



# Dijkstra Metro

Algorithme Dijkstra sert à résoudre le problème du plus court chemin dans un graph à poids non negatifs.

l'algorime fonctionne comme cela :

Soit un graphe avec une source (point de départ) noté S.

## 1. Initialisation

- $\text{dist}[S] = 0$
- Pour tout autre nœud  $v$  :  $\text{dist}[v] = +\infty$
- $\text{prev}[v] = \text{null}$  (pour reconstruire le chemin)
- Ensemble des nœuds "non traités" dans un tableau non traités

## 2. Choisir le prochain nœud

- Prendre le nœud  $u$  non traité avec la **plus petite**  $\text{dist}[u]$ . Qui devient le chemin optimal temporaire .
- Si  $\text{dist}[u] = +\infty$ , on peut arrêter : le reste est inatteignable.
- Marquer  $u$  comme **traité** (sa distance est maintenant optimale).

## 3. "Relaxer" les arêtes sortantes

Pour chaque voisin  $v$  de  $u$  avec un poids  $w(u,v)$  :

- Calculer  $\text{alt} = \text{dist}[u] + w(u,v)$
- Pour  $\text{dist}[v]$  la valeur la plus courte enregistré jusqu'à là.
- Si  $\text{alt} < \text{dist}[v]$  :
  - mettre  $\text{dist}[v] = \text{alt}$
  - mettre  $\text{prev}[v] = u$
  - mettre à jour  $v$  dans la file de priorité

## 4. Répéter

- Refaire 2) puis 3) jusqu'à ce que :

- tous les nœuds soient traités, ou
- tu aies traité la **destination** (si tu ne veux qu'un seul chemin), ou
- la file soit vide.

## 5 . Reconstruire un chemin (optionnel)

Pour obtenir le chemin vers un nœud  $T$  :

- partir de  $T$ , puis suivre  $\text{prev}[T]$ ,  $\text{prev}[\text{prev}[T]]$ , ... jusqu'à  $S$ , et inverser la liste

Et voilà pour l'algorithme concernant ça complexité:

Deux cas :

Avec une file de priorité (heap) : tas binaire

- $\text{extract-min}$  :  **$O(\log V)$**
- $\text{decrease-key}$  :  **$O(\log V)$**  (ou équivalent en pratique)
- tu fais  $\text{extract-min}$  environ  **$V$  fois** →  **$V * \log V$**
- tu peux faire une mise à jour par arête (au pire)  **$E$  fois** →  **$E * \log V$**

Total :  **$O((V + E) \log V)$**

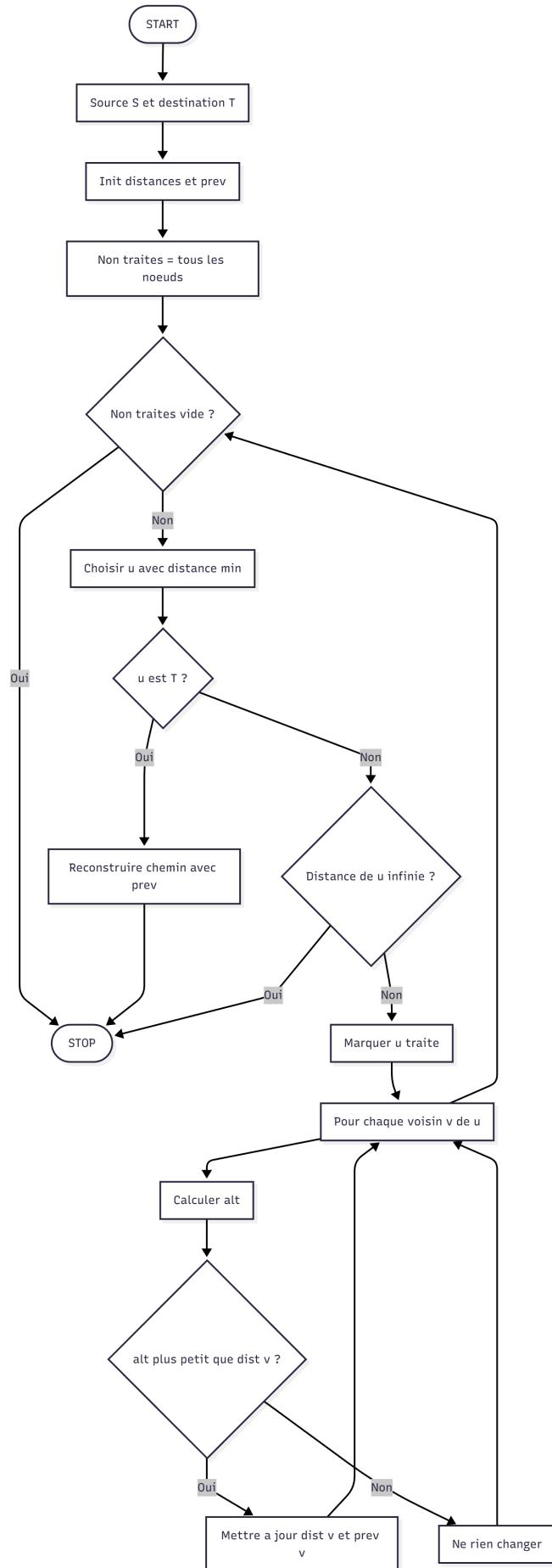
Sans heap : recherche linéaire du minimum

structure pas optimisé

- Tu mets tous les nœuds **pas encore visités** dans un **tableau**.
- À chaque tour, pour choisir  $u$ , tu dois **parcourir tout le tableau** et comparer les  $\text{dist}[\dots]$  pour trouver **le plus petit**.
- Comme tu regardes potentiellement  **$V$  nœuds** à chaque tour, ça coûte  **$O(V)$**  par tour (un balayage complet).

