INSTITUT POLYTECHNIQUE DE PARIS

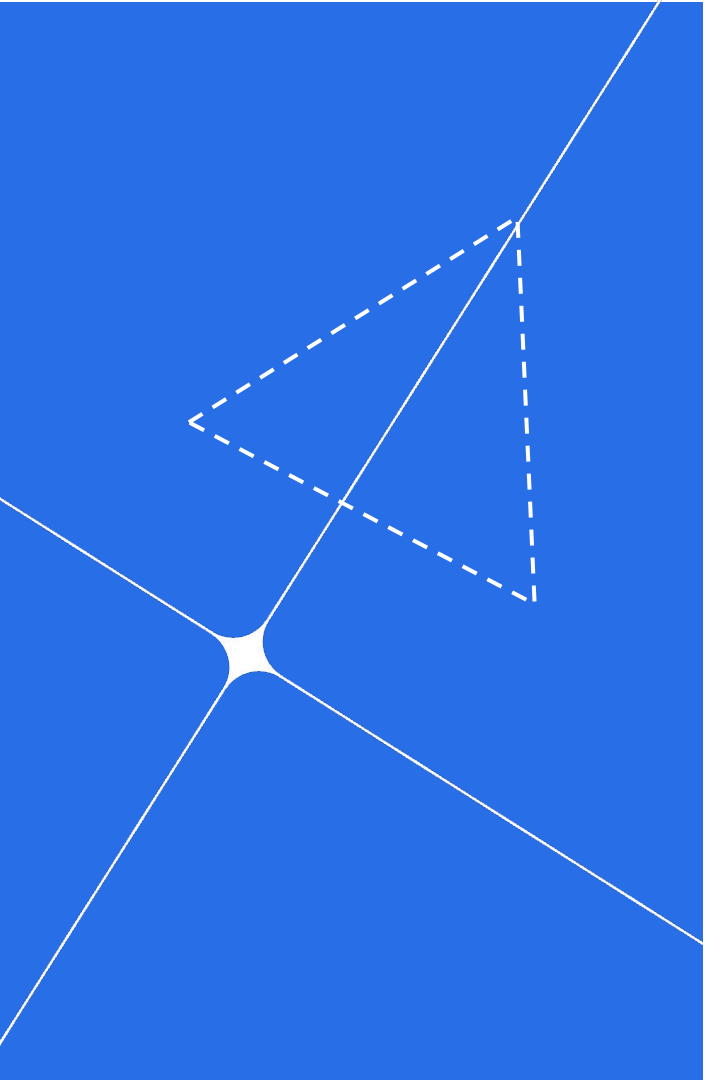
TELECOM SudParis

Introduction TP4

Julien Romero

Dans ce TP…

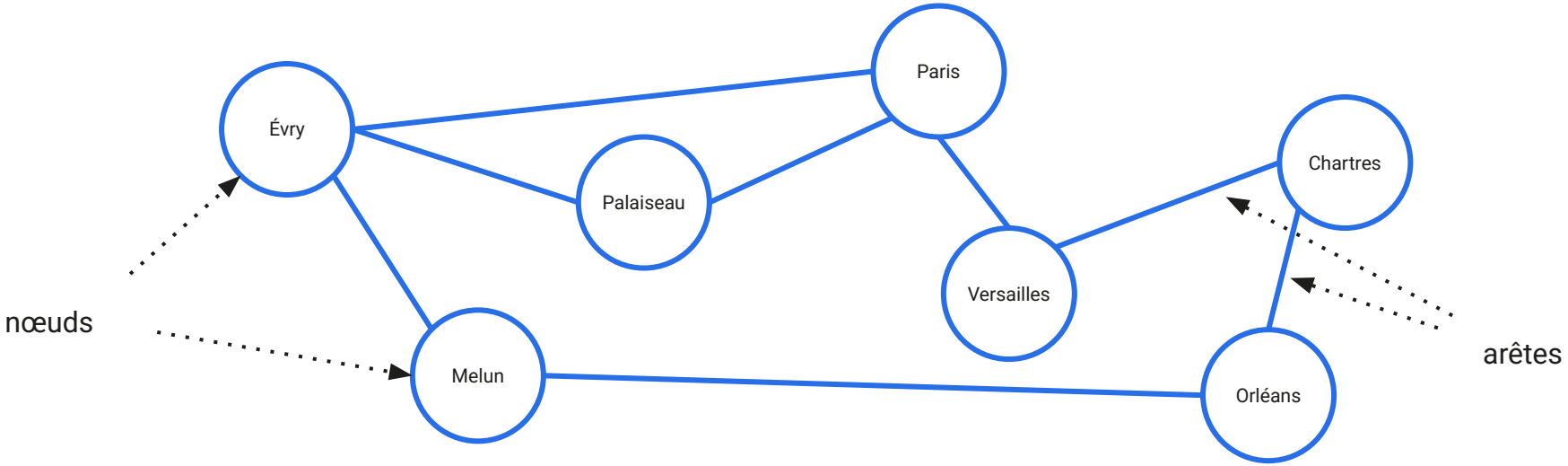
- Nous allons implémenter une nouvelle structure de données : les **graphes**
- Nous allons calculer le plus court chemin dans un graphe pondéré grâce à l'**algorithme de Dijkstra**
- Pour les plus avancés, nous allons implémenter une **file de priorité** avec un **tas binaire**



