IN104

Rapport du projet "Un jour au ski"

Tom SAVARD et Alexandre LARRY

23/05/2023

Table des matières

1 Analyse du problème

1.1 Présentation du problème

Le problème se présente sous la forme d'une carte comprenant un ensemble de pistes, chaque piste étant associée à un nombre représentant le plaisir qu'elle procure. Il est possible que certaines pistes génèrent un plaisir négatif si elles sont considérées comme ennuyeuses. Les croisements des pistes sont numérotés, avec le point de départ étant le croisement numéro 0.

L'objectif est de trouver l'itinéraire qui permet de maximiser la quantité totale de plaisir tout au long de la journée. Il est permis de parcourir plusieurs fois la même piste, offrant ainsi un plaisir infini. De plus, il est également possible de décider de ne pas skier du tout si cela s'avère être la meilleure solution.

1.2 Difficultés

La présence de cycles négatifs dans les entrées du problème peut poser un défi lors de la résolution. La détection de ces cycles est cruciale pour éviter des boucles infinies et des résultats erronés. L'algorithme de détection de cycles de Bellman-Ford peut être utilisé pour cette tâche contrairement à celui de Dijkstra (initiallement envisagé).

En ce qui concerne l'implémentation en C, il est important de choisir des structures de données efficaces pour représenter les connexions entre les pistes. L'utilisation de structures telles que la matrice d'adjacence ou la liste d'adjacence peut faciliter la détection des cycles négatifs et minimiser la complexité temporelle de l'algorithme.

2 Explication de la solution

2.1 Méthodes utilisées

L'algorithme de Bellman-Ford a été choisi pour résoudre le problème de la journée de ski en raison de sa capacité à gérer les poids négatifs et les cycles négatifs, ce que l'algorithme de Dijkstra ne permet pas. Il a donc fallu d'autre part adapter l'algorithme Dijkstra pour obtenir des résultats sur les entrées avec des cycles négatifs. Ces difficultés avaient été évoquées en IN103.

2.2 Architecture du projet

Pour le dossier programme bellman :

— `main_bellman.c` : Ce fichier contient la fonction `main` qui est le point d'entrée du programme. Il initialise les données d'entrée, appelle les fonctions nécessaires pour résoudre le problème de l'itinéraire de ski, et affiche le résultat.
— `bellman.c` : Ce fichier contient les fonctions liées à l'algorithme de Bellman-Ford utilisé pour trouver l'itinéraire optimal. Il comprend notamment la fonction `bellman` qui implémente l'algorithme de Bellman-Ford et renvoie la quantité maximale de plaisir obtenue.
— `matrice.c` : Ce fichier contient les fonctions liées à la manipulation de la matrice représentant les connexions entre les croisements des pistes. Il peut inclure des fonctions pour initialiser la matrice, récupérer les poids des pistes, etc.

— `lecture.c` : Ce fichier contient les fonctions de lecture des données d'entrée à partir d'un fichier ou d'une autre source. Il peut comprendre des fonctions pour extraire les informations sur les croisements, les pistes et les plaisirs associés.
— `utils.c (dijkstra, glouton et brute)` Ce fichier contient de simples fonctions telles que la recherche de l'indice du max d'une liste, l'affichage d'une liste ou encore le nombre de True dans un tableau binaire. Enfin, test utils. est un test unitaire sur la fonction de recherche du max. À voir en fin du readme.

