{s(t+1)} - H^{(2)}_{t+1})}{(H_{s(t+1)} - H_{t+1})^3} + \frac{v(s)}{H_{s(t+1)} - H_{t+1}}) \ln n + O(1).$$

où $\bar{a}(s) = h + 2s_0/s$ et $v(s) = 2s_0(s-2s_0)/s^2$ sont la moyenne et la variance du coût de traversée d'un nœud s-aire de l'arbre.

Ce résultat généralise l'expression obtenue par Baeza-Yates [BY87] dans l'analyse d'arbres s-aire non réduits ($t = 0$). Les tableaux de valeurs de l'annexe D montrent toutefois l'intérêt limité des arbres s-aires dans le cas général où les nœuds sont eux-mêmes des arbres binaires, c'est-à-dire pour un tri opérant uniquement par comparaisons.

Nœuds aléatoires

On suppose maintenant que les nœuds s-aires des arbres de recherche ont une forme aléatoire déterminée par une distribution de probabilité $P(a_1, \ldots, a_s)$; $P(a_1, \ldots, a_s)$ donne la probabilité d'atteindre chacun des s fils d'un nœud avec des coûts respectifs $a_1, a_2, \ldots, a_s$. Les équations de récurrence sur le coût CI de la recherche infructueuse s'obtiennent facilement à partir du cas précédent en introduisant la pondération $P(a_1, \ldots, a_s)$. On obtient ainsi la même équation sur la série caractéristique $CI(z, u)$ avec simplement un polynôme $N(u)$ un peu plus général. On trouve en effet l'équation différentielle :

$$PI_{s,u}(\Theta).(zCI(z, u)) = Q_M(z, u)$$

où

$$PI_{s,u}(x) = (-1)^{s(t+1)}\binom{x}{s(t+1)} + (-1)^t N(u)\binom{x}{t+1}, \tag{IV.29}$$

avec maintenant :

$$N(u) = \sum_{a_1,a_2,\ldots,a_s} \sum_{i=1}^{s} P(a_1, a_2, \ldots, a_s)\frac{u^{a_i}}{s}. \tag{IV.30}$$

On constate que le polynôme $N(u)$ est exactement le polynôme caractéristique de la distribution de probabilité du coût de traversée d'un nœud. En particulier, la fonction $\ln(N(e^y))$ est la série génératrice exponentielle des cumulants $\kappa_p^{\text{nœud}}$ de cette distribution.

Les propriétés des racines de $PI_{s,u}(x)$ au voisinage de $u = 1$ et en particulier le lemme IV.1 p. 59, s'étendent sans problème au cas des nœuds quelconques car on a $N(1) = 1$. La résolution donne alors comme précédemment

$$z\,CI(z, u) = \sum_{i=0}^{s(t+1)} B_i(u)(1-z)^{\alpha_i(u)} + R_M(z, u) \tag{IV.31}$$

où les $B_i(u)$ sont les constantes d'intégration et les $\alpha_i(u)$ sont les racines de $PI_{s,u}$. On en déduit alors le comportement asymptotique

$$\begin{aligned} CI_n(u) &= \frac{B_0(u)}{\Gamma(-\alpha_0(u))} n^{-\alpha_0(u)-1}(1 + o(1)) \\ \text{ou} \qquad \ln(CI_n(u)) &= -(\alpha_0(u) + 1) \ln n + O(1), \end{aligned} \tag{IV.32}$$

avec la racine $\alpha_0(u)$ caractérisée par :

$$\ln(N(e^y)) = \sum_{i=t+2}^{s(t+1)} \ln(\frac{-\alpha_0(e^y) - 1 + i}{i}), \tag{IV.33}$$

et $\alpha_0(1) = -1$. Ceci permet donc d'énoncer le théorème général suivant :

Théorème IV.1 *La distribution de probabilité des coûts de la recherche infructueuse dans un arbre s-aire réduit avec des nœuds et des feuilles quelconques, est caractérisée par des cumulants dont la forme générale est :*

$$\kappa_p(n) = k_p \ln n + O(1).$$

Les coefficients k_p sont déterminés par la récurrence :

$$\sum_{i=t+2}^{s(t+1)} L_p(\frac{k_1}{i}, \frac{k_2}{i}, \ldots, \frac{k_p}{i}) = \kappa_p^{n\oe ud} \qquad p \geq 1,$$

où $\kappa_p^{n\oe ud}$ est le cumulant d'ordre p de la distribution de probabilité du coût de traversée d'un nœud s-aire de l'arbre.

Remarque : La même méthode s'applique au calcul des séries $CS(z,u)$ et $CS_n(u)$ liées à la recherche avec succès. On retrouve notamment les formules IV.22 et IV.23 p. 62 du cas binaire avec des sommes étendues aux $(s-1)(t+1)$ racines de $PI_{s,u}(x)$. On obtient donc exactement le même résultat pour les coûts de la recherche avec succès dans un arbre s-aire réduit général. La différence entre les deux types de coût n'affecte en pratique que les termes constants dans les cumulants. Ces termes sont alors fortement dépendants de l'implantation utilisée en particulier au niveau des feuilles.

Théorème IV.2 *Les distributions normées des coûts de la recherche infructueuse et de la recherche avec succès, dans des arbres s-aires réduits de grande taille, converge vers une distribution gaussienne.*

Remarque sur la récurrence : La formule de récurrence du théorème IV.1 peut s'inverser explicitement pour donner k_p en fonction des cumulants $\kappa_i^{\text{nœud}}$ pour $i \leq p$. D'un point de vue fonctionnel, l'équation se traduit en effet par :

$$\sum_p k_p \frac{y^p}{p!} = F^{-1}(\sum_p \kappa_p^{\text{nœud}} \frac{y^p}{p!}) \qquad \text{avec} \qquad F(x) = \sum_{i=t+2}^{s(t+1)} \ln(1 + \frac{x}{i}) = \sum_k f_k \frac{x^k}{k!}. \tag{IV.34}$$

En utilisant la formule de Lagrange avec les polynôme de Bell $B_{p,i}$ (cf. annexe A), on en déduit alors l'expression directe :

$$k_p = \sum_i B_{p,k}(\kappa_1^{\text{nœud}}, \ldots, \kappa_k^{\text{nœud}}) G_k \tag{IV.35}$$

avec $G_1 = 1$,

$$G_k = \sum_{j=1}^{k-1} (-1)^j f_1^{-k-j} B_{k+j-1,j}(0, f_2, \ldots, f_j) \tag{IV.36}$$

et

$$f_j = (-1)^{j-1}(j-1)!(H_{s(t+1)}^{(j)} - H_{t+1}^{(j)}). \tag{IV.37}$$

3. Distribution des coûts pour le tri Quicksort.

Nous allons montrer dans cette section comment on détermine les moments associés à la distribution d'un coût sur l'algorithme Quicksort. En général, on n'obtient pas obligatoirement une équation fonctionnelle simple sur la série génératrice caractéristique à deux variables $C(z,u)$. Nous verrons sur l'exemple du nombre de clés échangées lors du partitionnement comment on peut toutefois arriver à déterminer les premiers moments de la distribution en travaillant avec les mêmes outils que pour les coûts moyens. On retrouve entre autres des équations différentielles en $PQ_{s,t}(\Theta)$ pour caractériser les séries génératrices des moments.

Dans le cas du nombre de comparaisons, on obtient une équation fonctionnelle implicite sur la série génératrice $C(z,u)$ pour le tri avec médiane et multipartitionnement. Cette équation, qui introduit un couplage entre les variables z et u, ne permet pas de déterminer globalement l'ensemble des moments ; elle traduit en particulier très mal les phénomènes d'annulation qui apparaissent quand on centre la distribution et qui constituent le point essentiel dans la caractérisation de la loi limite. La caractérisation de $C(z,u)$ permet toutefois de calculer de proche en proche les moments et les cumulants de la distribution du nombre de comparaisons. Ce calcul qui s'automatise facilement en utilisant un système de calcul formel (cf. annexe E) donne, en fonction des paramètres s et t, les expressions des cumulants de l'annexe C.

3.1. Equations générales.

Pour une fonction de coût C liée au tri Quicksort, on considère la probabilité $C_{n,k}$ de trier les objets de taille n avec un coût égal à k. En reprenant les notations précédentes ($P_{p,q}$ donnée par la formule II.1 p. 18), la structure de l'algorithme induit, dans le cas binaire $s=2$, la récurrence suivante sur les $C_{n,k}$:

$$C_{n,k} = \sum_{p+q=n-1} P_{p,q} \left(\sum_{i+j+r=k} D(p,q,r) C_{p,i} C_{q,j} \right) \qquad (n > M) \tag{IV.38}$$

où $D(p,q,r)$ est la probabilité pour que le coût du partitionnement soit égal à r si les sous-listes créées sont de taille p et q.

Pour que le coût sur une liste de taille n soit k, il faut en effet que cette liste se partitionne en deux sous-listes de taille p et q et de coût i et j avec $n=p+q+1$ et $k=i+j+r$ où r est le coût du partitionnement. Par rapport au coût de la recherche, on voit donc apparaître dans la récurrence un couplage donné par le terme $D(p,q,r)$ entre le coût k et la taille des listes ou des sous-listes. L'autre différence avec le coût de la recherche est dans la forme produit qui traduit le fait que l'on doit compter simultanément le coût des appels récursifs sur les deux sous-listes.

La relation précédente se traduit simplement sur les séries caractéristiques de la distribution pour les objets de taille n, par la récurrence :

$$C_n(u) = \sum_{p+q=n-1} P_{p,q} D_{p,q}(u) C_p(u) C_q(u) \qquad (n > M) \tag{IV.39}$$

$$\text{avec} \qquad D_{p,q}(u) = \sum_r D(p,q,r) u^r .$$

$D_{p,q}(u)$ est ainsi le polynôme caractéristique de la distribution du coût d'un partitionnement d'une liste de taille $n=p+q+1$ en sous-listes de tailles p et q.

La relation précédente s'étend facilement au tri avec multipartitionnement d'ordre s pour donner la récurrence caractéristique :

$$C_n(u) = \sum_{p_1+\cdots+p_s=n+1-s} P_{p_1,\ldots,p_s} D_{p_1,\ldots,p_s}(u) C_{p_1}(u)\cdots C_{p_s}(u) \qquad (n > M), \tag{IV.40}$$

où $D_{p_1,\ldots,p_s}(u)$ est le polynôme caractéristique de la distribution du coût d'un partitionnement en s sous-listes de taille p_i.

Dans le cas général, cette récurrence ne permet pas d'obtenir une équation fonctionnelle simple sur la série génératrice à deux variables $C(z,u)$. Pour obtenir une telle équation, on aimerait en particulier que la fonction $D_{p,q}(u)$ puisse se factoriser sous une forme $D_{p,q}(u) = f_n(u)D_p(u)D_q(u)$.

On peut toutefois, en considérant les dérivées successives prises en $u = 1$ de l'équation IV.39, transformer les récurrences obtenues sur les quantités $C_n^{(i)}(1)$ en une suite d'équations fonctionnelles sur les séries génératrices des moments factoriels :

$$\hat{M}_i(z) = \sum_n C_n^{(i)}(1) z^n = \left.\frac{\partial^i}{\partial u^i} C(z,u)\right|_{u=1}. \tag{IV.41}$$

On obtient alors des équations du même type que celles rencontrées lors de la détermination des coûts moyens, c'est-à-dire de la forme :

$$PQ_{s,t}(\Theta).\hat{M}_i(z) = SM_i(z) \tag{IV.42}$$

où $PQ_{s,t}$ est le polynôme caractéristique défini au chapitre III et où le second membre $SM_i(z)$ dépend des moments $\hat{M}_j(z)$ d'ordre inférieur ($j < i$) et de la distribution $D_{p,q}(u)$. On voit notamment apparaître dans ce second membre les moments du coût D de partitionnement et des convolutions entre cette distribution de D et la distribution des coûts C. C'est la détermination des séries en z associées à ces derniers termes qui peut poser problème ; elle s'effectue en général à la main et au cas par cas. Cette méthode permet par exemple de déterminer la variance du nombre de clés échangées pendant le partitionnement, comme on le verra dans la section suivante.

Nous allons considérer maintenant un cas particulier qui permet d'obtenir une équation fonctionnelle sur les séries $C(z,u)$ associées au nombre de comparaisons ou au nombre de partitionnements pendant le tri. On suppose que le coût $CP(\sigma)$ d'un partitionnement est indépendant de la permutation σ à partitionner et qu'il est linéaire en fonction de la taille de σ. Pour les permutations de taille n, on pose alors $CP(n) = a(n-k)+b$ avec $k = s(t+1)-1$.

Le polynôme $D_{p,q}(u)$ caractéristique du coût d'un partitionnement prend donc la forme :

$$D_{p,q}(u) = u^b \left(u^a\right)^{n-k}. \tag{IV.43}$$

En reportant dans l'équation IV.39, on obtient alors, dans le cas binaire $s = 2$, la récurrence :

$$\binom{n}{k} C_n(u) = u^b \sum_{p+q=n-1} \binom{p}{t}\binom{q}{t} C_p(u) C_q(u) \left(u^a\right)^{(p-t)+(q-t)} \qquad (n > M) \tag{IV.44}$$

qui se traduit simplement sur la série $C(z,u)$ par l'équation :

$$\frac{1}{k!}\frac{\partial^k}{\partial z^k} C(z,u) = u^b \left(\frac{1}{t!}\frac{\partial^t}{\partial Z^t} C(Z,u)\bigg|_{Z=zu^a} \right)^2 + Q_{M-k}(z,u) \tag{IV.45}$$

où Q_{M-k} est un polynôme de degré $M-k$ en z.

Cette équation caractéristique sur $C(z,u)$ se généralise simplement au cas du tri avec multipartitionnement. En reprenant les récurrences caractéristiques, on constate que le paramètre s n'intervient que comme le degré des convolutions dans les seconds membres des équations. En écrivant le polynôme caractéristique D sous la forme

$$D_{p_1,\ldots,p_s}(u) = u^a \prod_{i=1}^{s} u^{p_i - t},$$

on a alors la proposition suivante :

Proposition IV.5 *La distribution de probabilité d'une fonction de coût C telle que le coût d'un partitionnement $CP(\sigma) = a(|\sigma| - k) + b$ soit indépendant de σ, est caractérisée, pour le tri avec médiane et multipartitionnement, par l'équation :*

$$\frac{1}{k!}\frac{\partial^k}{\partial z^k} C(z,u) = u^b \left(\frac{1}{t!}\frac{\partial^t}{\partial Z^t} C(Z,u) \bigg|_{Z = zu^a} \right)^s + Q_{M-k}(z,u), \tag{IV.46}$$

avec $C(z,1) = 1/(1-z)$.

On obtient donc une équation synthétique pour $C(z,u)$ qui contient toute l'information sur la distribution des coûts. Cette équation implicite est toutefois difficile à étudier directement à cause du couplage entre les variables z et u si $a \neq 0$. Les méthodes de transformées de Mellin (cf. [Fla85b]), qui permettent souvent une analyse asymptotique dans le cas de telles couplages, restent ici inefficaces à cause de la présence simultanée dans l'équation IV.46 de dérivations suivant z et de la forme produit (puissance). La dérivation empêche en effet de prendre un logarithme pour éliminer la puissance.

Le cas particulier $a = 0$ est étudié par H. M. Mahmoud et B. Pittel [MP89] dans l'analyse d'arbres s-aires non réduits. Il correspond par exemple à l'étude de la place occupée par un arbre s-aire de recherche ou au nombre de multipartitionnements dans le processus de tri rapide. Le comportement des distributions est très différent de celui observé dans le cas $a \neq 0$ et dépend en particulier fortement des valeurs du paramètre s.

3.2. Variance du nombre de d'échanges de clés.

On considère ici un algorithme de tri Quicksort opérant sur des tableaux de valeurs. De manière usuelle, le partitionnement s'effectue alors en déplaçant certaines clés à l'intérieur du tableau comme le montre l'exemple II.2 p. 14. Cette procédure qui est quasiment optimale dans le cas binaire devient toutefois inefficace dans le cas du multipartitionnement. Nous nous limitons donc au cas binaire $s = 2$.

Considérons le nombre B_n d'échanges de clés tel qu'il est défini par R. Sedgewick [Sed80]. Il correspond à la procédure de l'exemple II.2 p. 14 modifiée de telle sorte que l'on déplace simultanément une clé vers la fin de la liste et une clé vers le début de la liste. De plus, dans le cas de la médiane, on ne compte dans B_n que les clés déplacées lors du partitionnement proprement-dit et non les déplacements liés aux $k = 2t + 1$ éléments de l'échantillon pour la médiane.

Le problème pour le calcul des moments de B_n vient du fait que le nombre de clés déplacées à chaque partitionnement dépend de la permutation considérée ; on n'a donc pas d'équation générale sur la série génératrice à deux variables correspondante. Avec les notations précédentes, la distribution du coût d'un partitionnement en sous-listes de tailles p et q est définie par :

$$D(p,q,r+1) = \frac{\binom{p-t}{r}\binom{q-t}{r}}{\binom{p-q-2t}{p-t}}. \tag{IV.47}$$

On échange en effet $r+1$ clés si on a initialement deux ensembles de r valeurs mal placées parmi les $p-t$ et les $q-t$ positions des éléments de chaque sous-listes.

Le polynôme caractéristique de la distribution du coût d'une étape de l'algorithme est alors :

$$D_{p,q}(u) = \sum_r \frac{\binom{p-t}{r}\binom{q-t}{r}}{\binom{p-q-2t}{p-t}} u^{r+1}. \tag{IV.48}$$

On peut en particulier calculer le nombre moyen d'échanges au cours d'un partitionnement d'une permutation de taille n :

$$B_1(n) = \sum_{p+q=n-1} P_{p,q} D'_{p,q}(1) = 1 + \frac{t+1}{2(2t+3)}(n-2t-2), \tag{IV.49}$$

et le moment d'ordre 2 :

$$B_2(n) = \sum_{p+q=n-1} P_{p,q} D''_{p,q}(1) = 2B_1(n) - 2 + \frac{(t+1)(t+2)(n-2t-3)(n-2t-4)}{4(2t+3)(2t+5)}. \tag{IV.50}$$

Remarque : En posant $D_{p,q}(u) = uQ(u)$, les dérivées en $u=1$ de $D_{p,q}(u)$ sont données par :

$$D^{(j)}_{p,q}(1) = Q^{(j)}(1) + jQ^{(j-1)}(1) \qquad \text{avec} \qquad Q^{(j)}(1) = j!\frac{\binom{p-t}{j}\binom{q-t}{j}}{\binom{p-q-2t}{j}}.$$

Revenons à la relation de récurrence IV.39 p. 67 sur les séries $C_n(u)$. Par dérivation en $u=1$ et en tenant compte de la symétrie entre p et q, on obtient de façon générale les relations :

$$C'_n(1) = \sum_{p+q=n-1} P_{p,q}(D'_{p,q}(1) + C'_p(1) + C'_q(1)) = B_1(n) + 2\sum_{p+q=n-1} P_{p,q} C'_p(1),$$

$$C''_n(1) = \sum_{p+q=n-1} P_{p,q}\left(D''_{p,q}(1) + 4D'_{p,q}(1)C'_p(1) + 2C''_p(1) + 2C'_p(1)C'_q(1)\right). \tag{IV.51}$$

La première relation redonne la récurrence sur le coût moyen telle qu'on l'a rencontrée dans le chapitre III. Pour la deuxième équation qui permet de calculer la variance, le passage aux séries génératrices en z ne pose pas de problème (cf. lemme de traduction p. 42), sauf pour le terme croisé $\sum P_{p,q} D'_{p,q}(1) C'_p(1)$. D'autre part, on voit apparaître pour $C''_n(1)$ la même forme de récurrence que pour $C'_n(1)$ qui se traduit donc sur la série génératrice $\hat{M}_2(z)$ par une équation différentielle du type rencontré pour l'analyse en moyenne. A partir de l'équation IV.51, on déduit ainsi de façon générale la propriété suivante :

Proposition IV.6 *Les moments d'ordre 2 dans le tri Quicksort binaire sont donnés par l'équation différentielle :*

$$PQ_{2,t}(\Theta).\hat{M}_2(z) = -\binom{\Theta}{2t+1}(B_2(z) + 4G(z)) + 2(1-z)\left(\binom{\Theta}{t}\hat{M}_1(z)\right)^2,$$

avec

$$B_2(z) = \sum_n z^n B_2(n), \qquad B_2(n) = \sum_{p+q=n-1} P_{p,q} D''_{p,q}(1),$$

$$G(z) = \sum_n z^n G_n, \qquad G_n = \sum_{p+q=n-1} P_{p,q} D'_{p,q}(1) C'_p(1).$$

Rappels :

$$P_{p,q} = \frac{\binom{p}{t}\binom{q}{t}}{\binom{p+q+1}{2t+1}},$$

$$\binom{\Theta}{i} F(z) = (-1)^i (1-z)^i \frac{F^{(i)}(z)}{i!} = (-1)^i (1-z)^i \sum_n z^{n-i}\binom{n}{i} F_n.$$

Le problème est donc de déterminer une forme close ou un développement dans l'échelle $X^r \ln^k X$ pour la fonction $G(z)$. Dans le cas du nombre d'échanges B_n, la forme de $D_{p,q}(u)$, et notamment le fait que

$$D'_{p,q}(1) = 1 + (p-t)(q-t)/(p+q-2t),$$

permet d'obtenir facilement $G(z)$ sous la forme $G(z) = I(z) + J(z)$ avec :

$$I_n = \sum_{p+q=n-1} \frac{\binom{p}{t}\binom{q}{t}}{\binom{p+q+1}{2t+1}} C'_p(1),$$

$$J_n = \sum_{p+q=n-1} \frac{t+1}{2} \frac{\binom{p}{t+1}\binom{q}{t+1}}{\binom{p+q+1}{2t+2}} C'_p(1). \tag{IV.52}$$

D'où en utilisant la règle de traduction III.4 p. 42

$$-\binom{\Theta}{2t+1} I(z) = (-1)^t \binom{\Theta}{t}\hat{M}_1(z),$$

$$\binom{\Theta}{2t+2} J(z) = (-1)^{t+1} \frac{(t+1)z}{2(1-z)} \binom{\Theta}{t+1}\hat{M}_1(z). \tag{IV.53}$$

On obtient ainsi le corollaire suivant pour le calcul de la variance du nombre d'échanges de clés :

Corollaire IV.3 *Pour le nombre d'échanges B_n, le second moment est caractérisé par :*

$$PQ_{2,t}(\Theta).\hat{M}_2(z) = -\binom{\Theta}{2t+1} B_2(z) + 2(1-z)\left(\binom{\Theta}{t}\hat{M}_1(z)\right)^2 + 4(-1)^t\binom{\Theta}{t}\hat{M}_1(z) - 4U(z),$$

où $U(z)$ est une solution de :

$$(\Theta - 2t - 1)U(z) = (-1)^{t+1} \frac{z(t+1)^2}{1-z} \binom{\Theta}{t+1}\hat{M}_1(z).$$

La résolution ne pose donc plus aucun problème avec les règles d'intégration décrites au chapitre III.

Proposition IV.7 *La variance du nombre B_n d'échanges de clés dans le tri avec médiane est :*

$$Var(B_n) = \left(V_t - \frac{k_{2,t}(t+1)}{2(2t+3)^2(2t+4)} + \frac{1}{4(2t+3)(2t+5)} \right) n^2 + O(n \ln n + n^{2b_{2,t}}),$$

avec

$$V_t = \frac{(t+1)^2 k_{2,t}^2}{4(2t+3)^2} \left(\frac{(2t+3)H^{(2)}_{2t+2} - (t+2)H^{(2)}_{t+1}}{t+1} - \frac{\pi^2}{6} \right),$$

$$k_{2,t} = \frac{1}{H_{2t+2} - H_{t+1}} \quad et \; b_{2,t} < 1 .$$

Le coefficient V_t correspond exactement à la variance que l'on obtiendrait en considérant que le coût d'un partitionnement n'est pas une variable aléatoire mais est fixe pour toutes les permutations (cf. proposition IV.8 p. 74). Le tableau de valeur IV.1 montre entre autres que la variance est en réalité bien inférieure à ce que donne cette constante V_t. On peut aussi noter, comme on le reverra pour le nombre de comparaisons, le gain important apporté sur la variance par l'utilisation de la médiane pour partitionner.

t	0	1	2	3	4	$t \to \infty$
V_t	.01167	.00459	.00243	.00150	.00102	$r^2 / 64t^2$
$[n^2]Var(B_n)$	.000563	.000309	.000197	.000136	.000100	$(2-r)^2 / 64t^2$

avec $\quad r = 1/\ln 2 = 1.4426950.$

Tableau IV.1 : Variance du nombre de clés échangées.

Les mêmes techniques permettraient de déterminer les moments d'ordre supérieur dans le cas de la distribution du nombre d'échanges ; $P_{p,q}D^{(i)}_{p,q}$ se décompose toujours convenablement suivant p et q pour permettre le passage aux séries génératrices en z sur les termes croisés.

3.3. Expressions des cumulants du nombre de comparaisons

Pour déterminer les p premiers moments ou cumulants de la distribution, on dérive p fois suivant la variable u l'équation caractéristique IV.46 p. 69 sur la fonction $C(z,u)$. En se plaçant en $u = 1$, on obtient alors une équation différentielle qui détermine la série $\hat{M}_p(z)$ en fonction des moments d'ordre inférieur $\hat{M}_i(z)$ pour $i < p$.

Plaçons nous pour simplifier dans le cas $CP_n = n - k$ ($b = 0$, $a = 1$); après dérivation, l'équation IV.46 p. 69 donne pour le moment d'ordre p la relation suivante :

$$\hat{M}_p^{(k)}(z) = \frac{k!}{(t!)^s} \sum_{\sum_{j=1}^s i_j + p_j = p} \binom{p}{i_1, \ldots, i_s, p_1, \ldots, p_s} \prod_{j=1}^s z^{i_j} \hat{M}_{p_j}^{(t+i_j)}(z) + Q_{M-k}(z). \qquad \text{(IV.54)}$$

En faisant passer, dans le premier membre, les moments d'ordre p qui apparaissent s fois dans le second membre, on obtient alors exactement le même type d'équation différentielle que pour l'analyse en moyenne :

$$\frac{\hat{M}_p^{(k)}(z)(1-z)^k}{k!} - s\frac{\hat{M}_p^{(t)}(z)(1-z)^t}{t!} = SM_p(z) + (1-z)^k Q_{M-k}(z)\,, \qquad \text{(IV.55)}$$

avec le second membre donné par

$$SM_p(z) = (1-z)^k \sum_{p_i \neq p} \binom{p}{i_1, \ldots, p_s} \prod_{j=1}^s \frac{z^{i_j} \hat{M}_{p_j}^{(t+i_j)}(z)}{t!}. \qquad \text{(IV.56)}$$

En introduisons le polynôme caractéristique $PQ_{s,t}(\Theta)$ défini dans le chapitre précédent, on a donc :

$$PQ_{s,t}(\Theta).\hat{M}_p(X) = SM_p(z) + Q_M(X), \qquad \text{(IV.57)}$$

où Q_M est un polynôme de degré M.

Dans le cas plus générale où a et b sont quelconques, la dérivation de l'équation caractéristique sur $C(z,u)$ donne pour les séries des moments des relations de la même forme avec des expressions plus complexes des second membres $SM_p(z)$. Pour le paramètre b, on a par exemple

$$SM_p^{(b\neq 0)}(z) = \sum_{j=0}^p \binom{p}{j}\binom{b}{j} j!SM_{p-j}^{(b=0)}(z).$$

On peut en déduire alors que le coefficient b n'intervient pas dans la composante dominante des moments, et peut donc être ignoré si on ne s'intéresse qu'à un équivalent asymptotique des moments. D'autre part, en reprenant la définition de la série génératrice $C(z,u)$ et son équation caractéristique, on montre facilement que les moments d'ordre p pour $a \neq 1$ sont obtenus en multipliant par a^p les moments d'ordre p pour $a = 1$. L'équation IV.57 avec le second membre défini par l'équation IV.56 permet donc de déterminer le comportement dominant de la distribution dans le cas le plus général où a et b sont quelconques.

On détermine alors de proche en proche les séries $\hat{M}_p(X)$ et les moments successifs de la distribution à partir des règles d'intégration décrites au chapitre III. La résolution ne pose pas de problèmes particuliers et s'automatise facilement à l'aide d'un langage de calcul formel comme par exemple MAPLE (cf. annexe E). On obtient ainsi les formules de la section C.4. p. 133 qui donnent, en fonction des paramètres s et t, les expressions asymptotiques pour les 6 premiers cumulants de la distribution du nombre de comparaisons (cf. annexe D pour des tables de valeurs).

Le calcul manuel ou automatique des moments successifs est toutefois vite limité par la taille des expressions mises en jeu dans les calculs intermédiaires comme au niveau des résultats. Il est en particulier très utile de pouvoir, comme le permet la résolution, éliminer au fur et à mesure les termes qui n'introduisent pas de composantes dominantes dans les moments. Si on ne s'intéresse qu'à un équivalent asymptotique des moments, on va par exemple omettre les termes z^{i_j} dans l'équation IV.54.

Le calcul complet donne ainsi la formule suivante pour la variance du nombre de comparaisons dans le tri :

Proposition IV.8

$$\begin{aligned} Var(C_n) &= a^2 k_{s,t}^2 \left(\frac{(st+s+1)H^{(2)}_{s(t+1)} - (t+2)H^{(2)}_{t+1}}{(s-1)(t+1)} - \frac{\pi^2}{6} \right) n^2 \\ &- {}^2k_{s,t}^3 (H^{(2)}_{s(t+1)} - H^{(2)}_{t+1}) n H_n + O(n + n^{2b_{s,t}}), \end{aligned} \tag{IV.58}$$

avec

$$k_{s,t} = \frac{1}{H_{s(t+1)} - H_{t+1}} \text{ et } b_{s,t} < 1\,.$$

Les valeurs du coefficient du terme dominant en n^2, données par la table D.8 p. 140, montrent notamment le grand intérêt de la médiane dans le choix de la clé de partitionnement. La variance est en effet divisée par un facteur 4 si on utilise la médiane de $2s-1$ éléments et encore par un facteur 2 avec médiane de $3s-1$. Quand t tend vers l'infini, la variance décroît comme $1/t^2$ et donne un écart type proportionnel à la taille des listes qui est équivalent à $a/(\sqrt{2s}\ln(s)\,t)\,n$. On constate de plus que le cas le plus défavorable, $\approx a\,n^2/(2(s-1)(t+1))$, comparaisons décroît dans les mêmes proportions que l'écart type (pour t quelconque). Le choix de la clé de partitionnement comme médiane de $k = s(t+1)-1$ clés à donc pour effet de réduire de façon très significative l'influence en nombre et en importance des cas défavorables. Dans la pratique, la variante de la médiane peut ainsi être utilisée de façon effective même si le gain sur le coût moyen parait limité comme par exemple pour médiane de 5.

On remarque d'autre part que dans le cas $a = 0$ étudié par M. Mahmoud et B Pittel [MP89], c'est éventuellement les racines complexes du polynôme $PQ_{s,t}(x)$ qui vont déterminer l'équivalent asymptotique de la variance. La variance est ainsi linéaire pour les premières valeurs de s et t et tend à devenir quadratique pour s ou t tendant vers l'infini.

En reprenant de plus près l'équation IV.57 et notamment le caractère homogène du second membre SM_p, on obtient à priori une forme générale pour les moments de la distribution à n'importe quel ordre.

Proposition IV.9 *Les moments de la distribution du nombre de comparaisons avec $CP(n) = a(n-k)+b$ sont de la forme :*

$$M_p(n) = a^p Pol_p(\ln n) n^p + O\left((n \ln n)^{p-1}\right)$$

où Pol_p est un polynôme de degré p indépendant de a et b.

Les coefficients du polynôme Pol_p s'expriment de façon rationnelle à partir des valeurs de la fonction Zeta de Riemann : $\zeta(2), \ldots, \zeta(p)$.

Preuve : La preuve se fait par récurrence en supposant que les séries des moments pour $i < p$ sont de la forme :

$$\hat{M}_i(z) = \frac{P_i(\ln(1-z))}{(1-z)^{i+1}} + O\left(\frac{\ln^i(1-z)}{(1-z)^i}\right),$$

où P_i est un polynôme de degré i à coefficients rationnels.

Compte tenu de la règle de dérivation

$$\frac{d}{dz}\frac{\ln^j(1-z)}{(1-z)^i} = \frac{i\ln^j(1-z) - j\ln^{j-1}(1-z)}{(1-z)^{i+1}}$$

les termes $\hat{M}_i(z)^{(k)}$ vérifient donc :

$$\hat{M}_i(z)^{(k)} = \frac{P1_{i,k}(\ln(1-z))}{(1-z)^{i+1+k}} + O\left(\frac{\ln^i(1-z)}{(1-z)^{i+k}}\right).$$

où les $P1_{i,k}$ sont des polynômes de degré i à coefficients rationnels.

Le second membre de l'équation IV.57 s'écrit alors alors :

$$SM_p(z) = (1-z)^k \sum_{p_i \neq p} \binom{p}{i_1, \ldots, p_s} \prod_{j=1}^{s} \frac{z^{i_j} P1_{p_j, t+i_j}(\ln(1-z))}{t!(1-z)^{i_j+1+t+p_j}} + O\left(\frac{\ln^p(1-z)}{(1-z)^p}\right).$$

D'où

$$SM_p(z) = \frac{P2_p(\ln(1-z))}{(1-z)^{p+1}} + O\left(\frac{\ln^p(1-z)}{(1-z)^p}\right),$$

où $P2_p$ est un polynôme de degré p à coefficients rationnels.

Les règles de résolution données par la proposition III.4 p.42 assurent donc pour l'équation IV.57 une solution de la forme

$$\hat{M}_p(z) = \frac{P_p(\ln(1-z))}{(1-z)^{p+1}} + O\left(\frac{\ln^p(1-z)}{(1-z)^p}\right), \tag{IV.59}$$

où P_p est un polynôme de degré p à coefficients rationnels. En utilisant les lemmes de transfert et les expressions des termes généraux données en annexe, on a alors la proposition sur les moments $M_p(n)$. □

Compte tenu des formules de passage entre les différentes sortes de moment (cf. section A.8. p. 121), la proposition précédente s'applique sous la même forme aux moments centrés et aux cumulants de la distribution. Le calcul des premiers moments (cf. annexe C) montre toutefois des propriétés particulières pour les polynômes Pol_p qui implique une forme beaucoup plus restrictive pour les moments centrées et pour les cumulants. On voit notamment que le fait de centrer les moments à pour effet de rendre constants les polynômes en $\ln n$. On constate de plus dans le cas des cumulants que les coefficients prennent une forme très simple en fonction des valeurs $\zeta(i)$. La proposition IV.9 ne donne donc que peu d'information sur le comportement réel des moments et ne permet en particulier pas de montrer l'existence d'une loi limite pour le nombre de comparaisons. Compte-tenu des résultats de l'annexe C sur les premiers moments, on peut toutefois conjecturer le développement suivant pour les cumulants d'ordre quelconque.

Conjecture IV.1 *Les cumulants de la distribution du nombre de comparaisons ($CP_n = a(n-k)+b$) dans l'algorithme Quicksort avec médiane et multipartitionnement sont de la forme :*

$$\frac{\kappa_p(n)}{a^p} = a_0^{(p)}(n+1)^p + a_1^{(p)}(n+1)^{p-1} + \cdots + a_{p-2}^{(p)}(n+1)^2 + b_p n \ln n + O(n + n^{pb_{s,t}}),$$

où les coefficients $a_i^{(p)}$ et b_p sont indépendants de a, b et des constantes d'intégration qui apparaissent lors de la résolution (traitement des petites listes) et où $b_{s,t} = \max(-\Re(\alpha+1))$ étendu aux racines α de $PQ_{s,t}$ autres que -2, 0, 1, $\ldots, t-1$.