Les graphes

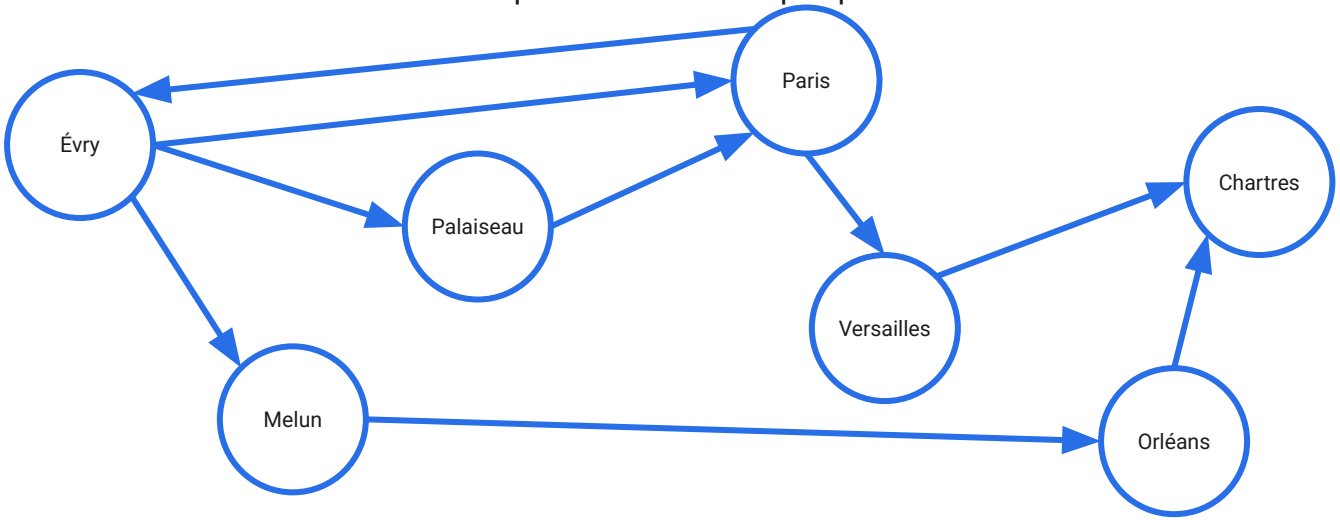
Les graphes

- Un graphe est une structure de données composée de nœuds connectés entre eux par des arêtes.
- Par exemple:
  - Dans un réseau social, un nœud est un utilisateur, et une connexion entre deux nœuds représente un lien d'amitié.
  - Dans un réseau routier, un nœud représente une ville, et une connexion entre deux nœuds une route.



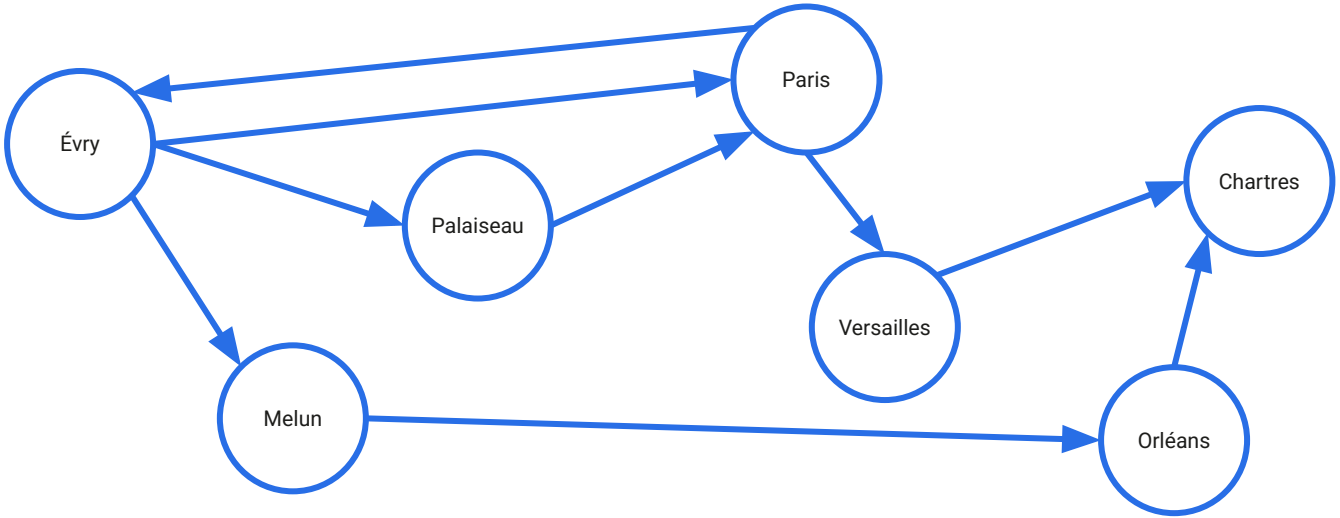
Les graphes - Graphe orienté

- Un graphe est dit **orienté** si les arêtes ont un sens de lecture.
- Lorsque le graphe est orienté, les arêtes sont représentées par des flèches.
  - Par exemple, dans le graphe ci-dessous, “Évry” est connecté à “Paris”.
- Notons que si un nœud A est connecté à un nœud B, nous n’avons pas forcément B connecté à A. Par exemple, sur X/Twitter la relation “Suit” est orientée mais pas forcément réciproque.