Et comme  $V + (V-1) + (V-2) + \dots + 1 = V(V+1)/2$  en notation O donne  $O(v^2)$  qui est la complexité sans heap.

Flowchart :



Voici ce que j'ai demandé à ChatGPT :

Donne-moi le graphe qui simulera mon trajet jusqu'au nouveau campus de l'ESIEA en passant par les lignes de métro. Voici les chemins possibles pour y arriver avec le format :

**ligne | direction | arrêt de départ | → temps (en min) → | arrêt d'arrivée.**

On ne prend en compte que le temps passé dans le métro.

Chemin possible 1 :

8 | Balard | Bonne Nouvelle | → 7 → | Invalides | → C | Massy | Invalides | → 16 -> | Ivry-sur-Seine |

Chemin possible 2 :

9 | Mairie de Montreuil | Bonne Nouvelle | → 4 → | Oberkampf | → 5 | Place d'Italie | Oberkampf | → 7 → | Gare d'Austerlitz | → C | Place d'Italie | Brétigny | → 6 → | Ivry-sur-Seine |

Fais une recherche sur le web pour ajouter d'autres moyens pour aller à la station Ivry-sur-Seine en métro/RER et ajoute-les à la liste.

Réponses

## Le graphe (prêt à "dijkstra-iser")

### Nœuds (stations)

Je te propose ces IDs :

- BN = Bonne Nouvelle
- INV = Invalides
- IVR = Ivry-sur-Seine (RER C)
- OBK = Oberkampf
- AUS = Gare d'Austerlitz
- SSD = Strasbourg-Saint-Denis
- SMN = Saint-Michel Notre-Dame
- REP = République
- BFM = Bibliothèque François Mitterrand

## Arêtes (trajets) — format

FROM --(ligne, direction, minutes)→ TO

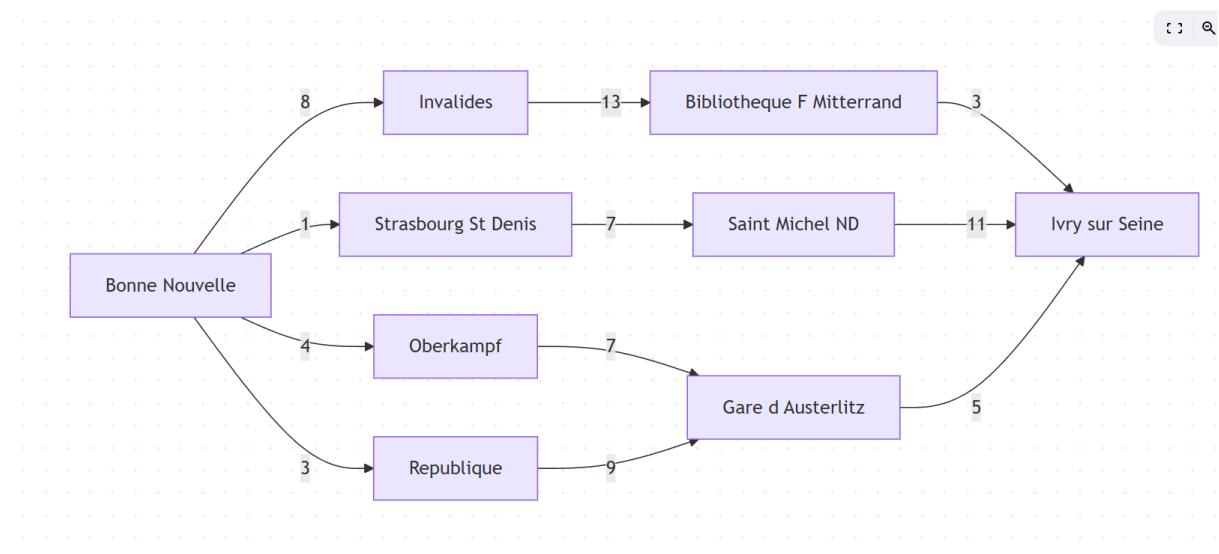
BN --(M8, "Balard", 8)→ INV  
BN --(M9, "Mairie de Montreuil", 4)→ OBK  
BN --(M9, "Mairie de Montreuil", 1)→ SSD  
BN --(M9, "Mairie de Montreuil", 3)→ REP

OBK --(M5, "Place d'Italie", 7)→ AUS  
REP --(M5, "Place d'Italie", 9)→ AUS  
SSD --(M4, "Bagneux – Lucie Aubrac", 7)→ SMN

AUS --(RER C, "via Bibliothèque F. Mitterrand", 5)→ IVR  
SMN --(RER C, "via Bibliothèque F. Mitterrand", 11)→ IVR

INV --(RER C, "vers Bibliothèque F. Mitterrand", 13)→ BFM  
BFM --(RER C, "vers Ivry-sur-Seine", 3)→ IVR

Voici le mermaid du graph pour qu'on y voit plus claire



Représentation graphique du graph sur carte RATP



Voici le résultat :

```
Ludo@LudoMusk:~/cours_prog/4A/Reseaux/TP_Dijkstra__bellmanford$ ./dijkstra
Chemin optimal (16 min) :
Bonne Nouvelle ->
Oberkampf ->
Gare d'Austerlitz ->
Ivry-sur-Seine
```

C'est donc ce chemin qui est le plus court.