HAMMAD Amir
SARRAZIN Alexandre

Projet Algorithme

Voici la carte avec les nombres que nous avons donnés à nos sommets.

Nous avons décidé de programmer notre projet en langage C car c'est le langage où nous sommes le plus à l'aise. Le Java aurait pu être une bonne option pour ce projet mais comme nous avons commencé le Java en début de semestre, nous avons moins de connaissances dessus. Le projet a été fait sous eclipse.
Pour compiler avec le terminal : gcc -Wall algo.c -o algo
Pour exécuter : ./algo

Présentation base de données :

Le fichier algo.h va nous servir à définir tout ce dont nous aurons besoin. Nous avons une énumération de chemins qui va contenir tous les types de chemins possibles, une enumération pour le niveau du skieur. Avec ses différentes énumérations, nous pouvons rajouter facilement un type de piste ou un autre moyen de transport pour se rendre d'un sommet à un autre(exemple : un hélicoptère).
Les define, présents à la fin du fichier algo.h, nous servent à écrire plus facilement les différents chemins lors de l'initialisation du plan des piste.

Il contient aussi les structures de notre projet. Nous avons mis en place une structure de chemin qui va contenir le début et l'arrivée d'un chemin(voir enumération chemin), le type de chemin sur lequel on est, son nom qui est un tableau de caractères et la longueur de notre chemin.
Nous avons aussi créer une structure de graphe qui va contenir un tableau d'arête(arête est une structure aussi) et le nombre de sommet de notre graphe. Cette structure arête va contenir un matrice d'adjacence de type chemin qui va contenir des chemins.
Nous avons décidé d'utiliser des structures avec des matrices et des tableaux car nous avons utilisé l'algorithme de Dijkstra. La particularité de cet algorithme est qu'il s'implémente bien avec les structures de données que nous avons choisi.

Pour nous la chose la plus compliqué à faire dans ce projet a été de trouver une bonne base de données et surtout de bien l'adapter au problème.

Exemple de la première idée que nous avons eu :

```
typedef struct{
	char nomP[30];
	char difficulte;
	int longueur;
}piste;

typedef struct{
	char nomR[30];
	int longueur;
	char type[30];
}remontee;

typedef struct {
	remontee r;
	piste p;
	int check;
}chemin;

typedef struct {
	int nombre_sommet;
	chemin ** tab;
} graphe;
```

Mais nous nous sommes rendus compte qu'il y avait un problème si deux pistes partent du même sommet et arrive au même sommet. En effet dans notre matrice, la deuxième piste aurait supprimé la première piste vu qu'elle irait dans la même case. Donc cette idée n'était pas du tout adéquate au projet.

Présentation du code :

La fonction ij2tab nous permet de convertir i et j en index d'un tableau.

Dans notre .c, nous avons initialiser toute nos pistes dans un tableau de chemin. Grâce aux defines, écrire nos chemins a été plus simple et plus rapide.

Au début du programme, nous devons initialiser notre graphe qui est une matrice d'arêtes. Le calloc nous permet de mettre toutes les cases à la valeur 0. Tant que le type des chemins n'étaient pas hors pistes, on ajoute les chemins contenus dans le tableau à notre graphe.

Nous avons crées une fonction pour afficher le graphe qui nous permet de voir toutes les pistes contenues dans le graphe mais nous l'avons mis en commentaire pour que l'affichage soit moins chargé.

Ensuite nous avons dû implémenter Dijkstra dans notre code, nous nous sommes servis de l'algorithme que nous avons vu en amphithéâtre et de ce que nous avons appris en td. En implémentant cette algorithme, nous avons aussi fait des tests à la main pour voir si l'algorithme trouvait bien le même chemin que nous. En implémentant Dijkstra, nous avons dû créer des fonctions annexes comme celle qui regarde si un sommet a déjà été visité et la fonction qui calcule la durée minimale.
La fonction int * parcours nous sert a inverser les sommets dans notre tableau. La première boucle va compter le nombre de sommets qu'il y a et la deuxième boucle va créer un tableau de sommets en partant de l'arrivée pour arriver au départ.
Pour afficher le parcours, nous devons faire comme dans la fonction pour afficher le graphe sauf que ici on doit prendre les sommets compris dans le parcours.

Dans le main, il y a les différentes questions que nous devons poser au skieur. Et l'affichage du parcours.

Ce qui nous a posé problème est l'affichage du parcours avec les noms de pistes car nous devons avoir une liste de sommets et regarder le sommet A avec le sommet B pour voir quel chemin se trouve entre les deux, puis regarder le sommet B avec le sommet suivant et ainsi de suite. Et nous devons aussi choisir la piste moins longue si il y a plusieurs pistes entres les mêmes sommets.

Problème que nous avons rencontrés:
- oubli d'écriture de piste ce qui nous donné des sommets inaccessibles
- nous avons essayé de faire au mieux pour donner les longueurs aux chemins.
- pour les sommets, nous ne savions pas si il fallait leur donner des noms ou juste des numéros. Nous avons opté pour des nombres.