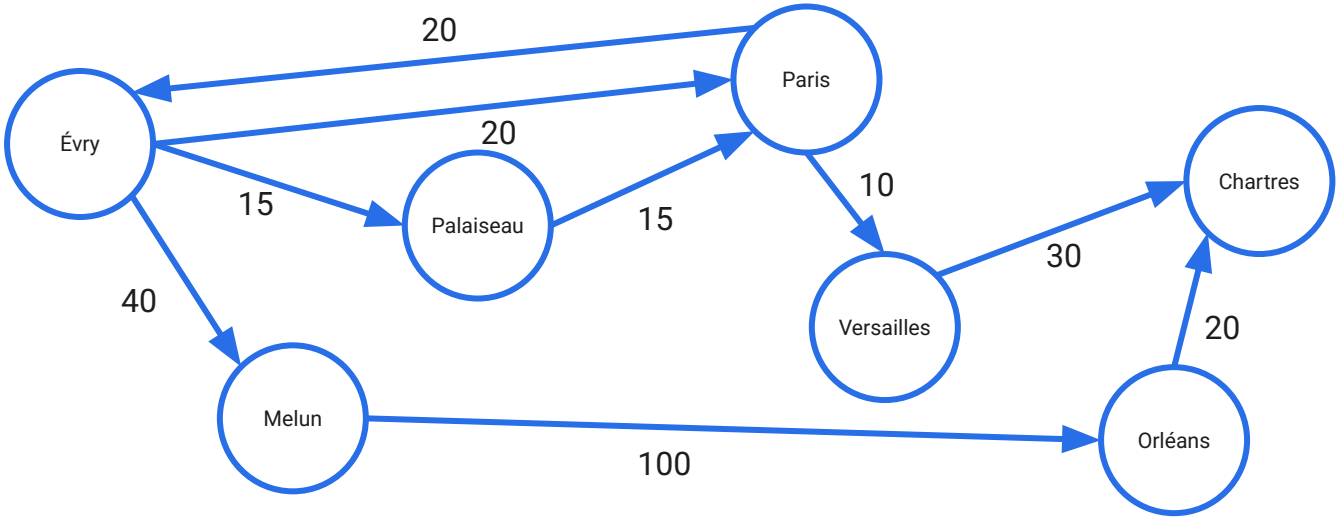
Les graphes - Voisins

- Les voisins d'un nœud A sont tous les nœuds connectés à A par une arête
  - Les voisins de "Évry" sont "Paris", "Palaiseau" et "Melun"
  - Les voisins de "Paris" sont "Évry" et "Versailles" ("Palaiseau" n'est pas un voisin à cause de la direction de l'arête)



Les graphes - Graphe pondéré

- Un graphe est dit **pondéré** si les arêtes ont un poids.
  - Par exemple, dans un réseau routier, le poids d'une arête représente la distance entre deux villes.



Algorithme de Dijkstra

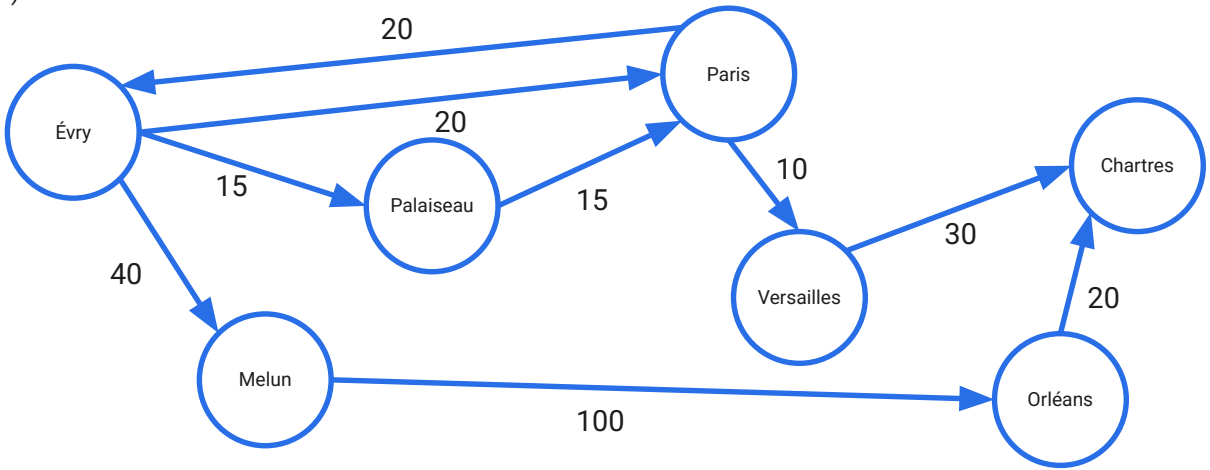
L'algorithme de Dijkstra

- L'algorithme de Dijkstra est un algorithme permettant de calculer le **plus court chemin** entre deux nœuds **A et B** dans un graphe pondéré (avec des poids positifs).
  - On note p(X, Y) le poids de l'arête entre X et Y.
- Le principe de l'algorithme est d'explorer les nœuds les uns après les autres en choisissant à chaque étape le nœud le plus proche que nous n'avons pas encore exploré.
- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - Tant qu'il y a des nœuds non explorés
    - Prendre le nœud inexploré C le plus proche de A.
    - If C == B, break
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - Tant qu'il y a des nœuds non explorés
    - Prendre le nœud inexploré C le plus proche de A
    - If C == B, break
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

