SABRY Soumaya
STEFF Alban

Rapport de Projet de Structure de données et algorithmes

Sujet :

« Création d'un arbre des trajets »

Alban STEFF & Soumaya SABRY
TD {N}

Table des matières

SABRY Soumaya
STEFF Alban

Explication de la structuration des données en mémoire :

```
typedef struct _connection
    {
        duree temps_trajet;
        int kilomettre;
        ville nom_connex;
        struct _connection* next ;

    }connection;

typedef struct
{
    ville nom_de_la_ville;
    connection* list_connections;
    int nb_de_connection;
}Case;
```

Pour récupérer les informations contenues dans les fichiers .csv, on utilise deux structures distinctes :

- connection : contient les informations relatives à une ville (la durée et la distance pour y aller depuis la ville précédente, le nom de la ville et la liste chaînée des connexions possibles)
- Case : permet de stocker dans un tableau les données concernant les connexions.

Note : la structure 'duree' contient simplement une variable heure et une variable minute et l'énumération 'ville' les noms des villes.

Explication de la structure créée pour l'arbre :

```
typedef struct _noeud
{
    Case position_actuelle;
    int distance_parcourue_cumule;
    duree duree_trajet_cumule;
    ville* villes_visitees; // tableaux de ville visitee
    int nb_villes_visitees;
    struct _noeud** suite_trajet;
} noeud;
```

L'avantage de cette structure c'est de pouvoir avoir toutes les informations nécessaires à l'utilisateur (distance totale du trajet, durée totale, toutes villes visitées en ordre) directement à la fin de l'arbre (dans les feuilles), sans avoir à parcourir à nouveau l'arbre en entier.

Cette structure permet de créer un nœud de l'arbre des trajets.

Elle contient une case du tableau ou est stoker nos donne des ficher csv, dans laquelle il y a le nom du nœud et la liste chaine de ses connexions.

Elle contient la distance parcourue pendant tout le trajet et la durée totale. De plus, on utilise un tableau de villes visitées pour savoir par où on est déjà passé.

Enfin, pour connaître la suite du trajet, on se sert d'un tableau de pointeurs (suite_trajet**), qui correspond au fils d'un nœud, ce qui permet d'allouer directement le trajet selon le nombre nécessaire des connexions du nœud et retire les villes déjà visitées.

Explication de la structure créée pour la récupération des trajets :

```
typedef struct
{
    int distance_parcourue_cumule;
    duree duree_trajet_cumule;
    ville* villes_visitees;
    int nb_villes_visitees;
} trajet ;
```

Cette structure permet de stocker un trajet :

Elle contient sa distance et durée, un tableau des villes visitées (bien en ordre) et le nombre de villes visitées pour ne pas dépasser la limite du tableau.

L'arbre est formé des feuilles qui peuvent être ou pas notre destination finale. Afin de comparer les distances et durées des trajets, on stocke tous les trajets qui mène a notre ville d'arrive dans un tableau de trajets.

SABRY Soumaya
STEFF Alban

Construction de l'arbre des trajets

On utilise d'abord une fonction pour créer la racine de l'arbre, soit la case avec la ville de départ. Les paramètres sont : «tab » correspondant au tableau de Case où est structuré toutes les données des fichiers csv, la ville de départ, la ville d'arrivée et taille maximale de l'arbre demandée à l'utilisateur ;

- noeud* creation_racine(Case* tab, ville destination_Final, ville villeDeDepart, int trajetMAX)

On utilise cette fonction pour créer la base de l'arbre et donc tout est initialisé à l'intérieur (la distance et le temps à zéro, le nombre de villes visitées à 1).

Ensuite, on remplit le tableau de pointeurs (ses fils) à l'aide de la fonction suivante :

- noeud* ajouter_noeud(Case* gdTab , noeud* parent , connection ville_a_ajouter, ville destination_Final , int trajetMAX)

Cette fonction permet d'ajouter un nouveau nœud dans l'arbre. Elle a pour objectif d'initialiser une structure nœud en tenant compte du nœud parent et donc en adaptant ses attributs et la suite du trajet selon ce qui s'est passé auparavant. C'est à dire qu'on va tester si dans le tableau de villes visitées, la ville qu'on veut ajouter est déjà dedans, et dans ce cas-là on n'ajoute pas, dans le cas contraire on ajoute. Ou si la taille de l'arbre atteint la taille maximale, ou le nom du nœud est égal a celui de la destination finale (cad nous somme arrivés) dans ce cas-là on n'ajoute pas, dans le cas contraire on ajoute.

Pour afficher l'arbre on a utilisé la fonction affiche arborescence ;

- void affichage_arb(noeud const * arbre, int offset);

Enfin on libère la mémoire de l'arbre en utilisant - void liberation_arbre(noeud * arbre);

Manipulation de l'arbre pour en tirer les informations

On utilise la logique de récursivité pour parcourir l'arbre en utilisant la condition si c'est une feuille (cad son pointeur de fils est NULL), et la condition que le nom de cette feuille est bien notre destination finale, on stocke les variables de cette feuille dans une ligne du tableau de la structure trajet mis en paramètre « trajet * tabDeTrajet », et on ajouter la « tailledutab » pour savoir le nombre de trajet à stoker au total, à l'aide de la fonction suivante :

- void interprete_arbre(noeud const * arbre, ville destination, trajet * tabDeTrajet, int* tailledutab);

Après avoir rempli notre tableau de trajet, on utilise la fonction suivante pour comparer toutes les distances / durées des trajets dans le tableau et afficher le plus court / rapide. Cette fonction gère aussi le cas où il y a des trajets différents mais avec le même distance / durée. En paramètre, on a « choix » pour le choix du plus court ou du plus rapide :

- void distance_duree_min(noeud * arbre, int choix, ville arrivee);

Finalement on utilise la fonction suivante pour récupérer le tableau de trajet et engendrer un entier aléatoire de 0 à la taille du tableau pour afficher le trajet aléatoire.