Chaque constante $a_i^{(p)}$ s'exprime rationnellement en fonction de $\zeta(p-i)$ et b_p est un rationnel.

On note que les moments centrés ont alors un développement de la même forme, sauf pour les coefficients $a_i^{(p)}$ qui dépendent rationnellement de toutes les valeurs $\zeta(2), \ldots, \zeta(p-i)$ et non seulement de $\zeta(p-i)$ comme pour les cumulants. Ceci explique entre autres pourquoi on utilise ici de préférence les cumulants.

Une partie de cette conjecture concernant le terme dominant en n^p du développement sera prouvée dans la suite (cf. section 4.2.). Le problème pour démontrer directement cette conjecture provient de ce que l'on sait pas traduire simplement, sur les séries génératrices en z, le fait de centrer la distribution. Pour la variance, on aimerait par exemple retrancher la série $M2(z) = \sum_n C_n'(1)^2 z^n$ à $\hat{M}_2(z)$ avant d'intégrer l'équation différentielle caractéristique ; on a malheureusement pas d'expression simple pour $M2(z)$ en fonction de $\hat{M}_1(z)$. D'autre part, les récurrences complexes que l'on obtient après intégration, par exemple sur les polynômes Pol_p, ne permettent pas de mettre en évidence les phénomènes d'annulation sur les termes en $\ln n$ qui apparaissent seulement au moment où l'on combine les termes généraux des séries pour obtenir les moments centrés.

4. Loi limite pour le nombre de comparaisons.

Nous nous intéressons, dans cette partie, uniquement au terme dominant des cumulants et des moments centrés de la distribution du nombre de comparaisons dans Quicksort. On montre l'existence d'une loi limite pour la variable aléatoire centrée et normée associée au nombre de comparaisons : $X_n = (C_n - \bar{C}_n)/\sqrt{Var(C_n)}$, ou de façon équivalente pour la variable aléatoire $Y_n = (C_n - \bar{C}_n)/n$.

Dans le cas de l'algorithme standard (binaire sans médiane), ce résultat a été prouvé récemment avec deux approches différentes par M. Régnier [Rég87] et par U. Rösler [Rös89]. La preuve de M. Régnier se fonde sur un argument probabiliste lié aux martingales et donne rapidement le résultat d'existence de la loi limite. Cette approche est toutefois "non constructive" et ne donne donc pas de caractérisation explicite de la distribution limite ou de ses moments. La preuve de U. Rösler utilise un argument de point fixe sur l'ensemble des distributions et correspond à un passage au continu sur les récurrences caractéristiques liées à la variable aléatoire Y_n. La loi $\mathcal{L}(C_n)$ du coût du tri sur les listes de taille n, est caractérisée sous la forme :

$$\mathcal{L}(C_n) = \mathcal{L}(CP_n + C_{Z_{n-1}} + C^*_{n-Z_n}). \tag{IV.60}$$

où Z_n est une variable aléatoire uniforme sur [1,..,n] qui correspond à la valeur de la clé de partitionnement, CP_n est le coût d'une étape de partitionnement, et C_i, C_i^* sont des variables aléatoires indépendantes et de même loi $\mathcal{L}(C_i) = \mathcal{L}(C_i^*)$.

En passant à la loi de $Y_n = (C_n - \bar{C}_n)/n$, on obtient la récurrence caractéristique :

$$\mathcal{L}(Y_n) = \mathcal{L}(Y_{Z_n-1}\frac{Z_n - 1}{n} + Y^*_{n-Z_n}\frac{n - Z_n}{n} + l_n(Z_n)), \tag{IV.61}$$

avec Z_n v.a. uniforme sur $[1, ..n]$, $l_n(i) = (CP_n + \bar{C}_{i-1} + \bar{C}_{n-i} - \bar{C}_n)/n$, et Y_i , Y_i^* indépendants et de même loi.

On montre alors que la loi de Y_n converge vers une loi limite $\mathcal{L}(Y)$ qui vérifie l'équation de point fixe :

$$\mathcal{L}(Y) = \mathcal{L}(\tau Y + (1 - \tau)Y^* + l(\tau)), \tag{IV.62}$$

avec τ v.a. uniforme sur $[0, 1]$, $l(\tau) = 1 + 2(\tau \ln \tau + (1 - \tau)\ln(1 - \tau)) \sim l_n(n\tau)$, Y,Y^* et τ indépendants, et $\mathcal{L}(Y) = \mathcal{L}(Y^*)$. τ correspond au passage à la limite de Z_n/n, et $l(\tau)$ provient alors de

$$\begin{aligned} l_n(n\tau) &\sim (n + 2n\tau H_{n\tau} + 2(n - n\tau)H_{n-n\tau} - 2nH_n)/n \\ &\sim 1 + 2(\tau \ln n\tau + (1 - \tau)\ln(n(1 - \tau)) - \ln n) \sim l(\tau). \end{aligned} \tag{IV.63}$$

On obtient ainsi une caractérisation de la loi limite qui permet de calculer les moments $M_r = \mathrm{E}\,Y^r$ par récurrence. On a en effet :

$$M_r = \mathrm{E}\,Y^r = \mathrm{E}(\tau Y + (1 - \tau)Y^* + l(\tau))^r = \mathrm{E} \sum_{i+j+k=r} (\tau Y)^i((1 - \tau)Y^*)^j l(\tau)^k. \tag{IV.64}$$

De part l'indépendance des variables aléatoires τ, Y et Y^*, on a donc :

$$M_r = \sum_{i+j+k=r} \binom{r}{i, j, k} M_i M_j \int_0^1 \tau^i(1 - \tau)^j l(\tau)^k d\tau. \tag{IV.65}$$

L'approche présentée ici, qui par certains côtés ressemble à la preuve de U. Rösler, permet de montrer l'existence de la loi limite dans le cas général du tri Quicksort avec médiane et multipartitionnement. Ce résultat implique alors trivialement pour les moments centrés et les cumulants un développement de la forme $\kappa_p(n) = a_p\, n^p + o(n^p)$.

En réinjectant ce résultat dans les équations différentielles caractéristiques des moments, nous allons voir que l'existence de la loi limite implique aussi une forme particulière simple pour les coefficients a_p des cumulants. On obtient ainsi les a_p sous la forme $a_p = (-k_{s,t})^p(A_p-(p-1)!\zeta(p))$ où les A_p sont des coefficients rationnels donnés par une récurrence. Ces résultats montrent donc la partie de la conjecture IV.1 concernant le premier terme en n^p du développement.

4.1. Existence d'une loi limite.

Pour montrer l'existence de la loi limite, nous allons considérer les récurrences qui définissent les moments centrés de la distribution en fonction de la taille n des listes. Pour simplifier les formules on se place dans le cas binaire $s = 2$ et on considère toujours n assez grand (en particulier $n > M$). La généralisation au cas s-aire ne pose toutefois pas de problème.

Les moments centrés $MC_r(n)$ sont définis par la série :

$$MC_n(y) = \sum_r MC_r(n)\frac{y^r}{r!} = C_n(e^y)e^{-\bar{C}_n y} \tag{IV.66}$$

où $\bar{C}_n$ désigne le coût moyen sur les listes de taille n. En reprenant la récurrence de base donnée par l'équation IV.39 p. 67, on a donc dans le cas binaire :

$$MC_n(y) = \sum_{p+q=n-1} P_{p,q}MC_p(y)MC_q(y)e^{(CP(n)+\bar{C}_p+\bar{C}_q-\bar{C}_n)y}\,, \tag{IV.67}$$

ou, pour le moment d'ordre r

$$MC_r(n) = \sum_{i+j+k=r}\ \sum_{p+q=n-1} P_{p,q}\binom{r}{i,j,k}MC_i(p)MC_j(q)(CP(n)+\bar{C}_p+\bar{C}_q-\bar{C}_n)^k\,. \tag{IV.68}$$

En regroupant les moments d'ordre r et en notant que $MC_0(n) = 1$, on obtient ainsi la récurrence :

$$MC_r(n) - 2\sum_{p+q=n-1} P_{p,q}MC_r(p) = V_r(n),$$

$$\text{avec}\qquad V_r(n) = \sum_{i,j\neq r}\ \sum_{p+q=n-1} P_{p,q}\binom{r}{i,j,k}MC_i(p)MC_j(q)(CP(n)+\bar{C}_p+\bar{C}_q-\bar{C}_n)^k\,. \tag{IV.69}$$

L'équation IV.69 correspond, sur les séries en z associées, à une équation différentielle du même type que pour le coût moyen avec médiane de k, c'est-à-dire :

$$PQ_{s,t}(\Theta).MC_r(z) = (-1)^k\binom{\Theta}{k}V_r(z) \qquad \text{avec} \qquad V_r(z) = \sum_n V_r(n)z^n\,. \tag{IV.70}$$

Cette équation ne permet pas de déterminer directement les séries $MC_r(z)$ dans la mesure où l'on peut difficilement obtenir le second membre $V_r(z)$ sous forme close à partir des moments centrés d'ordre inférieur. En procédant par récurrence sur r, on montre toutefois que les séries $V_r(z)$ se comportent comme $(1-z)^{-1-r}$; d'où l'on déduit l'existence de la loi limite en utilisant les propriétés de transfert de l'intégration (corollaire III.3 p. 43).

On introduit :

$$l(n,p,q) = \frac{CP(n)+\bar{C}_p+\bar{C}_q-\bar{C}_n}{n} \qquad \text{pour} \qquad p+q+1=n, \tag{IV.71}$$

où $CP(n)$ désigne le nombre moyen de comparaisons pour le partitionnement et $\bar{C}_n$ le coût moyen du tri.

Lemme IV.2 *Si on pose $CP(n) = an + o(n)$ et $\bar{C}_n = ak_{s,t}(n+1)\ln(n+1) + Cn + o(n)$, on a pour n tendant vers l'infini et p quelconque :*

$$l(n,p,q) = a\,l(\frac{p+1}{n+1}) + o(1) = O(1) \qquad \textit{avec} \qquad l(x) = 1 + k_{s,t}x\ln x + k_{s,t}(1-x)\ln(1-x)\,.$$

Preuve : En développant $l(n,p,q)$, on a :

$$l(n,p,q) = a + \frac{ak_{s,t}}{n}\left((p+1)\ln(p+1) + (q+1)\ln(q+1) - (n+1)\ln(n+1)\right) + C\frac{p+q-n}{n} + o(1).$$

D'où

$$l(n,p,q) = a\left(1 + \frac{k_{s,t}}{n}\left((p+1)\ln(\frac{p+1}{n+1}) + (q+1)\ln(\frac{q+1}{n+1})\right)\right) + o(1),$$

et donc le résultat. □

En pratique, on note que $l(x)$ varie entre $1 - 2\ln 2 \approx -0.4$ et 1.

Lemme IV.3 *Si f est une fonction différentiable sur $[0,1]$, on a pour n tendant vers l'infini :*

$$\sum_{p+q=n-1}^{n} P_{p,q} f(\frac{p}{n})(1+o(1)) = \frac{(2t+1)!}{t!\,t!}\int_0^1 x^t(1-x)^t f(x)dx + o(1).$$

Preuve : Ce résultat est en réalité une application simple de la formule sommatoire d'Euler (cf. section A.5. p. 118) pour la distribution de probabilité $P_{p,q}$. On a en effet, en passant au continu, pour $x \in [0,1]$:

$$P_{xn,n-1-xn} = \frac{\binom{xn}{t}\binom{(1-x)n-1}{t}}{\binom{n}{2t+1}} = \frac{(2t+1)!}{t!^2}\frac{x^t(1-x)^t}{n} + O(\frac{1}{n^2}).$$

On en déduit alors le lemme en appliquant la formule d'Euler à la somme $\sum_p g(p/n)/n$ avec la fonction $g(x) = nP_{xn,n-1-xn}f(x)(1+o(1))$. □

Lemme IV.4 *Si la suite f_n vérifie $f_n \sim n^r$ pour n tendant vers l'infini et $r > -1$, alors la série $f(z) = \sum_n f_n z^n$ converge avec un rayon de convergence égal à 1 et vérifie :*

$$f(z) \sim \frac{\Gamma(r+1)}{(1-z)^{r+1}}.$$

Preuve : Le lemme d'Abel (cf. [Die68]) permet d'assurer la convergence normale de la série $f(x)$ dans le disque de rayon 1. En considérant la série $f(x) - \Gamma(r+1)(1-z)^{-r-1}$, il reste donc à montrer que si une série $g(z) = \sum_n g_n z^n$ avec $g_n = o(n^r)$ converge dans le disque de rayon 1, alors on a $g(z) = o((1-z)^{-r-1}$. En effet,

$$\forall \epsilon \exists n_0 \text{ tel que } |g_n| < \epsilon n^r\,,$$

d'où

$$|g(z)| < \epsilon \sum_{n=0}^{\infty} n^r |z|^n + \sum_{n=0}^{n_0}(|g_n| - \epsilon n^r)|z|^n\,,$$

ce qui permet de conclure que $(1-|z|)^{r+1}g(z)$ tend uniformément vers 0 pour z tendant vers 1. □

Théorème IV.3 *La loi de la variable aléatoire $Y_n = (C_n - \bar{C}_n)/n$ associée au nombre de comparaisons dans le tri Quicksort avec médiane et multipartitionnement converge vers une loi limite pour n tendant vers l'infini.*

Les moments de la loi limite $M_r = E(Y^r)$ sont donnés dans le cas binaire $s = 2$ par la relation :

$$M_r = \frac{(2t+1)!}{t!^2} \sum_{i+j+k=r} \binom{r}{i,j,k} M_i M_j \int_0^1 x^{i+t}(1-x)^{j+t} l(x)^k dx\,,$$

avec

$$l(x) = k_{s,t} x \ln x + k_{s,t}(1-x)\ln(1-x) + 1.$$

Dans le cas du multipartitionnement d'ordre s, la relation devient :

$$M_r = \frac{(s(t+1)-1)!}{(t!)^s} \sum_{i_1+\cdots+i_s+k=r} \binom{r}{i_1,\ldots,i_s,k} M_{i_1}\cdots M_{i_s} A_{i_1,\ldots,i_s,k}$$

avec

$$A_{i_1,\ldots,i_s,k} = \int_{x_1+\cdots+x_{s-1}=1-x_s<1} x_1^{i_1+t}\cdots x_s^{i_s+t}\left(1 + k_{s,t}\sum_{j=1}^{s} x_j \ln x_j\right)^k dx_1\cdots dx_{s-1}.$$

Preuve : On raisonne par récurrence sur l'ordre du moment et on suppose que les moments centrés d'ordre $i < r$ sont de la forme $MC_i(n) = mc_i n^i + o(n^i)$ avec des coefficients mc_i constants.

La première étape consiste à montrer que la quantité $V_r(n)$ de l'équation IV.70 est alors de la forme $V_r(n) = v_r n^r + o(n^r)$ avec v_r constant. On a :

$$\frac{V_r(n)}{n^r} = \sum_{p+q=n-1} P_{p,q} \sum_{i,j\neq r} \binom{r}{i,j,k} \frac{MC_i(p)}{p^i}\frac{MC_j(q)}{q^j}\left(\frac{p}{n}\right)^i\left(\frac{q}{n}\right)^j l(n,p,q)^k.$$

Compte tenu du lemme IV.2 et de l'hypothèse de récurrence, l'expression précédente se met sous la forme :

$$\frac{V_r(n)}{n^r} = \sum_{p+q=n-1} P_{p,q} \sum_{i,j\neq r} \binom{r}{i,j,k} mc_i\, mc_j\, f_{i,j,k}\left(\frac{p}{n}\right)(1+o(1))\,,$$

où $f_{i,j,k}(x) = x^i(1-x)^j l(x)^k$ est une fonction continue et C^∞ sur $[0,1]$. On en déduit donc avec le lemme IV.3 que $V_r(n)/n^r$ tend vers une limite finie v_r quand n tend vers l'infini.

$$v_r = \frac{(2t+1)!}{t!^2} \sum_{i,j\neq r} \binom{r}{i,j,k} mc_i\, mc_j \int_0^1 x^{i+t}(1-x)^{j+t} l(x)^k dx.$$

On considère maintenant la série $V_r(z) = \sum V_r(n) z^n$ qui d'après le dernier lemme vérifie $V_r(z) \sim v_r r!/(1-z)^{r+1}$. Le corollaire III.3 p. 43 et les lemmes de transfert permettent alors d'intégrer l'équation différentielle IV.70 pour obtenir directement si $r > 1$:

$$MC_r(z) \sim c(r) v_r r!/(1-z)^{r+1} \qquad \text{et} \qquad MC_r(n) = c(r) v_r n^r + o(n^r)\ ;$$

$$\text{où} \qquad c(r) = \frac{\binom{r+k}{k}}{\binom{r+k}{k} - s\binom{r+t}{t}} \qquad (k = s(t+1)-1)$$

est un coefficient borné donné par les règles d'intégration du chapitre III.

On a montré par récurrence que tous les moments centrés sont de la forme $MC_i(n) = mc_i n^i + o(n^i)$ et donc que les moments de la variable aléatoire centrée normée Y_n convergent vers les constantes mc_i quand n tend vers l'infini. On a de plus la relation de récurrence pour le calcul des mc_i qui généralise la caractérisation donnée par U. Rösler.

Pour prouver complètement le théorème, il reste à montrer que les mc_i caractérisent bien de façon unique la loi de distribution limite. Pour cela, on utilise la condition suivante donnée par Feller [Fel71, p.514] qui assure l'unicité de la distribution si $\limsup_{n\to\infty} M_n^{1/n}/n = \lambda < \infty$. De la récurrence et du fait que $l(x) < 1$ et que $\int_0^1 x^i(1-x)^j dx < 1/(i+j+1)$, on déduit facilement la majoration

$$|mc_r| \leq c(r)\frac{(2t+1)!}{t!\,t!}\sum_{i,j\neq r}\binom{r}{i,j,k}|mc_i||mc_j|\frac{1}{i+j+1}.$$

Et comme $c(r)$ est borné, on obtient

$$\frac{|mc_r|}{r!} \leq C \sum_{\substack{i+j+k=r\\ i,j\neq r}} \frac{1}{(r-k+1)k!}\frac{|mc_i|}{i!}\frac{|mc_j|}{j!},$$

où C est une constante indépendante de r. Il est maintenant facile de montrer que $|mc_r| < A^r r!$ pour une valeur de A suffisamment grande telle que $\exp(1/A) < 1/C$. Par récurrence, on a en effet :

$$\frac{|mc_r|}{r\;!} \leq C \sum_{\substack{i+j+k=r\\ i,j\neq r}} \frac{A^{i+j}}{(r-k+1)k!} \leq C\sum_{k=0}^{r}\frac{A^{r-k}}{k!} \leq CA^r exp(1/A).$$

La formule de Stirling donne alors une borne finie pour $|mc_r|^{1/r}/r$ et permet de conclure à l'unicité de la loi limite ainsi caractérisée. □

Cette preuve s'étend sans problèmes particuliers au cas du multipartitionnement d'ordre s. La caractérisation des moments centrés est alors donnée par l'équation IV.70 en prenant pour $V_r(n)$ l'expression :

$$V_r(n) = \sum_{\substack{i_1+\cdots+i_s+k=r\\ i_j\neq r}} \sum_{p_1+\cdots+p-s=n+1-s} P_{p_1,\ldots,p_s}\binom{r}{i_1,\ldots,i_s,k}\prod_j MC_{i_j}(p)(CP(n)+\sum_k \bar{C}_{p_k}-\bar{C}_n)^k. \tag{IV.72}$$

La preuve s'effectue alors en remplaçant les différentes convolutions binaires de la preuve précédente par des convolutions à s termes; les lemmes IV.2 et IV.3 s'étendent eux par récurrence sur s. La caractérisation que l'on obtient sur les moments est toutefois d'un intérêt pratique limité dans la mesure où elle nécessite l'évaluation d'un grand nombre d'intégrales multiples complexes.

Dans le cas binaire, la relation de récurrence du théorème précédent permet de calculer numériquement les moments, pour t fixé, de façon relativement rapide. Elle sert aussi, en utilisant un système de calcul formel, pour déterminer l'expression exacte des moments limites en fonction du paramètre t et des valeurs entières de la fonction ζ. Les intégrales à calculer sont alors données par :

$$\int_0^1 x^i(1-x)^j \ln^k x \ln^l(1-x)dx = \left(\frac{d}{di}\right)^k \left(\frac{d}{dj}\right)^l \frac{\Gamma(i+1)\Gamma(j+1)}{\Gamma(i+j+2)}. \tag{IV.73}$$

où les dérivées de la fonctions Γ font apparaître les constantes $\zeta(p)$ (cf. [Köl85] pour le calcul d'intégrales de ce type). Ce calcul présente toutefois le défaut d'utiliser des expressions qui explosent très vite en taille. Nous allons établir dans la suite une récurrence légèrement plus simple qui permet de calculer directement les cumulants de la loi limite ; de plus, on obtient pour ces cumulants des expressions en fonction de t et des valeurs de la fonction ζ beaucoup plus simples que pour les moments centrés.

4.2. Forme particulière des cumulants.

Nous allons montrer que l'existence de la loi limite implique une forme simple sur le terme dominant des cumulants ou sur les cumulants de la loi limite. En réinjectant cette forme dans les équations caractéristiques sur les séries génératrices des moments, on obtient alors une récurrence permettant de calculer directement ces cumulants.

Conformément à la proposition IV.9 p. 74, on peut supposer que les séries génératrices des moments sont de la forme :

$$\hat{M}_p(z) = (-1)^p\, p!\, k_{s,t}^p \left(\frac{d}{dz}\right)^p \frac{B_p(\ln(1-z))}{1-z} + o(\frac{1}{(1-z)^{p+1}}), \tag{IV.74}$$

où $B_p = \sum_j B_{p,j} y^j/j!$ est un polynôme de degré p à coefficient rationnel. Le terme général de la série vérifie alors :

$$M_p(n) = (-1)^p n^p p!\, k_{s,t}^p \sum_{j=0}^{p} \frac{B_{p,j}}{j!} [z^n] \frac{\ln^j(1-z)}{1-z} + o(n^p). \tag{IV.75}$$

En utilisant les polynômes exponentiels (cf. annexe A), on peut exprimer le terme général des séries comme un polynôme en $H_n \sim \ln n$.

$$[z^n]\frac{\ln^j \frac{1}{1-z}}{1-z} = Y_j(H_n, -H_n^{(2)}, \ldots, (-1)^{j-1}(j-1)!H_n^{(j)}) = \sum_{l=0}^{j} \binom{j}{l} Y_{l,n} H_n^{j-l} \tag{IV.76}$$

avec

$$Y_{l,n} = Y_l(0, -H_n^{(2)}, \ldots, (-1)^{l-1}(l-1)!H_n^{(l)}). \tag{IV.77}$$

Compte tenu de la convergence des nombres harmoniques et de la forme des polynômes logarithmiques les valeurs $Y_{l,n}$ convergent :

$$Y_{l,n} = Y_l(0, -\zeta(2), \ldots, (-1)^{l-1}(l-1)!\zeta(l)) + O(\frac{1}{n}). \tag{IV.78}$$

Après quelques manipulations simples sur les sommes, on obtient ainsi les moments centrés pour les listes de taille n sous la forme :

$$MC_p(n) = \sum_{i=0}^{p} \binom{p}{i} M_i(n)(-M_1(n))^{p-i} = k_{s,t}^p\, n^p\, p! \sum_{r=0}^{p} (H_n)^r coef(p,r) + o(n^p).$$

$$\text{avec} \qquad coef(p,r) = (-1)^p \sum_{a+c=r} \sum_{d=0}^{p-r} \frac{(-1)^{d+a} Y_{d,n}}{a!\, c!\, d!} B_{a+p-r,a+d}. \tag{IV.79}$$

L'existence d'une loi limite implique que $MC_p(n) = O(n^p)$ et donc que $coef(p,r) = 0$ pour $r \geq 1$ et pour tout p. En prenant $r = p$, on obtient les relations $\sum_a (-1)^a B_{a,a}/(a!(p-a)!) = 0$, d'où on déduit $B_{p,p} = B_{0,0}$ pour tout p. Pour $r = p-1$, on déduit de même $B_{p,p-1} = B_{1,0}$. En continuant le processus, on montre ainsi que $B_{p,l} = B_{p-1,l-1} = B_{p-l,0}$ quelques soient l et p. Du fait de l'existence de la loi limite, les polynômes B_p, que l'on a utilisés pour décrire les séries des moments, vérifient donc la relation $B_p'(y) = B_{p-1}(y)$.

Compte tenu de la forme des polynômes $B_p(y)$, on peut maintenant poser

$$B_p(y) = \sum_{j=0}^{p} \frac{B_{p-j}}{(p-j)!j!}\, y^j, \tag{IV.80}$$

et introduire les coefficients A_p définis à partir des polynômes de Bell par

$$A_p = L_p(B_1, B_2, \ldots, B_p) \qquad \text{ou} \qquad B_p = Y_p(A_1, A_2, \ldots, A_p). \tag{IV.81}$$

En utilisant les propriétés des polynômes de Bell, l'équation IV.79 donne alors

$$p!\, coef(p,0) = \sum_{d=0}^{p} (-1)^{p-d} \binom{p}{d} Y_{d,n} B_{p-d} = Y_p(0, A_2 - H_n^{(2)}, \ldots, (-1)^p (A_p - (p-1)! H_n^{(p)})). \tag{IV.82}$$

En passant aux cumulants par $\kappa_p(n) = L_p(0, MC_2(n), \ldots, MC_p(n))$, on obtient alors facilement la proposition suivante :

Proposition IV.10 *Les cumulants du nombre de comparaisons dans le tri Quicksort avec médiane et multipartitionnement sont de la forme (pour $p > 1$)*

$$\kappa_p(n) = (-1)^p\, n^p\, k_{s,t}^p\, (A_p - (p-1)!\, \zeta(p)) + o(n^p)$$

où les A_p sont des constantes rationnelles qui ne dépendent que des paramètres s et t.

Nous allons maintenant voir comment on calcule, dans le cas binaire $s = 2$, les constantes B_i ou les A_i par récurrence. Pour cela, on réinjecte les polynômes $B_p(y)$ dans l'équation IV.54 p. 73 qui détermine les séries $\hat{M}_p(z)$. En spécifiant $y = 0$, les relations obtenues sur les polynômes $B_p(y)$ donnent alors la récurrence caractéristique sur les coefficients B_i.

Par dérivation de la formule IV.74, on obtient :

$$\hat{M}_p^{(j)}(z) = p!\, k_{s,t}^p \frac{(-1)^j (p+j)!}{(1-z)^{p+j+1}} \binom{De-1}{p+j} B_p(y)|_{y=\ln(1-z)} + o\left(\frac{1}{(1-z)^{p+j+1}}\right), \tag{IV.83}$$

où $De = d/dy$ est l'opérateur de dérivation usuel que l'on développe en utilisant les nombres de Stirling de première espèce:

$$i! \binom{De-1}{i} . B_p(y) = \sum_{k=0}^{i} s(i+1, k+1)\, De^k . B_p(y) = \sum_{k=0}^{\inf(i,p)} s(i+1,k+1) B_{p-k}(y). \tag{IV.84}$$

D'autre part, si l'on introduit la quantité $F(r)$ définie par :

$$F(r) = \frac{(-1)^t\, k_{s,t}^{-r}}{t!}\, X^{r+t+1} \sum_{a+b=r} \frac{\hat{M}_b^{(t+a)}(X)}{a!\, b!} = \sum_{a+b=r} \frac{(-1)^a\, k_{s,t}^b (r+t)!}{a!\, t!} \binom{De-1}{r+t} .B_b(y), \tag{IV.85}$$

la récurrence caractéristique IV.54 devient alors dans le cas binaire :

$$-\frac{(2t+1+p)!}{(2t+1)!} \binom{De-1}{2t+1+p} B_p(y) = \sum_{r=0}^{p} F(r)F(p-r). \tag{IV.86}$$

En développant et en se plaçant en $y = 0$, on obtient donc la proposition suivante qui donne une récurrence caractéristique pour calculer les coefficients B_i ou A_i.

Proposition IV.11 *Les constantes rationnelles A_p de la proposition précédente sont obtenues dans le cas binaire par :*

$$A_p = L_p(B_1, B_2, \ldots, B_p)$$

où les B_p sont donnés par la relation de récurrence :

$$\forall p \qquad \sum_{r=0}^{p} \frac{s(p+2t+2, r+1)}{(2t+1)!} \frac{B_{p-r}}{(p-r)!} + \sum_{r=0}^{p} F(r)F(p-r) = 0$$

avec

$$F(r) = \sum_{i=0}^{r} \frac{s(r+t+1, i+1)}{t!} G(r-i)$$

et

$$G(k) = \sum_{a=0}^{k} \frac{(-1)^a B_{k-a}}{a!\,(k-a)!\, k_{s,t}^a}.$$

Remarque : La proposition précédente se généralise au cas du multipartitionnement d'ordre s. La relation de récurrence devient alors :

$$\forall p \qquad \sum_{r=0}^{p} \frac{s(p+k+1, r+1)}{k!} \frac{B_{p-r}}{(p-r)!} = (-1)^k \sum_{r_1+\cdots+r_s=p} F(r_1)F(r_2)\cdots F(r_s) \tag{IV.87}$$

où $k = s(t+1)+1$ et F, G, A_i, B_i sont définis comme dans le cas binaire. On voit toutefois que comme pour la caractérisation sur les moments centrés (théorème IV.3), la complexité du calcul effectif des moments augmente très vite avec s.

Cette récurrence s'automatise facilement et permet un calcul rapide des cumulants pour des valeurs de t fixées ; elle est aussi utilisée pour obtenir de façon plus générale les cumulants en fonction du paramètre t (cf. section E.2. p. 154). La section D.6. p. 145 donne les valeurs numériques ainsi obtenues pour les 40 premiers cumulants et moments centrés dans le cas standard et dans les cas médiane de 3, 5 et 7.

Ce calcul des cumulants dans le cas binaire fait apparaître des phénomènes assez particuliers, qu'il est encore difficile d'interpréter à partir des caractérisations que l'on a données dans ce chapitre. La première constatation est que l'on a un phénomène d'élimination important entre les termes A_p et $(p-1)!\zeta(p)$ qui sont du même ordre de grandeur. Pour fixer les idées, si on prend $p = 40$ dans le cas standard, A_{40} et 39! valent de l'ordre de 10^{47} alors que la différence $A_p - (p-1)!\zeta(p)$ est de l'ordre de 10^{12}. Il semble donc que les cumulants de la loi limite $\kappa_p = \lim \kappa_p(n)/n^p$ obéissent en valeur absolue à une majoration beaucoup plus fine que celle en $(k_{s,t}\, p/e)^p$ qui apparaît à première vue.

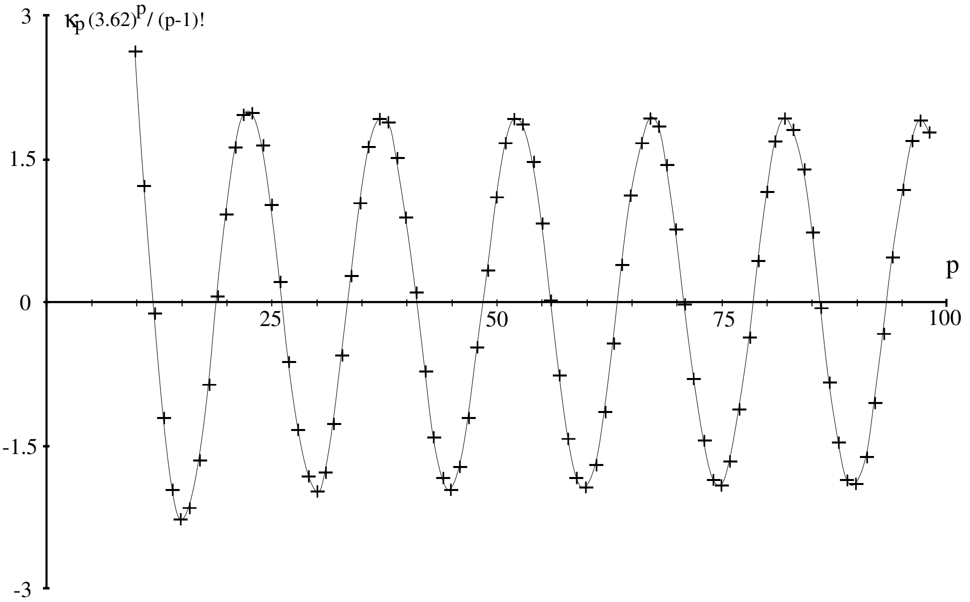


Figure IV.1 : Variations périodiques des cumulants en fonction de l'ordre (cas standard).

L'autre phénomène étonnant concerne le signe des cumulants. On constate en effet une grande régularité du signe qui, à partir du cinquième cumulant, est successivement positif puis négatif sur des intervalles de longueur 7 ou 8. Si on fait une analyse plus fine des valeurs numériques obtenues pour les cumulants limites, il semble que ces cumulants aient un comportement périodique que l'on pourrait représenter approximativement sous la forme :

$$\kappa_p = \lim_n \frac{\kappa_p(n)}{n^p} \approx C \sin(\frac{2\pi(p-p_0)}{T})\frac{(p-1)!}{\lambda(t)^p} \qquad \text{pour} \quad p > p_0, \tag{IV.88}$$

avec une période $T \approx 15$, un déphasage $p_0 \approx 18.8$ et une amplitude $C \approx 2$ qui ne varient pratiquement pas suivant les valeurs de t (pour $t = 0, 1, 2$). Les constantes $\lambda(t)$ croissent par contre rapidement avec t et valent expérimentalement $\lambda(0) \approx 3.62$ dans le cas standard, $\lambda(1) \approx 6.42$ pour médiane de 3 et $\lambda(2) \approx 8.9$ pour médiane de 5.

La figure IV.1 montre ce comportement périodique pour les 100 premiers cumulants limites κ_p dans le cas standard $t = 0$. La figure représente en fonction de p les valeurs de $\kappa_p(\lambda(t))^p/(p-1)!$

où les κ_p sont calculés à partir de la récurrence caractéristique. On constate alors la très grande régularité des cumulants en fonction de p ; la formule précédente semble donc donner une très bonne approximation des cumulants même pour des valeurs de p faibles. Les figures obtenues pour médiane de 3 et de 5 sont alors quasiment identiques avec simplement de très légères différences pour les premiers moments.

Une approche possible pour prouver ce comportement très sympathique des cumulants est d'essayer de faire une analyse asymptotique selon p en utilisant les séries génératrices en u associées aux moments ou aux cumulants de la loi limite :

$$M(u) = \sum_p M_p \frac{u^p}{p!} \qquad K(u) = \sum_p \kappa_p \frac{u^p}{p!} = \ln M(u). \tag{IV.89}$$

La caractérisation donnée par le théorème IV.3 se traduit alors simplement sur les séries $M(u)$ et $K(u)$. Dans le cas standard, on obtient ainsi :

$$M(u) = \int_0^1 M(xu)M((1-x)u)e^{u\,l(x)}dx \tag{IV.90}$$

ou

$$\exp(K(u)) = \int_0^1 \exp(K(xu) + K((1-x)u) + u\,l(x))dx. \tag{IV.91}$$

En introduisant une transformée de Laplace sur $M(u)$, on obtient alors une belle équation implicite qui malheureusement n'a pas encore permis de faire l'analyse asymptotique que l'on recherche :

$$-\mathcal{L}'(r+1) = \mathcal{L}(r)^2 \qquad \text{avec} \qquad \mathcal{L}(r) = \int_0^\infty M(u)e^{2u\ln u}e^{-r\,u}du. \tag{IV.92}$$

Les différentes approches que l'on a essayée pour obtenir une caractérisation globale des moments de la distribution limite afin d'en faire une étude asymptotique, redonnent toujours le même genre d'équation. On voit en effet apparaître une équation différentielle non linéaire qui se rapproche d'une équation de Ricatti avec malheureusement un couplage ou un décalage (pour $L(r)$) qui gêne énormément l'analyse. L'approximation IV.88 reste donc d'un point de vue théorique un très beau résultat à démontrer.