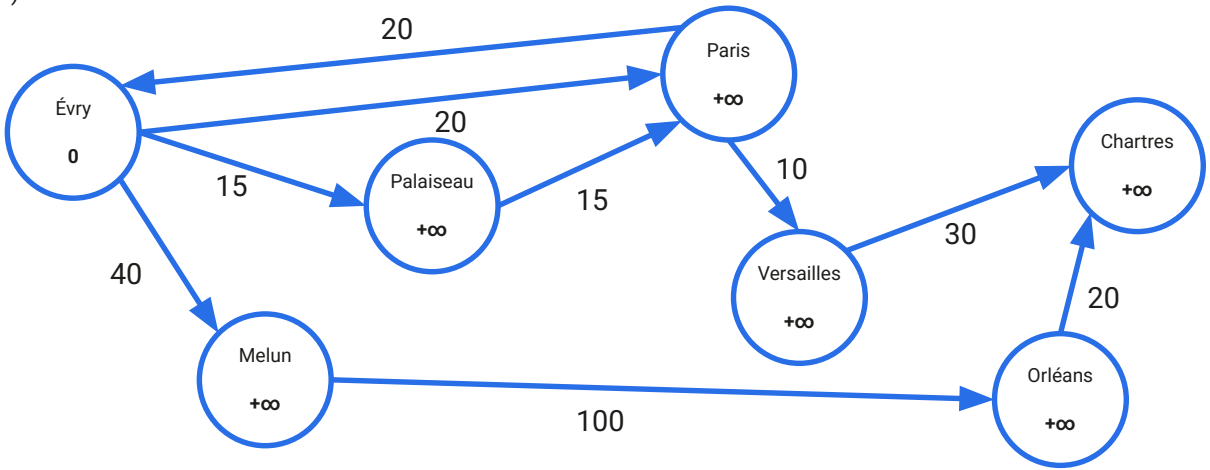
Distance entre Évry et Chartres ?



L’algorithme de Dijkstra

- **Algorithme**:
  - **Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l’instant entre A et X**
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - Tant qu’il y a des nœuds non explorés
    - Prendre le nœud inexploré C le plus proche de A
    - If C == B, break
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

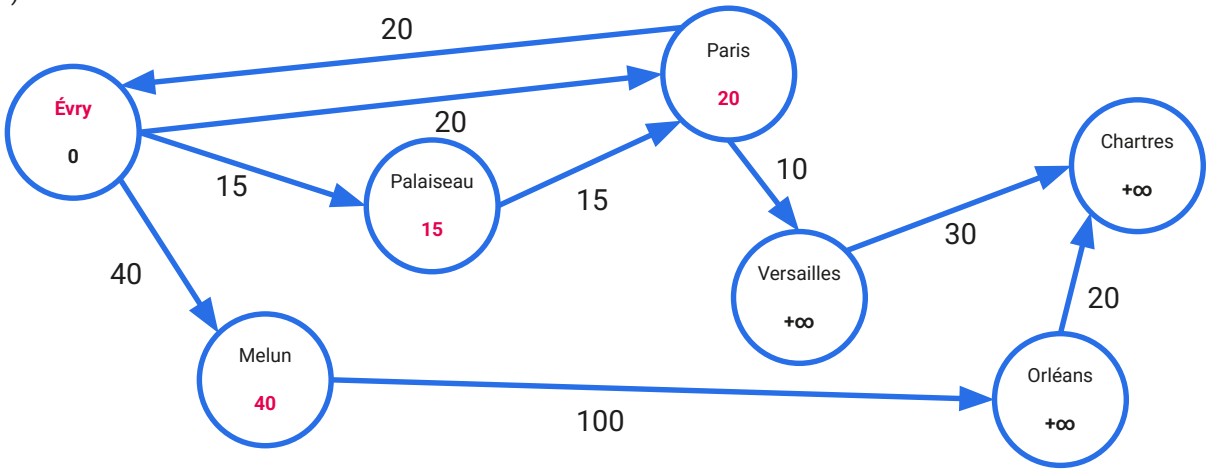
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - **Pour chaque voisin V de A (la source)**
    - **Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)**
  - Tant qu'il y a des nœuds non explorés
    - Prendre le nœud inexploré C le plus proche de A
    - If C == B, break
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

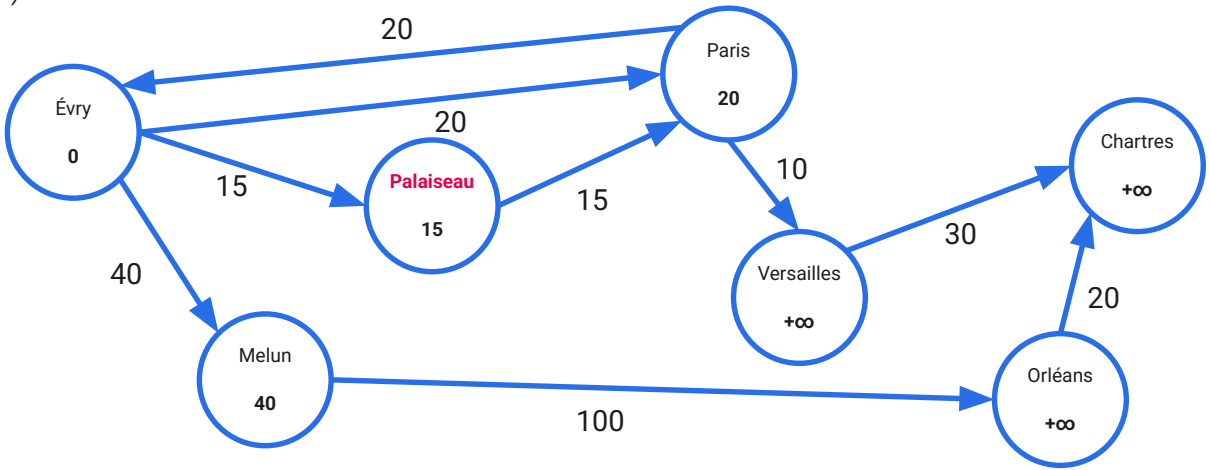
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - **Prendre le nœud inexploré C le plus proche de A**
    - If C == B, break
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