- void trajet_alea(noeud * arbre, ville destination);

SABRY Soumaya
STEFF Alban

Description du menu

Pour le choix des villes on utilise un curseur* qui bouge avec les « key up » et « key down » du clavier, en arrivant à la ville souhaiter, on appuie sur « Entre ». Pour cela ***uniquement***, nous avons utilisé les bibliothèques *<conio.h>* et *<windows.h>*.

Ensuite on demande à l'utilisateur la hauteur maximale de l'arbre des trajets dont son choix est borné en 2 et 23 (c'est le nombre des villes). Avec ses informations, on construire notre arbre, et on propose à l'utilisateur tous les choix présentés ci-dessus :

1) Un trajet aléatoire est affiché entre la ville de départ et la ville d'arrivée.
2) Tous les trajets existants sont affichés, avec toujours la condition de 8 villes visitées maximum par défaut (paramètre pouvant être changé par l'utilisateur).
3) Le trajet le plus court en kilomètres
4) Le trajet le plus court en temps
5) Chaque trajet est affiché sur un arbre des trajets que l'utilisateur peut lire

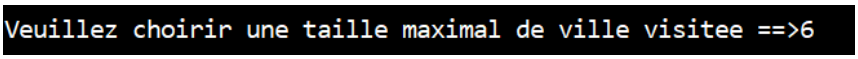
Comme mesure de sécurité, on a fait une fonction « saisir_entier_borne » pour gère tout cas ou l'utilisateur saisi un chiffre pas parmi les choix ou saisi une lettre au lieu d'un chiffre, le programme ne bug pas, il redemande la question.
On utilise souvent system("cls") pour effacer la console et bien présenter le programme.
Enfin, tout le menu est mis dans un WHILE pour refaire un autre parcouru si souhaitez sinon ça quitte.

*c'était un code proposé par Julien PYTEL -un collègue de la promo – et il l'a mis sur Yammer.

Exemple d'affichage

On commence par demander la ville de départ et celle d'arrivée en utilisant un curseur :

Ensuite, on demande la taille maximale de l'arbre ;



SABRY Soumaya
STEFF Alban

Ce qui nous mène aux fonctionnalités proposées décrites ci-dessus ;

```
VOICI LES FONCTIONNALITES PROPOSEES PAR CE PROGRAMME :
-> 1 : Affichage d'un trajet aleatoire
-> 2 : Affichage de tous les trajets possibles
-> 3 : Affichage du ou des trajets les plus courts (en distance)
-> 4 : Affichage du ou des trajets les plus rapides (en temps)
-> 5 : Affichage de l'arbre des trajets
Faites votre choix !
```

1- Trajet Aléatoire

```
Vous avez choisi : Affichage d'un trajet aleatoire

Distance du trajet : 1168 km    Duree totale : 12H22    Trajet : Caen, Amiens, Paris, Bourges, Clermont_Ferrand, Montpellier
```

2- Tout trajet possible

```
Vous avez choisi : Affichage de tous les trajets possibles
Distance | Duree        |            Trajet
-------------------------------------------------------------------------------------------------
1168     | 12H22        | Caen, Amiens, Paris, Bourges, Clermont_Ferrand, Montpellier

1214     | 12H51        | Caen, Amiens, Paris, Dijon, Lyon, Montpellier

951      | 09H53        | Caen, Le_Mans, Tours, Bourges, Clermont_Ferrand, Montpellier

1143     | 11H46        | Caen, Le_Mans, Paris, Bourges, Clermont_Ferrand, Montpellier

1189     | 12H15        | Caen, Le_Mans, Paris, Dijon, Lyon, Montpellier

Tous les trajets possibles ont ete affiches !
```

5- l'arbre

```
Vous avez choisi : Affichage de l'arbre des trajets
|-Caen
        |-Amiens
                |-Lille
                        |-Reims
                                |-Metz
                                        |-Strasbourg
                                |-Paris
                                        |-Bourges
                                        |-Dijon
                                        |-Le_Mans
                |-Paris
                        |-Bourges
                                |-Clermont_Ferrand
                                        |-Bordeaux
                                        |-Lyon
                                        |-Montpellier
                                |-Tours
                                        |-Le_Mans
                                        |-Nantes
                                        |-Niort
                        |-Dijon
                                |-Lyon
                                        |-Clermont_Ferrand
                                        |-Grenoble
                                        |-Marseille
                                        |-Montpellier
                                |-Strasbourg
                                        |-Metz
                        |-Le_Mans
```

3-Trajet le plus court

```
Vous avez choisi : Affichage du ou des trajets les plus courts (en distance)

Distance du trajet : 951 km     Duree totale : 09H53    Trajet : Caen, Le_Mans, Tours, Bourges, Clermont_Ferrand, Montpellier
```

4-Trajet le plus rapide

```
Vous avez choisi : Affichage du ou des trajets les plus rapides (en temps)

Distance du trajet : 951 km     Duree totale : 09H53    Trajet : Caen, Le_Mans, Tours, Bourges, Clermont_Ferrand, Montpellier
```

Après avoir saisi une des cinq options, le programme demande si l'utilisateur veux refaire un autre essai et si oui ou non avec les mêmes villes déjà choisies :

```
Voulez vous effectuer un autre parcours ? Tapez 1 si oui et 0 sinon
1
Voulez vous garder les villes de depart et d'arrivee ? Tapez 1 si oui et 0 sinon
1
```

Enfin, le programme tourne en boucle jusqu'à ce que l'utilisateur choisisse de ne pas refaire un autre parcours.