4.3. Simulations.

Pour conclure, nous donnons avec les figures IV.2 et IV.3, la forme obtenue expérimentalement pour les distributions du coût en nombre de comparaisons dans le cas de l'algorithme Quicksort standard et de la variante médiane de trois. La figure IV.2 représente, à la même échelle, les deux fonctions de densité que l'on obtient en comptant explicitement le nombre de comparaisons C_n dans la procédure Quicksort pour un échantillon de 100000 et 50000 permutations aléatoires de 200 clés. On effectue en réalité un tri partiel, c'est-à-dire que l'on ne trie pas les petites listes de taille inférieure ou égale à $M = 10$. La simulation consiste alors à compter pour chaque valeur possible C du coût, le nombre de permutations qui ont nécessitées C comparaisons pour être triées.

La figure IV.3 reprend les deux mêmes fonctions de densité en les centrant par rapport à leur valeur moyenne 1316 et 1067. On a effectué de plus un léger lissage (on représente la probabilité $prob(X \leq \text{coût} < X + 5)/5$) et pris une échelle double sur les probabilité pour le cas médiane de trois.

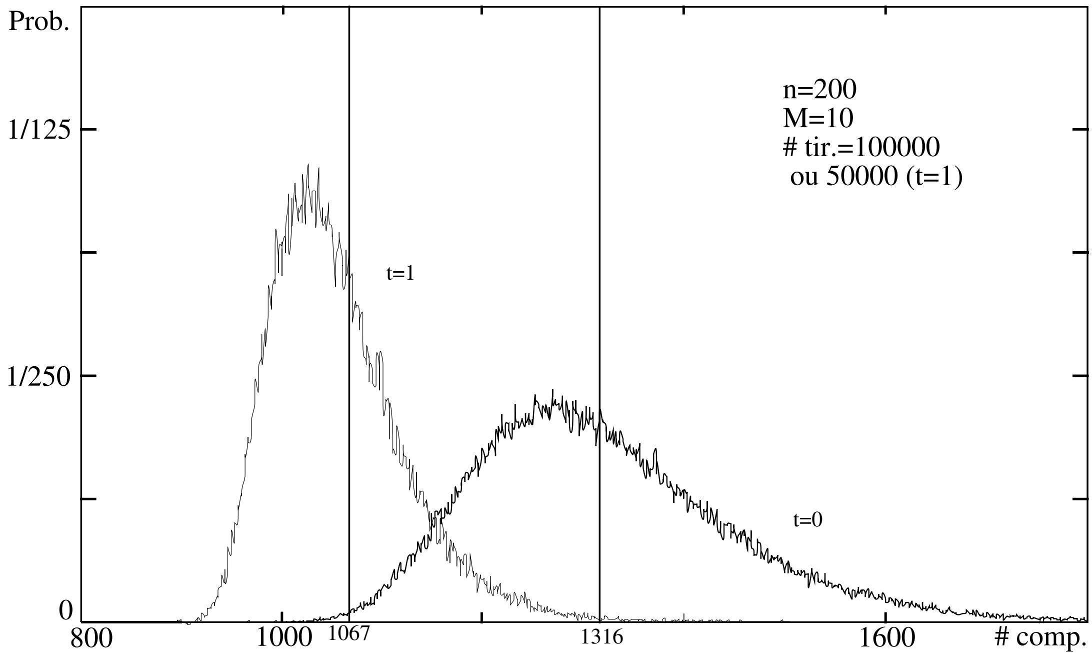


Figure IV.2 : Densités de probabilité dans le cas standard et avec médiane de 3.

Ces figures mettent en évidence le gain important apporté par la variante médiane de trois ; on voit en effet apparaître le facteur 2 gagné sur l'écart-type de la distribution. On constate d'autre part une nette dissymétrie des distributions qui est sans doute la raison principale pour laquelle la loi limite n'est pas gaussienne. La valeur maximale de la fonction de densité est en particulier sensiblement inférieure à la valeur moyenne et ceci indépendamment de la taille des listes ou du paramètre t. De façon approximative, la distribution semble avoir une forme triangulaire pour la partie inférieure à la valeur maximale de la densité, et une forme exponentielle plus étendue pour les grandes valeurs du coût. Intuitivement, le coût d'un tri est donc plus souvent en dessous de sa valeur moyenne mais le gain, au maximum $0.6n\ln n$, reste limité par rapport aux cas défavorables qui, bien que rares, peuvent coûter jusqu'a $n^2/2$ comparaisons. Si on cherche à faire converger vers la loi limite la distribution centrée normée, la branche supérieure de la distribution va ainsi s'étendre vers l'infini beaucoup plus vite (en n) que la branche inférieure (en $-\ln n$).

D'un point de vue pratique, on constate enfin que la distribution reste relativement bien concentrée autour de sa valeur moyenne. De façon plus précise, si on utilise l'inégalité de Tchebicheff avec le moment d'ordre 4, on obtient une borne sur la probabilité de trier une liste avec une erreur relative donnée par rapport à la valeur moyenne. On a ainsi dans le cas standard :

$$Prob(\frac{|C_n - \bar{C}_n|}{\bar{C}_n} > E) < \frac{0.046122}{(E\ln(n/M))^4}. \tag{IV.93}$$

Ainsi, pour $n = 1000$ et $M = 10$, le coût d'un tri ne s'éloigne pas de plus de 21% de sa valeur moyenne dans plus de 95% des cas ; avec une erreur relative de 30%, on englobe alors au moins

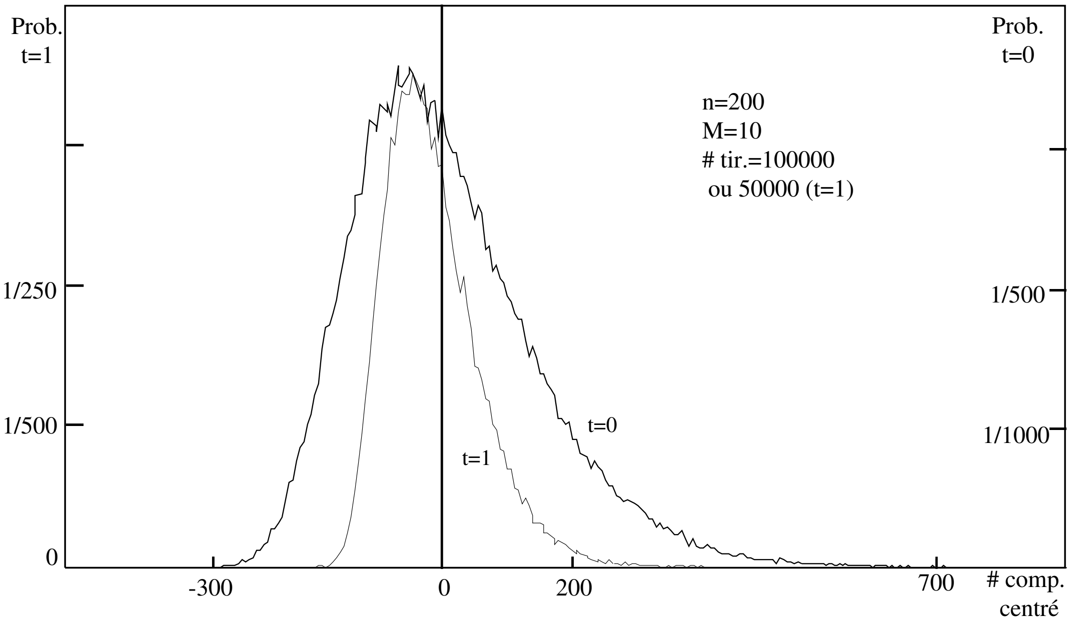


Figure IV.3 : Densités de probabilité centrées : cas standard et médiane de 3.

99% des cas.

Chapitre V

Quicksort avec clés répétées.

Dans les chapitres précédents, nous avons toujours supposé que les clés à trier étaient toutes distinctes. Pour de nombreux problèmes pratiques, il est toutefois utile de considérer le cas où des clés identiques peuvent apparaître à plusieurs reprises dans la liste à trier. Cette possibilité correspond à la situation òu la tri est effectué sur différents champs d'une clé ou sur des sous-clés. Ce chapitre présente une analyse en moyenne du tri Quicksort en considérant que les données sont distribuées selon la statistique des mots aléatoires sur un alphabet de taille fixée m, et non plus selon la statistique des permutations.

Dans le cas des clés répétées, on peut considérer plusieurs algorithmes de partitionnement qui diffèrent sur la manière de traiter, à chaque étape, les clés de la liste égales à la clé de partition. Nous étudions ici trois variantes qui consistent respectivement à mettre les clés égales dans une des deux sous-listes, à les répartir uniformément dans chaque sous-liste ou à les trier directement.

Dans les trois cas, on obtient des formules explicites pour les coûts moyens soit sur l'ensemble des mots aléatoires, soit pour chaque profil possible de mot, c'est-à-dire pour des nombres fixés de répétitions de chaque clé. On utilise pour cela des outils comparables à ceux développés pour les clés distinctes dans les chapitres précédents : séries génératrices, produit partitionnel, opérateur d'enracinement. La structure de l'algorithme se traduit alors par des relations de récurrence sur des séries génératrices à m variables dans le cas de l'analyse par profil, ou à une et deux variables pour l'analyse globale sur les mots. On peut en particulier déterminer directement les moyennes sur l'ensemble des mots sans passer par les formules sommatoires obtenues pour chaque profil. La partie analytique de résolution des équations sur les séries génératrices est sensiblement plus complexe que dans le cas de clés toutes distinctes. La difficulté provient essentiellement du fait que l'on n'a plus de forme close pour les séries génératrices, ou pour les coûts moyens qui s'expriment sous forme de sommations. De plus la présence du paramètre supplémentaire, m, augmente rapidement la complexité des expressions mises en jeu et des résultats.

A partir des expressions des coûts moyens, on peut toutefois déduire des développements asymptotiques quand n et m tendent vers l'infini, soit indépendamment, soit simultanement et dans un rapport $\lambda = n/m$ fixé. On a ainsi différentes expressions simplifiées, pour les valeurs moyennes des coûts, qui correspondent aux différents comportements possibles des paramètres m et n. Pour de grandes valeurs de m, on verra en particulier comment s'effectue la convergence vers le cas limite des clés distinctes.

Une partie des résultats, et notamment sur l'étude par profil, se rapproche des analyses faites par R. Sedgewick [Sed77b] qui donne des bornes pour le coût du tri avec clés répétées, et par W. Burge [Bur76] qui étudie des longueurs de branches dans des arbres binaires de recherche. Outre la description formelle de la structure des algorithmes, la particularité de l'analyse réside principalement dans le calcul direct des moyennes sur l'ensemble des mots, sans passer par l'expression par profil, et dans l'analyse asymptotique en fonction du taux de répétition de clés $\lambda = n/m$ fixé.

1. Statistique des mots.

Nous allons définir dans cette partie la notion de mot aléatoire pour décrire l'ensemble des listes que l'on veut trier. Nous introduisons les différentes séries génératrices que l'on utilise pour la description et l'analyse d'algorithme du type Quicksort dans le cas des clés répetées. Nous présentons enfin brièvement le modèle des mots "sans trou" qui constitue une alternative aux mots aléatoires classiques. Nous verrons notamment les possibilités de passage entre ces deux modèles.

1.1. Modèle probabiliste.

On suppose ici que les listes à trier peuvent contenir des clés identiques répétées plusieurs fois, ce qui correspond au cas où la dimension de l'ensemble des valeurs possibles pour les clés n'est pas infiniment grande devant la taille des listes considérées. Il apparaît alors naturel de décrire la distribution des données de l'algorithme en utilisant la statistique des mots aléatoires définis sur un alphabet de taille finie m. Comme dans le cas des clés distinctes, la valeur des clés n'intervient pas directement, on suppose donc pour simplifier que les clés prennent leurs valeurs dans l'ensemble $[1..m]$.

Ce modèle probabiliste sur les données consiste alors à générer les listes en effectuant n tirages indépendants et uniformes parmi les m valeurs possibles des clés. La configuration limite où m tend vers l'infini doit donc naturellement correspondre au cas des clés distinctes analysé dans les chapitres précédents.

Hypothèse H.IV *Les listes de taille n sont considérées comme des mots équiprobables sur un alphabet à m lettres : chaque mot a une probabilité $1/m^n$ d'apparaître.*

A chaque mot ainsi formé, on associe le vecteur $\tilde{p} = (p_1, p_2, \ldots, p_m)$, appelé profil, tel que p_i est le nombre de clés égales à i dans le mot. Les p_i sont donc positifs ou nuls, et leur somme notée $|\tilde{p}|$ est égale à la taille n du mot. Le nombre de mots qui correspondent à un même profil $\tilde{p}$ est alors donné par le coefficient multinomial

$$M(\tilde{p}) = \binom{|\tilde{p}|}{p_1, p_2, \ldots, p_m} = \frac{(p_1 + p_2 + \cdots + p_m)!}{p_1! p_2! \ldots p_m!}. \tag{V.1}$$

En sommant $M(\tilde{p})$ sur les profils de taille n, on retrouve alors facilement le nombre total de mots sur m lettres, c'est-à-dire m^n.

Pour simplifier l'écriture des équations, on introduit les notations suivantes :

- le tilde dénote un vecteur de dimension m : $\tilde{X} = (X_1, X_2, \ldots, X_m)$
- la taille du vecteur est $|\tilde{X}| = \sum_i X_i$
- la factorielle est le produit des factorielles : $\tilde{X}! = \prod_i X_i!$
- de même, pour la puissance : $\tilde{X}^{\tilde{p}} = \prod_i X_i^{p_i}$.

1.2. Séries génératrices.

Pour travailler avec la statistique des mots en fonction de leur profil, on introduit des séries génératrices à m variables; chaque variable correspond à une des m lettres de l'alphabet. A une fonction de coût C définie sur les mots, on associe ainsi les séries génératrices par profil définies de la manière suivante :

Définition V.1 *Les séries génératrices exponentielles et ordinaires de coût moyen associées à une fonction de coût C sont respectivement :*

$$C(X_1, \ldots, X_m) = C(\tilde{X}) = \sum \frac{C(p_1, \ldots, p_m)}{M(\tilde{p})} \frac{\tilde{X}^{\tilde{p}}}{\tilde{p}!} = \sum C(p_1, \ldots, p_m) \frac{\tilde{X}^{\tilde{p}}}{|\tilde{p}|!},$$

et

$$\hat{C}(X_1, \ldots, X_m) = \hat{C}(\tilde{X}) = \sum \frac{C(p_1, \ldots, p_m)}{M(\tilde{p})} \tilde{X}^{\tilde{p}},$$

où $C(p_1, \ldots, p_m)$ est le coût cumulé sur l'ensemble des mots aléatoires de profil $\tilde{p} = (p_1, \ldots, p_m)$.

La quantité $C(p_1, \ldots, p_m)/M(\tilde{p})$ représente donc exactement la valeur moyenne de la fonction de coût C sur les mots de profil $\tilde{p}$.

Les séries de dénombrement des mots ($C = 1$) s'obtiennent alors facilement en prenant $C(p_1, \ldots, p_m) = M(\tilde{p})$. On a ainsi

$$M(X_1, \ldots, X_m) = e^{X_1 + \cdots + X_m} \qquad \text{et} \qquad \hat{M}(X_1, \ldots, X_m) = \frac{1}{(1 - X_1) \cdots (1 - X_m)}. \tag{V.2}$$

On note que dans les deux cas, les séries génératrices ordinaires ou exponentielles à m variables sont égales au produit des séries $\hat{M}(X_i)$ ou $M(X_i)$ sur chacune des variables.

Comme dans le cas des fonctions à une seule variable, le passage entre les séries génératrices ordinaires et les séries exponentielles s'effectue par une transformation de Laplace. De façon usuelle, la transformée de Laplace $\mathcal{L}$ d'une fonction à plusieurs variables est définie par

$$\mathcal{L}(C)[s_1, \ldots, s_m] = \int_0^\infty \cdots \int_0^\infty e^{-(s_1 X_1 + \cdots + s_m X_m)} C(X_1, \ldots, X_m) dX_1 \cdots dX_m. \tag{V.3}$$

Pour les séries génératrices exponentielles et ordinaires telles qu'on les a définies, on vérifie alors aisément que l'on a

$$\hat{C}(X_1, \ldots, X_m) = \frac{1}{X_1 X_2 \cdots X_m} \mathcal{L}(C)[\frac{1}{X_1}, \ldots, \frac{1}{X_m}]. \tag{V.4}$$

Proposition V.1 *En notant $\hat{\mathcal{L}}$ l'opérateur du type Laplace qui transforme une série génératrice exponentielle en la série génératrice ordinaire associée, $\hat{C}(\tilde{X}) = \hat{\mathcal{L}}(C(\tilde{X}))$, on a :*

- *$\hat{\mathcal{L}}$ est linéaire et admet une transformée inverse.*
-
$$\hat{\mathcal{L}}\left(X_j C(\ldots, X_j, \ldots)\right) = X_j \frac{d}{dX_j}\left(X_j \hat{\mathcal{L}}\left(C(\ldots, X_j, \ldots)\right)\right).$$
-
$$\hat{\mathcal{L}}\left(X_j \frac{d}{dX_j} C(\ldots, X_j, \ldots)\right) = X_j \frac{d}{dX_j} \hat{\mathcal{L}}\left(C(\ldots, X_j, \ldots)\right).$$
-
$$\hat{\mathcal{L}}\left(C(X_1, \ldots, X_k) D(Y_1, \ldots, Y_l)\right) = \hat{\mathcal{L}}\left(C(X_1, \ldots, X_k)\right) \hat{\mathcal{L}}\left(D(Y_1, \ldots, Y_l)\right),$$
si $\{X_1, \ldots, X_k\} \cap \{Y_1, \ldots, Y_l\} = \emptyset$ (Fubini).

Ces propriétés faciles à démontrer permettent dans la suite d'écrire les équations caractéristiques liées à l'algorithme Quicksort en utilisant aussi bien les séries génératrices exponentielles que les séries génératrices ordinaires. Pour la résolution explicite des systèmes, les séries exponentielles $C(X_1, \ldots, X_m)$ semblent toutefois plus adaptées en général.

Dans l'analyse, nous allons nous intéresser aux coûts moyens du tri sur l'ensemble des mots de taille n à m lettres. Au lieu de passer par l'analyse des coûts par profil telle qu'elle est faite par W. Burge [Bur76] et R. Sedgewick [Sed77b], on va notamment déterminer directement le comportement moyen sur l'ensemble des mots aléatoires. Pour cela, on introduit les séries génératrices à une ou deux variables X et Y qui correspondent respectivement aux paramètres n et m.

Définition V.2 *Pour une fonction de coût C, si $C_{m,n}$ est le coût total sur l'ensemble des m^n mots de taille n, on définit les séries génératrices*

$$C_m(X) = \sum_n C_{m,n} \frac{X^n}{n!},$$

et

$$C(X, Y) = \sum_m C_m(X) Y^m = \sum_{n,m} C_{m,n} \frac{X^n Y^m}{n!}.$$

Le coût total $C_{m,n}$ est obtenu à partir des coûts $C(p_1, \ldots, p_m)$ par :

$$C_{m,n} = \sum_{|\tilde{p}|=n} C(p_1, \ldots, p_m). \tag{V.5}$$

On a donc la relation suivante entre les séries génératrices sur les mots et les séries par profil :

$$C_m(X) = C(X_1, \ldots, X_m)\big|_{X_1 = \cdots = X_m = X}. \tag{V.6}$$

En particulier pour les séries de dénombrement des mots, on trouve bien :

$$M_m(X) = e^{mX} = \sum m^n \frac{X^n}{n!} \qquad \text{et} \qquad M(X,Y) = \frac{1}{1 - Y\exp(X)}. \tag{V.7}$$

Nous verrons dans la section suivante, que la structure de l'algorithme Quicksort ou des arbres de recherche avec clés répétées, peut dans la plupart des cas être caractérisée par une récurrence sur les séries $C_m(X)$, ou par une équation différentielle simple sur les séries de coût à deux variables $C(X,Y)$.

Comme dans le cas des clés distinctes, on introduit la série génératrice CP associée au coût d'une étape de partitionnement. On suppose ici que le coût moyen d'une étape de partitionnement est indépendant du profil ou de la taille m de l'alphabet considéré. On se limite ainsi au cas où le coût moyen d'un partitionnement est linéaire de la forme : $\bar{C}P(n,m) = a(n-1) + b$ et $CP = 0$ pour $n = 0$ ou $n = 1$.

Les séries génératrices associées valent alors :

$$CP(X_1, \ldots, X_m) = a\hat{X}(e^{\hat{X}} - 1) + (b-a)(e^{\hat{X}} - 1 - \hat{X}) \quad \text{avec} \quad \hat{X} = \sum_{i=1}^{m} X_i, \tag{V.8}$$

et

$$CP(X,Y) = a\frac{XYe^X}{(1-Ye^X)^2} + (b-a)(\frac{1}{1-Ye^X} - \frac{1}{1-Y}) - b\frac{XY}{(1-Y)^2}. \tag{V.9}$$

1.3. Mots sans trou.

D'un point de vue pratique, le paramètre m qui intervient dans la statistique des mots aléatoires n'apparaît pas toujours très naturellement. Pour une liste donnée, on connaît en effet facilement son profil ou le nombre de clés distinctes dans le mot mais pas obligatoirement le nombre de “clés absentes” liées au modèle probabiliste. Il peut donc apparaître intéressant de considérer la statistique des mots sans trou sur m lettres, c'est-à-dire des mots contenant exactement m clés distinctes.

Pour une fonction de coût donnée et une taille de liste n fixée, on note $C(m)$ la valeur moyenne sur l'ensemble des mots à m lettres et $C_{st}(m)$ la moyenne correspondante sur les mots sans trou (supposés équiprobables) ayant exactement m lettres. Les mots aléatoires sur m lettres étant obtenus comme ré union des mots sur exactement k lettres distinctes pour $k \leq m$, on a la relation suivante entre les deux valeurs moyennes :

$$C(m)m^n = \sum \binom{m}{k} C_{st}(k)R(k,n), \tag{V.10}$$

où $R(k,n)$ désigne le nombre de mots sans trou sur k lettres et de longueur n.

La récurrence précédente s'inverse de façon classique sous la forme

$$C_{st}(m)R(m,n) = \sum (-1)^k \binom{m}{k} C(m-k)(m-k)^n. \tag{V.11}$$

D'où l'on déduit, en prenant $C(k) = 1$, la valeur de $R(m,n)$:

$$R(m,n) = \sum_k (-1)^k \binom{m}{k} (m-k)^n = m! S(n,m). \tag{V.12}$$

On retrouve ainsi une propriété combinatoire usuelle des nombres de Stirling de seconde espèce (cf. [Com70] et section A.4. p. 118).

A titre d'exemple, on peut calculer le nombre moyen de clés différentes $\lambda(m)$ dans un mot aléatoire sur m lettres. On a en effet pour le nombre de lettres différentes dans les mots sans trou : $\lambda_{st}(m) = m$. On en déduit donc en utilisant les propriétés des nombres de Stirling :

$$\lambda(m) = \sum_k \binom{m}{k} \frac{k! S(n,k)\, k}{m^n} = \sum_k \binom{m}{k} \frac{k!(S(n+1,k) - S(n,k-1))}{m^n} = \frac{m^{n+1} - m(m-1)^n}{m^n}. \tag{V.13}$$

D'où pour le nombre moyen de clés distinctes dans un mot aléatoire

$$\lambda(m) = m(1 - (1 - \frac{1}{m})^n). \tag{V.14}$$

Ce genre d'étude est à rapprocher des distributions de probabilité obtenues par exemple dans le problème des anniversaires (cf. [FGT87]).

On a ainsi une correspondance simple entre les valeurs moyennes des paramètres pour les statistiques des mots avec ou sans trou. Au niveau des séries génératrices, cette correspondance s'exprime de manière synthétique à condition de considérer des séries génératrices exponentielles en Y. En introduisant

$$\check{C}(X,Y) = \sum_{n,m} C_{m,n} \frac{X^n Y^m}{n!\, m!}$$

où $C_{m,n}$ désigne le coût total, on a en effet :

$$\check{C}(X,Y) = \check{C}_{st}(X,Y) e^Y . \tag{V.15}$$

Le passage aux séries génératrices $C(X,Y)$ introduites précédemment est alors possible par une transformation de Laplace en Y. Dans la suite de ce travail, on se contente de l'analyse en moyenne sur les mots aléatoires dans la mesure où elle correspond à une description combinatoire et à des résultats plus simples. Compte tenu des équations ou des formules obtenues pour $C(X,Y)$, il est en particulier difficile d'espérer pouvoir travailler sur les séries $\check{C}(X,Y)$ pour avoir directement la correspondance avec les mots sans trou.

2. Analyse en moyenne pour trois algorithmes de base.

2.1. Algorithmes.

De manière générale, un algorithme de partitionnement de type Quicksort pour des clés répétées crée, à chaque étape, deux sous-listes contenant respectivement des clés inférieures ou égales et des clés supérieures ou égales à la clé de partition. Il existe ainsi différentes stratégies pour traiter les clés égales à l'élément de partition : on peut par exemple les mettre toutes dans la même sous-liste, ou bien les répartir dans chaque sous-liste, ou bien encore en trier le maximum. Ces choix possibles correspondent à des comportements sensiblement différents de l'algorithme, et doivent donc être analysés séparément. Nous nous limitons ici à l'analyse de trois algorithmes particuliers qui donnent notamment des bornes minimales et maximales sur les coûts moyens que l'on peut espérer dans le cas des clés répétées.

Quel que soit l'algorithme choisi, le partitionnement d'un mot de profil $\tilde{p} = (p_1, \ldots, p_m)$ autour d'une clé i, produit deux mots de profil $\tilde{q} = (p_1, \ldots, p_{i-1}, q)$ et $\tilde{r} = (r, p_{i+1}, \ldots, p_m)$ avec $q + r < p_i$. Un algorithme est donc caractérisé par la distribution des valeurs de q et r. Les trois cas que l'on analyse sont représentés par la figure V.1.

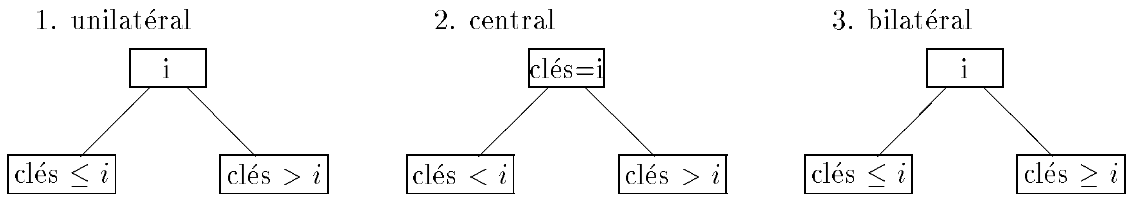


Figure V.1 : Stratégies de partitionnement.

Le premier algorithme consiste à mettre dans la même sous-liste toutes les clés égales à la clé de partition. Il correspond donc au partitionnement le plus défavorable avec $q = p_i - 1$ et $r = 0$ (ou de façon totalement équivalente $q = 0$ et $r = p_i - 1$). Le grand avantage de cet algorithme est qu'il s'implante de la même façon que le tri pour des clés distinctes et nécessite un nombre minimum de comparaisons puisque l'on ne teste pas l'égalité des clés. Les clés égales à i sont en effet confondues avec les clés strictement inférieures (ou strictement supérieures).

Le cas le plus favorable est illustré par le deuxième algorithme. Il consiste à prendre $q = r = 0$ à chaque étape, c'est-à-dire à trier systématiquement les clés égales à la clé de partitionnement i. Dans la pratique, ceci nécessite toutefois de tester à la fois l'ordre et l'égalité des clés et correspond donc, en général, à un coût de partitionnement à chaque étape sensiblement plus élevé.

Le dernier algorithme que l'on analyse correspond d'une certaine manière à un cas moyen. On suppose que les clés égales à la clé de partitionnement sont toutes réparties uniformément dans les deux sous-listes ; c'est-à-dire que q et r sont uniformément distribués sur l'intervalle $[0..p_i - 1]$ avec $q + r = p_i - 1$. D'un point de vue pratique, cette hypothèse donne une assez bonne approximation du cas où l'on effectue le partitionnement sur des listes représentées par un tableau et que l'on laisse systématiquement en place les clés égales à i. Comme on va le voir dans l'analyse, cet algorithme présente de plus la particularité d'être totalement insensible aux répétitions des clés. Les coûts moyens ne dépendent donc que de la taille des listes n et sont donnés par les mêmes formules que pour des clés toutes distinctes.

2.2. Descriptions combinatoires.

L'analyse de ces trois algorithmes présente de nombreuses similitudes avec l'étude faite pour des clés distinctes dans le chapitre III. La description combinatoire de la structure des algorithmes utilise notamment les mêmes opérateurs de produit partitionnel et d'enracinement. Les classes de structures combinatoires que l'on introduit sont toutefois légèrement différentes dans la mesure où elles sont paramétrées par la taille m de l'alphabet sous-jacent.

On définit ainsi la classe $\mathbf{M_m}$ constituée des mots sur un alphabet de m lettres. La fonction taille sur cette classe est donnée par le nombre total n de lettres de chaque mot. La structure combinatoire des algorithmes de tri avec clés répétées s'exprime alors non plus par une simple équation formelle, mais par une relation de récurrence donnant $\mathbf{M_m}$ à partir des classes $\mathbf{M_i}$ pour $i \leq m$. Les opérateurs formels sont ceux déjà utilisés dans le cas des clés distinctes, à l'exception de l'opérateur d'enracinement qui est marqué par la valeur i de la clé enracinée ; on note alors l'enracinement $\circ_{[i]}$. Ce marquage des valeurs de clés est essentiel pour obtenir des relations sur les séries génératrices à m variables dans lesquelles on différencie chacune des lettres.

Définition V.3 *Pour un alphabet A donné, on appelle enracinement marqué d'une classe de structure combinatoire $\mathbf{C}$, la classe*

$$\circ_{[i]}(\mathbf{C}) = A \times \mathbf{C}$$

avec la fonction taille définie par

$$|(a,x)|_{\circ_{[i]}(\mathbf{C})} = |x|_{\mathbf{C}} + 1.$$

Plaçons nous dans le cas du premier algorithme où les clés égales à la clé i de partitionnement sont toutes mises dans la même sous-liste. Un mot de $\mathbf{M_m}$ se décompose alors en un mot de $\mathbf{M_i}$ et un mot de $\mathbf{M_{m-i}}$; les profils dans chaque sous-mot sont complètement déterminés par le profil initial.

D'autre part, si on considère les mécanismes de partitionnement et de shuffle, ils ne sont conditionnés que par les tailles des listes et sont insensibles à la répétition des clés, car les sous-listes n'ont pas de clés communes. En reprenant le cas des clés distinctes, on voit donc que la construction $o_{[i]}(\mathbf{M_i} \otimes \mathbf{M_{m-i}})$ décrit exactement l'ensemble des mots sur m lettres qui se partitionnent autour de la clé i en deux mots données de $\mathbf{M_i}$ et $\mathbf{M_{m-i}}$. L'ensemble de tout les mots sur m lettres est alors décrit en prenant la réunion des ensembles précédents pour i allant de 1 à m.

On en déduit ainsi la décomposition formelle des classes de dénombrement $\mathbf{M_m}$:

$$\mathbf{M_m} = \sum_{i=1}^{m} o_{[i]}(\mathbf{M_i} \otimes \mathbf{M_{m-i}}) + \mathbf{M_0}. \qquad \text{(V.16)}$$

De la même manière que pour les permutations, on obtient alors facilement une équation caractéristique sur les classes de coût associées $\mathbf{C}(\mathbf{M_m})$.

Proposition V.2 *Pour le tri Quicksort avec partitionnement unilatéral des clés égales, la classe de coût* $\mathbf{C}(\mathbf{M_m})$ *associée est décrite formellement par*

$$\mathbf{C}(\mathbf{M_m}) = \sum_{i=1}^{m} o_{[i]}(\mathbf{C}(\mathbf{M_i}) \otimes \mathbf{M_{m-i}} + \mathbf{M_i} \otimes \mathbf{C}(\mathbf{M_{m-i}})) + \mathbf{CP}(\mathbf{M}).$$

Le passage aux séries génératrices par profil ne pose maintenant plus de problème et est donné par le lemme et la définition suivante.

Définition V.4 *On note* Δ*, l'opérateur différentiel défini par*

$$\Delta F(X_1, \ldots, X_m) = \sum_{i=1}^{m} \frac{\partial}{\partial X_i} F(X_1, \ldots, X_m).$$

Lemme V.1 *Si les* $\mathbf{G}_i$ *sont* m *classes de structures combinatoires, on a*

$$\mathbf{F} = \sum_{i=1}^{m} o_{[i]} \mathbf{G_i} \Longrightarrow \Delta F(X_1, \ldots, X_m) = \sum_{i=1}^{m} X_i G_i(X_1, \ldots, X_m).$$

On en déduit donc

Proposition V.3 *Le tri Quicksort avec partitionnement unilatéral des clés égales est caractérisé par les équations suivantes sur les séries génératrices exponentielles par profil :*

- *Série de dénombrement des mots :*

$$\Delta M(X_1, \ldots, X_m) = \sum_{j=1}^{m} X_j M(X_1, \ldots, X_j) M(X_{j+1}, \ldots, X_m),$$

- *Série des coûts :*

$$\begin{aligned} \Delta C(X_1, \ldots, X_m) &= \Delta CP(X_1, \ldots, X_m) + \sum_{j=1}^{m} X_j C(X_1, \ldots, X_j) M(X_{j+1}, \ldots, X_m) \\ &\quad + \sum_{j=1}^{m} X_j M(X_1, \ldots, X_j) C(X_{j+1}, \ldots, X_m). \end{aligned} \tag{V.17}$$

On note, à titre indicatif, la traduction des équations précédentes sous forme de récurrences sur les valeurs moyennes par profil (cf. [Sed77b]) :

$$M(p_1, p_2, \ldots, p_m) = \sum_{j=1}^{m} \binom{n-1}{p_1 + \cdots + p_j - 1} M(p_1, \ldots, p_j - 1) M(p_{j+1}, \ldots, p_m), \tag{V.18}$$

et

$$\begin{aligned} C(p_1, \ldots, p_m) &= CP(p_1, \ldots, p_m) + \sum_{j=1}^{m} \binom{n-1}{p_1 + \cdots + p_j - 1} C(p_1, \ldots, p_j - 1) M(p_{j+1}, \ldots, p_m) \\ &\quad + \sum_{j=1}^{m} \binom{n-1}{p_1 + \cdots + p_j - 1} M(p_1, \ldots, p_j - 1) C(p_{j+1}, \ldots, p_m). \end{aligned} \tag{V.19}$$

En utilisant les définitions et les propriétés des séries décrites au début de ce chapitre, on en déduit maintenant les équations caractéristiques sur les séries génératrices ordinaires à m variables et sur les séries à une ou deux variables.