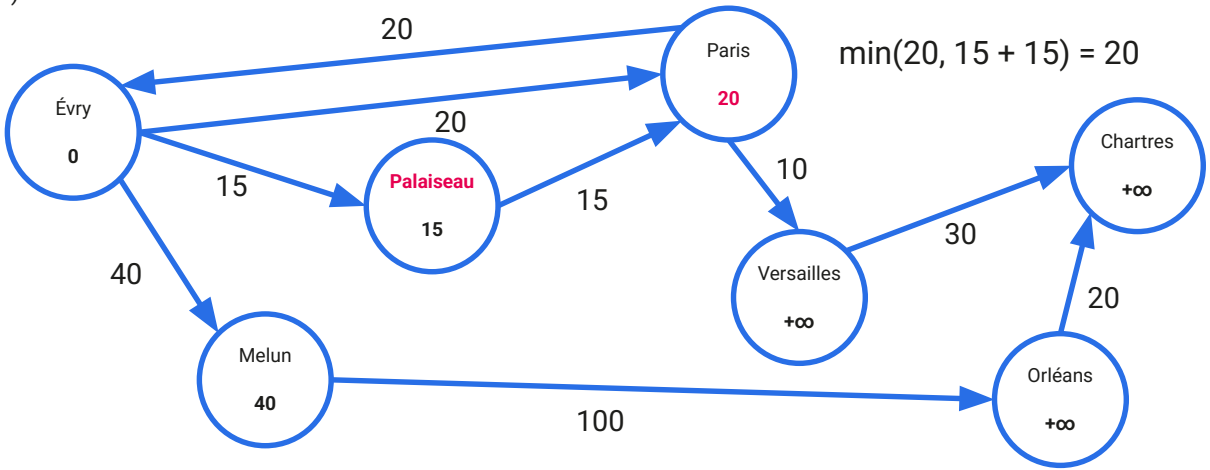
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - Prendre le nœud inexploré C le plus proche de A
    - If C == B, break
    - **Pour chaque voisin V de C**
      - **Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))**
  - return dist(B)

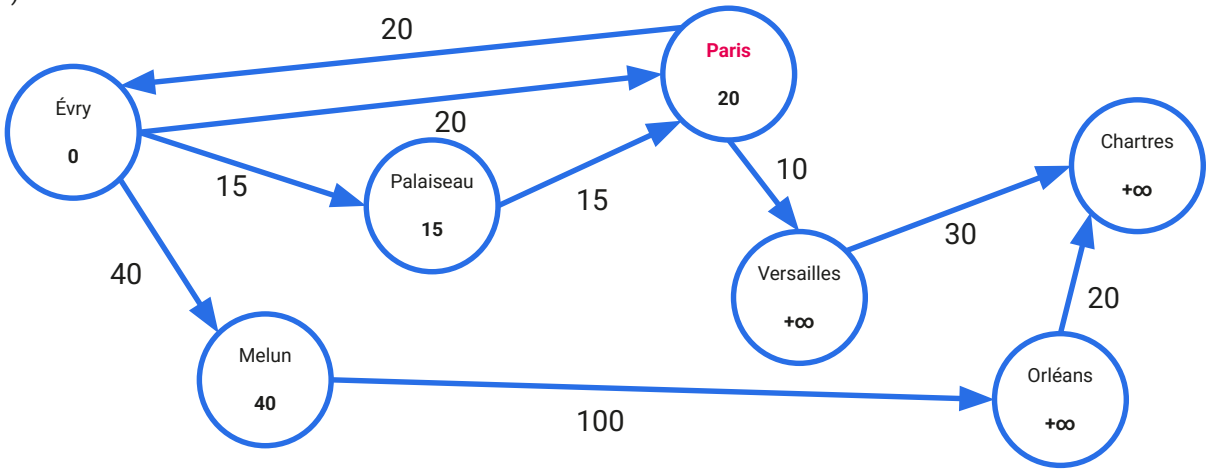
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - **Prendre le nœud inexploré C le plus proche de A**
    - If C == B, break
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

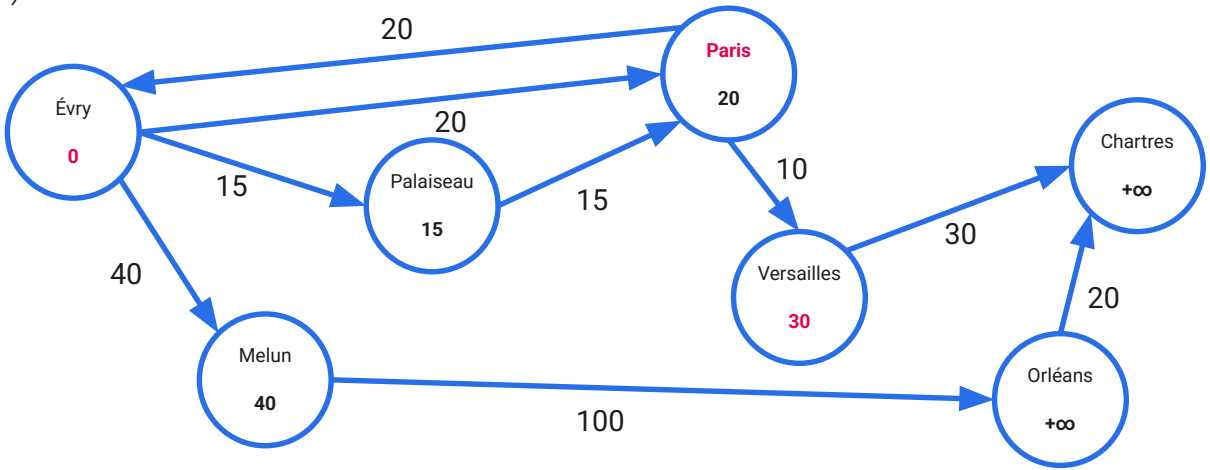
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - Prendre le nœud inexploré C le plus proche de A
    - If C == B, break
    - **Pour chaque voisin V de C**
      - **Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))**
  - return dist(B)

