Rapport d'investigation sur l'algorithme de plus court chemin pour le réseau de la RTC

1.Formulation de la tâche

Ce TP se concentre spécifiquement sur l'optimisation d'un algorithme de plus court chemin. Les étudiants sont invités à remplacer l'implémentation existante de l'algorithme de Dijkstra dans la classe Graphe par une version substantiellement plus efficace. L'objectif est d'améliorer significativement les performances en termes de temps d'exécution, tout en respectant les exigences du professeur. En résumé, ce TP combine des aspects pratiques liés à la programmation et à l'optimisation d'algorithmes avec une composante réflexive, incitant les étudiants à documenter leurs choix, observations et conclusions dans un rapport structuré.

2.Examen des options possibles

Plusieurs algorithmes de plus court chemin existent :

- Dijkstra : Algorithme simple à implémenter avec une complexité en temps de O(|V| + |E| log |V|), où |V| est le nombre de sommets et |E| le nombre d'arcs. Cependant, il peut être inefficace pour les graphes denses.
- Bellman-Ford : Algorithme plus complexe avec une complexité en temps de O(|V| |E|), mais efficace pour les graphes avec cycles négatifs.
- Floyd-Warshall : Algorithme encore plus complexe avec un temps d'exécution en O(|V|^3), mais efficace pour trouver tous les chemins les plus courts entre tous les sommets du graphe.

Choix de l'algorithme :

Dans notre cas, on a un réseau RTC très dense donc l'utilisation de Floyd-Warshall va être trop complexe et prendra beaucoup de temps O(|V|^3). On sait aussi que ce réseau ne peut pas contenir des cycles négatifs donc l'utilisation de Bellman-Ford n'est pas nécessaire, et on essayons Bellman-Ford avec le tri acyclique ça me donne une amélioration de deux fois mais ceci n'est pas assez rapide ni efficace comparer à Dijkstra. Compte tenu des caractéristiques du réseau de la RTC, dense et sans cycles négatifs, Dijkstra semble être le choix le plus approprié et moins complexe.

3.Implémentation finale

L'implémentation optimisée de l'algorithme de Dijkstra comprend les étapes suivantes :

Initialisation des distances : Tous les sommets sont initialisés à une distance maximale sauf l'origine, dont la distance est mise à zéro.

Boucle principale :

a. Trouver le sommet non visité avec la distance la plus courte.

b. Marquer le sommet trouvé comme visité.

c. Pour chaque voisin non visité du sommet trouvé :

- Mettre à jour la distance du voisin si elle est plus courte que la distance actuelle plus le poids de l'arc.

Reconstruction du chemin : À partir du sommet de destination, remonter les sommets parents jusqu'à l'origine.

Améliorations de l'algorithme :

Pour optimiser l'algorithme de Dijkstra, les changements suivants ont été apportés :

Remplacement de la File de Priorité : La file de priorité standard a été remplacée par une implémentation manuelle d'un tas binaire.

Intégration de la Classe BinaryHeapNode : Une nouvelle classe a été ajoutée pour encapsuler les informations nécessaires dans chaque nœud du tas binaire.

Restructuration des Opérations sur le Tas Binaire : Deux fonctions, pushToBinaryHeap et popFromBinaryHeap, ont été introduites pour gérer les opérations d'insertion et d'extraction du minimum dans le tas binaire.

Ces optimisations améliorent collectivement l'efficacité de l'algorithme de Dijkstra, le rendant adapté au calcul de chemins en temps réel dans le réseau de transport de la RTC. En minimisant les calculs inutiles et en utilisant des structures de données efficaces, l'algorithme optimisé fournit des calculs d'itinéraires plus rapides et plus réactifs.

4.Résultats et analyse

L'algorithme optimisé de Dijkstra a montré des performances significativement améliorées par rapport à l'implémentation originale telles que :

Pour la moyenne du temps d'exécution sur 200 itinéraires :

L'algorithme optimisé donne une moyenne de 789.77 microsecondes comparé à 33410.5 microsecondes de l'algorithme original : ce qui est 33410.5/789.77=42.3 fois plus rapide.

Optimisé :

```
La moyenne du temps d'exécution sur 200 itinéraires est de 789.77 microsecondes

RUN FINISHED; exit value 0; real time: 29s; user: 630ms; system: 29s
```

Original :

```
La moyenne du temps d'exécution sur 200 itinéraires est de 33410.5 microsecondes

RUN FINISHED; exit value 0; real time: 36s; user: 160ms; system: 36s
```

Cette optimisation peut se voir aussi dans le temps d'exécution du plus court chemin prenant par exemple le test 0 :

```
Test numéro 1
station du point origine = 1-3963 - Dandrieu (lat:46.8039, long:-71.306)
station du point destination = 1-1133 - St-Jean/1133 (lat:46.8145, long:-71.2096)
distance = 7.42667 kilomètres

====================
     ITINÉRAIRE
====================

Heure de départ du point d'origine: 07:30:00
Rendez vous à la station 1-5256 - Lahaye (lat:46.8168, long:-71.3063)
De cette station, prenez l'autobus numéro 279 à l'heure 07:55:55 Vers Colline Parlementaire (Sud)
et arrêtez-vous à la station 1-1190 - D'Youville (lat:46.8117, long:-71.2154) à l'heure 08:16:00
Déplacez-vous à pieds de cette station au point destination
Heure d'arrivée à la destination: 08:22:27
Durée du trajet: 0 heures, 52 minutes, 27 secondes
Temps d'exécution de l'algorithme de plus court chemin: 444 microsecondes
```

Optimisé :

Original :

```
Test numéro 1
station du point origine = 1-3963 - Dandrieu (lat:46.8039, long:-71.306)
station du point destination = 1-1133 - St-Jean/1133 (lat:46.8145, long:-71.2096)
distance = 7.42667 kilomètres

====================
     ITINÉRAIRE
====================

Heure de départ du point d'origine: 07:30:00
Rendez vous à la station 1-5256 - Lahaye (lat:46.8168, long:-71.3063)
De cette station, prenez l'autobus numéro 279 à l'heure 07:55:55 Vers Colline Parlementaire (Sud)
et arrêtez-vous à la station 1-1190 - D'Youville (lat:46.8117, long:-71.2154) à l'heure 08:16:00
Déplacez-vous à pieds de cette station au point destination
Heure d'arrivée à la destination: 08:22:27
Durée du trajet: 0 heures, 52 minutes, 27 secondes
Temps d'exécution de l'algorithme de plus court chemin: 7690 microsecondes
```

Dans le code original on trouve 7690 microsecondes dans le code amélioré on trouve 444 microsecondes. Nous remarquant une amélioration de 17.3 fois.

Ceci se reproduit pour tous les autres tests avec plus au moins cette même efficacité, par manque d'espace je n'inclurai pas les autres tests.

5.Conclusions et recommandations

L'algorithme de Dijkstra optimisé est un choix efficace pour trouver le plus court chemin dans le réseau de la RTC. Des améliorations supplémentaires pourraient inclure l'utilisation de techniques de partitionnement du graphe et d'heuristique pour réduire le nombre d'arcs explorés, utiliser une structure de données de type arbre binaire de recherche pour stocker les sommets non solutionnés. Cela permettrait de trouver le sommet avec la distance la plus courte en temps constant.

Utiliser une stratégie de relâchement des arcs qui tienne compte de la distance du sommet source au sommet adjacent. Cela pourrait permettre de réduire le nombre d'itérations de l'algorithme.