Corollaire V.1 *Dans le cas du tri avec partitionnement unilatéral, on a :*

- $$\Delta\hat{M}(X_1,\ldots,X_m) = \sum_{j=1}^{m} X_j \frac{d}{dX_j}(X_j\hat{M}(X_1,\ldots,X_j))\hat{M}(X_{j+1},\ldots,X_m),$$
$$M'_m(X) = \sum_{j=1}^{m} M_j(X)M_{m-j}(X) \qquad et \qquad \frac{d}{dX}M(X,Y) = M(X,Y)^2 - M(X,Y).$$

- $$\begin{aligned}\Delta\hat{C}(X_1,\ldots,X_m) &= \Delta\hat{CP}(X_1,\ldots,X_m) + \sum_{j=1}^{m}\frac{X_j}{1-X_j}\hat{M}(X_1,\ldots,X_j)\hat{C}(X_{j+1},\ldots,X_m)\\ &\quad + \sum_{j=1}^{m} X_j\frac{d}{dX_j}(X_j\hat{C}(X_1,\ldots,X_j))\hat{M}(X_{j+1},\ldots,X_m),\end{aligned}$$
$$C'_m(X) = C'_{0\,m}(X) + 2\sum_{j=0}^{m} C_j(X)e^{(m-j)X} - C_m(X),$$
$$\frac{d}{dX}C(X,Y) = \frac{d}{dX}CP(X,Y) + 2M(X,Y)C(X,Y) - C(X,Y). \tag{V.20}$$

La description combinatoire est très similaire pour le deuxième algorithme où les clés égales à la clé de partitionnement sont mises en place et n'apparaissent plus dans aucunes des deux sous-listes sur lesquelles on appelle récursivement le tri. Le partitionnement d'un mot de $\mathbf{M_m}$ autour de la clé i donne cette fois-ci un mot de $\mathbf{M_{i-1}}$ et un mot de $\mathbf{M_{m-i}}$. En réalité, il faut aussi considérer un troisième mot de $\mathbf{M_1}$ qui contient les clés égales à i. L'appel récursif de la procédure de tri ne s'effectue bien sûr que sur les deux premiers mots. Comme précédemment, l'enracinement de la clé i sur le produit partitionnel des trois sous-mots décrit exactement l'ensemble des antécédents possibles par le partitionnement, et ceci parce que les différents sous-mots n'ont pas de valeurs de clés communes. On obtient ainsi facilement la description formelle suivante.

Proposition V.4 *Pour le tri Quicksort avec partitionnement central des clés égales, la décomposition structurelle des mots suit le schéma :*

$$\mathbf{M_m} = \sum_{i=1}^{m} o_{[i]}(\mathbf{M_{i-1}} \otimes \mathbf{M_1} \otimes \mathbf{M_{m-i}}) + \mathbf{M_0}.$$

Pour une classe de coût $\mathbf{C}(\mathbf{M_m})$ associée au tri sur des mots aléatoires à m lettres, on a alors

$$\mathbf{C}(\mathbf{M_m}) = \sum_{i=1}^{m} o_{[i]}(\mathbf{C}(\mathbf{M_{i-1}}) \otimes \mathbf{M_1} \otimes \mathbf{M_{m-i}} \ + \ \mathbf{M_{i-1}} \otimes \mathbf{M_1} \otimes \mathbf{C}(\mathbf{M_{m-i}})) + \mathbf{CP}(\mathbf{M}).$$

On déduit alors simplement les équations caractéristiques sur les séries génératrices qui sont assez proches de celles obtenues dans la proposition V.3 et le corollaire V.1 pour le premier algorithme.

Proposition V.5 *Le tri Quicksort avec partitionnement central des clés égales, est caractérisé par les équations suivantes sur les séries génératrices exponentielles par profil :*

- *Séries des mots :*

$$\Delta M(X_1,\ldots,X_m) = \sum_{j=1}^{m} X_j e^{X_j} M(X_1,\ldots,X_{j-1}) M(X_{j+1},\ldots,X_m),$$

$$\Delta \hat{M}(X_1,\ldots,X_m) = \sum_{j=1}^{m} \frac{X_j}{(1-X_j)^2} \hat{M}(X_1,\ldots,X_{j-1}) \hat{M}(X_{j+1},\ldots,X_m),$$

$$M'_m(X) = \sum_{j=1}^{m} e^X M_{j-1}(X) M_{m-j}(X) \qquad et \qquad \frac{d}{dX} M(X,Y) = Y e^X M(X,Y)^2.$$

- *Séries des coûts :*

$$\begin{aligned} \Delta C(X_1,\ldots,X_m) &= \Delta CP(X_1,\ldots,X_m) + \sum_{j=1}^{m} X_j e^{X_j} C(\ldots,X_{j-1}) M(X_{j+1},\ldots) \\ &\quad + \sum_{j=1}^{m} X_j e^{X_j} M(\ldots,X_{j-1}) C(X_{j+1},\ldots), \end{aligned} \tag{V.21}$$

$$\begin{aligned} \Delta \hat{C}(X_1,\ldots,X_m) &= \Delta \hat{CP}(X_1,\ldots,X_m) + \sum_{j=1}^{m} \frac{X_j}{(1-X_j)^2} \hat{C}(\ldots,X_{j-1}) \hat{M}(X_{j+1},\ldots) \\ &\quad + \sum_{j=1}^{m} \frac{X_j}{(1-X_j)^2} \hat{M}(\ldots,X_{j-1}) \hat{C}(X_{j+1},\ldots), \end{aligned}$$

$$C'_m(X) = C'_{0\,m}(X) + 2 \sum_{j=1}^{m} C_{j-1}(X) e^{(m-j+1)X},$$

$$\frac{d}{dX}(C(X,Y) - CP(X,Y)) = 2Y e^X M(X,Y) C(X,Y) = 2M(X,Y)C(X,Y) - 2C(X,Y). \tag{V.22}$$

Comme dans le premier cas, on peut noter les récurrences induites sur les valeurs moyennes par profil qui sont alors de la forme :

$$M(p_1,\ldots,p_m) = \sum_{j=1}^{m} \binom{n-1}{p_1+\cdots+p_{j-1}, p_j - 1, p_{j+1}+\cdots+p_m} M(p_1,\ldots,p_{j-1}) M(p_{j+1},\ldots,p_m). \tag{V.23}$$

Le dernier algorithme est traité de manière légèrement différente dans la section suivante.

2.3. Cas de la répartition uniforme des clés égales.

Pour le dernier algorithme qui répartit uniformément les clés égales sur chacune des sous-listes, on n'a pas de description combinatoire simple utilisant les opérateurs d'enracinement et de produit partitionnel. Le problème vient du produit partitionnel qui ne décrit plus l'opération de shuffle voulue et ceci à cause de la présence de clés identiques dans chacun des deux sous-mots à mélanger. Plutôt que d'introduire un opérateur spécifique et peu naturel pour traiter ce cas particulier, on a préféré appuyer ici l'analyse à partir de la récurrence sur les valeurs moyennes par profil. Les équations caractéristiques sur les séries génératrices sont alors déduites de ces récurrences.

Nous montrerons que le comportement de l'algorithme, et par là-même les coûts moyens, sont totalement insensibles à la répétition des clés. Ces coûts ne dépendent ainsi que de la taille totale de la liste n, et pas de la taille de l'alphabet m. Les résultats correspondent donc exactement aux formules obtenues dans le chapitre III pour des clés distinctes.

Pour obtenir la récurrence de base sur les valeurs moyennes par profil, nous allons considérer l'algorithme sous la forme suivante.

La première étape consiste à prendre la première clé de la liste comme clé de partitionnement. Pour un mot de profil $\tilde{p} = (p_1, \ldots, p_m)$ et de taille $n = |\tilde{p}|$, on obtient de cette manière la clé j avec une probabilité p_j/n.

On tire ensuite une valeur i uniformément dans l'intervalle $[1..p_j]$. On note que les couples (j, i) sont alors équiprobables avec une probabilité égale à $1/n$. On décide que les $i-1$ premières clés égales à j dans la liste sont traitées comme les clés strictement inférieures et donc placées dans la sous-liste gauche. Les $p_j - i$ clés égales à j qui restent sont mises avec les clés supérieures dans la sous-liste droite.

La dernière étape de l'algorithme consiste alors à partitionner la liste en tenant compte du choix effectué pour les clés égales et en préservant l'ordre des clés sur chacune des sous-listes.

Pour deux sous-listes de profils $\tilde{p} = (p_1, \ldots, p_j)$ et $\tilde{p}' = (p'_j, \ldots, p'_m)$, l'opération de mélange induite par l'algorithme consiste en un shuffle classique avec une contrainte supplémentaire sur les clés j. Il faut en effet que, dans le mot produit par shuffle, les p_j clés j provenant du premier mot soient avant les p'_j clés j du deuxième mot.

De manière plus formelle, l'ensemble des mots antécédents de deux mots donnés, pour un partitionnement autour d'une clé j, est décrit non pas en prenant tout les éléments du produit partitionnel, mais en considérant des classes de congruence sur le produit partitionnel. La congruence consiste à confondre les mélanges ou shuffles qui placent les clés j aux mêmes positions.

Pour les deux mots de profils $\tilde{p}$ et $\tilde{p}'$, on ne compte donc qu'une fois les $\binom{p_j+p'_j}{p_j}$ éléments du produit partitionnel qui correspondent à des positions identiques des $p_j + p'_j$ clés j.

La récurrence caractéristique sur les valeurs moyennes par profil, est maintenant obtenue en considérant les différentes valeurs possibles des couples (j, i) qui déterminent la valeur de la clé de partitionnement et la répartition des clés égales dans chaque sous-liste.

Proposition V.6 *La structure de l'algorithme Quicksort avec répartition uniforme des clés égales sur chaque sous-liste est décrite par la récurrence*

$$M(p_1,\ldots,p_m) = \sum_{j=1}^{m}\sum_{i=1}^{p_j} \frac{1}{p_j} \frac{\binom{n-1}{p_1+\cdots+p_{j-1}+i-1}}{\binom{p_j-1}{i-1}} M(p_1,\ldots,p_{j-1},i-1)M(p_j-i,p_{j+1},\ldots,p_m).$$

Le rapport des deux coefficients multinomiaux correspond à l'opération de mélange avec contrainte sur les clés égales. Le terme $1/p_j$ donne lui la probabilité de chaque répartition $i-1, p_j-i$ des clés j.

La proposition précédente permet d'obtenir facilement des relations de récurrence sur les séries génératrices ordinaires par profil pour le dénombrement, ou pour une fonction de coût C définie sur les mots.

Proposition V.7 *Le tri Quicksort avec partitionnement uniforme des clés égales est décrit par les équations suivantes sur les séries génératrices ordinaires par profil :*

$$\Delta.\hat{M}(X_1,\ldots,X_m) = \sum_{j=1}^{m} X_j \hat{M}(X_1,\ldots,X_j)\hat{M}(X_j,\ldots,X_m),$$

et

$$\begin{aligned}\Delta.\hat{C}(X_1,\ldots,X_m) &= \Delta.\hat{CP}(X_1,\ldots,X_m) + \sum_{j=1}^{m} X_j \hat{C}(X_1,\ldots,X_j)\hat{M}(X_j,\ldots,X_m)\\ &\quad + \sum_{j=1}^{m} X_j \hat{M}(X_1,\ldots,X_j)\hat{C}(X_j,\ldots,X_m).\end{aligned}$$

On trouve ainsi des équations assez proches de celles obtenues pour les deux autres algorithmes p. 98 et 99. Ici, on n'a toutefois pas de traduction simple sur les séries génératrices exponentielles par profil ou les séries à une ou deux variables. Le problème vient de la présence de la clé j, ou de la variable X_j, dans les deux sous-listes. Ceci empêche en effet d'utiliser la formule de Fubini pour obtenir, par transformation de Laplace, une forme simple sur les séries génératrices exponentielles $C(X_1,\ldots,X_m)$.

Nous allons montrer maintenant que l'algorithme avec répartition uniforme des clés égales dans les sous-listes est totalement insensible à la répétition des clés. A chaque étape de partitionnement, on a en effet la même distribution uniforme des tailles des sous-listes produites, et ceci indépendamment du profil considéré. Les coûts moyens ne dépendent alors que de la taille totale des listes n et sont égaux à ceux obtenus dans le chapitre III pour des clés distinctes. La fin de cette section va consister à caractériser sur les séries génératrices cette propriété d'indépendance vis-à-vis de la taille de l'alphabet m. Ceci permet notamment d'obtenir les coûts moyens attendus à partir de l'équation précédente sur les séries génératrices ordinaires par profil.

Pour une fonction de coût C donné, le critère d'indépendance vis-à-vis du profil se traduit par la relation

$$C(p_1,\ldots,p_m) = C(p_1+\cdots+p_m)\binom{p_1+\cdots+p_m}{p_1,\ldots,p_m}. \tag{V.24}$$

Le coût moyen sur tout les mots de profil $\tilde{p} = (p_1,\ldots,p_m)$ est ainsi égal au coût pour le mot sur une seule clé de taille totale $|\tilde{p}| = p_1+\cdots+p_m$.

On a alors, par définition de la série génératrice ordinaire associée,

$$\hat{C}(X_1, \ldots, X_m) = \sum C(p_1 + \cdots + p_m)\tilde{X}^{\tilde{p}}.$$

En reprenant un résultat de Knuth (cf. [Knu68] ex. 1.2.3.33), on en déduit facilement la forme suivante pour les séries génératrices par profil.

$$\hat{C}(X_1, \ldots, X_m) = \sum_{j=1}^{m} \hat{C}(X_j) \prod_{i \neq j} \frac{X_j}{X_j - X_i}. \tag{V.25}$$

On exprime ainsi la série à m variables à partir de la série $\hat{C}(X)$ sur une seule variable. Cette série $\hat{C}(X)$ correspond de plus exactement à la série génératrice ordinaire des coûts moyens en fonction du seul paramètre qui intervient, c'est-à-dire la taille totale n des listes.

Réciproquement, il est facile de vérifier qu'une fonction génératrice par profil qui s'exprime sous la forme de l'équation V.25 correspond à une fonction de coût qui est insensible aux répétitions de clés; c'est-à-dire dont les valeurs moyennes par profil ne dépendent que de la taille totale des mots, n. Le critère d'indépendance des coûts vis-à-vis des répétitions de clés correspond donc exactement à la relation V.25 sur les séries génératrices ordinaires par profil.

Revenons à l'équation caractéristique donnée par la proposition V.7. Pour les fonctions CP et M, on sait déjà qu'elles vérifient le critère d'indépendance. Nous allons maintenant supposer qu'il en est aussi de même pour la fonction de coût C.

Lemme V.2 *Si les séries $\hat{C}$, $\hat{CP}$ et $\hat{M}$ vérifient la relation donnée par la formule V.25, l'équation caractéristique de la proposition V.7 sur la série de coût se réécrit sous la forme*

$$\sum_{j=1}^{m} \left(\hat{C}'(X_j) - \hat{CP}'(X_j) - 2\hat{C}(X_j)\hat{M}(X_j)\right) \prod_{i \neq j} \frac{X_j}{X_j - X_i} = 0.$$

Cette formule traduit la conservation de la propriété d'indépendance par dérivation ou par shuffle. Elle s'obtient facilement par des manipulations simples sur les sommes et les produits, et en utilisant le lemme suivant.

Lemme V.3 *Pour un ensemble de m variables X_i, et pour deux entiers a et b avec $1 \leq a < b \leq m$, on a l'identité algébrique suivante :*

$$\sum_{j=a}^{b} X_j \prod_{\substack{k=1 \\ k \neq a}}^{j} \frac{X_a}{X_a - X_k} \prod_{\substack{k=j \\ k \neq b}}^{m} \frac{X_b}{X_b - X_k} = 0\,.$$

Preuve : On constate d'abord que les termes où apparaissent les variables X_i pour $i \notin [a \ldots b]$ se factorisent dans la somme. Pour démontrer le lemme, il suffit donc de vérifier l'identité dans le cas où $a = 1$ et $b = m$. En multipliant par le facteur non identiquement nul $\prod_2^m (X_1 - X_k)/X_1$, le lemme se met alors sous la forme

$$R(X_1, \ldots, X_m) = \sum_{j=1}^{m} X_j \prod_{k=j}^{m-1} \frac{X_m(X_1 - X_{k+1})}{X_1(X_m - X_k)} = 0 \qquad \text{si} \quad m > 1.$$

Cette identité se vérifie facilement dans le cas $m = 2$. Nous allons maintenant montrer que si $m > 2$, l'expression $R(X_1, \ldots, X_m)$ est totalement indépendante de X_2. On regroupe pour cela les termes où apparaissent la variable X_2, c'est-à-dire pour $j = 1$ et $j = 2$ dans la sommation, et constater que cette variable s'élimine.

En posant

$$G(j) = \prod_{k=j}^{m-1} \frac{X_m(X_1 - X_{k+1})}{X_1(X_m - X_k)},$$

on a en effet, pour $m > 2$,

$$\begin{aligned} R(X_1, \ldots, X_m) &= \sum_{j=3}^{m} X_j G(j) + X_2 G(2) + X_1 G(2) \frac{X_m(X_1 - X_2)}{X_1(X_m - X_1)} \\ &= \sum_{j=3}^{m} X_j G(j) + G(2) \frac{X_1(X_m - X_2)}{X_m - X_1} \\ &= \sum_{j=3}^{m} X_j G(j) + G(3) \frac{X_m(X_1 - X_3)}{X_m - X_1}. \end{aligned}$$

$G(j)$ étant indépendant de X_2 pour $j \geq 3$, $R(X_1, \ldots, X_m)$ l'est aussi. Le lemme est alors obtenu par récurrence car on a

$$R(X_1, X_2, \ldots, X_{m-1}, X_m) = R(X_1, 0, \ldots, 0, X_m) = R(X_1, X_m) = 0.$$

□

On voit donc maintenant que si $\hat{C}(X)$ est solution de $\hat{C}'(X) = \hat{C}\hat{P}'(X) + 2\hat{C}(X)\hat{M}(X)$, la série $\hat{C}(X_1, \ldots, X_m)$ définie par la formule V.25 est solution de l'équation caractéristique du tri donnée dans la proposition V.7. De plus, l'équation donnant $\hat{C}(X)$ correspond exactement à l'équation caractéristique donnant les coûts moyens dans le cas de l'algorithme avec des clés toutes distinctes. On a donc pour l'algorithme de tri, avec répartition uniforme des clés égales, des coûts moyens par profil qui ne dépendent que de la taille totale des listes et qui sont égaux aux coûts moyens sur les permutations de même taille.

2.4. Résolution par profil.

Dans cette partie, on va montrer comment on résout, pour les deux premiers algorithmes, les équations caractéristiques sur les séries génératrices à m variables. Ceci permet d'obtenir des formules pour les coûts moyens en fonction de chaque profil possible pour les données de l'algorithme. Ces expressions sont semblables à celles obtenues par W. Burge [Bur76] et R. Sedgewick [Sed77b] dans leurs analyses; l'objectif est ici simplement de montrer la résolution plus synthétique obtenue en utilisant les séries génératrices.

On introduit les opérateurs E_1 et E_m définis par :

$$\begin{aligned} E_1.C(X_1, X_2, \ldots, X_m) &= C(X_1, X_2, \ldots, X_m) - M(X_1)C(X_2, \ldots, X_m), \\ E_m.C(X_1, X_2, \ldots, X_m) &= C(X_1, X_2, \ldots, X_m) - M(X_m)C(X_1, \ldots, X_{m-1}). \end{aligned} \quad (V.26)$$

Ces opérateurs permettent d'éliminer les sommes dans le second membre des équations caractéristiques sur les profils, et font disparaître la récurrence sur la taille de l'alphabet m. Cette élimination provient de la relation de dérivation :

$$\Delta(E_1.C(X_1,\ldots)) = \Delta(C(X_1,\ldots)) - M(X_1)\Delta(C(X_2,\ldots)) - X_1 e^{X_1} C(X_2,\ldots). \qquad \text{(V.27)}$$

On considère dans un premier temps le cas du partitionnement central. En reprenant la relation de récurrence V.21, on obtient facilement avec l'opérateur E_1 :

$$\Delta(E_1.C(X_1,\ldots)) = E_1.\Delta(CP(X_1,\ldots)) + \sum_{j=1}^{m} X_j e^{X_j} E_1.C(X_1,\ldots,X_j)M(X_{j+1},\ldots X_m), \qquad \text{(V.28)}$$

et en utilisant E_m :

$$\Delta(E_m.E_1.C(X_1,X_2,\ldots,X_m)) = E_m.E_1.\Delta(CP(X_1,X_2,\ldots,X_m)). \qquad \text{(V.29)}$$

On a ainsi une équation simple pour la série $C(X_1,\ldots,X_m)$ en fonction de la série de coût $CP(X_1,\ldots,X_m)$ pour chaque partitionnement. Pour simplifier les expressions, on ne considère qu'une forme simplifiée pour la fonction de coût CP. On se limite à la composante principale dans les coûts moyens quand le coût d'un partitionnement est exactement égal à la taille de la liste. La série $CP(X_1,X_2,\ldots,X_m)$ est alors égale à Xe^X avec $X = \sum X_i$. L'équation précédente devient alors

$$\Delta(E_m.E_1.C(X_1,X_2,\ldots,X_m)) = 2X_1X_m e^{X_1+\cdots+X_m} = 2\sum_{\tilde{p}} p_1 p_m \frac{\tilde{X}^{\tilde{p}}}{\tilde{p}!}. \qquad \text{(V.30)}$$

L'opérateur Δ est difficile à intégrer directement sur les séries génératrices à m variables. Toutefois si on passe aux termes généraux de ces séries, l'intégration consiste simplement à diviser chaque terme général par la taille totale du profil. On a donc

$$E_m.E_1.C(X_1,X_2,\ldots,X_m) = 2\sum_{\tilde{p}} \frac{p_1p_m}{p_1+\cdots+p_m}\frac{\tilde{X}^{\tilde{p}}}{\tilde{p}!}. \qquad \text{(V.31)}$$

En utilisant les définitions des opérateurs E_1 et E_m, on montre que

$$C(X_1,X_2,\ldots,X_m) = \sum_{1\le j<k\le m} E_k.E_j.C(X_j,\ldots,X_k)e^{X_1+\cdots+X_j+X_{k+1}+\cdots+X_m} + \sum_i C(X_i). \qquad \text{(V.32)}$$

Ceci permet donc de remonter l'équation précédente pour obtenir facilement le coût moyen du tri sur les mots de profil $p_1,\ldots,p_m$ donné :

$$\bar{C}_{p_1,\ldots,p_m} = 2\sum_{j<k}\frac{p_jp_k}{p_j+\cdots+p_k} + \sum_j \bar{C}_{p_j}. \qquad \text{(V.33)}$$

Le cas du tri avec partitionnement unilatéral se résout de manière très semblable. A partir de la récurrence V.17, on obtient en effet une équation caractéristique similaire sur les séries génératrices.

$$\Delta(e^{-X_m} E_m.E_1.C(X_1,\ldots,X_m)) = e^{-X_m} E_m.E_1.\Delta(CP(X_1,\ldots,X_m)) = 2X_1X_m e^{X_1+\cdots+X_{m-1}}. \tag{V.34}$$

On en déduit alors

$$E_m.E_1.C(X_1,X_2,\ldots,X_m) = 2\sum_{\tilde{p}} \frac{p_1p_m}{p_1+\cdots+p_{m-1}+1}\frac{\tilde{X}^{\tilde{p}}}{\tilde{p}!}, \tag{V.35}$$

et donc pour les coûts moyens

$$\bar{C}_{p_1,\ldots,p_m} = 2\sum_{j<k} \frac{p_jp_k}{1+p_j+\cdots+p_{k-1}} + \sum_j \bar{C}_{p_j}. \tag{V.36}$$

A partir de ces résultats, on peut obtenir par sommation les valeurs moyennes sur l'ensemble des mots, indépendamment du profil (cf. [Sed77b]). Cette approche est toutefois un peu lourde au niveau des sommations à effectuer. Nous allons voir dans la suite qu'il est possible d'obtenir directement ces coûts moyens sans passer par la résolution par profil mais en résolvant les équations différentielles sur les séries $C_m(X)$ ou $C(X,Y)$.

2.5. Coûts moyens sur l'ensemble des mots.

Cette section concerne la détermination directe des coûts moyens sur l'ensemble des mots pour les algorithmes de tri avec partitionnement unilatéral ou central des clés égales. Nous allons pour cela résoudre les équations caractéristiques sur les séries à deux variables qui ont été obtenues dans le corollaire V.1 p. 98 et la proposition V.5 p. 99.

Tri avec partitionnement unilatéral

La série de coût à deux variables est caractérisée par l'équation différentielle V.20 p. 98, qui s'intègre en utilisant les méthodes classiques sous la forme :

$$C(X,Y) = \frac{e^X R(X,Y)}{(1-Ye^X)^2} \qquad \text{avec} \qquad \frac{d}{dX}R(X,Y) = \frac{(1-Ye^X)^2}{e^X}\frac{d}{dX}CP(X,Y). \tag{V.37}$$

En prenant la fonction de coût CP telle qu'elle est définie au début de ce chapitre par $\bar{CP}(n,m) = a(n-1)+b$ ou par l'équation V.9 p. 93, on a

$$\frac{d}{dX}CP(X,Y) = a\frac{XYe^X(1+Ye^X)}{(1-Ye^X)^3} + b\left(\frac{Ye^X}{(1-Ye^X)^2} - \frac{Y}{(1-Y)^2}\right). \tag{V.38}$$

D'où on déduit simplement par intégration

$$C(X,Y) = \frac{e^XY}{(1-Ye^X)^2}\left(\frac{aX^2}{2} + bX + 2aY\int_0^X \frac{xe^x}{(1-Ye^x)}dx - b\frac{1-e^{-X}-2XY+Y^2(e^X-1)}{(1-Y)^2}\right). \tag{V.39}$$

On n'a pas de forme close pour $C(X,Y)$ sauf à utiliser la fonction dilogarithme définie comme étant une primitive de $\ln(1-t)/t$. Le fait d'avoir une forme intégrale dans $C(X,Y)$ n'est toutefois pas très gênant dans la mesure où, en passant au terme général de la série, cette opération de primitive se traduit par une simple division sur le terme général. On verra de plus qu'ici l'intégrale a une forme close dès que l'on extrait le terme général de la série en Y ; le calcul de la fonction $C_m(X)$ est ainsi rendu aisé.

La détermination de la série des coûts sur les mots de taille m, $C_m(X)$, consiste à prendre le terme en Y^m dans l'équation précédente. Compte tenu des développements

$$\frac{Ye^X}{(1-Ye^X)^2} = \sum_m me^{mX}Y^m \qquad et \qquad \frac{Yxe^x}{(1-Ye^x)} = \sum_{m>0} xe^{mX}Y^m, \tag{V.40}$$

on a donc

$$\begin{aligned} C_m(X) &= me^{mX}(\frac{aX^2}{2}+bX)+2a\sum_{j=1}^{m}(m-j)e^{(m-j)X}\int_0^X xe^{jx}dx \\ &\quad +bm(e^{mX}-1)+2b\sum_{j=0}^{m}e^{(m-j)X}\,(2j-m+j(m-j)X). \end{aligned} \tag{V.41}$$

On note que l'intégrale peut se calculer explicitement. On a en effet

$$\int_0^X xe^{jx}dx = \frac{1-e^{jX}+jXe^{jX}}{j^2}, \tag{V.42}$$

mais ceci n'apporte aucune simplification dans la première sommation. On peut par contre obtenir une formule explicite pour la deuxième somme qui s'exprime à partir de sommes géométriques et de leurs dérivées. Le calcul du terme général de la série et donc du coût moyen est toutefois plus rapide en utilisant la forme des équations V.41 et V.42. Ce coût moyen du tri pour des listes de taille n sur m lettres est donné par

$$\bar{C}_{m,n} = \frac{n!}{m^n}[X^n]C_m(X),$$

ce qui se calcule facilement à partir des relations :

$$\frac{n!}{m^n}[X^n]X^pe^{kX} = \frac{n(n-1)\dots(n-p+1)}{k^p}(\frac{k}{m})^n.$$

Proposition V.8 *Le coût moyen du tri des mots de taille n sur un alphabet de m lettres, dans le cas de l'algorithme avec partitionnement unilatéral des clés égales, vaut*

$$\bar{C}_{m,n} = a\frac{n(n-1)}{2m}+2a\frac{N_{m,n}}{m}+b(n+m)+2b\sum_{j=0}^{m}(2j-m+jn)(1-\frac{j}{m})^n,$$

avec

$$N_{m,n} = \sum_{j=1}^{m}(1-\frac{j}{m})\frac{(1-\frac{j}{m})^n-1+n\frac{j}{m}}{(\frac{j}{m})^2} = \sum_{j=1}^{m-1}\sum_{k=1}^{n}(n-k)(\frac{j}{m})^k.$$

La contribution principale dans le coût moyen provient des deux premiers termes liés à la constante de temps a. On voit en particulier apparaître un terme en $n^2/2m$ qui provient du traitement peu efficace des mots sur une lettre. En effet, l'algorithme tel qu'on l'a présenté ne sait pas détecter si une sous-liste ne contient que des clés égales. Dans ce cas, on continue la même procédure de tri en n'éliminant qu'une seule clé à chaque partitionnement. Pour une liste de p clés égales, le coût du tri vaut ainsi $a\,p(p-1)/2$. De manière intuitive sur les mots aléatoires, le terme en $n^2/2m$ provient donc du tri des m listes sur une seule clé qui apparaissent en fin de récursion et qui ont une taille moyenne de l'ordre de n/m.

Le comportement du coût moyen $\bar{C}_{m,n}$, en fonction des paramètres n et m, est étudié plus en détail dans la section 3.. On cherchera notamment des expressions asymptotiques pour la quantité $N_{m,n}$, quand m et n tendent vers l'infini indépendamment ou dans un rapport fixé.

Tri avec partitionnement central des clés égales

La résolution de l'équation caractéristique V.22 p. 99 pour l'algorithme avec partitionnement central est très semblable à celle du cas précédent. L'intégration donne ainsi

$$C(X,Y) = \frac{R(X,Y)}{(1-Ye^X)^2} \qquad \text{avec} \qquad \frac{d}{dX}R(X,Y) = (1-Ye^X)^2\frac{d}{dX}CP(X,Y). \tag{V.43}$$

Compte tenu de l'expression de $\frac{d}{dX}CP(X,Y)$, on a donc

$$\begin{aligned} C(X,Y) &= \frac{aY}{(1-Ye^X)^2}\left(\int_0^X \frac{2xe^x}{(1-Ye^x)}dx + e^X(1-X) - 1\right) \\ &+ \frac{bY}{(1-Ye^X)^2}\left(\frac{e^X - 1 - X - \frac{Y^2(e^X-1)^2}{2}}{(1-Y)^2} + e^X - 1\right). \end{aligned} \tag{V.44}$$

D'où, en passant au terme général en Y

$$\begin{aligned} C_m(X) &= ae^{mX}\left(2\sum_{j=1}^{m}(m+1-j)\frac{jX-1+e^{-jX}}{j^2} - m(X-1+e^{-X})\right) \\ &- b\left(e^{mX} + m\frac{e^{(m-1)X}+1}{2} + \sum_{j=0}^{m}(2j-m-1+j(m-j+1)X)e^{(m-j)X}\right) \end{aligned} \tag{V.45}$$

Ceci donne donc une expression pour le coût moyen très voisine de celle obtenue pour le tri avec partitionnement unilatéral.

Proposition V.9 *Le coût moyen du tri des mots de taille n sur un alphabet de m lettres, dans le cas de l'algorithme avec partitionnement des clés égales, vaut*

$$\begin{aligned} \bar{C}_{m,n} &= 2am\sum_{j=1}^{m}(1-\frac{j-1}{m})\frac{(1-\frac{j}{m})^n - 1 + n\frac{j}{m}}{j^2} + am\left(1 - \frac{n}{m} - (1-\frac{1}{m})^n\right) \\ &- b(1+\frac{m}{2}(1-\frac{1}{m})^n) - b\sum_{j=0}^{m-1}(2j-m-1+nj+\frac{nj}{m-j})(1-\frac{j}{m})^n. \end{aligned}$$

3. Etude asymptotique.

Les propositions V.8 et V.9 donnent pour les coûts moyens des expressions peu explicites quant au comportement pratique des deux algorithmes. Nous allons chercher ici à simplifier ces formules en les exprimant sous forme de développements asymptotiques. On distingue pour cela différentes configurations possibles des données déterminées par les valeurs relatives des paramètres n et m ou par le taux moyen de répétition de clés $\lambda = n/m$.

On analyse d'abord le cas où la taille de l'alphabet m est faible par rapport à la taille n des listes à trier; c'est-à-dire si l'on a un très fort taux de répétition des clés. Pratiquement, ceci correspond donc à une analyse asymptotique pour n tendant vers l'infini, à m fixé.

On s'intéresse ensuite au cas où les clés égales n'apparaissent qu'en nombre très limité ou avec une faible probabilité; c'est-à-dire si m tend vers l'infini avec n fixé. On obtiendra, en utilisant la formule sommatoire d'Euler, un développement des coûts moyens en puissance de $1/m$. On retrouve ainsi, avec le premier terme de développement, l'expression des coûts moyens dans le cas des clés distinctes. On a de plus, avec les termes suivants, une mesure de la perturbation due à un faible taux de répétition de clés, par rapport au modèle des clés distinctes.

La dernière configuration que l'on considère est le cas moyen où les deux paramètres m et n sont du même ordre de grandeur. On cherche alors à exprimer le coût moyen en fonction de la taille n des listes et du taux de répétition $\lambda = n/m$. L'analyse asymptotique porte sur n tendant vers l'infini pour des valeurs de λ fixées. La validité des résultats est alors assurée pour des valeurs de λ ni trop grande, ni trop petite.

Ces analyses asymptotiques étant très voisines pour les deux algorithmes avec partitionnement unilatéral ou central des clés égales, elles ne sont développées que dans le premier cas. Pour le tri avec partitionnement central, nous nous contentons d'énoncer les résultats.

3.1. Alphabet de faible cardinalité.

On suppose ici que la taille n des listes est grande par rapport à la taille m de l'alphabet. En pratique, ceci correspond par exemple au cas où l'on trie un ensemble d'objets selon un critère peu sélectif.

Si on reprend l'expression du coût moyen donnée par la proposition V.8 p. 106, on peut négliger les termes de la forme $(k/m)^n$ pour k allant de 0 à $m-1$. Les termes restants se somment alors facilement en utilisant les nombres harmoniques. Pour le tri avec partitionnement unilatéral, le coût moyen se réécrit donc sous la forme

$$\bar{C}_{m,n} = a\frac{n(n-1)}{2m} + 2a\sum_{j=1}^{m}\frac{(m-j)(nj-m)}{mj^2} + b(n-m) + R_{m,n}, \tag{V.46}$$

avec

$$R_{m,n} = 2am\sum_{j=1}^{m}\frac{(1-\frac{j}{m})^{n+1}}{j^2} + 2bm\sum_{j=1}^{m}((n+2)\frac{j}{m}-1)(1-\frac{j}{m})^n. \tag{V.47}$$

Les différents termes dans les deux sommes du reste $R_{m,n}$ décroissent rapidement par rapport à j et donnent directement un développement asymptotique exact selon les termes exponentiellement décroissants $(k/m)^n$ pour $k < m$. On peut aussi majorer grossièrement le reste $R_{m,n}$ en majorant chaque terme de la somme par le premier terme obtenu pour $j = 1$. On obtient alors le coût moyen du tri avec partitionnement unilatéral des clés égales sous la forme

$$\bar{C}_{m,n} = a\frac{n(n-1)}{2m} + 2a\left((n+1)H_m - n - mH_m^{(2)}\right) + b(n-m) + R_{m,n}, \qquad \text{(V.48)}$$

avec

$$|R_{m,n}| \leq 2m^2(1-\frac{1}{m})^n(|a| + |b|(n+1)) \qquad \text{si } m \leq n+2. \qquad \text{(V.49)}$$

De la même façon pour le tri avec partitionnement central, on obtient

$$\bar{C}_{m,n} = 2a\left((n+1)H_m + \frac{n}{m}H_m - \frac{3}{2}n + \frac{1}{2}m - (m+1)H_m^{(2)}\right) + bm + R_{m,n} \qquad \text{(V.50)}$$

avec

$$|R_{m,n}| \leq 2m^2(1-\frac{1}{m})^n(|a| + |b|n) \qquad \text{si } m \leq n+2. \qquad \text{(V.51)}$$

Dans le premier cas, la composante dominante du coût moyen est donnée par le terme $n(n-1)/2m$ qui provient du tri inefficace des mots sur une seule lettre. Hormis ce terme quadratique qu'il faut chercher à éliminer dans une implantation pratique, on retrouve dans le coût moyen des composantes linéaires très voisines pour les deux algorithmes. On voit notamment apparaître le terme $2(n+1)H_m$ qui est à rapprocher du $2(n+1)H_n$ obtenu pour le tri des clés distinctes.