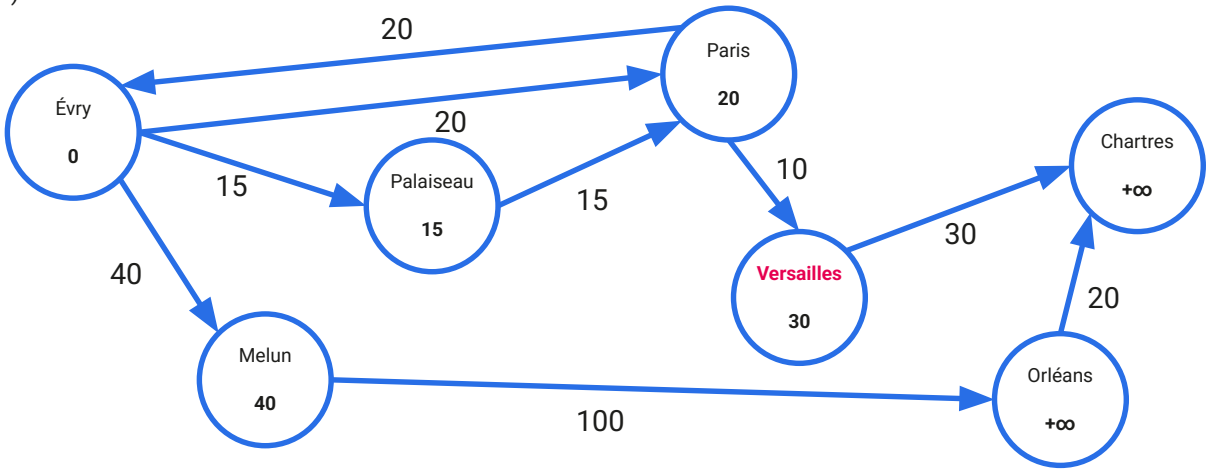
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - **Prendre le nœud inexploré C le plus proche de A**
    - If C == B, break
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

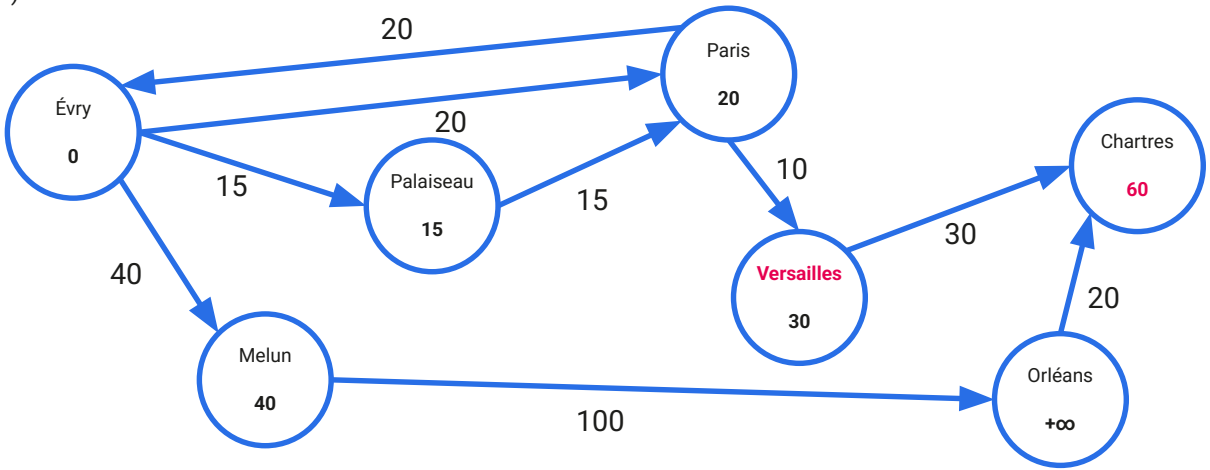
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - Prendre le nœud inexploré C le plus proche de A
    - If C == B, break
    - **Pour chaque voisin V de C**
      - **Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))**
  - return dist(B)