L'organisation est analogue pour le dossier programme dijkstra

3 Présentation des résultats

3.1 Performance

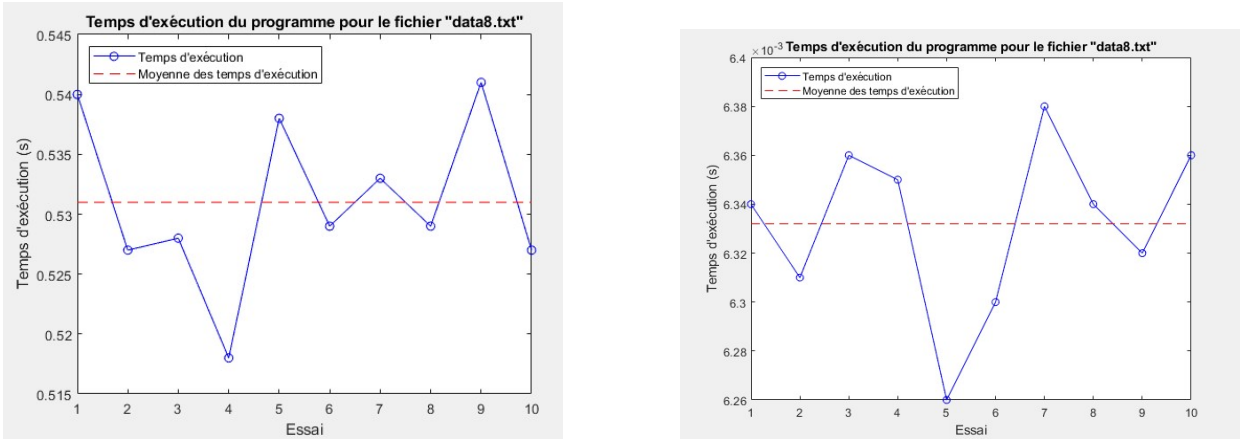

FIGURE 1 – Résultats sur l'algorithme utilisant bellman (à gauche) et dijkstra (à droite)

3.2 Analyse

Pour la performance temporelle, on observe un écart net entre les deux implémentations. Les deux algorithmes s'appuient sur la même méthode. Cependant le «dijkstra » (nom de base mais qui au final s'éloigne de la méthode de dijkstra) parcours les sommets de proches en proches tandis que « bellman » regarde toutes les pistes imaginables. Ainsi l'algorithme dijkstra permet de minimiser les étapes de calculs en gardant uniquement les pistes pertinentes.

Pour ce qui est de la complexité spatiale, seul le premier algorithme a des résultats convaincants. En effet, on observe parfois un problème d'allocation mémoire avec le second dont on n'a pas réussi à détecter la source.

4 Extension (problème du voyageur)

Le problème se présente sous la forme d'une carte comprenant un ensemble de villes, chaque ville étant associée à des coordonnées. L'objectif est de trouver l'itinéraire qui permet de minimiser la distance tout en passant une et une seule fois par chaque ville. Lors de cette étude, uniquement des graphes complets ont été pris en compte. La distance entre les villes est celle de la norme L2.

4.1 Méthodes possibles

L'interet de cette extension réside dans son absence de résolution « parfaite ». En effet, le premier problème était de classe P or celui-ci est NP-Complet. Ce problème amène donc à une question d'optimisation et de balance entre performance temporelle et performance en précision.

Il existe différentes méthodes classiques pour résoudre ce type de problème. On peut notamment citer la méthode gloutonne, la brute force ou encore les algorithmes génétiques.

4.2 Notre choix pour répondre au problème

Ici deux algorithmes sont étudiés : le glouton et la brute force. Ce choix est motivé par leur balance complexité/précision qui sont totalement opposées. En effet, l'algorithme glouton fonctionne sans mémoire et recherche à chaque itération l'optimal local. Il favorise donc la complexité à la précision.

À l'opposé, la brute force considère toutes les combinaisons possibles avant de déterminer celle permettant la distance minimale. On a donc la solution la plus précise possible cependant la complexité et en $O(n!)$. Le temps de calculs explose totalement à peine la dizaine de ville dépassée.

4.3 Résultats

L'algorithme brute force n'a pas pu être achevé. Cependant le glouton est opérationnel. Comme attendu on obtient un résultat très rapide qui pour des cas simples présente une solution tout à fait correcte. En revanche, cet algorithme ne garantit en rien la précision dans des situation plus complexes.