Plaçons nous dans le cas de l'algorithme avec partitionnement central des clés égales. Si la taille de l'alphabet est faible, les arbres associées au tri d'une liste ont en effet une profondeur bornée par la taille de l'alphabet. Ainsi au lieu d'avoir environ n nœuds comme dans le cas des clés distinctes, les arbres ont m nœuds internes et donc une profondeur moyenne de l'ordre de $2H_m$. Intuitivement, les termes $2(n+1)H_n$ pour les clés distinctes et $2(n+1)H_m$ pour les clés répétées proviennent de la même manière du coût de partitionnement pour tout les nœuds d'un même niveau de l'arbre $\simeq n$, multiplié par le nombre moyen de niveau.

Pour préciser le domaine de validité des approximations précédentes sur les coûts moyens, le tableau V.1 donne quelques valeurs de l'expression $(1-1/m)^n$ qui intervient dans les termes d'erreur.

	$n = 10$	$n = 50$	$n = 100$	$n = 500$
$m = 5$	0.11	$1.4\,10^{-5}$	$2.0\,10^{-10}$	$3.5\,10^{-49}$
$m = 10$	0.35	$5.2\,10^{-3}$	$2.65\,10^{-5}$	$1.32\,10^{-23}$
$m = 50$	0.82	0.36	0.13	$4.1\,10^{-5}$

Tableau V.1 : Valeurs de $(1-1/m)^n$.

3.2. Clés presque toutes distinctes.

On suppose ici que les clés à trier correspondent à un alphabet de grande cardinalité. Les listes contiennent alors peu de clés identiques qui ne perturbent que légèrement le comportement du tri par rapport au cas des clés toutes distinctes. Les sommations qui apparaissent dans l'expression des coûts moyens peuvent facilement se mettre sous la forme $\sum_{j=1}^{m} f(j/m)$. On en déduit alors des développements asymptotiques en puissance de $1/m$ en utilisant la formule sommatoire d'Euler (cf. section A.5. p. 118). Le cas qui nous intéresse se limite ici à la somme de puissance d'entier; c'est-à-dire pour $f(x) = x^k$. En introduisant les nombres de Bernoulli B_n, la formule d'Euler donne alors le développement

$$\sum_{j=1}^{m}(\frac{j}{m})^N = \frac{m}{N+1} + \frac{1}{2} + \sum_{i=1}^{r} \frac{B_{2i}}{2i}\binom{N}{2i-1}\frac{1}{m^{2i-1}} + R_r, \tag{V.52}$$

où le reste R_r est majoré par

$$|R_r| \leq (r+\frac{1}{2})B_{2r}\binom{N}{2r+1}\frac{1}{m^{2r+1}}. \tag{V.53}$$

Il est important de noter que le reste R_r devient nul pour $r \geq N/2$, et donc que le développement précédent, étendu jusqu'au terme en $1/m^N$ ($r = \lfloor N/2 \rfloor$), donne une expression exacte. On retrouve ainsi l'expression de la somme de puissance N^{ieme} d'entier comme un polynôme de degré $N+1$ en la variable borne de sommation (cf. polynômes de Bernoulli dans Comtet [Com70]).

Ainsi pour l'expression $N_{m,n}$ qui intervient dans le coût moyen donné par la proposition V.8 p. 106, on a

$$N_{m,n} = \sum_{k=1}^{n}\sum_{j=1}^{m-1}(n-k)(\frac{j}{m})^k = \sum_{k=1}^{n} m\frac{n-k}{k+1} - \sum_{k=1}^{n}\frac{n-k}{2} + \sum_{i=1}^{\lfloor n/2 \rfloor}\frac{B_{2i}}{2i\,m^{2i-1}}\sum_{k=2i}^{n}(n-k)\binom{k}{2i-1}. \tag{V.54}$$

On peut alors effectuer les sommations sur l'indice k pour obtenir le développement

$$N_{m,n} = m(n+1)(H_{n+1}-1) - mn - \frac{n(n-1)}{4} + \sum_{i=1}^{\lfloor n/2 \rfloor}\frac{B_{2i}}{2i\,m^{2i-1}}\left(\binom{n+1}{2i+1} - n + 2i - 1\right). \tag{V.55}$$

L'autre sommation qui apparaît dans le coût moyen de la proposition V.8 se traite de la même manière. On a en effet

$$\begin{aligned}
\sum_{j=0}^{m}((n+2)\frac{j}{m}-1)(1-\frac{j}{m})^n &= \sum_{j=0}^{m}(n+1)(\frac{j}{m})^n - (n+2)(\frac{j}{m})^{n+1} \\
&= -\frac{1}{2} + \sum_{i=1}^{n/2}\frac{B_{2i}}{m^{2i-1}}\left(\binom{n+1}{2i} - \binom{n+2}{2i}\right) - \frac{B_{n+1}(n+2)}{m^n} \\
&= -\frac{1}{2} - \sum_{i=1}^{n/2}\frac{B_{2i}}{m^{2i-1}}\binom{n+1}{2i-1} - \frac{B_{n+1}(n+2)}{m^n}
\end{aligned} \tag{V.56}$$

En regroupant l'ensemble, on obtient donc un développement en puissance de $1/m$, exact à l'ordre n, pour le coût moyen du tri avec partitionnement unilatéral :

$$\begin{aligned}\bar{C}_{m,n} &= 2a\left((n+1)H_n - 2n\right) + b\frac{2n-1}{3} + 2a\sum_{i=2}^{n-1}\frac{B_i}{i\,m^i}\left(\binom{n+1}{i+1} + i - n - 1\right) \\ &\quad -2b\sum_{i=2}^{n-2}\binom{n+1}{i+1}\frac{B_{i+2}}{m^i} - 2b\frac{(n+2)B_{n+1}}{m^{n-1}}. \qquad (V.57)\end{aligned}$$

En se limitant aux premiers termes du développement, on a ainsi

$$\bar{C}_{m,n} = 2a\left((n+1)H_n - 2n\right) + b\frac{2n-1}{3} + a\frac{(n-1)(n-2)(n+3)}{36m^2} + b\frac{\binom{n+1}{3}}{15m^2} + O(\frac{1}{m^4}). \quad (V.58)$$

Pour l'autre algorithme avec partitionnement central, le développement exact à l'ordre n s'obtient de la même façon. L'expression complète étant toutefois assez longue à écrire, nous nous contentons ici des premiers termes du développement. On a alors

$$\begin{aligned}\bar{C}_{m,n} &= 2a\left((n+1)H_n - 2n\right) + b\frac{2n-1}{3} + a\frac{2nH_{n-1} - (n+2)(n-1)}{m} + b\frac{n(5-3n)}{12m} \\ &\quad -a\frac{(n-2)(n-1)(15-7n)}{36m^2} - b\frac{(n-2)(n-1)(31-14n)}{180m^2} + O(\frac{1}{m^3}). \qquad (V.59)\end{aligned}$$

Comme on pouvait s'y attendre pour les deux algorithmes, le terme constant du développement, $2a((n+1)H_n - 2n) + b(2n-1)/3$, est exactement le coût moyen du tri pour des clés toutes distinctes. Ainsi, plus la taille de l'alphabet est grande, plus les comportements des deux algorithmes de tri avec clés répétées se rapprochent du cas des clés distinctes étudié dans les chapitres précédents. On voit toutefois qu'il y a une grande différence entre les deux algorithmes dans la manière d'approcher le cas limite. Pour le tri avec partitionnement unilatéral des clés égales, le coût moyen est sensiblement supérieur à sa valeur limite, mais avec une convergence assez rapide en $1/m^2$. On constate de plus que le terme en an^2/m, qui provenait essentiellement du tri inefficace des listes sur une seule clé, disparaît complètement pour des listes à faible taux de répétition de clés. L'algorithme avec partitionnement central des clés égales converge lui plus lentement, en $1/m$, vers le cas limite, mais avec des valeurs du coût moyen qui sont inférieures à la valeur limite.

3.3. Taux de répétition fixé.

Après avoir vu les deux cas extrêmes où les clés sont presque toutes identiques ou presque toutes distinctes, nous allons supposer maintenant que les paramètres m et n varient dans une proportion raisonnable. On exprime alors les coûts moyens en fonctions des paramètres m et $\lambda = n/m$, et faire une analyse asymptotique pour m tendant vers l'infini avec λ fixé.

Nous allons introduire la quantité $F_k(\lambda, m)$ définie par la sommation suivante :

$$F_k(\lambda, m) = \sum_{j=1}^{m} j^k (1 - \frac{j}{m})^{\lambda m}. \tag{V.60}$$

En reprenant la proposition V.8 ou les équations V.47 et V.48 p. 109, le coût moyen, dans le cas du partitionnement unilatéral pour des listes de taille $n = \lambda m$, s'exprime simplement en fonction des quantités $F_k(\lambda, m)$. On a en effet

$$\begin{aligned} \bar{C}_{m,\lambda m} \quad = \quad & a\left(\frac{\lambda(\lambda m - 1)}{2} + 2(\lambda m + 1)H_m + 2m(F_{-2}(\lambda, m) - H_m^{(2)} - 2\lambda) - 2F_{-1}(\lambda, m)\right) \\ & + b\left(\lambda m - m - 2mF_0(\lambda, m) + 2(m\lambda + 2)F_1(\lambda, m)\right). \end{aligned} \tag{V.61}$$

L'analyse du coût moyen consiste donc maintenant à étudier le comportement asymptotique des quantités $F_k(\lambda, m)$ quand m tend vers l'infini avec λ fixé. L'idée, pour obtenir ce développement asymptotique, est de faire un développement de Taylor de chaque terme d'indice j de la somme $F_k(\lambda, m)$ et de sommer ensuite sur j chacun des termes obtenus. Cette opération est correcte si l'on montre que les restes de Taylor, pour chaque composante d'indice j, se somment pour donner un reste valide sur le développement des $F_k(\lambda, m)$.

Proposition V.10 *Pour $\lambda > 0$ fixé, on a le développement asymptotique*

$$F_k(\lambda, m) = G_k(\lambda) - \frac{\lambda}{2m} G_{k+2}(\lambda) + \frac{\lambda}{m^2}\left(\frac{\lambda G_{k+4}(\lambda)}{8} - \frac{G_{k+3}(\lambda)}{3}\right) + O(\frac{1}{m^3}),$$

avec

$$G_k(\lambda) = \sum_{j=1}^{\infty} j^k e^{-\lambda j}.$$

Remarque : Les quantités $G_k(\lambda)$, avec k entier, s'obtiennent par intégrations ou dérivations à partir de la série géométrique $G_0(\lambda) = 1/(e^\lambda - 1)$. On a en effet

$$G_k(\lambda) = -G'_{k-1}(\lambda). \tag{V.62}$$

Les $G_k(\lambda)$ s'expriment ainsi à partir des fonctions usuelles pour $k \geq -1$, et à partir des fonctions polylogarithmes (cf. [AS73]) pour $k \leq -2$. La fonction $G_{-2}(\lambda)$ dont on a besoin correspond en particulier au dilogarithme :

$$\operatorname{dilog}(x) = \int_0^x \frac{-\ln(1-x)}{x} dx = \sum_{j=1}^{\infty} \frac{x^j}{j^2}. \tag{V.63}$$

On a alors $G_{-2}(\lambda) = \operatorname{dilog}(e^{-\lambda})$, ce qui pour de faibles valeurs de λ se développe en

$$G_{-2}(\lambda) = \operatorname{dilog}(e^{-\lambda}) = \frac{\pi^2}{6} + \lambda(\ln(\lambda) - 1) - \frac{\lambda^2}{4} + O(\lambda^3). \tag{V.64}$$

La preuve de la proposition précédente découle des deux lemmes suivants.

Lemme V.4 *Pour tout k et pour $\lambda > 0$ fixé, les séries $G_k(\lambda)$ convergent avec un reste exponentiellement décroissant :*

$$G_k(\lambda) = \sum_{j=1}^{\infty} j^k e^{-\lambda j} = \sum_{j=1}^{m} j^k e^{-\lambda j} + O(m^k e^{-\lambda m}).$$

Lemme V.5 *Pour $\lambda > 0$ et $m > j$, on a*

$$f(j) = (1 - \frac{j}{m})^{\lambda m} = \exp(-\lambda \sum_{p=1}^{\infty} \frac{j^p}{pm^{p-1}}) = e^{-\lambda j}\left(1 - \frac{\lambda j^2}{2m} + \frac{3\lambda^2 j^4 - 8\lambda j^3}{24m^2} + O(\frac{j^9}{m^3})\right).$$

Preuves : Le premier lemme correspond à une majoration simple du reste des séries géométriques G_k. Le deuxième résultat est un développement de Taylor classique de la fonction $f(j)$ avec seulement une majoration non triviale du reste en fonction de la valeur de j. On l'obtient à partir de

$$A(j) = \sum_{p=2}^{\infty} \frac{j^p}{pm^{p-1}} \leq \frac{j^2}{2m} + \frac{1}{3}\sum_{p=3}^{\infty} \frac{j^p}{m^{p-1}} \leq \frac{j^2}{2m} + \frac{j^3}{3m(m-j)} \leq \frac{j^3}{m},$$

qui donne la majoration voulue dans la mesure où $f(j) = \exp(-\lambda j)\exp(-A(j))$, et où le reste du développement de Taylor à l'ordre k de $\exp(-A(j))$ est majoré par $A(j)^k/k!$ en valeur absolue.

Le développement pour $F_k(\lambda, m) = \sum_{j=1}^{m} j^k f(j)$ est maintenant obtenu en sommant les termes du développement des $f(j)$, et en remplaçant les sommes convergentes $\sum_{j=1}^{m} j^r \exp(-\lambda j)$ par leur limite. Ceci est valide puisque les restes des développements des $f(j)$ se somment de façon convergente pour donner un reste en $O(1/m^3)$ sur $F_k(\lambda, m)$. □

Remarque : On note que le développement de la proposition précédente s'étend facilement à n'importe quel ordre. De façon générale, on peut exprimer ce développement, en utilisant les polynômes de Bell, sous la forme :

$$F_k(\lambda, m) = G_k(\lambda) + \sum_{p=1}^{r} \frac{\sum_{j=1}^{p} \frac{(-\lambda)^j}{(p+j)!} B_{p,j}(0, 1!, 2!, \ldots, p!) G_{k+p+j}(\lambda)}{m^p} + O(\frac{1}{m^{r+1}} + e^{-\lambda m}). \quad \text{(V.65)}$$

En injectant dans l'expression du coût moyen donnée par l'équation V.61, les développements des $F_k(\lambda, m)$ pour $k = -2, -1, 0, 1$, on obtient ainsi la proposition suivante.

Proposition V.11 *Le coût moyen du tri de listes de taille $n = \lambda m$ avec partitionnement unilatéral des clés égales s'exprime sous la forme*

$$\begin{aligned}\bar{C}_{m,\lambda m} &= 2am\left(\lambda(H_m - 1) + \frac{\lambda^2}{4} + \text{dilog}(L) - \frac{\pi^2}{6}\right) \\ &+2a\left(H_m - \frac{\lambda}{4} - \frac{\lambda L}{2(1-L)} + \ln(1-L) + 1\right) \\ &+bm(\frac{\lambda(L^2+1)}{(1-L)^2} - \frac{1+L}{1-L}) + bL\frac{4(1-L)^2 + \lambda(1-L^2) - \lambda^2(1+4L+L^2)}{(1-L)^4} + O(\frac{1}{m}),\end{aligned}$$

en posant $L = \exp(-\lambda)$.

On obtient de la même manière le coût moyen pour le deuxième algorithme de partitionnement.

Proposition V.12 *Pour le tri avec partitionnement central des clés égales, le développement asymptotique du coût moyen est*

$$
\begin{aligned}
\bar{C}_{m,\lambda m} &= 2am\left(\lambda(H_m - \frac{3}{2}) + \mathrm{dilog}(L) - \frac{\pi^2}{6} + \frac{1-L}{2}\right) + bm\left(\frac{L^2-L+2}{2(1-L)} - \frac{L\lambda}{(1-L)^2}\right) \\
&\quad +2a\left((\lambda+1)H_m + \mathrm{dilog}(L) - \frac{\pi^2}{6} - \frac{L\lambda(1+L)}{4(1-L)} + \ln(1-L) + 1\right) \\
&\quad -bL\frac{4(1-L)(1-L^2) + \lambda(1-L^2)(5-4L+L^2) - 2\lambda^2(1+4L+L^2)}{4(1-L)^4} + O(\frac{1}{m}),
\end{aligned}
$$

avec $L = \exp(-\lambda)$.

Pour les deux algorithmes, on trouve ainsi asymptotiquement le même comportement dominant en $2a\lambda m H_m = 2anH_m$. Dans le cas où $m < n$, ce terme principal s'interprète intuitivement comme dans la section 3.1. par le fait que m borne le nombre de partitionnements.

D'un point de vue pratique, le calcul de valeurs exactes pour $n, m < 500$ montre que les formules des deux propositions précédentes donnent des erreurs très faibles ($< 1\%$) pour des valeurs du facteur λ allant de $1/10$ à 5. Hors de ces bornes pour le taux de répétition λ, il convient alors d'appliquer les expressions données dans les sections précédentes.

4. Problèmes ouverts

Nous venons de voir que, même si les calculs et les expressions sont plus lourdes que pour la statistique des permutations, il est possible d'exprimer raisonnablement les coûts moyens dans le cas d'algorithmes de tri sur des listes avec répétitions de clés. La question est maintenant de savoir si l'on peut étendre cette analyse pour les différentes variantes décrites au chapitre II ou pour la détermination de la variance et des moments d'ordre supérieur. L'exploration de ces problèmes montre qu'en réalité, on est très vite bloqué par la complexité des formules ou par l'absence de forme close sur les séries génératrices.

Si on s'intéresse au calcul des moments, il est possible d'obtenir, de manière similaire au cas des clés distinctes, des équations implicites sur la série génératrice caractéristique de la distribution. En introduisant, $C_{m,n,k}$, la probabilité que le coût soit k pour le tri de mots de taille n sur m lettres, on définit la série

$$C(X,Y,u) = \sum C_{m,n,k} \frac{M^n}{n!} X^n Y^m u^k. \tag{V.66}$$

Dans le cas du partitionnement unilatéral, cette série vérifie

$$\frac{\partial}{\partial X} C(X,Y,u) = u^b(C^2(Xu^a, Y, u) - C(Xu^a, Y, u)). \tag{V.67}$$

De même pour le partitionnement central, on obtient l'équation caractéristique

$$\frac{\partial}{\partial X} C(X,Y,u) = u^b C^2(Xu^a, Y, u) Y exp(Xu^a). \tag{V.68}$$

On trouve ainsi des relations très voisines de l'équation IV.46 p. 69 obtenue pour la distribution avec la statistique des permutations. Par dérivations successives suivant u, on peut alors en déduire des équations différentielles pour les séries des moments. Ces équations sont exactement du même type que celles obtenues pour les séries de coûts moyens, et on peut donc exprimer les solutions sous forme intégrale à partir des seconds membres. Le problème vient de la complexité du second membre de ces équations différentielles.

Si l'on veut par exemple calculer la variance, on voit apparaître, dans le second membre, le carré de la série génératrice du coût moyen. Compte tenu de l'expression complexe obtenue pour cette série de coût moyen (équation V.39 p. 105), il semble alors vain d'essayer d'obtenir des expressions raisonnables pour les termes généraux des séries de moments. Il faut rajouter à cela, qu'une fois le second moment obtenu, il resterait encore à lui retrancher le carré de la moyenne pour obtenir la variance; on peut alors prévoir un important phénomène d'élimination sur les composantes dominantes de la variance.

L'utilisation du calcul formel ne permet pas de débloquer fondamentalement la situation. Il n'offre en effet, pour le moment, que peu de moyens effectifs pour la détermination des termes généraux des séries utilisées ou le traitement et la simplification des formules sommatoires mises en jeu. Notre approche semble donc exclure la possibilité d'obtenir pour la variance ou les moments des expressions utilisables dans la pratique.

Le même type de blocage apparaît si l'on essaye d'étendre l'analyse des coûts moyens aux variantes du tri comme par exemple la médiane de trois éléments. Dans ce cas le premier problème est d'obtenir les équations sur les séries génératrices. Pour l'algorithme avec partitionnement central, ceci peut se faire facilement en reprenant les outils utilisés jusqu'a présent. L'autre algorithme avec partitionnement unilatéral n'a par contre pas de traduction simple sur les séries génératrices à une ou deux variables.

Si on obtient une équation caractéristique sur le coût moyen, le second problème est de la résoudre. Pour le tri avec médiane de trois et partitionnement centrale des clés égales, la structure de l'algorithme donne l'équation différentielle

$$\begin{aligned} C^{(3)}(X,Y) &= CP^{(3)}(X,Y) + 2Ye^X C(X,Y)M(X,Y) \\ &\quad + 6Ye^X \left(C'(X,Y)M(X,Y) + C(X,Y)M'(X,Y) + 2C'(X,Y)M'(X,Y)\right) \end{aligned} \tag{V.69}$$

où les dérivations sont des dérivations partielles suivant X. Ceci se simplifie alors sous la forme

$$R^{(2)}(X,Y) + 6\frac{Ye^X}{1-Ye^X}R'(X,Y) + 6\left(\frac{Ye^X}{1-Ye^X}\right)^2 R(X,Y) = (1-Ye^X)^2 CP^{(3)}(X,Y), \tag{V.70}$$

avec

$$R(X,Y) = ((1-Ye^X)^2 C(X,Y))'.$$

Les méthodes classiques de résolution ne permettent pas d'obtenir les solutions $R(X,Y)$ ou $C(X,Y)$ sous une forme utilisable dans la pratique. En particulier, pour l'équation différentielles sans second membre, les solutions ne s'expriment que sous forme de fonctions hypergéométriques en la variable $Z = Ye^X$.

Les problèmes que l'on vient d'esquisser montrent les limites de notre approche qui consiste à déterminer, sous une forme relativement explicite, les séries génératrices introduites. Pour sortir de ces limites, qui semblent assez naturelles, il convient donc de développer des méthodes permettant d'extraire les informations utiles sur le terme général directement à partir des caractérisations implicites sur les séries génératrices. Cette d'approche correspond par exemple au théorème d'inversion de Lagrange applicable pour un certain type d'équation implicite. Dans le cas qui nous intéresse, c'est-à-dire pour des équations différentielles, les résultats de ce genre restent pour le moments très limités.

Annexe A

Notations et Définitions.

Cette annexe donne quelques définitions et propriétés de base des quantités combinatoires que l'on utilise dans notre analyse. Pour plus de précision on peut se reporter à Comtet [Com70] ou [Com74].

1. Nombres complexes.

Les parties réelles et imaginaires sont notées : $x = \Re(x) + \imath\Im(x)$.

2. Coefficients binomiaux.

$$\binom{r}{k} = \begin{cases} 1 & \text{si } k = 0 \\ 0 & \text{si } k < 0 \\ \frac{r(r-1)\cdots(r-k+1)}{k!} & \text{sinon} \end{cases} \tag{A.1}$$

$$\binom{r+1}{k+1} = \binom{r}{k} + \binom{r}{k+1} \qquad \binom{-r}{k} = (-1)^k \binom{r+k-1}{k} \tag{A.2}$$

3. Nombres harmoniques d'ordre p.

$$H_n = \sum_{i=1}^{n} \frac{1}{i} = \ln n + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + O(\frac{1}{n^4}) \tag{A.3}$$

$$H_n^{(p)} = \sum_{i=1}^{n} \frac{1}{i^p} = \zeta(p) - \frac{1}{(p-1)n^{p-1}} + \frac{1}{2n^p} + O(\frac{1}{n^{p+1}}) \qquad \text{pour } p > 1 \tag{A.4}$$

avec $\gamma \approx 0.5772$ la constante d'Euler et ζ la fonction Zeta de Riemann.

4. Nombres de Stirling.

On note $S(n,k)$ les nombres de Stirling de seconde espèce, $s(n,k)$ les nombres de Stirling de première espèce et $\bar{s}(n,k) = |s(n,k)| = (-1)^{n+k}s(n,k)$.

On a alors les propriétés :

$$
\begin{array}{ll}
x^n = \displaystyle\sum_k k!S(n,k)\binom{x}{k} & n!\displaystyle\binom{x}{n} = \sum_k s(n,k)x^k \\
S(n,n) = S(0,0) = 1,\ S(n,0) = S(0,k) = 0 & s(n,n) = s(0,0) = 1,\ s(n,0) = s(0,k) = 0 \\
S(n,k) = kS(n-1,k) + S(n-1,k-1) & \bar{s}(n,k) = (n-1)\bar{s}(n-1,k) + \bar{s}(n-1,k-1) \\
S(n+k,n) = \displaystyle\sum_{1\le i_1\le\cdots\le i_k\le n} i_1 i_2\cdots i_k & \bar{s}(n,n-k) = \displaystyle\sum_{1\le i_1<\cdots<i_k<n} i_1 i_2\cdots i_k \\
\exp(u(e^t-1)) = \displaystyle\sum_{n,k} S(n,k)u^k\frac{t^n}{n!} & (1+t)^u = \displaystyle\sum_{n,k} s(n,k)u^k\frac{t^n}{n!} \\
S(n,k) = \displaystyle\frac{1}{k!}\sum_{j=0}^{k}(-1)^j\binom{k}{j}(k-j)^n & \displaystyle\sum_k S(n,k)s(k,m) = \sum_k s(n,k)S(k,m) = \delta_{n,m} \\
\displaystyle\sum_k S(n+1,k+1)s(k,m) = \binom{n}{m} & \displaystyle\sum_k s(n+1,k+1)S(k,m) = (-1)^{n+m}\frac{n!}{m!} \\
\displaystyle\sum_k \binom{n}{k}S(k,m) = S(n+1,m+1) & \displaystyle\sum_k s(n+1,k+1)\binom{k}{m} = s(n,m) \\
\displaystyle\sum_k (-1)^{n-k}\binom{n}{k}S(k+1,m+1) = S(n,m) & \displaystyle\sum_k \bar{s}(n,k)\binom{k}{m} = \bar{s}(n+1,m+1)
\end{array}
\tag{A.5}
$$

$$\Theta = t\,\frac{d}{dt} \quad\Longrightarrow\quad \Theta^n = \sum_{k=1}^{n} S(n,k)t^k\left(\frac{d}{dt}\right)^k \quad\Longleftrightarrow\quad \binom{\Theta}{k} = \frac{t^k\left(\frac{d}{dt}\right)^k}{k!} \tag{A.6}$$

$$F(\ln x)^{(n)} = x^{-n}\sum_k s(n,k)F^{(k)}(\ln x) \tag{A.7}$$

5. Formule sommatoire d'Euler.

Pour une fonction f, C^∞ sur un intervalle $[a,b]$, on a le développement :

$$\sum_{i=a}^{b} f(i) = \int_a^b f(t)dt + \frac{f(a)+f(b)}{2} + \sum_{h=1}^{r}\frac{B_{2h}}{2h!}(f^{(2h-1)}(b) - f^{(2h-1)}(a)) + R_r\,, \tag{A.8}$$

avec la majoration du reste

$$|R_r| \le \frac{(r+1/2)B_{2r}}{(2r+1)!}\int_a^b |f^{(2r+1)}(t)|dt. \tag{A.9}$$

Les B_r sont les nombres de Bernoulli et valent

$$B_0 = 1, \quad B_1 = -\frac{1}{2}, \quad B_2 = \frac{1}{6}, \quad B_4 = -\frac{1}{30}, \quad B_6 = \frac{1}{42}, \quad B_8 = -\frac{1}{30} \ldots, \quad B_{2i+1} = 0. \tag{A.10}$$

Sous une autre forme, pour une fonction f, C^∞ sur un intervalle $[0,1]$, on a :

$$\frac{1}{m}\sum_{i=0}^{m} f(\frac{i}{m}) = \int_0^1 f(t)dt + \frac{f(a)+f(b)}{2m} + \sum_{h=1}^{r} \frac{B_{2h}}{2h!m^{2h}}(f^{(2h-1)}(b) - f^{(2h-1)}(a)) + O(m^{2r+2}). \tag{A.11}$$

6. Polynômes de Bell.

Les polynômes de Bell partiels $B_{n,k}(x_1, x_2, \ldots, x_{n-k+1})$ sont définis par le développement en série double :

$$\exp(u\sum_{n\geq 0} x_n \frac{t^n}{n!}) = \sum_{n,k\geq 0} B_{n,k}(x_1, x_2, \ldots, x_{n-k+1})\frac{t^n}{n!}u^k. \tag{A.12}$$

Pour une fonction $f(u)$, on pose $f(u) = \sum f_k \frac{u^k}{k!}$. On a alors les propriétés suivantes :

$$\text{Faa Di Bruno :} \qquad h = f \circ g \quad \Longrightarrow \quad h_n = \sum_{i=1}^{n} f_i B_{n,i}(g_1, g_2, \ldots, g_{n-i+1}) \tag{A.13}$$

$$\text{Inversion de Lagrange :} \qquad f_n^{<-1>} = \sum_{k=1}^{n-1} \binom{-n}{k} k! f_1^{-n-k} B_{n-1,k}(\frac{f_2}{2}, \frac{f_3}{3}, \ldots) \tag{A.14}$$

$$\text{Expression exacte :} \quad B_{n,k}(x_1, x_2, \ldots) = \sum_{\substack{c_1+c_2+c_3+\ldots=k \\ c_1+2c_2+3c_3+\ldots=n}} \frac{n!}{c_1!c_2!\ldots(1!)^{c_1}(2!)^{c_2}\ldots} x_1^{c_1} x_2^{c_2} \ldots \tag{A.15}$$

Le nombre de monômes de $B_{n,k}$ est donc le nombre de partitions de n à k sommants.

Les polynômes de Bell complets (ou exponentiels) $Y_n(x_1, x_2, \ldots, x_n)$ et les polynômes logarithmiques $L_n(x_1, x_2, \ldots, x_n)$ sont définis par les développement en série :

$$\exp(\sum_{n>0} x_n \frac{t^n}{n!}) = \sum_{n\geq 0} Y_n(x_1, x_2, \ldots, x_n)\frac{t^n}{n!}, \tag{A.16}$$

$$\ln(1 + \sum_{n>0} x_n \frac{t^n}{n!}) = \sum_{n\geq 0} L_n(x_1, x_2, \ldots, x_n)\frac{t^n}{n!}. \tag{A.17}$$

De ces définitions on déduit les propriétés principales des polynômes de Bell :

$$
\begin{array}{ll}
B_{n,n} = x_1^n \quad B_{n,1} = x_n \quad B_{0,0} = 1 & B_{n,k}(1!, 2!, \ldots) = \binom{n-1}{k-1}\frac{n!}{k!} \\
B_{n,k}(abx_1, ab^2x_2, \ldots) = a^k b^n B_{n,k}(x_1, x_2, \ldots) & B_{n,k}(1, 1, \ldots) = S(n,k) \\
B_{n,k}(X+Y) = \sum_{\substack{i\leq k\\ l\leq n}} \binom{n}{l} B_{l,i}(X)B_{n-l,k-i}(Y) & B_{n,k}(0!, 1!, 2!, \ldots) = \bar{s}(n,k) \\
B_{n,k}(\frac{x_2}{2}, \frac{x_3}{3}, \ldots) = \frac{n!}{(n+k)!}B_{n+k,k}(0, x_2, x_3, \ldots) & k\, B_{n,k} = \sum_{i=k-1}^{n-1}\binom{n}{i} x_{n-i}B_{i,k-1} \\
L_n = \sum_{k=1}^{n}(-1)^{k-1}(k-1)!B_{n,k}, \quad L_0 = 0 & Y_n = \sum_{k=1}^{n} B_{n,k}, \quad Y_0 = 1 \\
L_n(1, 1, \ldots) = \delta_{n,1} & Y_n(0!, 1!, 2!, \ldots, n!) = n! \\
L_n(1!, 2!, \ldots, n!) = (n-1)! & Y_n(X+Y) = \sum_k \binom{n}{k} Y_k(X)Y_{n-k}(Y) \\
L_n(\ldots, x_n) = x_n + L_n(\ldots, x_{n-1}, 0) & Y_n(\ldots, x_n) = x_n + Y_n(\ldots, x_{n-1}, 0)
\end{array}
\tag{A.18}
$$

7. Termes généraux de séries.

La notation $[z^n]f(z)$ désigne le terme général d'ordre n de la série $f(z)$. On a en particulier :

$$
\begin{array}{ll}
[z^n](1+z)^r = \binom{r}{n} & [z^n]\frac{1}{(1-z)^{k+1}} = \binom{n+k}{k} \\
[z^n]e^z = \frac{1}{n!} & [z^n]\ln\frac{1}{1-z} = \frac{1}{n} \\
[z^n](e^z-1)^k = \frac{k!}{n!}S(n,k) & [z^n]\ln^k\frac{1}{1-z} = \frac{k!}{n!}s(n,k) \\
[z^n]\frac{\ln\frac{1}{1-z}}{(1-z)^{k+1}} = \binom{n+k}{k}(H_{n+k}-H_k) & [z^n](1-z)^p\ln\frac{1}{1-z} = \frac{(-1)^p}{(p+1)\binom{n}{p+1}}
\end{array}
\tag{A.19}
$$

$$
[z^n]\frac{\ln\left(\frac{1}{1-z}\right)^p}{1-z} = \frac{p!}{n!}|s(n+1,p+1)| = Y_p(H_n, -H_n^{(2)}, \cdots, (-1)^{k+1}(k-1)!H_n^{(k)}, \cdots). \tag{A.20}
$$

Les coefficients $C_k^i(n)$ définis par :

$$
C_k^i(n) = [z^n]\frac{\ln^k\frac{1}{1-z}}{(1-z)^i} \tag{A.21}
$$

sont obtenus par dérivation de la formule précédente. On a :

$$\frac{d^{i-1}}{dz^{i-1}}\frac{ln^k\frac{1}{1-z}}{1-z} = \sum_j |s(i,j+1)|j!\binom{k}{j}\frac{ln^{k-j}\frac{1}{1-z}}{(1-z)^i}. \tag{A.22}$$

D'où le récurrence

$$C_k^i(n) = \binom{n+i-1}{i-1}Y_k(H_{n+i-1}, -H_{n+i-1}^{(2)}, \cdots) - \sum_{j=1}^{inf(i-1,k)} \frac{k!|s(i,j+1)|}{(i-1)!(k-j)!}C_{k-j}^i(n). \tag{A.23}$$

8. Valeurs typiques d'une variable aléatoire.

- Moyenne : $\mu = E(X)$.
- Moments : $M_p = E(X^p)$, $\psi(t) = \sum_i M_i \frac{t^i}{i!} = E(e^{tX})$.
- Moments centrés : $MC_p = E((X-\mu)^p)$, $\psi^*(t) = E(e^{t(X-\mu)}) = e^{-t\mu}\psi(t)$.
- Cumulants : κ_p , $\gamma(t) = \sum_i \kappa_i \frac{t^i}{i!} = \ln(\psi(t))$.
- Moments factoriels : $\hat{M}_p = E(X(X-1)(X-2)\cdots(X-p+1))$.