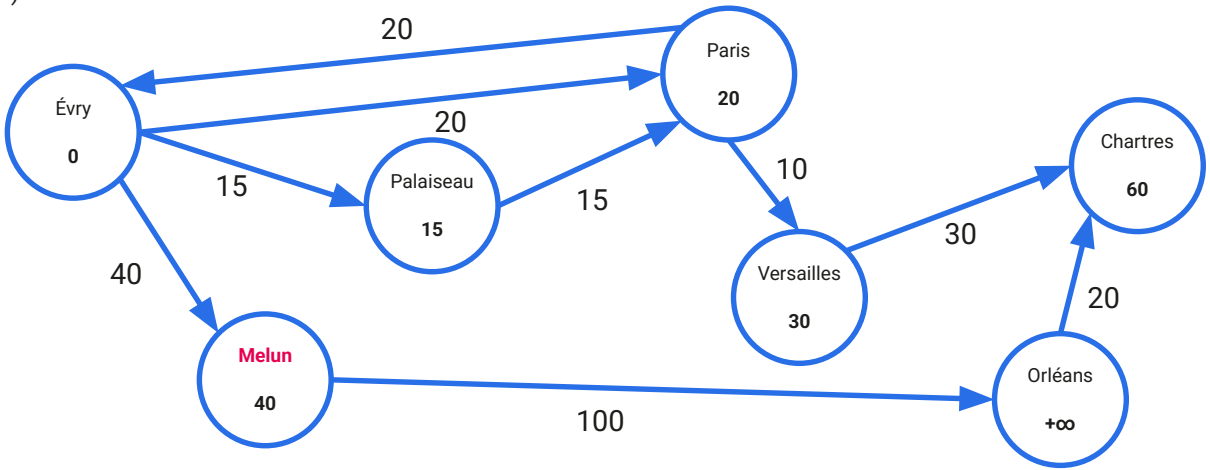
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - **Prendre le nœud inexploré C le plus proche de A**
    - If C == B, break
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

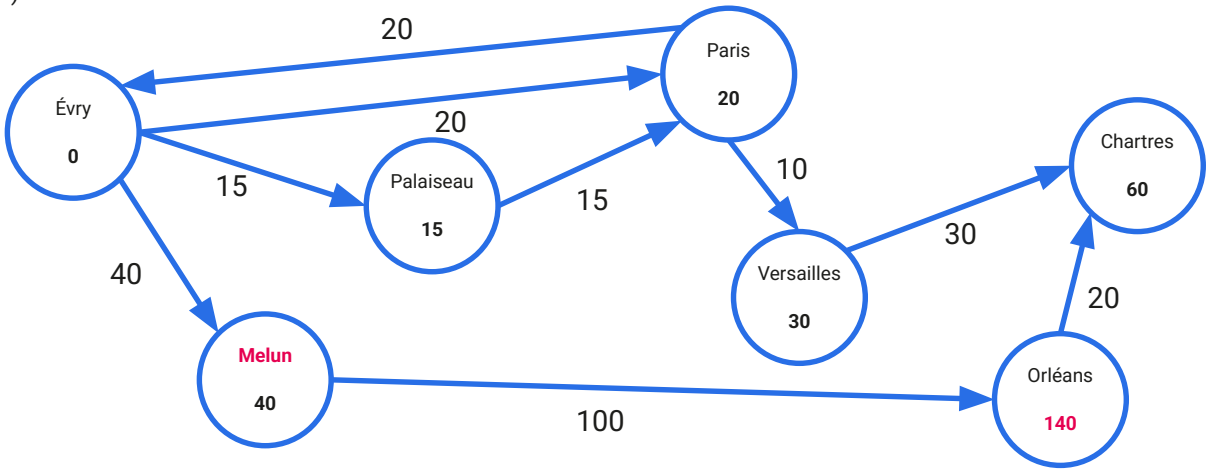
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - Prendre le nœud inexploré C le plus proche de A
    - If C == B, break
    - **Pour chaque voisin V de C**
      - **Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))**
  - return dist(B)

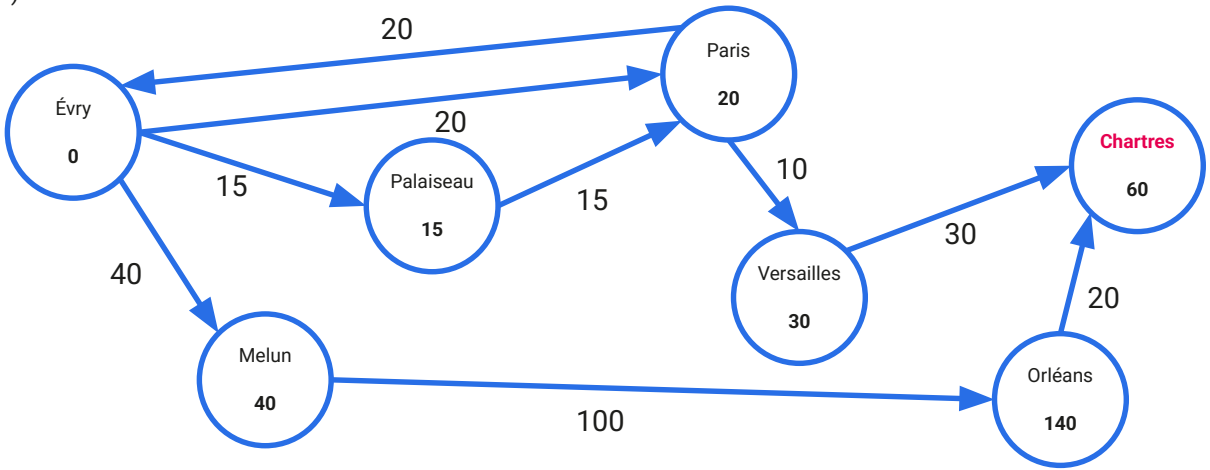
Distance entre Évry et Chartres ?



L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - **Prendre le nœud inexploré C le plus proche de A**
    - If C == B, break
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

Distance entre Évry et Chartres ?

