Comparaison des trois algorithmes

Présentation des algorithmes :

Algorithme	Description
Dijkstra	Cet algorithme permet de calculer les plus courts chemins depuis un sommet source vers tous les autres.Il s'applique en cas de pondérations positives uniquement.
Bellman-Ford	Cet algorithme permet de calculer les plus courts chemins depuis un sommet source.Bellman-Ford gère les arêtes à poids négatifs mais pas les cycles négatifs.
Floyd-Warshall	Cet algorithme permet de calculer les plus courts chemins entre tous les couples de sommets. Il fonctionne avec des poids négatifs (et pas de cycles négatifs).

Complexité des algorithmes :

V : nombre de sommets et E : nombre d'arêtes

Algorithme	Complexité temps (graphes denses)	Complexité temps (graphes clairsemés)	Complexité espace
Dijkstra	O((V + E) log V)	O(E log V)	O(V)
Bellman-Ford	O(V × E)	O(V × E)	O(V)
Floyd-Warshall	$O(V^3)$	$O(V^3)$	$O(V^2)$

Définition : Un graphes clairsemés est un graphe qui contient peu d'arêtes par rapport au nombre de sommets possibles.

Rapidité d’exécution :

- **Dijkstra** est **très rapide** sur les graphes avec **beaucoup de sommets** et les graphes qui contient des **poids positifs uniquement**.

- **Bellman-Ford** est plus **lent**, car il répète le parcours du graphe V fois(le nombre de sommets). Il est utile si des arêtes négatives existent.

- **Floyd-Warshall** est **très lent** pour les grands graphes, mais efficace si on veut tous les plus courts chemins, surtout pour **des graphes de petite taille**.

Pertinence selon le cas d’application :

Contexte d’application	Algorithme recommandé	Justification
Plus court chemin depuis un seul sommet (poids positifs)	**Dijkstra**	Plus rapide et optimisé
Présence de poids négatifs sans cycles négatifs	**Bellman-Ford**	Gère les poids négatifs
Tous les plus courts chemins (dense, petite taille)	**Floyd-Warshall**	Donne toutes les distances entre paires de sommets
Détection de cycle négatif	**Bellman-Ford**	Peut signaler un cycle négatif
Graphes très grands et clairsemés	**Dijkstra**	rapide et optimisé

Bilan et recommandation :

- Pour des **grands graphes sans poids négatif**, **Dijkstra** est le plus adapté.

- En cas de **poids négatif**, **Bellman-Ford** est le seul applicable parmi les deux premiers.

- Pour **des besoins de calcul complet de toutes les paires de chemins les plus courts**, et pour un **graphe de taille moyenne**, **Floyd-Warshall** est plus simple à implémenter.

Pour notre cas d'étude :

Dijkstra est le plus rapide et le plus adapté dans ce contexte.

Critère	Analyse
Poids positifs	Les trajets entre stations de métro ont toujours un temps ou une distance positive. Dijkstra fonctionne donc parfaitement.
Source unique	On cherche le chemin le plus court entre deux points (cuisinier → client). Cas idéal pour Dijkstra.
Graphes peu denses	Le réseau de métro est plutôt clairsemé (chaque station n'a que quelques connexions). Dijkstra est très efficace.
Pas de poids négatifs	Le réseau de métro est plutôt clairsemé (chaque station n'a que quelques connexions). Dijkstra est très efficace.
Pas besoin de tous les chemins	On veut un trajet point-à-point, pas tous les couples. Floyd-Warshall serait trop lent et inutilement complet.

Algorithme	Complexité
Dijkstra	O(E log V)
Bellman-Ford	O(V × E)
Floyd-Warshall	$O(V^3)$

Conclusion :

Dijkstra est le plus rapide et le plus adapté dans notre contexte.