Le passage entre les différents moments est donné par les formules :

$$\begin{aligned}
M_n &= \sum_i \binom{n}{i}\mu^i MC_{n-i} & MC_n &= \sum_i \binom{n}{i}(-1)^i\mu^i M_{n-i} \\
M_n &= Y_n(\kappa_1, \kappa_2, \ldots, \kappa_n) & \kappa_n &= L_n(M_1, M_2, \ldots, M_n) \\
MC_n &= Y_n(0, \kappa_2, \ldots, \kappa_n) & \kappa_n &= L_n(0, MC_2, \ldots, MC_n) \\
M_n &= \sum_k S(n,k)\hat{M}_k & \hat{M}_n &= \sum_k s(n,k)M_k
\end{aligned} \tag{A.24}$$

Annexe B

Polynômes caractéristiques.

Le comportement du tri ou de la recherche dans un arbre s-aire avec médiane est caractérisé par les polynômes :

$$PQ_{s,t}(x) = (-1)^k \binom{x}{k} - s(-1)^t \binom{x}{t} \quad \text{avec} \quad k = s(t+1) - 1\,. \tag{B.1}$$

Les racines α_i de ces polynômes font en particulier apparaître des termes en $O(N^{-\Re(\alpha_i)-1})$ pour le coût moyen ou pour les moments d'ordres supérieurs.

De façon plus générale, on considère aussi les polynômes $PG_{s,t}(x)$ correspondants à l'algorithme avec médiane et sélection non uniforme de la clé de partitionnement pour des arbres s-aires (cf. section III.4. p. 51) :

$$PG_{s,k}(x) = (-1)^k \binom{x}{k} - \sum_{t_1+\cdots+t_s=k+1-s} P_{t_1,t_2,\ldots,t_s} \left[\sum_{i=1}^{s} (-1)^{t_i} \binom{x}{t_i} \right] \tag{B.2}$$

où $P_{t_1,t_2,\ldots,t_s}$ décrit une distribution de probabilité, c'est à dire $P_{t_1,t_2,\ldots,t_s} \geq 0$ et $\sum P_{t_1,t_2,\ldots,t_s} = 1$.

1. Racines de $PQ_{s,t}(x)$.

Lemme B.1 $0, 1, \ldots, t-1$ *sont des racines triviales de* $PQ_{s,t}(x)$.

Les autres racines sont alors caractérisées par :

$$PT(x) = \frac{(t-x)(t+1-x)\cdots(k-x-1)}{(t+2)(t+3)\cdots(k+1)} = \prod_{i=t}^{k-1} \frac{i-x}{i+2} = 1\,. \tag{B.3}$$

Lemme B.2 -2 *est racine de* $PQ_{s,t}(x)$, *et toutes les autres racines ont une partie réelle* $\Re(x)$ *strictement supérieure à* -2.

Preuve : soit x un nombre complexe tel que $\Re(x) \leq -2$. On a alors

$$\forall i > 0 \quad |i - x| \geq i + 2 \quad \text{avec égalité ssi } x = -2\,.$$

d'où

$$|PT(x)| = \prod_{i=t}^{k-1} \frac{|i-x|}{i+2} < 1 \qquad \text{si } x \neq -2$$

et donc x n'est racine que si $x = -2$. □

Lemme B.3 *$t+k+1 = (s+1)(t+1)-1$ est racine de $PQ_{s,t}(x)$ si $(s+1)(t+1)$ est un entier pair. Toutes les racines (sauf éventuellement $t+k+1$) vérifient $\Re(x) < t+k+1$.*

La preuve est identique à celle du lemme précédent mais peut aussi être déduite des propriétés de symétrie du polynôme $PT(x)$:

$$PT(x) = (-1)^{(k+t)} PT(t+k-1-x)\,. \tag{B.4}$$

En particulier si $(s+1)(t+1)$ est pair, les racines de $PQ_{s,t}(x)$ sont symétriques par rapport à l'axe $\Re(x) = \frac{t+k-1}{2}$.

Lemme B.4 *Les seules racines réelles de $PQ_{s,t}(x)$ sont -2 et éventuellement $t+k+1$.*

Preuve : soit x un réel :

- Si $-2 < x < t$, on a
$$\forall i \in [t, k-1] \quad 0 < i - x < i+2$$
d'où en passant au produit $|PT(x)| < 1$ et x n'est pas racine.
- De même par symétrie, si $k-1 < x < k+t+1$, x ne peut être racine.
- Si $t < x < k-1$, en notant $\lfloor x \rfloor$ la partie entière de x, on a
$$|PT(x)| = \frac{(x-t)(x-1-t)\cdots(x-\lfloor x\rfloor)(1+\lfloor x\rfloor - x)(2+\lfloor x\rfloor)\cdots(k-1-x)}{(t+2)\cdots(k-1)}\,.$$
Chaque terme du produit est positif et peut être majoré par l'entier supérieur, on a donc
$$|PT(x)| < \frac{(\lfloor x\rfloor + 1 - t)\,!(k-1-\lfloor x\rfloor)\,!}{(t+2)\cdots(k-1)} = 1/\binom{k+1}{t+1\,,\ k-1-\lfloor x\rfloor\,,\ \lfloor x\rfloor+1-t} < 1\,.$$
- Il est clair enfin que t et $k-1$ ne sont pas racines.

□

Lemme B.5 *Le polynôme $PQ_{s,t}(x)$ n'a pas de racines multiples.*

Preuve : Soit x une racine double, on a

$$PQ'_{s,t}(x) = 0 \Rightarrow A(x) = \frac{1}{t-x} + \frac{1}{t+1-x} + \cdots + \frac{1}{k-1-x} = 0\,.$$

On a alors pour la partie imaginaire $\Im(A(x)) = \Im(x)\sum_i 1/|i-x|^2 = 0$ et donc x doit être réel. Ceci implique que $t < x < k-1$ car sinon les termes de la somme sont tous de même signe. On a donc contradiction avec le lemme précédent qui interdit les racines réelles entre t et $k-1$. □

On récapitule les résultats précédents par le lemme :

Lemme B.6 (Localisation des racines de $PQ_{s,t}(x)$)
Les racines de $PQ_{s,t}(x)$ sont :

- $0, 1, \ldots, t-1$.
- -2
- $(s+1)(t+1)-1$ *si* $(s+1)(t+1)$ *est un entier pair.*
- *Des complexes* $\alpha_i(s,t)$ *et leurs conjugués* $\overline{\alpha_i}(s,t)$ *pour* $i \in [1 \ldots \lfloor \frac{k-t-1}{2} \rfloor]$ *avec*

$$-2 < \Re(\alpha_i(s,t)) = \Re(\overline{\alpha_i}(s,t)) < (s+1)(t+1)-1.$$

Toutes les racines sont des racines simples.

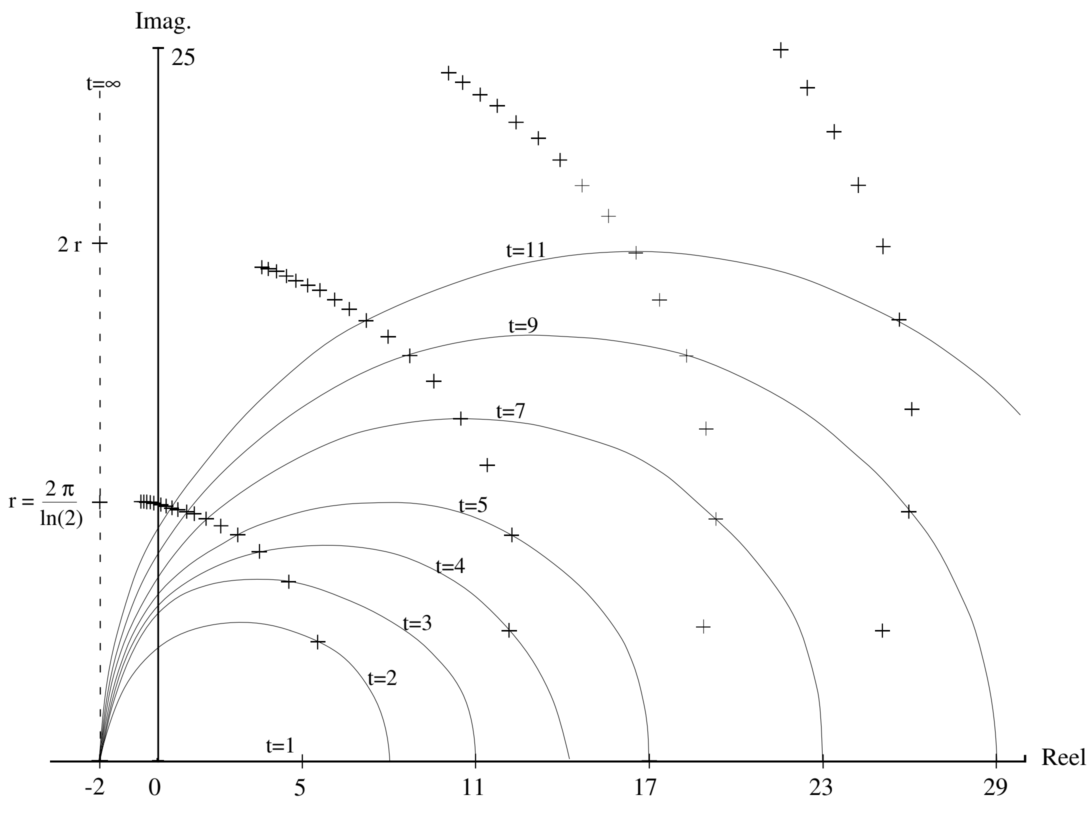


Figure B.1 : Racines de $PQ_{s,t}(x)$ pour $s = 2$ et $t < 21$.

La figure B.1 montre le comportement étonnement régulier des racines non triviales $\alpha_i(s,t)$ de $PQ_{s,t}(x)$ pour les valeurs successives de t dans le cas $s = 2$: chaque courbe correspond aux racines pour une valeur de t fixée.

2. Equivalent quand $t \to \infty$.

On appelle $\alpha_k(s,t)$ la k^{ieme} racine complexe de plus petite partie réelle et de partie imaginaire positive. On note alors la racine conjuguée $\alpha_{-k}(s,t) = \overline{\alpha_k}(s,t)$ et $\alpha_0(s,t) = -2$.

Le comportement de $\alpha_k(s,t)$ pour s fixé et $t \to \infty$ est alors donné par le lemme suivant :

Lemme B.7 *Pour s fixé et en posant $c = 2k\pi/\ln s$ pour $k \in \mathbf{Z}$, on a le développement*

$$\begin{aligned}\alpha_k(s,t) &= (-2+\imath c) + (s-1)\frac{c(c+\imath)}{2s\ln(s)\,t} - (s-1)c^2\frac{3(1-s)+\ln(s)(3s+1)}{4s^2\ln^2(s)\,t^2} \\ &\quad + \imath c(s-1)\frac{2c^2(s+1)\ln(s)+3(1-s)(2c^2-1)-\ln(s)(7s+1)}{12s^2\ln^2(s)\,t^2} + O(\frac{1}{t^3})\,.\end{aligned} \tag{B.5}$$

Comme le montre la figure B.1, les racines tendent lentement vers l'axe $\Re(x) = -2$ quand $t \to \infty$. A la limite, les racines complexes de $PQ_{s,t}$ induisent donc des termes quasiment linéaires dans les coûts, mais pour des valeurs de t raisonnables dans la pratique ($t < 15$) les composantes introduites dans les coûts restent en $O(1/n)$.

Pour être plus précis, on introduit la valeur $b_{s,t} = \max(-\Re(\alpha+1))$ étendu aux racines α de $PQ_{s,t}$ autres que $-2, 0, 1, \ldots, t-1$. Les composantes introduites dans les coûts moyens par les racines autres que -2 sont alors en $O(n^{b_{s,t}})$, et $b_{s,t} < 1$ indépendamment de s et t. Le tableau B.1 donne de plus les premières valeurs de $b_{s,t}$ en fonction des paramètres s et t.

$b_{s,t}$	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
s=2	$-\infty$	-6	-6.5	-5.5	-4.53	-3.75	-3.14	-2.66	-2.28	-1.96	-1.70
s=3	-4	-3.5	-2.25	-1.49	-1.01	-0.68	-0.44	-0.26	-0.12	-0.01	0.08
s=4	-3.5	-1.76	-0.88	-0.42	-0.13	0.06	0.19	0.29	...	...	...
s=5	-2.5	-0.92	-0.29	0.03	0.22	0.35	...	...	...	...	...
s=6	-1.76	-0.46	0.03	0.27	...	...	...	...	...	...	...

Tableau B.1 : Exposants principaux liés aux racines complexes de $PQ_{s,t}$.

Pour obtenir la formule du lemme précédent, on caractérise les racines non triviales de $PQ_{s,t}(x)$ en utilisant la fonction Gamma d'Euler. L'équation B.3 s'écrit alors :

$$\frac{\Gamma(s(t+1)-1-x)}{\Gamma(s(t+1))} = s\,\frac{\Gamma(t-x)}{\Gamma(t+1)}\,. \tag{B.6}$$

Le développement de la fonction $\ln\Gamma(z)$,

$$\ln\Gamma(z) = 2\,\imath k\pi + (z-1/2)\ln z - z + \ln(2\pi)/2 + \sum_{h=1}^{n}\frac{B_{2h}\,z^{1-2h}}{2h\,(2h-1)} + O(z^{-2n-1})\,,$$

donne pour les racines une équation de la forme :

$$-\ln(s)(x+2) + 2\imath k\pi + P_2(x)/t + P_3(x)/t^2 + \cdots = 0$$

où les $P_i(x)$ sont des polynômes de degré i.

On en déduit alors de proche en proche les termes successifs de l'équation B.5.

3. Racines de $PG_{s,k}(x)$.

Lemme B.8 *-2 est une racine simple de $PG_{s,k}(x)$ et les autres racines vérifient $-2 < \Re(x) < 2k-2$.*

Preuve :

- On vérifie facilement que -2 est racine car

$$PG_{s,k}(-2) = k+1 - \sum_{t_1+\cdots+t_s=k+1-s} P_{t_1,t_2,\ldots,t_s}\left[\sum_{i=1}^{s} t_i + 1\right] = 0.$$

- Le polynôme $PG_{s,k}(x)$ peut se mettre sous la forme

$$PG_{s,k}(x) = (-1)^k \binom{x}{k}\left[1 - \sum_{t_1+\cdots+t_s=k+1-s} P_{t_1,t_2,\ldots,t_s}\sum_{i=1}^{s} B_{t_i}(x)\right]$$

  avec

$$B_j(x) = \frac{j+1}{k+1}\,\frac{(k+1)k\cdots(j+2)}{(k-1-x)(k-2-x)\cdots(j-x)}.$$

  Soit une racine x telle que $\Re(x) \leq -2$ et $x \neq -2$, on a

$$\forall j \in [0..k-1] \qquad |B_j(x)| < \frac{j+1}{k+1}$$

  d'où pour chaque s-uplet $t_1, t_2, \ldots, t_s$ sommant à $k+1-s$

$$\left|\sum_{i=1}^{s} B_{t_i}(x)\right| < \sum_{i=1}^{s}\frac{t_i+1}{k+1} = 1$$

  et donc x ne peut être racine.

- La preuve est identique pour $\Re(x) < 2k-2$.

- -2 n'est pas racine double, car par concavité de nH_n on a

$$PG'_{s,k}(-2) = -(k+1)H_{k+1} + \sum_{t_1+\cdots+t_s=k+1-s} P_{t_1,t_2,\ldots,t_s}\left[\sum_{i=1}^{s}(t_i+1)H_{t_i+1}\right] < 0\,.$$

□

4. Dérivés de binomiaux.

On considère les polynômes :

$$P_k(x) = (-1)^k\binom{x}{k} = \binom{k-1-x}{k}. \qquad \text{(B.7)}$$

Lemme B.9 *Les dérivés de $P_k(x)$ s'expriment en fonction des nombres harmoniques généralisés*

$$S_k^{[p]}(x) = \sum_{i=0}^{k-1} \frac{1}{(i-x)^p} \tag{B.8}$$

par

$$P_k^{(p)}(x) = P_k(x) Y_p(-S_k^{[1]}(x), -1!\, S_k^{[2]}(x), -2!\, S_k^{[3]}(x), \ldots, -(p-1)! S_k^{[p]}(x)) \,. \tag{B.9}$$

où les $Y_p(\ldots)$ désignent les polynômes de Bell complets (ou exponentiels).

Preuve : On a de façon générale la propriété :

$$f(x+y) = \sum_{p=0}^{\infty} f^{(p)}(x) \frac{y^p}{p!} \,.$$

En exprimant les $S_k^{[p]}(x)$ sous la forme :

$$(p-1)!\, S_k^{[p]}(x) = \sum_{i=0}^{k-1} \ln^{(p)}(1/(i-x))$$

on a donc :

$$\begin{aligned}
-\sum_{p=1}^{\infty} (p-1)!\, S_k^{[p]}(x) \frac{y^p}{p!} &= -\sum_{i=0}^{k-1} \sum_{p=1}^{\infty} \ln^{(p)}\left(\frac{1}{i-x}\right) \frac{y^p}{p\,!} \\
&= -\sum_{i=0}^{k-1} \left[\ln \frac{1}{i-x-y} - \ln \frac{1}{i-x} \right] \\
&= \ln \prod_{i=0}^{k-1} \frac{x+y-i}{x-i} = \ln \frac{P_k(x+y)}{P_k(x)} \\
&= \ln \sum_{p=0}^{\infty} \frac{P_k^{(p)}(x)}{P_k(x)} \frac{y^p}{p\,!} \,.
\end{aligned}$$

D'où le résultat par définition des polynômes de Bell Y_p. □

Annexe C

Equations sur les moments.

1. Coûts moyens exacts dans le cas binaire.

On considère l'algorithme de tri Quicksort dans le cas binaire $s = 2$ avec médiane de $2t+1$ clés. On effectue de plus un tri par insertion pour les listes de taille inférieure ou égale à un paramètre M donné.

Les formules suivantes donnent pour $t = 0, 1, 2, 3$ les valeurs exactes des coûts moyens pour les quantités caractéristiques de l'algorithme introduites par R. Sedgewick [Sed80].

Pour un paramètre A, le coût moyen est complètement déterminé par la donnée de :

- AP_n : coût moyen d'un partitionnement sur les listes de taille $n > M$.
- AF_n : désigne le coût moyen du tri (par insertion) sur les listes de taille $n \leq M$.

On introduit les racines complexes z_2 et z_3 du polynôme caractéristique $PQ_{2,t}$ pour $t = 2$ et $t = 3$, et l'on pose :

$$B_i^{n,M} = (-1)^{n-M} \frac{\binom{z_i}{n}}{\binom{z_i}{M+1}} \qquad (i = 2, 3)\,, \tag{C.1}$$

avec

$$z_2 = \frac{11 + \imath\sqrt{71}}{2}, \qquad z_3 = \frac{9 + \imath\sqrt{159}}{2}.$$

1. Nombre de comparaisons entre clés pendant les partitionnements (on ne compte pas la

recherche de la médiane) $CF_n = 0, CP_n = n+1-2t$:

$$\bar{C}_n = \left| \begin{array}{ll} t=0 & 2(n+1)(H_{n+1} - H_{m+2} + \frac{1}{2}) \\ t=1 & \frac{12}{7}(n+1)(H_{n+1} - H_{M+2} - \frac{1}{7} - \frac{2}{M+2}) + \frac{4(3M-1)}{49}\frac{\binom{M+1}{6}}{\binom{n}{6}} + n + 3 \\ t=2 & \frac{60}{37}(n+1)(H_{n+1} - H_{M+2} - \frac{15}{74} - \frac{4}{M+2}) + n + 5 \\ & \quad - \frac{2}{1369}\Re\left((225M - 586 + \imath\frac{(1155M-2278)}{\sqrt{71}})B_2^{n,M}\right) \\ t=3 & \frac{840}{533}(n+1)(H_{n+1} - H_{M+2} - \frac{251}{1066} - \frac{6}{M+2}) + \frac{420(2M-9)}{6929}\frac{\binom{M+1}{12}}{\binom{n}{12}} \\ & \quad + n + 7 - \frac{12}{1681}\Re\left((35M - 176 + \imath\frac{455M-1796}{\sqrt{159}})B_3^{n,M}\right) \end{array} \right. \tag{C.2}$$

2. Nombre d'appels à la procédure de partitionnement $AF_n = 0, AP_n = 1$:

$$\bar{A}_n = \left| \begin{array}{ll} t=0 & 2\frac{n+1}{M+2} - 1 \\ t=1 & \frac{12}{7}\frac{n+1}{M+2} - 1 + \frac{2}{7}\frac{\binom{M+1}{6}}{\binom{n}{6}} \\ t=2 & \frac{60}{37}\frac{n+1}{M+2} - 1 - \frac{2}{37}\Re\left((7 + \imath\frac{31}{\sqrt{71}})B_2^{n,M}\right) \\ t=3 & \frac{840}{533}\frac{n+1}{M+2} - 1 + \frac{70}{533}\frac{\binom{M+1}{12}}{\binom{n}{12}} - \frac{12}{41}\Re\left((1 + \imath\frac{11}{\sqrt{159}})B_3^{n,M}\right) \end{array} \right. \tag{C.3}$$

3. Nombre d'échanges de clés pendant le partitionnement $BF_n = 0, BP_n = \frac{(t+1)n-2t^2+4}{2(2t+3)} =$ nombre optimal d'échanges nécessaires $+1$. D'où

$$\forall t \qquad \bar{B}_n = \frac{(t+1)\bar{C}_n + (t+3)\bar{A}_n}{2(2t+3)} \tag{C.4}$$

4. Nombre de clés réellement insérées (pour les petites listes) $DF_n = n - H_n, DP_n = 0$:

$$\bar{D}_n = \left| \begin{array}{ll} t=0 & (n+1)(1-2\frac{H_{M+1}}{M+2}) \\ t=1 & (n+1)(1-\frac{12}{7}\frac{H_{M+1}+1/3}{M+2}) + \frac{5-6H_{M+1}}{21}\frac{\binom{M+1}{6}}{\binom{n}{6}} \\ t=2 & (n+1)(1-\frac{60}{37}\frac{H_{M+1}+13/30}{M+2}) \\ & -\frac{1}{1110}\Re\left((299-420H_{M+1}+\imath\frac{1007-1860H_{M+1}}{\sqrt{71}})B_2^{n,M}\right) \\ t=3 & (n+1)(1-\frac{840}{533}\frac{H_{M+1}+101/210}{M+2}) + \frac{253-420H_{M+1}}{3198}\frac{\binom{M+1}{12}}{\binom{n}{12}} \\ & -\frac{4}{1435}\Re\left((71-105H_{M+1}+\imath\frac{571-1155H_{M+1}}{\sqrt{159}})B_3^{n,M}\right) \end{array} \right. \tag{C.5}$$

5. Nombre de clés déplacées pendant l'insertion $EF_n = n(n-1)/4, EP_n = 0$:

$$\bar{E}_n = \left| \begin{array}{ll} t=0 & (n+1)\,\frac{M(M-1)}{6(M+2)} \\ t=1 & (n+1)\,\frac{(2M+1)(3M-4)}{35(M+2)} - \frac{3M^2+15M-2}{140}\frac{\binom{M+1}{6}}{\binom{n}{6}} \\ t=2 & (n+1)\,\frac{45M^2-34M-38}{259(M+2)} \\ & +\frac{1}{259}\Re\left((8M^2+40M-1+\imath\frac{59+230M+46M^2}{\sqrt{71}})B_2^{n,M}\right) \\ t=3 & (n+1)\,\frac{280M^2-199M-258}{1599(M+2)} - \frac{5(3+20M+4M^2)}{1599}\frac{\binom{M+1}{12}}{\binom{n}{12}} \\ & +\frac{1}{1476}\Re\left((35M^2+175M-6+\imath\frac{3(258+875M+175M^2)}{\sqrt{159}})B_3^{n,M}\right) \end{array} \right. \tag{C.6}$$

2. Coûts moyens asymptotiques avec termes linéaires.

On reprend les quantités définies dans la section précédente dans le cas s-aire avec s quelconque. Les formules suivantes donnent les coûts moyens avec une erreur en $O(n^{b_{s,t}})$ où $b_{s,t}$ est le maximum de $-\Re(\alpha_i)-1$ étendu aux racines de $PQ_{s,t}(x)$ autres que -2 et $0, 1, ..., t-1$; on a toujours $b_{s,t} < 1$ (cf. annexe B et chapitre III).

On introduit la quantité $k_{s,t} = 1/(H_{s(t+1)} - H_{t+1})$ dont les premières valeurs sont tabulées dans l'annexe D p. 138.

1. Nombre de comparaisons entre clés pendant les partitionnements (on ne compte pas la recherche de la médiane) $CP_n = a(n+1) + b$ avec par exemple $a = \bar{a}(s)$ et $b = \bar{a}(s)(2 -$

$s(t+1))$:

$$\bar{C}_n = ak_{s,t}(n+1)\left(H_{n+1} - H_{M+2} + \frac{1}{2k_{s,t}} + k_{s,t}\frac{H^{(2)}_{s(t+1)} - H^{(2)}_{t+1}}{2}\right) + b\bar{A}_n + O(n^{b_{s,t}}). \quad (C.7)$$

2. Nombre d'appels à la procédure de partitionnement :

$$\bar{A}_n = (n+1)\frac{k_{s,t}}{M+2} - \frac{1}{s-1} + O(n^{b_{s,t}}). \quad (C.8)$$

3. Nombre de clés insérées (pour les petites listes) :

$$\bar{D}_n = (n+1)\left(1 + \frac{s}{M+2} - \frac{(s-1)k_{s,t}}{M+2}(H_{M+1} + \frac{2t+1}{t+1})\right) + O(n^{b_{s,t}}). \quad (C.9)$$

4. Nombre de clés déplacées pendant l'insertion :

$$\bar{E}_n = (n+1)\left(k_{s,t}(\frac{s-1}{2(M+2)} + \frac{(M+3)(s-1)(t+1)}{4(s(t+1)+1)}) - 1\right) + O(n^{b_{s,t}}). \quad (C.10)$$

3. Cumulants pour la recherche.

On considère le coût de la recherche dans un arbre s-aire avec médiane de $k = s(t+1) - 1$ éléments. Les cumulants, donnés par le théorème IV.1 p. 66, sont alors de la forme :

$$\kappa_i(n) = KR_i(s,t)\ln n + O(1). \quad (C.11)$$

On pose :

$$\bar{a}(s) = h + 2\frac{s_0}{s} \quad , \quad l(s) = 2\frac{s_0}{s} \quad \text{pour} \quad s = 2^h + s_0 \quad \text{avec} \quad 0 < s_0 \le 2^h \quad (C.12)$$

$$\text{et} \quad D_i = \frac{H^{(i)}_{s(t+1)} - H^{(i)}_{t+1}}{H_{s(t+1)} - H_{t+1}}, \qquad l_1 = \frac{\bar{a}(s)}{H_{s(t+1)} - H_{t+1}}, \qquad l_i = \frac{L_i(l(s), l(s), \ldots, l(s))}{H_{s(t+1)} - H_{t+1}}. \quad (C.13)$$

Si les nœuds s-aires sont des arbres binaires complets, on a pour les cumulants :

$$\begin{aligned}
KR_2 &= l_2 + l_1^2 D_2 \qquad (C.14)\\
KR_3 &= l_3 - 2l_1^3 D_3 + 3l_2 l_1 D_2 + 3l_1^3 D_2^2\\
KR_4 &= l_4 + 6l_1^4 D_4 - 12 l_2 l_1^2 D_3 + (4l_3 l_1 + 3l_2^2) D_2 - 20 l_1^4 D_3 D_2 + 18 l_2 l_1^2 D_2^2 + 15 l_1^4 D_2^3\\
KR_5 &= l_5 - 24 l_1^5 D_5 + 60 l_2 l_1^3 D_4 - 10 l_1(3l_2^2 + 2l_3 l_1) D_3 + 40 l_1^5 D_3^2 + 90 l_1^5 D_4 D_2 - 210 l_1^5 D_2^2 D_3\\
&\quad - 200 l_2 l_1^3 D_3 D_2 + 15 l_1(3l_2^2 + 2l_3 l_1) D_2^2 + 5(l_4 l_1 + 2l_3 l_2) D_2 + 150 l_2 l_1^3 D_2^3 + 105 l_1^5 D_2^4\\
KR_6 &= l_6 + 120 l_1^6 D_6 - 360 l_2 l_1^4 D_5 + 30 l_1^2(4l_3 l_1 + 9l_2^2) D_4 - 30(l_2^3 + 4l_3 l_2 l_1 + l_4 l_1^2) D_3\\
&\quad - 420 l_1^6 D_4 D_3 + 600 l_2 l_1^4 D_3^2 + (15 l_4 l_2 + 10 l_3^2 + 6 l_5 l_1) D_2 - 504 l_1^6 D_5 D_2 + 1575 l_2 l_1^4 D_2^4\\
&\quad - 100 l_1^2(4l_3 l_1 + 9l_2^2) D_3 D_2 + 1120 l_1^6 D_3^2 D_2 + 45(l_2^3 + 4l_3 l_2 l_1 + l_4 l_1^2) D_2^2 + 1260 l_1^6 D_2^2 D_4\\
&\quad - 3150 l_2 l_1^4 D_2^2 D_3 + 75 l_1^2(4l_3 l_1 + 9l_2^2) D_2^3 - 2520 l_1^6 D_3 D_2^3 + 1350 l_2 l_1^4 D_4 D_2 + 945 l_1^6 D_2^5
\end{aligned}$$

Les mêmes formules s'appliquent pour des nœuds quelconques en posant :

$$l_i = \frac{\kappa_i^{\text{nœud}}}{H_{s(t+1)} - H_{t+1}},$$

où les $\kappa_i^{\text{nœud}}$ sont les cumulants de la distribution du coût de traversée de chaque nœud.

Dans le cas où $s = 2^k$ ou si le coût de traversée d'un nœud est fixe (indépendant du nœud ou du fils dans le nœud), on a alors $l_i = 0$ pour $i > 1$; d'où les formules plus simples :

$$\begin{aligned}
KR_2 &= l_1^2 D_2 \qquad (C.15)\\
KR_3 &= l_1^3(3D_2^2 - 2D_3)\\
KR_4 &= l_1^4(6D_4 - 20D_3D_2 + 15D_2^3)\\
KR_5 &= l_1^5(40D_3^2 - 24D_5 + 90D_4D_2 - 210D_2^2D_3 + 105D_2^4)\\
KR_6 &= l_1^6(120D_6 - 420D_4D_3 - 504D_5D_2 + 1120D_3^2D_2 + 1260D_2^2D_4 - 2520D_3D_2^3 + 945D_2^5)\\
KR_7 &= l_1^7(1260D_4^2 + 10395D_2^6 - 720D_7 - 9072D_2^2D_5 - 2240D_3^3 + 3360D_6D_2 + 2688D_5D_3\\
&\quad - 15120D_3D_4D_2 + 25200D_2^2D_3^2 - 34650D_2^4D_3 + 18900D_4D_2^3)\\
KR_8 &= l_1^8(120960D_3D_5D_2 - 20160D_6D_3 + 75600D_2^2D_6 + 50400D_3^2D_4 - 415800D_2^2D_4D_3\\
&\quad - 25920D_2D_7 + 56700D_4^2D_2 - 123200D_3^3D_2 + 554400D_3^2D_2^3 + 311850D_2^4D_4\\
&\quad - 540540D_3D_2^5 - 18144D_5D_4 - 166320D_5D_2^3 + 135135D_2^7 + 5040D_8)
\end{aligned}$$

Les premières valeurs de ces moments sont données par les tableaux de l'annexe D p. 138

4. Cumulants pour le tri.

Les cumulants de la distribution du nombre de comparaisons dans Quicksort avec médiane de k clés et avec nœud s-aire sont de la forme (si $CP(n) = an + b$) :

$$\kappa_i(n) = \frac{(-1)^i a^i}{(H_{s(t+1)} - H_{t+1})^i}\left[A_i - (i-1)!(\zeta(i) - H_{t+1}^{(i)})\right] n^i + O(n^{i-1} + n\ln n + n^{ib_{s,t}}), \quad (C.16)$$

avec des constantes A_i qui ne dépendent que des paramètres s et t.

On pose :

$$k = s(t+1)-1, \qquad u = s(t+1)^2 = (k+1)(t+1), \qquad v = (s+1)(t+1) = (k+1)+(t+1) \quad (C.17)$$

$$\text{et} \quad d_i = H_{s(t+1)}^{(i)} - H_{t+1}^{(i)}, \qquad R_i = (i-1)!\left[\binom{k+i}{i-1} - \binom{t+i}{i-1}\right] \qquad \text{pour } i > 1. \quad (C.18)$$

Les cumulants sont alors caractérisés par les rationnels :

$$\begin{aligned}
A_2 &= \frac{(k+2)d_2}{R_2} \\
A_3 &= \frac{2(k+3)(k+2)d_3}{R_3} - \frac{3d_2}{R_3} \\
A_4 &= \frac{6(k+4)(k+3)(k+2)d_4}{R_4} + \frac{6(t+2)(k+2)d_2^2}{R_4R_2} - \frac{R_2(36d_2+8(9+5v+3u)d_3)}{R_4R_3} \\
A_5 &= \frac{24(k+5)(k+4)(k+3)(k+2)d_5}{R_5} + \frac{120(t+3)(t+2)(k+3)(k+2)d_2d_3}{R_5R_3} \\
&\quad - \frac{40(3v^2+135+97v-3v^3+21vu+87u)R_2^2d_3}{R_4R_3R_5} - \frac{360(13+3v)R_2^2d_2}{R_4R_3R_5} \\
&\quad - \frac{30(125v^2+120+173v+41v^3+5v^4+2uv^3-96u-39vu+uv^2-2vu^2)R_2d_2^2}{R_4R_3R_5} \\
&\quad - \frac{60(60v+65+3u^2+13v^2+32u+13vu)R_2d_4}{R_5R_4} \\
A_6 &= 120(k+6)(k+5)(k+4)(k+3)(k+2)d_6/R_6 \\
&\quad +1080(t+4)(t+3)(t+2)(k+4)(k+3)(k+2)d_2d_4/R_6R_4 \\
&\quad +240(t+3)(t+2)(k+3)(k+2)(15+7v+3u)R_2d_3^2/R_6R_3^2 \\
&\quad +5400(6vu-118v^2+22u-551v-890-9v^3)R_2^3d_2/R_3R_6R_4R_5 \\
&\quad +120(t+2)(k+2)(32+22v^2+4v^3+32v+uv^2-7vu-41u-u^2)d_2^3/R_2R_6R_4 \\
&\quad -288(5u^3+596vu+100uv^2+77v^3+650v^2+1450+850u \\
&\qquad +125u^2+1735v+40vu^2)R_2d_5/R_6R_5 \\
&\quad +1200(6v^5-4050-1927vu+24vu^2+57v^4+55v^3-4345v \\
&\qquad +84u^2-444uv^2-1147v^2-39uv^3-3030u)R_2^3d_3/R_3R_6R_4R_5 \\
&\quad +900(12u^3+3uv^4-18u^2v^2-2250-25uv^3+60v^4+7v^5+68v^3-96vu^2 \\
&\qquad -810v^2-222u^2-2340u-1827vu-2715v-431uv^2)R_2^2d_4/R_6R_4R_5 \\
&\quad +180(15197uv^2+uv^7-493uv^5-2342uv^4+180vu^2-12150-25005v \\
&\qquad -24378v^3-7151v^4-180u^3+515u^2v^3+55980u+2u^3v^3+1510u^2v^2 \\
&\qquad -156u^3v+40u^2v^4-161v^5-3u^2v^5-31uv^6-22u^3v^2+96v^7+7v^8 \\
&\qquad -2403uv^3+54171vu-3330u^2-35610v^2+416v^6)R_2^3d_2^2/R_6R_4R_3^2R_5 \\
&\quad -720(29400-38817uv^2+12uv^7+621uv^5-404uv^4-12312vu^2+76500v \\
&\qquad -9180u^2+97380v^2+75966v^3+38066v^4+12276v^5+1770u^3-33390u \\
&\qquad +16u^4v-17u^3v^3-6u^3v^4+1203u^3v-435u^2v^4-2679u^2v^3 \\
&\qquad -7859u^2v^2-11266uv^3-58607vu+2u^2v^6+4u^4v^2-17u^2v^5 \\
&\qquad +155uv^6+218u^3v^2+282v^7+14v^8+2468v^6)R_2^3d_3d_2/R_6R_4R_3^2R_5
\end{aligned} \tag{C.19}$$

Les premières valeurs de ces cumulants ainsi que les moments centrés correspondant sont tabulées en annexe D, p. 140.