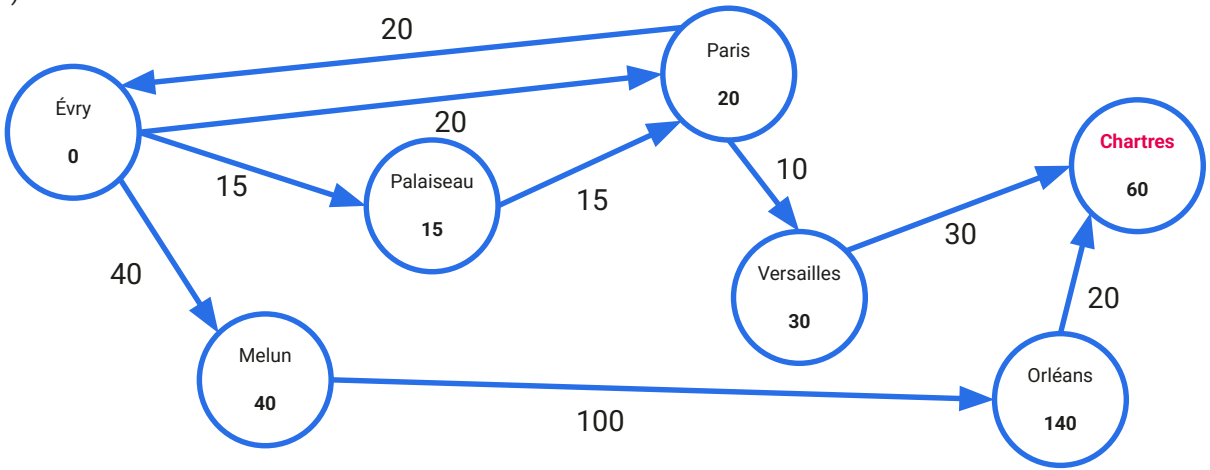


L'algorithme de Dijkstra

- **Algorithme**:
  - Initialiser les distances à tous les nœuds à +∞ (sauf A qui est à 0). dist(X) représente la meilleure distance trouvée pour l'instant entre A et X
  - Pour chaque voisin V de A (la source)
    - Mettre à jour la distance entre A et V comme étant dist(V) = p(A, V)
  - **Tant qu'il y a des nœuds non explorés**
    - Prendre le nœud inexploré C le plus proche de A
    - **If C == B, break**
    - Pour chaque voisin V de C
      - Mettre à jour la distance entre A et V comme étant dist(V) = min(dist(V), dist(C) + p(C, V))
  - return dist(B)

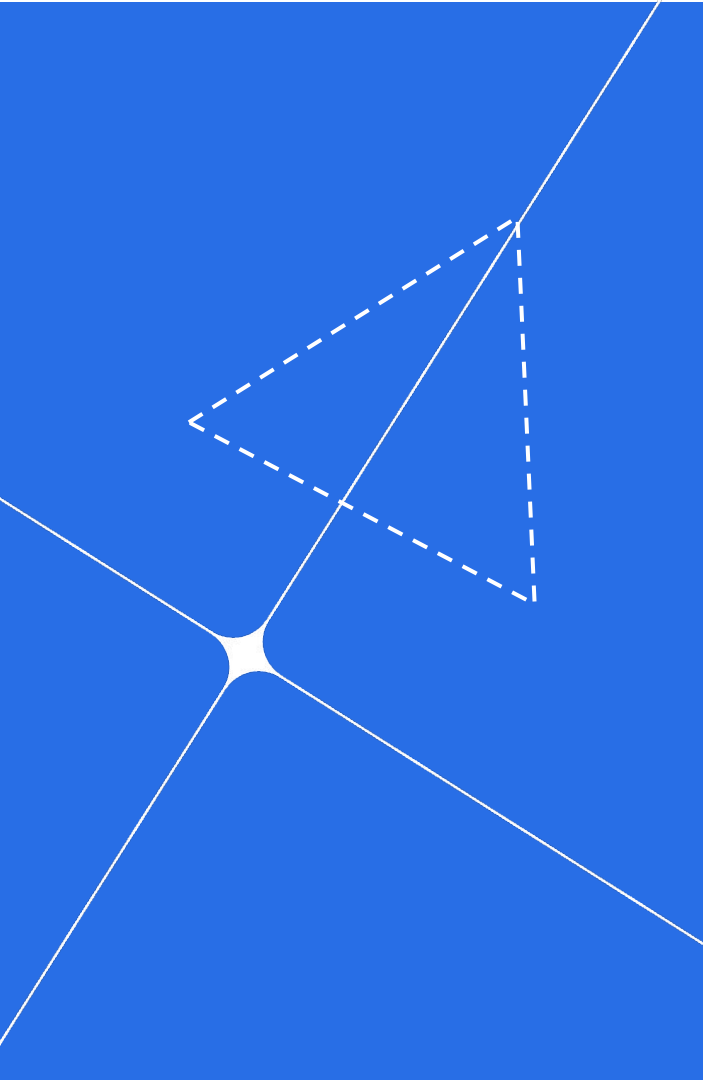
Distance entre Évry et Chartres ?

Résultat: 60



Les files de priorité

- Une étape importante de Dijkstra est de “Prendre le nœud inexploré C le plus proche de A”
  - Pour cela, on utilise une file de priorité.
- Une **file de priorité** est une structure de donnée permettant d’accéder aux éléments suivant un ordre dépendant du poids (ou priorité) de chaque élément.
  - Dans Dijkstra, le poids est représenté par la distance à la source.
- Plusieurs implémentations possibles :
  - Avec un tableau que l’on parcours entièrement : première solution du TP.
  - Avec un tas binaire (arbre binaire + tas, plus efficace) : deuxième solution du TP.
  - Avec un tas de Fibonacci (plus efficace encore, non vu dans le TP).



En route vers le TP !