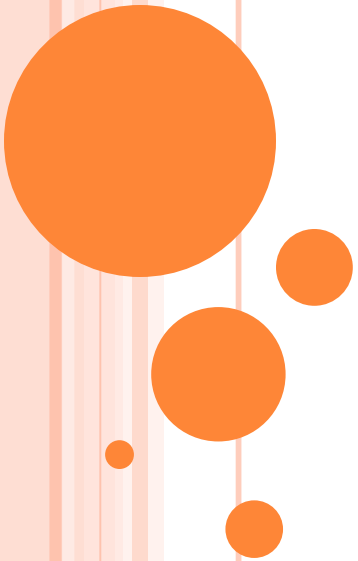
Rapport du projet d'algorithmique

METRO C'EST TROP !



Membres du groupe :

- LABCHRI Koceila ------- 219 19 396
- AOUCHICHE Salah------ 219 21 706
- MAAMRI Fouzi ---------- 219 19 839

Plan du rapport :

- Présentation globale du projet.
- Mise au point sur les différents fichier du projet.
- Eclaircissement sur les structures utilisées.
- Mise en avant des fonctions clés du projet.
- Exemple pratique de fonctionnement.
- Croquis de l'interface graphique à réaliser et explications.
- Conclusion.

Présentation globale du projet :

- Le projet que nous avons réaliser à pour but de tiré part de l'algorithme de Dijkstra vu en cours et l'implémenter sur un problème réel qui est « La circulation des métro parisiens ».
- Nous avons donc pour ce fait programmer un code en C ayant pour but de stocker chaque sommet des différentes lignes de métro sous forme d'un graphe connexe, pour ensuite appliquer notre algorithme du plus court chemin afin de trouver la solution optimale.

Mise au point sur les différents fichier du projet :

- Metro.txt : contient la liste des différents sommets fournis et des arêtes, auxquels nous avant rajouté ces informations :

➡ Pour chaque sommet on associe un numéro de ligne.

Si la station correspond à une des lignes (3bis ou 7bis) on note (30 , 70) respectivement.

➡ Pour chaque arête on associe les deux terminus correspondant au trajet.

Dans le cas d'un changement de ligne (on est toujours dans la même station) on insère deux terminus représentatif (1111 , 1111).

- Lecture.h: contient les signatures de fonctions utilisés dans le fichier lecture.c.
- Lecture.c : contient la partie du code qui se chargera de la lecture du fichier metro.txt et du remplissage du graphe.

- Graphe.h : contient les différentes structures manipulés dans notre conception du graphe (sommet, arc , graphe) en plus de la structure Dijkstra et une structure de liste ‘chemin’ représentant notre plus court chemin. Contient aussi les signatures des fonctions manipulés dans graphe.c .
- Graphe.c : contient une liste de fonctions de manipulations de structures (initialisation , libération de mémoire , primitives d’utilisation d’une liste …) et notre fonction clé du plus court chemin renvoyant notre structure Dijkstra rempli.

Eclaircissement sur les structures utilisées.

```
struct sommet{
    int num_station; // identifiant du sommet
    char nom_station[TAILLE_NOM];
    int num_metro ;
};
```

Comme décrit dans metro.txt un sommet contiendra un num_station , un nom_station et un num_metro.

```
struct arc {
    int successeur_finale ; // identifiant du succésseur finale
    int poid ;
};
```

Ici la structure arc nous servira surtout à créer une matrice avec comme indices les identifiants de nos sommets (en effet c'est un champ dans la structure graphe).

struct graphe {

SOMMET *tab_sommets;

ARC **tab_arcs; // notre matrice dont on vient de parler.

int nb_sommets;

}

-Cette structure représente le cœur du projet en effet on stock tout nos sommets dans le tab_sommets, les arcs dans le champs tab_arc, et le nombre de nos sommets (utilisation technique) dans le champs correspondant.

struct chemin{

SOMMET station;

struct chemin *suiv;

}

-Pour la structure chemin il n' y a pas grand chose à notifier, celle-ci servira a stocker le plus court chemin que l'on récupérera avec notre algorithme de Dijkstra sous forme d'une liste.

struct dijkstra{

int id_debut ;

int id_fin ;

CHEMIN *plus_court; // le plus court chemin

int duree;

}

-La structure Dijkstra représente l'aboutissement final du projet elle prend 2 champs représentant l'indice du sommet de départ et le sommet d'arrivé et en appliquant notre algorithme sur notre graphe on abouti au remplissage du champs « plus_court » et du calcul de la durée du trajet par la même occasion.

Mise en avant des fonctions clés du projet:

- Les fonctions de remplissage de graphe dans lecture.c :

  GRAPHE lecture_sommet(char *fichier, GRAPHE g);

  GRAPHE lecture_arcs(char *fichier,GRAPHE g) ;

- La fonction de calcul du plus court chemin divisé en 2 partie :

  DIJKSTRA court_chemin (GRAPHE g, int deb , int fin)

Consiste essentiellement à trouver le plus court chemin menant d'un point 'deb' à un point 'fin', en effet, fin de fonction on dispose d'un tableau des pères qu'il nous suffit de parcourir et d'ajouter à chaque fois le sommet correspondant dans notre liste CHEMIN pour faire sortir notre chemin. Ce traitement est fait dans une fonction à part :

DIJKSTRA remplir_chemin(DIJKSTRA D , GRAPHE g,int tab_pere[]);

- void affichage_trajet(DIJKSTRA D,GRAPHE G);

  Fonction d'affichage qu'on a implémenté pour remédier à l'absence de l'application graphique (même si ce n'est pas le même niveau de rendu).

Exemple pratique de fonctionnement :

- Après avoir lancé le makefile dans le terminale, l'utilisateur devra saisir deux noms de station, si le nom de station est dupliqué plusieurs fois il devra spécifier le numéro de ligne correspondant à sa requête.
- Le programme affichera alors le chemin plus court de la station de départ à la station d'arrivée tout en calculant la durée.

Croquis de l'interface graphique à réaliser :

- Nous n'avons malheureusement pas eu le temps d'implémenter notre interface graphique par soucis de manque de temps en vu des différents projets à faire dans les différents modules.
- L'interface graphique que nous voulions implémenter se serait devisé donc en 3 parties essentielles :

  Saisie de la station de départ et d'arrivée équipée d'une liste déroulante permettant à l'utilisateur de sélectionner directement la station souhaitée.

  Un plan du métro parisien avec une vue plus détaillée sur notre chemin à parcourir une fois le résultat du plus court chemin obtenu.

  Illustration du chemin à parcourir avec itinéraire détaillé avec un avertissement lors des changements de lignes, et affichage de la durée totale et spécifique entre les différentes stations.

  La rubrique information nous permettrait par exemple d'afficher des avertissement sur les différentes perturbations qui pourraient survenir dans notre trajet.

Conclusion :

- Nous avons dans ce projet traiter tout les points demandés par l'énoncé mise à part l'implémentation de l'interface graphique.
- Nous espérons que ce rapport ait été suffisamment complet pour vous permettre de comprendre les détails du projet sans pour autant vous plonger dans chaque détail du code.
- Nous avons par ailleurs rajouter des commentaires un peu partout dans le code, nous espérons qu'ils sont assez clair.