Pour les constantes R_i, on a :

$$\begin{aligned}
R_2 &= (t+1)(s-1) \\
R_3 &= R_2(3+v) \\
R_4 &= R_2(11+v^2-u+6v) \\
R_5 &= R_2(5+v)(v^2-2u+5v+10) \\
R_6 &= R_2(274+225v-85u+u^2+85v^2+15v^3+v^4-30uv-3uv^2)
\end{aligned} \tag{C.20}$$

5. Développement des cumulants pour le tri.

5.1. Cas général

Avec les notations de la section précédente, les premiers cumulants de la distribution du nombre de comparaisons se développe sous la forme :

$$\begin{aligned}
d_1\kappa_1(n) &= n\ln n + O(n) \\
d_1^2\kappa_2(n) &= \left(H_{t+1}^{(2)} - \zeta(2) + A_2\right)(n+1)^2 - d_2\kappa_1(n) + O(n) \\
d_1^3\kappa_3(n) &= \left(2(\zeta(3) - H_{t+1}^{(3)}) - A_3\right)(n+1)^3 + 3(2(k+2)(t+2)d_3 - (2v+3)d_2)\frac{(n+1)^2}{R_3} \\
&\quad -3d_2d_1\kappa_2(n) + (3d_2 - 2d_3)\kappa_1(n) + O(n) \\
d_1^4\kappa_4(n) &= \left(6(H_{t+1}^{(4)} - \zeta(4)) + A_4\right)(n+1)^4 \\
&\quad -12(t+2)(k+2)\left((v^2+3v+2-u)d_2^2 + 3(2v+u+4)d_4R_2\right)\frac{(n+1)^3}{R_4R_3} \\
&\quad +48(22v-2u+17+2v^3+uv^2+uv-u^2+11v^2)d_3\frac{R_2^2(n+1)^3}{R_3^2R_4} \\
&\quad -12(3u+11v+3v^2+21+v^3-uv)d_2\frac{R_2^2(n+1)^3}{R_3^2R_4} \\
&\quad +6(t+2)(k+2)(v^3+3v^2-uv+4v+3u)d_2^2\frac{(n+1)^2}{R_4R_3} \\
&\quad +6(t+2)(k+2)(7v^2+20v+33-18u)d_4\frac{R_2(n+1)^2}{R_4R_3} \\
&\quad -8(9u+66v^3+9u^2+217v^2+342v+9v^4+198+3uv^3-3u^2v)d_3\frac{R_2^2(n+1)^2}{R_4R_3R_3} \\
&\quad +12(76v^2+141v-18u+2v^4-6uv-2uv^2+18v^3+99)d_2\frac{R_2^2(n+1)^2}{R_4R_3R_3} \\
&\quad -6d_2d_1^2\kappa_3(n) - \left(3d_2^2 - 12d_2d_1 + 8d_3d_1\right)\kappa_2(n) - 2(3d_4 - 8d_3 + 6d_2)\kappa_1(n) + O(n)
\end{aligned} \tag{C.21}$$

5.2. Cas standard

Les formules suivantes donnent les développements des premiers cumulants pour le nombre de comparaisons dans le cas du tri binaire sans médiane ($t = 0, s = 2$).

$$\kappa_2(n) = (7 - 4\zeta(2))(n+1)^2 - 2n\ln n + O(n) \tag{C.22}$$

$$\kappa_3(n) = (16\zeta(3) - 19)(n+1)^3 + 12(\zeta(2) - 2)(n+1)^2 + 14n\ln n + O(n)$$

$$\begin{aligned}\kappa_4(n) &= (\frac{937}{9} - 96\zeta(4))(n+1)^4 + 4(\frac{83}{3} - 24\zeta(3))(n+1)^3 \\ &\quad +4(\frac{581}{9} - 31\zeta(2))(n+1)^2 - 154n\ln n + O(n)\end{aligned}$$

$$\begin{aligned}\kappa_5(n) &= (768\zeta(5) - \frac{85981}{108})(n+1)^5 + 5(192\zeta(4) - \frac{3767}{18})(n+1)^4 \\ &\quad +5(272\zeta(3) - \frac{33199}{108})(n+1)^3 + 5(364\zeta(2) - \frac{1555}{2})(n+1)^2 \\ &\quad +2342n\ln n + O(n)\end{aligned}$$

$$\begin{aligned}\kappa_6(n) &= (\frac{21096517}{2700} - 7680\zeta(6))(n+1)^6 + (\frac{715909}{60} - 11520\zeta(5))(n+1)^5 \\ &\quad +37(\frac{283897}{540} - 480\zeta(4))(n+1)^4 + (\frac{1017821}{36} - 25440\zeta(3))(n+1)^3 \\ &\quad +(\frac{33918593}{450} - 34684\zeta(2))(n+1)^2 - 45618n\ln n + O(n)\end{aligned}$$

$$\begin{aligned}\kappa_7(n) &= (92160\zeta(7) - \frac{7527245453}{81000})(n+1)^7 + 7(23040\zeta(6) - \frac{3164887}{135})(n+1)^6 \\ &\quad +7(38400\zeta(5) - \frac{643638773}{16200})(n+1)^5 + 7(58560\zeta(4) - \frac{17401679}{270})(n+1)^4 \\ &\quad +14(42104\zeta(3) - \frac{51909109}{1125})(n+1)^3 + 7(116068\zeta(2) - \frac{57461741}{225})(n+1)^2 \\ &\quad +O(n\ln n)\end{aligned}$$

$$\begin{aligned}\kappa_8(n) &= (\frac{19281922400989}{14883750} - 1290240\zeta(8))(n+1)^8 \\ &\quad +(\frac{368828081011}{141750} - 2580480\zeta(7))(n+1)^7 \\ &\quad +4(\frac{250115717573}{212625} - 1155840\zeta(6))(n+1)^6 \\ &\quad +4(\frac{52123591}{27} - 1868160\zeta(5))(n+1)^5 + \cdots\end{aligned}$$

Annexe D

Tables

Nous donnons dans cette annexe des tables de valeurs, pour les coefficients caractéristiques des moyennes et de moments, en fonction des paramètres s et t. Ces tables sont obtenues par application des formules de l'annexe C et en utilisant les procédures Maple de l'annexe E.

1. Valeurs moyennes des coûts.

Les coûts moyens pour le tri ou la recherche dans un arbre s-aire avec médiane de $s(t+1)-1$ clés sont caractérisés par les constantes :

$$k_{s,t} = \frac{1}{H_{s(t+1)} - H_{t+1}} \quad \text{et} \quad K_{s,t} = \frac{\bar{a}(s)}{H_{s(t+1)} - H_{t+1}}. \tag{D.1}$$

$\bar{a}(s)$ est le coût moyen du parcours d'un nœud de l'arbre considéré comme un arbre binaire optimum (complet) à s feuilles, d'où :

$$\bar{a}(s) = h + 2\frac{s_0}{s} \quad \text{pour} \quad s = 2^h + s_0 \quad \text{avec} \quad 0 < s_0 \leq 2^h. \tag{D.2}$$

La constante $k_{s,t}$ correspond au cas où l'on compte un coût unitaire indépendant de s pour le parcours d'un nœud de l'arbre $\bar{a}(s) = 1$.

s	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\bar{a}(s)$	1	1.66	2	2.4	2.66	2.85	3	3.22	3.4	3.54	3.66	3.77	3.86	3.93

Tableau D.1 : Profondeur moyenne d'un arbre binaire complet.

$k_{s,t}$	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	2	1.714	1.6216	1.5760	1.5489	1.5309	1.5181	1.5086	1.4905
s = 3	1.2	1.052	1.0044	.98051	.96628	.95684	.95012	.94509	.93552
s = 4	.923	.8211	.78748	.77077	.76080	.75417	.74945	.74591	.73918
s = 5	.779	.6998	.67345	.66032	.65248	.64725	.64353	.64074	.63543
s = 6	.689	.6237	.60177	.59080	.58423	.57986	.57674	.57440	.56995
s = 7	.627	.5709	.55187	.54235	.53664	.53284	.53012	.52809	.52421
s = 8	.582	.5317	.51477	.50629	.50120	.49781	.49539	.49358	.49011
s = 9	.546	.5012	.48588	.47819	.47357	.47049	.46829	.46665	.46350

Tableau D.2 : Moyenne : coût unitaire pour les nœuds.

$K_{s,t}$	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	2	1.714	1.6216	1.5760	1.5489	1.5309	1.5181	1.5086	1.4905
s = 3	2	1.754	1.6740	1.6342	1.6105	1.5947	1.5835	1.5751	1.5592
s = 4	1.846	1.642	1.5750	1.5415	1.5216	1.5083	1.4989	1.4918	1.4784
s = 5	1.870	1.679	1.6163	1.5848	1.5659	1.5534	1.5445	1.5378	1.5250
s = 6	1.839	1.663	1.6047	1.5755	1.5579	1.5463	1.5380	1.5317	1.5199
s = 7	1.793	1.631	1.5768	1.5496	1.5333	1.5224	1.5146	1.5088	1.4978
s = 8	1.746	1.595	1.5443	1.5189	1.5036	1.4934	1.4862	1.4807	1.4703
s = 9	1.761	1.615	1.5656	1.5408	1.5260	1.5160	1.5089	1.5036	1.4935

Tableau D.3 : Moyenne : nœud = arbre binaire complet.

2. Cumulants liés à la recherche.

Les cumulants de la longueur de cheminement dans un arbre s-aire de recherche avec médiane sont de la forme :

$$\kappa_i(n) = KR_i(s,t)\ln n + O(1)\,. \tag{D.3}$$

Les valeurs suivantes sont obtenues par application des formules de la section 3. de l'annexe C.

KR_2	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8
s = 2	2	.8746	.55554	.40635	.32013	.26403	.22462	.19544	.17296
s = 3	2	1.016	.72602	.58806	.50752	.45476	.41754	.38987	.36850
s = 4	1.332	.6143	.39822	.29441	.23348	.19343	.16510	.14400	.12768
s = 5	1.450	.7597	.54749	.44464	.38397	.34396	.31559	.29442	.27803
s = 6	1.299	.6821	.48980	.39599	.34046	.30376	.27769	.25823	.24314
s = 7	1.110	.5651	.39319	.30894	.25893	.22581	.20226	.18466	.17101
s = 8	.9363	.4523	.29833	.22257	.17748	.14759	.12631	.11039	.09804
s = 16	.6928	.3479	.23270	.17486	.14006	.11681	...	...	...

Tableau D.4 : Recherche : Variance.

KR_3	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8
s = 2	2	.4291	.18020	.09843	.06188	.04246	.03092	.02352	.01849
s = 3	2	.5385	.25481	.14555	.08967	.05629	.03427	.01875	.00725
s = 4	.8224	.1791	.07548	.04128	.02597	.01783	.01299	.00989	.00777
s = 5	1.084	.3556	.20867	.15058	.12028	.10187	.08958	.08082	.07427
s = 6	.7459	.1985	.08970	.04697	.02475	.01128	.00230	-.00410	-.00889
s = 7	.4653	.0946	.02684	.00166	-.01094	-.01836	-.02319	-.02658	-.02908
s = 8	.2960	.0601	.02431	.01297	.00802	.00544	.00393	.00297	.00233
s = 16	.0594	.0034	-.00061	-.00100	-.00090	-.00075	...	...	...

Tableau D.5 : Recherche : Cumulant d'ordre 3.

KR_4	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8
s = 2	2	.2178	.06170	.02545	.01285	.00737	.00461	.00307	.00215
s = 3	2	.2043	.00522	-.04599	-.06424	-.07188	-.07532	-.07689	-.07753
s = 4	.5861	.0716	.02142	.00910	.00468	.00272	.00171	.00115	.00081
s = 5	.9174	.1379	.02618	-.01045	-.02730	-.03665	-.04249	-.04644	-.04927
s = 6	.4438	.0071	-.03887	-.04872	-.05112	-.05144	-.05110	-.05056	-.04998
s = 7	.2605	.0205	.00112	-.00049	.00069	.00225	.00370	.00496	.00603
s = 8	.2061	.0328	.01092	.00490	.00260	.00154	.00099	.00067	.00047
s = 16	.1294	.0269	.00967	.00449	.00243	.00146	...	...	...

Tableau D.6 : Recherche : Cumulant d'ordre 4.

KR_5	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8
s = 2	2	.108	.02007	.00612	.00245	.00116	.00062	.00036	.00022
s = 3	2	.090	.02175	.02920	.04135	.05157	.05961	.06594	.07100
s = 4	.384	.016	.00230	.00054	.00017	$.66\,10^{-4}$	$.29\,10^{-4}$	$.14\,10^{-4}$	$.78\,10^{-5}$
s = 5	.615	-.065	-.09839	-.09757	-.09314	-.08885	-.08521	-.08221	-.07972
s = 6	.226	.007	.01953	.03046	.03781	.04290	.04660	.04940	.05158
s = 7	.131	.031	.03018	.02910	.02808	.02723	.02654	.02599	.02553
s = 8	.010	-.010	-.00344	-.00137	-.00064	-.00033	-.00019	-.00011	$-.73\,10^{-4}$
s = 16	-.121	-.023	-.00653	-.00245	-.00111	-.00057	...	...	...

Tableau D.7 : Recherche : Cumulant d'ordre 5.

3. Cumulants liés au tri.

Les cumulants du nombre de comparaisons dans Quicksort avec médiane et nœuds s-aires sont de la forme :

$$\kappa_i(n) = KQ_i(s,t)\, n^i + O(n^{i-1}) \quad \text{pour} \quad i > 2\,. \tag{D.4}$$

Les tables suivantes sont obtenues à partir des formules de la section 4. de l'annexe C.

KQ_2	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	.4202	.1148	.05301	.03043	.01972	.01382	.01021	.00786	.00419
s = 3	.3091	.0846	.03906	.02242	.01453	.01018	.00753	.00579	.00309
s = 4	.2081	.0571	.02642	.01517	.00984	.00689	.00510	.00392	.00209
s = 5	.1765	.0486	.02251	.01294	.00839	.00588	.00435	.00335	.00179
s = 6	.1454	.0402	.01863	.01072	.00695	.00488	.00361	.00278	.00148
s = 7	.1205	.0334	.01550	.00892	.00579	.00406	.00301	.00231	.00124
s = 8	.1011	.0281	.01306	.00752	.00489	.00343	.00254	.00195	.00104
s = 9	.0924	.0257	.01197	.00690	.00448	.00314	.00233	.00179	.00096

Tableau D.8 : Tri : Variance.

KQ_3	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	.2329	.0378	.01276	.00582	.00313	.00188	.00122	.00083	.00033
s = 3	.1482	.0236	.00785	.00354	.00189	.00113	.00073	.00050	.00020
s = 4	.0812	.0127	.00419	.00188	.00100	.00059	.00038	.00026	.00010
s = 5	.0623	.0096	.00315	.00141	.00075	.00044	.00028	.00019	$.76\,10^{-4}$
s = 6	.0456	.0070	.00227	.00101	.00054	.00032	.00020	.00014	$.54\,10^{-4}$
s = 7	.0336	.0051	.00166	.00073	.00039	.00023	.00015	$.99\,10^{-4}$	$.39\,10^{-4}$
s = 8	.0253	.0038	.00123	.00055	.00029	.00017	.00011	$.74\,10^{-4}$	$.29\,10^{-4}$
s = 9	.0215	.0032	.00104	.00046	.00024	.00014	$.91\,10^{-4}$	$.62\,10^{-4}$	$.24\,10^{-4}$

Tableau D.9 : Tri : Cumulant d'ordre 3 = Moment centré d'ordre 3.

KQ_4	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	.2080	.02011	.00506	.00187	.00085	.00044	.00025	.00016	$.47\,10^{-4}$
s = 3	.1176	.01105	.00268	.00096	.00043	.00022	.00012	$.75\,10^{-4}$	$.22\,10^{-4}$
s = 4	.0540	.00488	.00115	.00040	.00018	$.89\,10^{-4}$	$.50\,10^{-4}$	$.30\,10^{-4}$	$.89\,10^{-5}$
s = 5	.0385	.00336	.00077	.00027	.00012	$.58\,10^{-4}$	$.32\,10^{-4}$	$.19\,10^{-4}$	$.56\,10^{-5}$
s = 6	.0256	.00217	.00049	.00017	$.71\,10^{-4}$	$.35\,10^{-4}$	$.19\,10^{-4}$	$.11\,10^{-4}$	$.34\,10^{-5}$
s = 7	.0171	.00141	.00031	.00011	$.45\,10^{-4}$	$.22\,10^{-4}$	$.12\,10^{-4}$	$.74\,10^{-5}$	$.21\,10^{-5}$
s = 8	.0116	.00094	.00021	$.69\,10^{-4}$	$.29\,10^{-4}$	$.14\,10^{-4}$	$.80\,10^{-5}$	$.48\,10^{-5}$	$.13\,10^{-5}$
s = 9	.0094	.00074	.00016	$.53\,10^{-4}$	$.22\,10^{-4}$	$.11\,10^{-4}$	$.62\,10^{-5}$	$.37\,10^{-5}$	$.10\,10^{-5}$

Tableau D.10 : Tri : Cumulant d'ordre 4.

KQ_5	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	.2401	.01349	.00254	.00077	.00030	.00014	$.69\,10^{-4}$	$.38\,10^{-4}$	$.92\,10^{-5}$
s = 3	.1214	.00678	.00122	.00035	.00013	$.57\,10^{-4}$	$.28\,10^{-4}$	$.15\,10^{-4}$	$.35\,10^{-5}$
s = 4	.0475	.00255	.00043	.00012	$.43\,10^{-4}$	$.18\,10^{-4}$	$.91\,10^{-5}$	$.48\,10^{-5}$	$.10\,10^{-5}$
s = 5	.0322	.00164	.00027	$.71\,10^{-4}$	$.25\,10^{-4}$	$.10\,10^{-4}$	$.51\,10^{-5}$	$.27\,10^{-5}$	$.59\,10^{-6}$
s = 6	.0198	.00096	.00015	$.39\,10^{-4}$	$.13\,10^{-4}$	$.57\,10^{-5}$	$.27\,10^{-5}$	$.14\,10^{-5}$	$.30\,10^{-6}$
s = 7	.0122	.00056	$.86\,10^{-4}$	$.22\,10^{-4}$	$.76\,10^{-5}$	$.31\,10^{-5}$	$.15\,10^{-5}$	$.78\,10^{-6}$	$.16\,10^{-6}$
s = 8	.0077	.00034	$.51\,10^{-4}$	$.12\,10^{-4}$	$.44\,10^{-5}$	$.18\,10^{-5}$	$.85\,10^{-6}$	$.44\,10^{-6}$	$.92\,10^{-7}$
s = 9	.0059	.00025	$.37\,10^{-4}$	$.93\,10^{-5}$	$.31\,10^{-5}$	$.12\,10^{-5}$	$.60\,10^{-6}$	$.31\,10^{-6}$	$.65\,10^{-7}$

Tableau D.11 : Tri : Cumulant d'ordre 5.

KQ_6	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	.3301	.01051	.00147	.00036	.00012	$.48\,10^{-4}$	$.22\,10^{-4}$	$.11\,10^{-4}$	$.21\,10^{-5}$
s = 3	.1472	.00491	.00066	.00015	$.48\,10^{-4}$	$.18\,10^{-4}$	$.82\,10^{-5}$	$.40\,10^{-5}$	$.69\,10^{-6}$
s = 4	.0492	.00162	.00020	$.45\,10^{-4}$	$.13\,10^{-4}$	$.50\,10^{-5}$	$.21\,10^{-5}$	$.10\,10^{-5}$	$.16\,10^{-6}$
s = 5	.0320	.00100	.00012	$.24\,10^{-4}$	$.72\,10^{-5}$	$.25\,10^{-5}$	$.10\,10^{-5}$	$.50\,10^{-6}$	$.79\,10^{-7}$
s = 6	.0185	.00054	$.61\,10^{-4}$	$.12\,10^{-4}$	$.34\,10^{-5}$	$.12\,10^{-5}$	$.50\,10^{-6}$	$.23\,10^{-6}$	$.35\,10^{-7}$
s = 7	.0107	.00029	$.31\,10^{-4}$	$.61\,10^{-5}$	$.17\,10^{-5}$	$.59\,10^{-6}$	$.24\,10^{-6}$	$.11\,10^{-6}$	$.16\,10^{-7}$
s = 8	.0063	.00016	$.16\,10^{-4}$	$.32\,10^{-5}$	$.88\,10^{-6}$	$.30\,10^{-6}$	$.12\,10^{-6}$	$.54\,10^{-7}$	$.82\,10^{-8}$
s = 9	.0047	.00011	$.11\,10^{-4}$	$.21\,10^{-5}$	$.58\,10^{-6}$	$.19\,10^{-6}$	$.79\,10^{-7}$	$.35\,10^{-7}$	$.53\,10^{-8}$

Tableau D.12 : Tri : Cumulant d'ordre 6.

4. Moments centrés pour le tri.

Les moments centrés correspondants à la distribution précédente sont de la forme :

$$\text{Moment centré}_i = MC_i(s,t)\, n^i + O(n^{i-1}) \qquad \text{(D.5)}$$

et sont obtenues en appliquant les polynômes de Bell complet sur l'équation D.4 page 140.

MC_4	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	.7379	.0597	.01349	.00465	.00202	.00102	.00057	.00034	.00010
s = 3	.4044	.0325	.00726	.00247	.00106	.00053	.00029	.00018	$.51\,10^{-4}$
s = 4	.1840	.0146	.00324	.00109	.00047	.00023	.00013	$.76\,10^{-4}$	$.22\,10^{-4}$
s = 5	.1320	.0104	.00229	.00077	.00033	.00016	$.89\,10^{-4}$	$.53\,10^{-4}$	$.15\,10^{-4}$
s = 6	.0891	.0070	.00153	.00051	.00022	.00011	$.58\,10^{-4}$	$.35\,10^{-4}$	$.10\,10^{-4}$
s = 7	.0607	.0047	.00103	.00034	.00015	$.72\,10^{-4}$	$.39\,10^{-4}$	$.23\,10^{-4}$	$.67\,10^{-5}$
s = 8	.0424	.0033	.00072	.00024	.00010	$.49\,10^{-4}$	$.27\,10^{-4}$	$.16\,10^{-4}$	$.46\,10^{-5}$
s = 9	.0350	.0027	.00059	.00020	$.83\,10^{-4}$	$.40\,10^{-4}$	$.22\,10^{-4}$	$.13\,10^{-4}$	$.38\,10^{-5}$

Tableau D.13 : Tri : Moment centré d'ordre 4.

MC_5	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	1.21	.0569	.00931	.00254	.00092	.00040	.00019	.00010	$.23\,10^{-4}$
s = 3	.579	.0267	.00428	.00114	.00041	.00017	$.83\,10^{-4}$	$.44\,10^{-4}$	$.96\,10^{-5}$
s = 4	.216	.0098	.00154	.00040	.00014	$.59\,10^{-4}$	$.28\,10^{-4}$	$.15\,10^{-4}$	$.32\,10^{-5}$
s = 5	.142	.0063	.00098	.00025	$.88\,10^{-4}$	$.36\,10^{-4}$	$.17\,10^{-4}$	$.92\,10^{-5}$	$.19\,10^{-5}$
s = 6	.086	.0037	.00057	.00015	$.51\,10^{-4}$	$.21\,10^{-4}$	$.10\,10^{-4}$	$.52\,10^{-5}$	$.11\,10^{-5}$
s = 7	.052	.0022	.00034	$.87\,10^{-4}$	$.30\,10^{-4}$	$.12\,10^{-4}$	$.59\,10^{-5}$	$.30\,10^{-5}$	$.65\,10^{-6}$
s = 8	.033	.0014	.00021	$.53\,10^{-4}$	$.18\,10^{-4}$	$.76\,10^{-5}$	$.36\,10^{-5}$	$.18\,10^{-5}$	$.39\,10^{-6}$
s = 9	.025	.0010	.00016	$.41\,10^{-4}$	$.14\,10^{-4}$	$.58\,10^{-5}$	$.27\,10^{-5}$	$.14\,10^{-5}$	$.30\,10^{-6}$

Tableau D.14 : Tri : Moment centré d'ordre 5.

MC_6	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	3.29	.08222	.00936	.00198	.00059	.00022	$.92\,10^{-4}$	$.43\,10^{-4}$	$.73\,10^{-5}$
s = 3	1.35	.03361	.00374	.00077	.00022	$.80\,10^{-4}$	$.33\,10^{-4}$	$.15\,10^{-4}$	$.25\,10^{-5}$
s = 4	.419	.01023	.00111	.00022	$.63\,10^{-4}$	$.22\,10^{-4}$	$.94\,10^{-5}$	$.43\,10^{-5}$	$.69\,10^{-6}$
s = 5	.255	.00611	.00065	.00013	$.36\,10^{-4}$	$.12\,10^{-4}$	$.52\,10^{-5}$	$.24\,10^{-5}$	$.37\,10^{-6}$
s = 6	.141	.00331	.00035	$.67\,10^{-4}$	$.18\,10^{-4}$	$.66\,10^{-5}$	$.26\,10^{-5}$	$.12\,10^{-5}$	$.19\,10^{-6}$
s = 7	.079	.00182	.00019	$.36\,10^{-4}$	$.10\,10^{-4}$	$.35\,10^{-5}$	$.14\,10^{-5}$	$.65\,10^{-6}$	$.10\,10^{-6}$
s = 8	.046	.00104	.00011	$.20\,10^{-4}$	$.56\,10^{-5}$	$.19\,10^{-5}$	$.79\,10^{-6}$	$.36\,10^{-6}$	$.55\,10^{-7}$
s = 9	.034	.00076	.00007	$.14\,10^{-4}$	$.40\,10^{-5}$	$.14\,10^{-5}$	$.57\,10^{-6}$	$.26\,10^{-6}$	$.39\,10^{-7}$

Tableau D.15 : Tri : Moment centré d'ordre 6.

5. Distribution limite pour le tri.

La distribution normée (variance = 1) du nombre de comparaisons converge vers une loi limite caractérisée par les cumulants et les moments centrés suivants :

$$\overline{\kappa}_i = \frac{KQ_i(s,t)}{KQ_2(s,t)^{i/2}} \qquad \overline{M}_i = \frac{MC_i(s,t)}{MC_2(s,t)^{i/2}}\,. \tag{D.6}$$

$\overline{\kappa}_3$	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	.85488	.97144	1.0457	1.0955	1.1310	1.1573	1.1777	1.1938	1.2270
s = 3	.86248	.95958	1.0173	1.0548	1.0810	1.1003	1.1150	1.1267	1.1504
s = 4	.85513	.93166	.97590	1.0044	1.0242	1.0387	1.0497	1.0584	1.0762
s = 5	.84078	.89965	.93368	.95566	.97095	.98217	.99074	.99750	1.0112
s = 6	.82331	.86771	.89404	.91126	.92332	.93221	.93902	.94440	.95537
s = 7	.80467	.83737	.85779	.87144	.88111	.88829	.89381	.89818	.90713
s = 8	.78589	.80909	.82491	.83583	.84368	.84955	.85410	.85771	.86515
s = 9	.76748	.78294	.79512	.80389	.81032	.81519	.81898	.82201	.82828

Tableau D.16 : Loi limite : Cumulant d'ordre 3 = Moment centré d'ordre 3.

$\overline{\kappa}_4$	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	1.1781	1.5238	1.8022	2.0177	2.1860	2.3199	2.4286	2.5183	2.7123
s = 3	1.2314	1.5423	1.7584	1.9127	2.0272	2.1152	2.1848	2.2412	2.3600
s = 4	1.2462	1.4937	1.6482	1.7529	1.8284	1.8853	1.9297	1.9653	2.0393
s = 5	1.2363	1.4200	1.5259	1.5955	1.6449	1.6818	1.7104	1.7332	1.7804
s = 6	1.2115	1.3396	1.4094	1.4546	1.4866	1.5104	1.5288	1.5436	1.5741
s = 7	1.1785	1.2607	1.3039	1.3320	1.3521	1.3671	1.3789	1.3883	1.4079
s = 8	1.1413	1.1865	1.2100	1.2260	1.2378	1.2469	1.2540	1.2599	1.2722
s = 9	1.1024	1.1181	1.1270	1.1343	1.1402	1.1450	1.1490	1.1523	1.1596

Tableau D.17 : Loi limite : Cumulant d'ordre 4.

$\overline{\kappa}_5$	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	2.0973	3.0161	3.9284	4.7483	5.4605	6.0734	6.6012	7.0578	8.1105
s = 3	2.2848	3.2520	4.0366	4.6485	5.1300	5.5158	5.8306	6.0918	6.6609
s = 4	2.4040	3.2580	3.8325	4.2330	4.5261	4.7495	4.9252	5.0670	5.3648
s = 5	2.4584	3.1370	3.5217	3.7656	3.9338	4.0570	4.1512	4.2256	4.3776
s = 6	2.4612	2.9588	3.1961	3.3327	3.4215	3.4839	3.5303	3.5662	3.6375
s = 7	2.4270	2.7622	2.8909	2.9556	2.9939	3.0191	3.0369	3.0502	3.0753
s = 8	2.3682	2.5666	2.6171	2.6336	2.6396	2.6417	2.6421	2.6419	2.6399
s = 9	2.2943	2.3813	2.3758	2.3601	2.3453	2.3329	2.3226	2.3141	2.2961

Tableau D.18 : Loi limite : Cumulant d'ordre 5.

$\overline{\kappa}_6$	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	4.4471	6.9280	9.8395	12.866	15.811	18.577	21.127	23.45	29.25
s = 3	4.9824	8.1021	11.131	13.744	15.934	17.764	19.305	20.61	23.55
s = 4	5.4551	8.6450	11.107	12.904	14.239	15.260	16.062	16.70	18.05
s = 5	5.8134	8.6512	10.397	11.492	12.220	12.732	13.109	13.39	13.94
s = 6	6.0330	8.3262	9.4450	10.037	10.381	10.595	10.736	10.83	10.98
s = 7	6.1218	7.8342	8.4671	8.7237	8.8318	8.8740	8.8847	8.881	8.827
s = 8	6.1033	7.2779	7.5550	7.5927	7.5584	7.5028	7.4434	7.385	7.237
s = 9	6.0043	6.7153	6.7392	6.6366	6.5170	6.4069	6.3048	6.227	6.028

Tableau D.19 : Loi limite : Cumulant d'ordre 6.

$\overline{M}_4$	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	4.1781	4.5238	4.8022	5.0177	5.1860	5.3199	5.4286	5.5183	5.7123
s = 3	4.2314	4.5423	4.7584	4.9127	5.0272	5.1152	5.1848	5.2412	5.3600
s = 4	4.2462	4.4937	4.6482	4.7529	4.8284	4.8853	4.9297	4.9653	5.0393
s = 5	4.2363	4.4200	4.5259	4.5955	4.6449	4.6818	4.7104	4.7332	4.7804
s = 6	4.2115	4.3396	4.4094	4.4546	4.4866	4.5104	4.5288	4.5436	4.5741
s = 7	4.1785	4.2607	4.3039	4.3320	4.3521	4.3671	4.3789	4.3883	4.4079
s = 8	4.1413	4.1865	4.2100	4.2260	4.2378	4.2469	4.2540	4.2599	4.2722
s = 9	4.1024	4.1181	4.1270	4.1343	4.1402	4.1450	4.1490	4.1523	4.1596

Tableau D.20 : Loi limite : Moment centré d'ordre 4.

$\overline{M}_5$	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	10.646	12.730	14.385	15.703	16.770	17.646	18.377	18.996	20.380
s = 3	10.909	12.847	14.209	15.196	15.940	16.518	16.981	17.358	18.164
s = 4	10.955	12.574	13.591	14.276	14.767	15.136	15.422	15.651	16.126
s = 5	10.866	12.133	12.858	13.322	13.643	13.878	14.058	14.200	14.490
s = 6	10.694	11.636	12.136	12.445	12.654	12.806	12.920	13.010	13.191
s = 7	10.473	11.135	11.468	11.670	11.805	11.901	11.974	12.032	12.146
s = 8	10.227	10.657	10.866	10.991	11.076	11.137	11.183	11.219	11.291
s = 9	9.9691	10.210	10.327	10.399	10.448	10.484	10.512	10.534	10.578

Tableau D.21 : Loi limite : Moment centré d'ordre 5.

$\overline{M}_6$	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 10
s = 2	44.427	54.222	62.807	70.133	76.392	81.770	86.425	90.483	99.994
s = 3	45.892	55.443	62.855	68.561	73.028	76.599	79.510	81.924	87.185
s = 4	46.461	54.730	60.354	64.286	67.154	69.328	71.027	72.390	75.226
s = 5	46.426	53.044	57.003	59.557	61.322	62.606	63.580	64.344	65.881
s = 6	45.984	50.949	53.578	55.160	56.205	56.941	57.486	57.905	58.721
s = 7	45.274	48.756	50.383	51.297	51.876	52.271	52.557	52.775	53.188
s = 8	44.398	46.621	47.510	47.969	48.243	48.423	48.549	48.641	48.786
s = 9	43.430	44.616	44.966	45.113	45.186	45.228	45.252	45.267	45.302

Tableau D.22 : Loi limite : Moment centré d'ordre 6.

6. Moments pour médiane de 1,3,5 et 7 avec s=2.

Les valeurs suivantes sont obtenues pour chaque valeur de t à partir de la récurrence caractéristique pour la partie rationnelle des cumulants dans le cas binaire (cf. proposition IV.11 page 84).

