Alexandre DUFORET, Titouan Ducourau, Jeanne Espinos

**Analyses Algorithmes – Plus Court Chemin**

**/ Dijkstra**

L’algorithme de Dijkstra part d’un sommet source et étend progressivement le calcul des distances minimales à l’aide d’une file de priorité. Il est optimisé pour les graphes à poids positifs comme le notre puisque les temps de transports sont des valeurs positives. Pour des graphes de très grande taille, l’optimisation avec le tas devient critique pour la performance.

**Complexité en temps: O(mlogn)**

**/ Bellman-Ford**

L’algorithme de Bellman-Ford passe sur toutes les arêtes du graphe à plusieurs reprises (jusqu’à n − 1 fois) afin de converger vers la solution optimale. Il peut gérer des arêtes de poids négatif et détecter des cycles négatifs. Moins efficace que Dijkstra dans le cas où tous les poids sont positifs.

La complexité linéaire en nombre de sommets multiplié par le nombre d’arêtes peut devenir prohibitive pour de grands graphes.

**Complexité en temps : O(n×m)**, ce qui est nettement moins performant que Dijkstra pour des graphes à poids positifs.

**/ Floyd-Warshall**

L’algorithme de Floyd-Warshall est un algorithme de programmation dynamique qui calcule les plus courts chemins entre toutes les paires de sommets. Il met à jour de manière itérative une matrice de distances. Permet de calculer simultanément les plus courts chemins entre toutes les paires de sommets. Simple à implémenter pour des graphes de petite taille ou denses. N’est pas le choix optimal si l’on ne recherche que les chemins depuis une source spécifique.

**Complexité en temps :** O(n3), ce qui est acceptable pour des graphes de petite ou moyenne taille mais devient vite prohibitif pour de grands graphes.

Dijkstra est clairement l’algorithme le plus efficace pour une recherche de plus courts chemins à partir d’un sommet source lorsque toutes les arêtes ont un poids positif.