KQ_i	cas standard	médiane de 3	médiane de 5	médiane de 7
2	.420263732607094	.1148876402827631	.0530100019574274	.030429803448537
3	.232910450553508	.0378289329044446	.0127627396962395	.005815346286146
4	.208080674841844	.0201135171157979	.0050643954576087	.001868320500827
5	.240145579737733	.0134935784789169	.0025416432283785	.000766981717687
6	.330098774245421	.0105057559683607	.0014656975947450	.000362542181298
7	.513193641464511	.0090960831672502	.0009250439759239	.000186678615026
8	.862978989608401	.0084513236615152	.0006164582215785	.000100504849489
9	1.49525017300110	.0080419449156541	.0004152570932891	.00005395668155
10	2.47498449000397	.0071808926906793	.0002590315840830	.00002632848241
11	3.17769375482215	.0045149889480990	.0001078666104054	.000008102364859
12	-.859132378280702	-.0029146042071731	-.000072456450440	-.000005489543896
13	-31.6125643877558	-.0215295458707870	-.0003186899539072	-.00001697320397
14	-183.289129169014	-.0645798238383224	-.0006720706087864	-.00002760023177
15	-822.661659613760	-.1567119865016804	-.0011675171458365	-.00003753889249
16	-3220.12148521433	-.3333504415080302	-.0017878744745589	-.00004544149804
17	-10848.7043486094	-.6010985328395709	-.0023246111253218	-.00004720012759
18	-26551.0458277133	-.7195730897782686	-.0020028083659631	-.00003339961002
19	8467.14554077612	.7060458526164791	.0014500894342344	.00001531322307
20	750153.038321758	9.525961023502868	.0139421982446374	.000135729222582
21	7194352.01360198	46.87172292561307	.0494271509128392	.000391842969298
22	51549716.2430050	180.5874160383280	.137309630594457	.000880007873573
23	315768693.706688	598.5492312418584	.328256934144499	.001701512105335
24	1663571898.94808	1684.586785174802	.666726303802823	.002812601058995
25	6821174892.18786	3460.221774968233	.990217887292621	.003495654906896
26	9980889525.59750	96.25873076741410	.0035138492476297	.000786074915988
27	-205306829117.961	-53452.51564274453	-7.87249575278135	-.014712177921717
28	-3310472445449.60	-409406.9169159608	-43.6290407479761	-.068825048778741
29	-34751594160427.7	-2265846.357658921	-174.445734513367	-.224430648052438
30	-302105458617234.	-10558706.13470346	-587.292294900249	-.612644035937462
31	-2257655323367641.	-41999402.99598232	-1688.93646907473	-1.43287766290185
32	$-.13829270542\,10^{17}$	-131111456.0269604	-3821.60534423792	-2.68518115684022
33	$-.52236951002\,10^{17}$	-179869181.6557800	-3891.15013702369	-2.65347321438346
34	$.24399179384\,10^{18}$	1672159281.172277	24609.0832648919	8.34476586892890
35	$.86376885465\,10^{19}$	21012610782.40290	226197.236060000	69.8886636149123
36	$.13009591639\,10^{21}$	160336689059.4511	1250362.38421335	318.946501492722
37	$.15294165137\,10^{22}$	995288497185.7464	5615244.84084379	1166.97691191065
38	$.15298650210\,10^{23}$	5267522277587.947	21504525.3477741	3641.83804299439
39	$.12898359620\,10^{24}$	22847414404636.04	67619493.6218510	9446.79815637732
40	$.80633838008\,10^{24}$	63443403994571.21	137549095.185462	16938.4414603502

Tableau D.23 : 40 premiers cumulants pour médiane de 1,3,5,7.

MC_i	cas standard	médiane de 3	médiane de 5	médiane de 7
2	.4202637326070943	.1148876402827631	.0530100019574274	.0304298034485370
3	.2329104505535086	.0378289329044446	.0127627396962395	.0058153462861464
4	.7379454896763864	.0597110267850226	.0134945763801881	.0046462393145775
5	1.218983732865909	.0569543468369831	.0093071717911764	.0025365801624135
6	3.297717794700013	.0822242776384209	.0093559465731371	.0019761717073546
7	8.648240250323654	.1207093912726384	.0097823985364617	.0016224819452917
8	26.77181315131407	.2050693987118127	.0117415910314550	.0015228257278308
9	89.03128054445473	.3781462506624708	.0153682242121401	.0015626752066090
10	324.5638026641834	.7622607864609576	.0219764213368562	.0017518577443934
11	1273.515546249036	1.657173553789679	.0339305567894081	.0021220922197902
12	5361.346948084962	3.865779627328726	.0562343488121827	.0027605249205363
13	24057.26040972797	9.617651333669289	.0994471335506235	.0038334705096548
14	114554.5306092381	25.40016469965256	.186759435960554	.0056552135836787
15	576456.5706978151	70.91413558467065	.370893888972571	.0088251093345086
16	3054790.711813408	208.5454232784044	.776097232218971	.0145147548989886
17	16994217.72188542	643.9740637205663	1.70567115939765	.0250794373103403
18	98975559.22502801	2082.200306779814	3.92609307906757	.0453948816299973
19	601994276.2225897	7032.035065621876	9.44101728538404	.0858568258581934
20	3815351815.359896	24749.89296198296	23.6640787070314	.1692908497288322
21	25147121175.67131	90599.07969485325	61.7005433625643	.3472896492843908
22	172055079694.4059	344301.1901392616	167.038168666385	.7398477306666699
23	1219989483679.547	1356108.530605186	468.748894129536	1.633991456438462
24	8951559709968.285	5527518.474596890	1361.43073399396	3.735442143791078
25	67872439832406.03	23282981.27510348	4086.66794343598	8.826779256247851
26	531111311406006.1	101218564.5324638	12661.8661506565	21.53091902634599
27	4284123733167363.	453605039.7445474	40444.5050779941	54.14996690140929
28	$.35583371664\,10^{17}$	2093207622.224706	133036.792388553	140.2564709425455
29	$.30401739112\,10^{18}$	9936143241.622512	450179.296896265	373.7531433344593
30	$.26693378611\,10^{19}$	48470682470.63773	1565605.97757259	1023.677248998209
31	$.24064508823\,10^{20}$	242777336135.9095	5590773.37091409	2879.150200830236
32	$.22256546434\,10^{21}$	1247502792550.024	20482736.4619382	8308.461854069570
33	$.21101193846\,10^{22}$	6571104346934.247	76928554.9211787	24580.27571007361
34	$.20493026927\,10^{23}$	35455052303576.47	295970282.924850	74496.90174341810
35	$.20372992196\,10^{24}$	195820847926572.4	1165643640.91192	231137.0079103208
36	$.20719049416\,10^{25}$	1106358556228228.	4696280925.60020	733655.8473118233
37	$.21541865147\,10^{26}$	6390279210447704.	19343814287.5202	2380864.097969246
38	$.22884513030\,10^{27}$	$.37711682372\,10^{17}$	81409229918.3079	7894751.821704122
39	$.24825872268\,10^{28}$	$.22725976670\,10^{18}$	349869231069.963	26733745.63892919
40	$.27488025703\,10^{29}$	$.13977531311\,10^{19}$	1534644777813.22	92398908.14401525

Tableau D.24 : 40 premiers moments centrés pour médiane de 1,3,5,7.

Annexe E

Sources MAPLE.

Nous allons donner dans cette annexe quelques procédures en MAPLE qui permettent le calcul des cumulants associés au tri Quicksort ou à la recherche dans les arbres réduits. Les résultats obtenus correspondent aux formules données dans les sections 3., 4. et 5. de l'annexe C et aux tables de valeurs de la section D.6. p. 145. De façon générale, on se reportera à l'annexe A pour valider un certain nombre des formules utilisées.

Nous donnons aussi à titre purement indicatif des temps d'exécution CPU qui correspondent à l'utilisation de MAPLE 4.1 sur un VAX8600.

1. Cumulants du nombre de comparaisons par intégration.

1.1. Terme Principal.

Cette partie contient les procédures de base pour la détermination de proche en proche des moments par intégration des équations différentielles caractéristiques en Θ. L'intégration et la dérivation s'effectuent conformément aux règles données par les propositions III.4 p. 42 et III.6 p. 44. On trouve aussi le calcul des valeurs particulières des dérivées du polynôme $PQ_{s,t}(x)$ données par le lemme B.9 p. 128 et la détermination des coefficients généraux des séries $\ln^p(1-z)/(1-z)^q$ obtenus à partir des équations de la section 7. de l'annexe A.

Les procédures suivantes correspondent plus particulièrement au calcul du terme dominant des cumulants du nombre de comparaisons en fonction des paramètres s et t. Le principe du calcul est décrit dans la section IV.3.3. et donne les résultats de la section C.4..

```
#    Polynômes logarithmiques et exponentiels (Bell) :
#      génére YP[i], LP[i] en fonction des variables x(k) pour i <= ord.
bellpoly:=proc (ord) local i;
  bell:=proc (n, k) options remember;
    if k=1 then x(n)
    else sum('binomial(n, l)*x(n-l)*bell(l, k-1)/k', 'l'=k-1..n-1)
    fi;
  end:
  for i to ord do
    YP[i]:=expand(sum('bell(i, k)', 'k'=1..i));
    LP[i]:=expand(sum('(-1)^(k-1)*(k-1)!*bell(i, k)', 'k'=1..i))
```

```
  od;
  YP[0]:=1; LP[0]:=1; bell:='bell'
end:

#    Nombres harmoniques d'ordre j : H_n^(j).
h:=proc (j, n)  options remember;   sum('1/p^j', 'p'=1..n) end:

#    Coef. gén. de séries : n1 = n + 1, z.i = H_n1^(i), z.i(-k) = H_(n1-k)^(i),
#       [x^n] ln^p(1/(1 - x))/(1 - x)^q = coefexact(p, q) = coefdom(p, q)n^(q-1)(1 + o(1)).

yh:=proc(k, i) option remember;
  eval(subs(x=proc(r) -h(r, n)*(-1)^r*(r-1)! end, n=i, YP[k]))
end:

yy:=proc(k) option remember;
  eval(subs(x=proc(r) -z.r*(-1)^r*(r-1)! end, YP[k]))
end:

zz:=proc(j, i) option remember;
  if j<0 then 0
  elif i=1 then   eval(subs(x=proc(r) -z.r(-1)*(-1)^r*(r-1)! end, YP[j]))
  elif i=2 then n1*yy(j)
  else ((n1+i-2)*zz(j, i-1)+j*zz(j-1, i-1))/(i-1)
  fi
end:

coefexact:=proc(j, i) option remember;
  if i=0 then j/n*eval(subs(x=proc(r) -z.r(-2)*(-1)^r*(r-1)! end, YP[j-1]))
  else sum('yh(k,i-1)*binomial(j,k)*coefexact(j-k,i)', 'k'=1..min(j,i-1));
  collect(zz(j,i)-",n1)
  fi
end:

coefdom:=proc (j, i)  options remember;
  if i=0 then j/n1*yy(j-1)
  else sum('yh(k,i-1)*binomial(j,k)*coefdom(j-k,i)', 'k'=1..min(j,i-1));
   expand(yy(j)/(i-1)!-")
  fi
end:

#    Valeurs des dérivées des polynômes d'intégration et de dérivation :
#         t1 = t + 1, h.i = H_t1^(i), d.i = H_st1^(i) - h.i,
#         derp1(r, i) = binomial(x, t)^(i) en x = r,
#         dpv(r, i) = P\Q(x)^(i), invp(r, i) = (1/P\Q(x))^(i) en x = r,
#         sauf dpv(2, i) = (P\Q(x)/(x - 2))^(i), invp(2, i) = ((x - 2)/P\Q(x))^(i) en x = r = 2.
```

```
`diff/polyn`:=proc(u, i, u) polyn(u, i+1) end:

derinv:=proc (i) options remember;
 if i=0 then 1/polyn(u, 0) else diff(derinv(i−1), u) fi
end:

hh:=proc(i, r) options remember;
  if r=1 then 1  else h.i+sum('1/(t1+k)^i', 'k'=1..r−2)−h(i, r−1)  fi
end:

dpv:=proc(r, i) option remember;
  if r=2 then (s*derp1(2, i+1)−subs(SUB, derp1(2, i+1)))/(i+1)
  else s*derp1(r, i)−subs(SUB, derp1(r, i))
  fi;
  expand("/s/t1)*s*t1;
end:

derp1:=proc (r, i) options remember;
  eval(subs(x=proc(k) −hh(k, p)*(k−1)! end, p=r, YP[i]));
  mf(collect("*product('t1+k-1', 'k'=1..r−1)/(r−1)!, LH));
end:

invp:=proc (r, i) options remember;  eval(subs(u=r, polyn=dpv, derinv(i)))  end:
```

\#   *map sur des expressions en* $Z = 1/X = 1/(1-z)$ *et* $Y = \ln(Z) = \ln(1/(1-z))$ *:*
\#   *pour dérivations, intégrations et coefficients des séries.*
\#   *applique(f,w) remplace les termes* Z^rY^k *de* w *par f(k,r).*

```
applique:=proc(f, w)
  map(proc(q) g(degree(q, Y), degree(q, Z))*subs(Y=1, Z=1, q) end, w+1);
  eval(subs(g=f, "−g(0, 0)))
end:

deriv1:=proc(k, r) # Dérivation par (Θ choose t).
  Z^r*sum('binomial(k, i)*derp1(r, i)*Y^(k-i)*(-1)^i', 'i'=0..k)
end:

antider:=proc(k, r) # Intégration par P\Q(Θ).
  if r=2
  then −Z^2*sum('binomial(k, i)*invp(2, i)*Y^(k-i+1)/(k-i+1)*(-1)^i', 'i'=0..k)
  else Z^r*sum('binomial(k, i)*invp(r, i)*Y^(k-i)*(-1)^i', 'i'=0..k)
  fi
end:
```

\#   *Réduction et normalisation des expressions.*

```
col2:=proc (x)  collect(x, LZ, distributed);   collect(map(factor, "), z1) end:

mfc:=proc(w)
  collect(w, LD, distributed);  tcoeff(", LD, 't');
  mf(""-"*t)+normal(t*")
end:

mf:=proc(w) map(normal, w) end:
norm1:=proc(w) Z^degree(w, Z)*Y^degree(w, Y); mfc(w/")*" end:
norm:=proc(w) collect(w, [Z, Y], distributed) end:
reduit:=proc(w) map(norm1, norm(w)) end:
conv:=proc(w) subs(ln X =-Y, X=1/Z, w) end:
deconv:=proc(w) subs(Z=1/X, Y=-ln X, w) end:

#    Détermination asymptotique à partir des séries c.i des moments tm[i],
#        cumulant tk[i] et moments centrés tmc[i].

gm:=proc (i)  tm[i]:=col2(applique(coefdom, c.i))  end:

gmc:=proc (i)
  subs(x=proc(j) tm[j] end, LP[i]);
  tk[i]:=col2(eval("));
  subs(x=proc(j) if j=1 then 0 else tk[j] fi end, YP[i]);
  tmc[i]:=col2(eval("))
end:
```

$\#$ *Calcul du second membre sm(i) à intégrer :*

$\#$ $gt.i = (-1)^t X \binom{\Theta}{t} c.i,$

$\#$ $valgt(i) = d/du^i (X^{t+1} C^{(t)}(zu, u))/t!$ *en* $u = 1,$

$\#$ $gtq(i) = d/du^i gt(zu, u)$ *en* $u = 1.$

```
gt0:=1:

gtq:=proc(q) option remember;
  sum('binomial(q,j)*(-1)^j*conv(diff(deconv(gt.(q-j)),X$j))','j'=1..q)+gt.q;
end:

valgt:=proc(r) option remember;
  sum('binomial(r,p)*product('t1+i-1', 'i'=1..p)*Z^p*gtq(r-p)', 'p'=0..r);
end:

`diff/gt`:=proc(p, u, u) gt(p+1, u) end:

sm:=proc(i) Z*eval(subs(gt=proc(p, u) valgt(p) end, s*gt.i-diff(gt(0, u)^s, u$i)))  end:

#    Intégration et calcul.
```

```
reval:=proc(i);
   SUB:=['h.l=h.l+d.l'$'l'=1..i, t1=s*t1];
   LH:=['h.l'$'l'=1..i];
   LD:=['d.l'$'l'=2..i, op(LH)];
   LZ:={'z.l'$'l'=1..i, op(LD)};
   gtq(i-1):=reduit(gtq(i-1));
   valgt(i-1):=reduit(valgt(i-1));
end:

int:=proc(i)
   reval(i);
   c.i:=reduit(applique(antider, norm(sm(i))));
   gt.i:=reduit(applique(deriv1, c.i)/Z);
end:

calcul:=proc(ord);
   bellpoly(ord);
   for j to ord do int(j); print(c.j) od;
   for j to ord do gm(j); gmc(j); print(tk[j]) od;
end:
```

En posant `d1:=1;` (simplification possible par homogénéité), le calcul des 5 premiers cumulants (`calcul(5);`) prend de l'ordre de $10mn$ CPU ; pour 4 cumulants, il suffit de $1mn30s$.

1.2. Calcul du développement des cumulants.

Cette section donne les modifications à effectuer pour obtenir le développement des cumulants jusqu'au terme linéaire. Ce développement correspond à la conjecture IV.1 p. 76 et aux résultats de la section C.5..

```
# Calcul exact avec sous-termes.

gtq:=proc(q) option remember;
 sum('binomial(q,j)*((1-Z)/Z)^j*conv(diff(deconv(gt.(q-j)),X$j))','j'=1..q)+gt.q
end:

valgt:=proc(r) option remember;
   sum('binomial(r, p)*product('t1+i-1', 'i'=1..p)*(Z-1)^p*gtq(r-p)', 'p'=0..r);
end:

col2:=proc (x)
   collect(x, {op(LZ), n1}, distributed);
   collect(map(factor, "), {z1, n1}, distributed)
end:

readlib(stir2):
gm:=proc(i)
```

```
  sum('c.j*stir2(i, j)', 'j'=1..i);
  tm[i]:=col2(applique(coefexact, reduit('')))
end:
```

Le calcul des 4 premiers cumulants prend de l'ordre de $14mn$ CPU.

1.3. Calcul avec t fixé.

On donne ici les modifications à apporter pour déterminer les cumulants avec des valeurs du paramètre t fixées.

```
t1:=...

hh:=proc(i, r) options remember;
  if r=1 then 1 else h.i+sum('1/(vt1+k)^i', 'k'=1..r-2)-h(i, r-1) fi;
end:

derpp1:=proc (r, i) options remember;
  eval(subs(x=proc(k) -hh(k, p)*(k-1)! end, p=r,YP[i]));
  ''*product('vt1+k-1', 'k'=1..r-1)/(r-1)!;
end:

derp1:=proc(r, i)  options remember; subs(SUB1, derpp1(r, i)); end:

dpv:=proc(r, i) options remember;
  if r=2 then (s*derpp1(2, i+1)-subs(SUB, derpp1(2, i+1)))/(i+1)
  else s*derpp1(r, i)-subs(SUB, derpp1(r, i))
  mf(collect('', LD));
end:

reval:=proc(i)
  SUB:=['h.l=h(l, t1)+d.l'$'l'=1..i, vt1=s*t1];
  SUB1:=['h.l=h(l, t1)'$'l'=1..i, vt1=t1];
  LD:=['d.l'$'l'=2..i];
  LZ:={'z.l'$'l'=1..i, op(LD)};
  gtq(i-1):=reduit(gtq(i-1));
  valgt(i-1):=reduit(valgt(i-1))
end:
```

Pour $t1 = t + 1 = 1$, on obtient ainsi les 7 premiers cumulants en $17mn$ ou les 5 premiers en $1mn$ CPU. Si de plus on fixe la valeur de s, il faut alors de l'ordre de 2 à 3 minutes pour obtenir les 10 premiers moments.

2. Cumulant du tri par récurrence dans le cas binaire $s = 2$.

Les procédures de cette section calculent dans le cas binaire le coefficient rationnel du terme dominant des cumulants du nombres de comparaisons dans Quicksort. On utilise la récurrence caractéristique de la proposition IV.11 p. 84 pour obtenir les résultats de la section D.6..

2.1. Cas où t est fixé.

```
# Cumulants cas binaire avec t fixé à partir récurrence caractéristique :
#     κ_i(n) = (-1)^i kt^i (i-1)!(a[i] - ζ(i))n^i,
#     b[p] = Y_p(···, a[i](i-1)!, ···)/p! avec Y_p polynôme de Bell complet,
#     a[p] = L_p(···, b[i]i!, ···)/(p-1)! avec L_p polynôme logarithmique.

nmax:=50:
t:=0: kt:=2:
# t:=1: kt:=12/7:
# t:=2: kt:=60/37:
# t:=3: kt:=840/533:

# Calcul des b[i].
readlib(stirl):
g:=proc (rk) option remember; sum('b[rk-j]/j!*(-1)^j/kt^j', 'j'=0..rk) end:
f:=proc (r) option remember; sum('stirl(r+t+1, k+1)/t!*g(r-k)', 'k'=0..r) end:
cc:=proc (p)
   sum('stirl(p+2+2*t,r+1)/(2*t+1)!*b[p-r]+f(r)*f(p-r)', 'r'=0..p)
end:

b[0]:=1: b[1]:=1:
for i from 2 to nmax do b[i]:=solve(cc(i), b[i]); f(i):=eval(f(i)); g(i):=eval(g(i)) od;

# Calcul des a[i] :
# bell1(n, k) = B_{n,k}(.., b[i]i!, ...)/n!.
bell1:=proc(n, k) option remember;
  if k=1 then b[n] else sum('b[n-l]*bell1(l,k-1)/k','l'=k-1..n-1) fi
end:
for i to nmax do a[i]:=i*sum('(-1)**(k-1)*(k-1)!*bell1(i, k)', k=1..i) od;

# Valeurs flottantes : kf=cumulant.
read zeta: # lecture d'une table z[i] de valeurs flottantes de zeta
Digits:=70:
for i from 2 to nmax do kf[i]:=evalf((-1)**i*(i-1)!*kt^i*(a[i]-z[i])) od;

# Moments centrés : mcf.
bell:=proc(n, k) option remember;
   if k=1 then kf[n] else sum('binomial(n,l)*kf[n-l]*bell(l, k-1)/k', 'l'=k-1..n-1) fi
end:
kf[1]:=0:
for i to nmax do mcf[i]:=sum('bell(i, k)', 'k'=1..i) od;
```

Le calcul des 50 premiers cumulants et moments centrés prend de 30 à $45mn$ CPU suivant les valeurs de t. Le même calcul en effectuant dès le départ les évaluations en flottant (et non en rationnel) prend environ un $9mn$ CPU.

2.2. Cas t quelconque.

\# *Cumulants cas binaire avec t quelconque (récurrence caractéristique) :*
\# $stirt(a,b) = |stir1(a+t1,b)|/(t1-1)! \quad 0 < b$,
\# $stir2t(a,b) = |stir1(a+2t1,b)|/(2t1-1)!$,
\# $s(b) = |stir1(t1,b)|/(t1-1)!$ *s'elimine dans la somme cc(i),*
\# $s1(b) = |stir1(t1+1,b)|/t1! = Y_{b-1}(\cdots,(-1)^p p! h.(p+1),\cdots)/(b-1)!$,
\# *avec* $t1 = t+1$, $h.r = H_{t1}^{(r)}$ *et* $d.r = H_{2t1}^{(r)} - h.r$,
\# $bell1(n,k) = B_{n,k}(\ldots,x(i)i!,\ldots)/n!$ *(idem YP1,LP1)*

```
bell1:=proc(n, k) option remember;
  if k=1 then x(n) else sum('x(n-l)*bell1(l, k-1)/k', 'l'=k-1..n-1) fi;
  expand(");
end:
YP1:=proc(i) sum('bell1(i, k)', 'k'=1..i) end:
LP1:=proc(i) sum('(-1)^(k-1)*(k-1)!*bell1(i, k)', 'k'=1..i) end:

s1:=proc(b) eval(subs(x=proc(r) -(-1)^r*h.r/r end, YP1(b-1))) end:

stirt:=proc(a, b) option remember;
  if b=1 then expand((t1+a-1)!/(t1-1)!)
  elif a=0 then s(b)
  elif a=1 then s1(b)*t1
  else stirt(a-1, b-1)+(t1+a-1)*stirt(a-1, b) fi;
  map(normal, collect(", ['h.i'$'i=1..b-1'], distributed));
end:

stir2t:=proc(a, b) subs('h.i=h.i+d.i'$'i'=1..b-1, t1=2*t1, stirt(a, b)); end:

g:=proc(rk) option remember; sum('(-1)^i*b[i]/(rk-i)!*d1^(rk-i)', 'i'=0..rk) end:
f:=proc(r) option remember; sum('stirt(r, k+1)*g(r-k)', 'k'=0..r) end:
cc:=proc(p)
  sum('(-1)^(p-r+1)*stir2t(p, r+1)*b[p-r]+f(r)*f(p-r)', 'r'=0..p)
end:

norm:=proc(w); map(normal, collect(w, T, distributed)) end:

calcul:=proc(n)
    for i from 2 to n do
         T:={'h.l'$'l'=1..i, 'd.l'$'l'=2..i};
         b[i]:=norm(solve(cc(i), b[i]));
```

```
        f(i):=norm(f(i));
        a[i]:=norm(eval(subs(x=proc(r) b[r] end, i*LP1(i))));
        print(a[i]);
    od;
end:

b[0]:=1: b[1]:=1:
```

On obtient ainsi en $8mn$ CPU, les formules en fonction de t pour les 7 premiers cumulants ($s = 2$).

3. Cumulant de la recherche.

Les procédures suivantes permettent de déterminer les coefficients du terme dominant des cumulants pour le coût de la recherche dans un arbre s-aire réduit. Ces procédures correspondent à la proposition IV.1 p. 60 et au théorème IV.1 p. 66 ; elles donnent les résultats de la section C.3..

3.1. Calcul par récurrence.

On utilise ici la décomposition de la récurrence caractéristique donnée par l'équation IV.18 p. 61.

 # Détermination des cumulants pour la recherche :
$\kappa(i) = kk[i] \ln n + O(1)$,
$d.i = H^{(i)}_{s(t+1)} - H^{(i)}_{t+1}$, $D.i = d.i/d1$,
$L(i) = L_i(cumulantsauxnoeuds)/d1$ *(donnés).*

```
bell:=proc(n, k) option remember;
  if k=1 then kk[n]
  else sum('binomial(n, l)*kk[n-l]*bell(l, k-1)/k', 'l'=k-1..n-1) ;
  expand(");
  fi
end:

D1:=1:
calcul:=proc(n)
  L(n)=sum('(-1)**(k-1)*(k-1)!*D.k*bell(n, k)', 'k'=1..n);
  kk[n]:=collect(solve(", kk[n]), {'D.k' $'k'=2..n}, distributed)
end:
```

#Arbres s−aires avec médiane : $s = 2^k + s0 \quad 0 < s0 < 2^k + 1$, $a(s) = k + 2s0/s$, $l(s) = 2s0/s$.

```
readlib(stir2):
L:=proc(n)
  if n=1 then a(s)/d1
  else -sum('(k-1)!*(-l(s))^k*stir2(n, k)', 'k'=1..n)/d1
```

```
  fi;
end:
#Arbres s-aires complets : s = 2^k.
L:=proc(n) if n=1 then a(s)/d1 else 0 fi; end:
```

On obtient de cette façon les 10 premiers cumulants en 19s CPU.

3.2. Calcul direct.

Les cumulants sont déterminés par l'expression directe donnée p. 66 dans la remarque finale de la section IV.2..

```
#    Cumulants pour la recherche : calcul direct.

bbell:=proc(n, k) option remember;
  if k=1 then x(n)
  else sum('binomial(n, l)*x(n-l)*bbell(l, k-1)/k', 'l'=k-1..n-1)
  fi;
  expand(")
end:

f:=proc(n, k) option remember;
  eval(subs(x=proc(n) (-1)^n*n!*D.(n+1)/(n+1) end, bbell(n, k)))
end:

g:=proc(i) option remember;
  if i=1 then 1 else  sum('(-1)^j*(i+j-1)!/(i-1)!*f(i-1, j)', 'j'=1..i-1) fi;
end:

calcul:=proc(n)
  eval(subs(x=proc(u) L(u) end, sum('bbell(n, i)*g(i)', 'i'=1..n)));
  kk[n]:=collect(", {'D.k' $'k'=2..n}, distributed)
end:
```

Les 10 premiers cumulants sont alors obtenus en 17s CPU. La méthode permet aussi de calculer directement un cumulant sans déterminer les cumulants d'ordre inférieur. On obtient ainsi le 10$^{\text{ème}}$ cumulant seul en 11s CPU dont 4s pour l'affichage.

Bibliographie

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[AHU83] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

[AS73] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions*. Dover, 1973.

[Ben74] E. A. Bender. Asymptotic methods in enumeration. *SIAM Review*, 16(4), October 1974.

[Ben75] E. A. Bender. An asymptotic expansion for the coefficients of some formal power series. *J. London Math. Soc.*, 9(2):451–458, 1975.

[BS84] G. G. Brown and B. O. Shubert. On random binary trees. *Math. op. res.*, 9(1):43–65, 1984.

[Bur72] W. Burge. An analysis of a tree sorting method and some properties of a set of trees. In *First USA-JAPAN Computer Conference*, 1972.

[Bur76] W. H. Burge. An analysis of binary search trees formed from sequences of nondistinct keys. *JACM*, 23(3):451–454, July 1976.

[BY87] A. Baeza-Yates. Some average measures in m-ary search trees. *Information Processing Letters*, 25:375–381, July 1987.

[CGG+88] B. W. Char, K. O. Geddes, G. H. Gonnet, M. B. Monagan, and S. M. Watt. *MAPLE: Reference Manual*. University of Waterloo, 1988. 5th edition.

[Com70] L. Comtet. *Analyse Combinatoire*. PUF, Paris, 1970. 2 volumes.

[Com74] L. Comtet. *Advanced Combinatorics*. Reidel, Dordrecht, 1974.

[CP88] W. Cunto and P. Poblete. Transforming multiway trees into a practical external data structure. *Acta Informatica*, 1988.

[Dar78] Gaston Darboux. Mémoire sur l'approximation des fonctions de très grands nombres, et sur une classe étendue de développements en série. *Journal de Mathématiques Pures et Appliquées*, pages 5–56,377–416, Février 1878.

[dB58] Nicolaas Govert de Bruijn. *Asymptotic Methods in Analysis.* North Holland, third edition, 1958. Reprinted by Dover, 1981.

[Die68] J. Dieudonné. *Calcul Infinitésimal.* Hermann, Paris, 1968.

[Fel71] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 2. John Wiley, 1971.

[FGT87] P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms, and self–organizing search. Technical Report 720, Institut National de Recherche en Informatique et en Automatique, August 1987. 17 pages. Submitted to *Discrete Applied Mathematics.*

[Fla85a] P. Flajolet. Ambiguity and transcendence. In Wilfried Brauer, editor, *Automata, Languages and Programming*, volume 194 of *Lecture Notes in Computer Science*, pages 179–188. Springer Verlag, 1985. Proceedings of 12th ICALP Colloquium, Nafplion, Greece, July 1985.

[Fla85b] P. Flajolet. Elements of a general theory of combinatorial structures. In Lothar Budach, editor, *Fundamentals of Computation Theory*, volume 199 of *Lecture Notes in Computer Science*, pages 112–127. Springer Verlag, 1985. Proceedings of FCT'85, Cottbus, GDR, September 1985 [Invited lecture].

[Fla87] P. Flajolet. Mathematical tools for automatic program analysis. Research Report 603, INRIA, 1987.

[FO90] Philippe Flajolet and Andrew M. Odlyzko. Singularity analysis of generating functions. *SIAM Journal on Discrete Mathematics*, 3(1), February 1990. To appear. (Also available as INRIA Research Report 826, 1987, 25 pages).

[Foa74] D. Foata. *La série génératrice exponentielle dans les problèmes d'énumération.* S.M.S. Montreal University Press, 1974.

[FP86] Philippe Flajolet and Claude Puech. Partial match retrieval of multidimensional data. *Journal of the ACM*, 33(2):371–407, 1986.

[Fra76] J. Françon. Arbres binaires de recherche : Propriétés combinatoires et applications. *RAIRO informatique théorique et applications*, 10(12):35–50, 1976.

[FSt87] P. Flajolet, J-M. Steyaert A Complexity Calculus for Recursive Programs Over Tree Structures. *Mathematical System Theory* 19:301–331, 1987.

[FSZ89a] P. Flajolet, B. Salvy, and P. Zimmermann. Lambda–Upsilon–Omega: An assistant algorithms analyzer. In T. Mora, editor, *Applied Algebra, Algebraic Algorithms and Error–Correcting Codes*, volume 357 of *Lecture Notes in Computer Science*, pages 201–212, 1989. (Proceedings AAECC'6, Rome, July 1988).

[FSZ89b] P. Flajolet, B. Salvy, and P. Zimmermann. Lambda–Upsilon–Omega: The 1989 Cookbook. Research Report 1073, Institut National de Recherche en Informatique et en Automatique, August 1989. 116 pages.

[GJ83] I. P. Goulden and D. M. Jackson. *Combinatorial Enumeration.* John Wiley, New York, 1983.

[GK81] D. H. Greene and D. E. Knuth. *Mathematics for the analysis of algorithms.* Birkhauser, Boston, 1981.

[Gre83] D. H. Greene. *Labelled formal languages and their uses.* PhD thesis, Stanford University, 1983.

[Hen70] P. Henrici. Methods of search for solving polynomial equations. *Journal of the ACM*, 17(2):273–283, April 1970.

[Hen89] Pascal Hennequin. Combinatorial analysis of quicksort algorithm. *RAIRO, Theoretical Informatics and Applications*, 23(3):317–333, 1989.

[Hoa62] C. A. R. Hoare. Quicksort. *Computer J.*, 5(1):10–15, 1962.

[IR80] A. Itai and M. Rodeh. Modified binary search trees. Technical Report 182, Dept. of Computer Science, Technion-IIT, Haifa, Israel, August 1980.

[Joy81] A. Joyal. Une théorie combinatoire des séries formelles. *Advances in Mathematics*, 42(1):1–82, 1981.

[Knu68] D. E. Knuth. *The Art of Computer Programming*, volume 1: Fundamental Algorithms. Addison-Wesley, 1968.

[Knu73] D. E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, 1973.

[Köl85] K. S. Kölbig. Explicit evaluation of certain definite integrals involving powers of logarithms. *Journal of Symbolic Computation*, 1:109–114, 1985.

[Lou87] G. Louchard. Exact and asymptotic distributions in digital and binary search trees. *RAIRO informatique théorique et applications*, 21(4):479–495, 1987.

[Mah86] H. Mahmoud. On the average internal path length of search trees. *Acta Arithmetica*, 23:111–117, 1986.

[Moo70] J. W. Moon. Counting labelled trees. In *Canadian Mathematical Monographs N.1.* William Clowes and Sons, 1970.

[MP85] J. I. Munro and P. V. Poblete. The analysis of a fringe heuristic for binary search trees. *Journal of Algorithms*, 6:336–350, 1985.

[MP89] H. M. Mahmoud and B. Pittel. Analysis of the space of search trees under the random insertion algorithm. *J. Algorithms*, 10:52–75, 1989.

[Rég87] M. Régnier. A limiting distribution for quicksort. Technical Report 739, INRIA, oct 1987. To appear in RAIRO informatique théorique et applications.

[Rös89] U. Rösler. A limit theorem for quicksort. *RAIRO informatique théorique et applications*, 1989. To appear.

[Sed77a] R. Sedgewick. The analysis of quicksort programs. *Acta Informatica*, 7:327–355, 1977.

[Sed77b] R. Sedgewick. Quicksort with equal keys. *SIAM Journal on Computing*, 6(2):240–267, June 1977.

[Sed80] R. Sedgewick. *Quicksort.* Garland Pub. Co., New York, 1980. Reprint of Ph.D. thesis, Stanford University, 1975.

[Sed83] R. Sedgewick. *Algorithms.* Addison-Wesley, Reading, Mass., 1983.

[Sir71] L. Sirovich. *Techniques of Asymptotic Analysis.* Springer Verlag, 1971.

[Sta78] Richard Peter Stanley. Generating functions. In G-C. Rota, editor, *Studies in Combinatorics,* M.A.A. Studies in Mathematics, Vol. 17., pages 100–141. The Mathematical Association of America, 1978.

[Ste84] J-M. Steyaert. Structure et complexité des algorithmes. Thèse d'Etat à l'Université de Paris VII, 1984.

[Tit39] E. C. Titchmarsh. *The Theory of Functions.* Oxford University Press, 2nd edition, 1939.

[VF89] J. Vitter and P. Flajolet. Analysis of algorithms and data structures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity, chapter 9. North Holland, 1989. To appear. Available as INRIA Research Report 718, August 1987, 67pages. Also available as Technical Report, CS-87-20, Brown University 1987, 96 pages; revised 1989.

[Vie76] G. Viennot. Quelques algorithmes de permutation. In *Astérisque*, pages 38–39, 275–293. Société Mathématique de France, 1976.

[WW76] A. Walter and D. Wood. Locally balanced binary trees. *Computer J.*, 19(4):322–325, 1976.