Diego Castillo
Timothée Kocev
PET-C

Rapport final projet info 2018

1/Implantation **1**
1.1.Etat du logiciel 1
1.1.1.GUIDE DES RÉFÉRENCES DE LIBRAIRIES 1
1.2.Tests effectués 2
1.2.1 Test lecturefichier/afficheGraphe 2
1.2.2. test du Dijkstra 2
1.4.Les optimisations et les extensions réalisées 5
2/Suivi **5**
1.Problèmes rencontrés 5
2.Planning effectif 5
3.Qu'avons nous appris et que faudrait il de plus ? 5
4.Suggestion d'améliorations du projet 6
3/Conclusion **6**

1/Implantation

1.1.Etat du logiciel

Construction du graphe :
La fonction qui construit un graphe à partir de la lecture des fichier .csv et .txt fonctionne. Elle a été testée et éprouvée sur les petits fichier comme les gros.
Fonction de test : une fonction à été réalisée pour tester la bonne construction du graphe
Réseau routier américain :
Le calcul du plus court chemin a été implanté à l'aide de l'algorithme de Dijkstra. Il est fonctionnel sur les petits fichiers (monfichier.txt,graphe2.txt). En revanche sur les graphes volumineux, l'exécution du Dijkstra rencontre des problèmes au niveau de l'allocation en mémoire.
Métro et RER Parisien :
Suite à divers contretemps la gestion du métro n'est pas assurée

1.1.1.GUIDE DES RÉFÉRENCES DE LIBRAIRIES

Dans cette rubrique retrouvez les prototypes des fonctions réalisées lors de ce projet

graphe.h

L_ARC **creerListeArc**();
int **listeArcVide**(L_ARC lArc)
void **visualiserListeArc**(L_ARC)
L_ARC **ajoute_arc**(T_ARC e,L_ARC lArc)
L_ARC **supprime_arc**(L_ARC lArc) ;
void **libererListeArc**(L_ARC lArc);

void **afficherGraphe**(GRAPH G, int nbsommets,int nbarcs);
GRAPH **lectureFichier**(const char* filename,int* nbrS,int *nbrA);
int **nbSommets**(const char *filename);

liste.h

/* ici on n'entre pas dans les détails, ce sont les même fonction qu'en TD ! avec typedef T_SOMMET ELEMENT
* /

Liste **supprimen**(int n, Liste l)

dijkstra.h

void **affichePcc**(Liste PCC, int nbrs);
Liste **dijkstra**(GRAPH G, int nbrs, T_SOMMET root, T_SOMMET target);
T_SOMMET **sommetDePccMin**(Liste *C,unsigned int* PCC)
void **AffichageDijkstragraphe2**(Liste l1,T_SOMMET a,T_SOMMET b);
void **equivalentnumero**(GRAPH G,T_SOMMET* a,T_SOMMET* b,int c,int d,int nbrs);

1.2.Tests effectués

Évidemment au cours du projet de très nombreux tests ont été réalisé, dans cette rubrique on a sélectionné l'essentiel, à savoir les tests planifiés lors de notre préparation.

1.2.1 Test lecturefichier/afficheGraphe

Le premier objectif stratégique fixé était celui de construire un graphe valide et de s'assurer qu'il soit complet. (que toute l'information utile des fichiers soit stockée en mémoire vive)
La fonction lectureFichier() réalise la construction du graphe la fonction afficheGraphe() quant à elle ne sert qu'à la vérification manuelle de la bonne construction du graphe, c'est pourquoi il est préférable d'exécuter ce test sur des petits fichiers de test

Ce test se compile à l'aide de la commande shell **"make partie1"**

Exemple d'exécution :

```
8 12
Sommets du graphe
0   0.5 0.95    M1  Aaa
1   0.1 0.7     M1  Baa
2   0.5 0.7     M1  Caa
3   0.9 0.7     M1  Daa
4   0.1 0.35    M1  Eaa
5   0.5 0.35    M1  Faa
6   0.9 0.35    M1  Gaa
7   0.1 0.05    M1  Haa
Arêtes du graphe : noeud1 noeud2 valeur
0   1   5
0   2   20
0   3   40
1   2   10
1   4   7
1   5   20
2   3   10
2   5   10
4   5   10
4   7   40
6   5   20
7   5   10
```

FIchier d'exécution

```c
#include"graphe.h"
#include<stdio.h>
#include<stdlib.h>

int main(){
        printf("debut de la fonction main\n");
        int nbrs;
        int nbarc;
        /*nbrs=nbSommets("monfichier.txt");*/
        GRAPH G;
        G=lectureFichier("monfichier.txt", &nbrs,&nbarc);
        printf("lectureFichier effectuée\n");
        printf("nbrs= %d\n", nbrs);
        printf("nbrarcs= %d\n", nbarc);
        if (G==NULL)
        {
                return 1;
        }
        else
        {
                printf("contenu du graphe :\n");
                afficherGraphe(G, nbrs, nbarc);
        }
}
```

Main pour le test

1.2.2. test du Dijkstra

Afin de suivre le bon déroulement du Dijkstra, un ensemble de marqueurs ont été implantés afin de suivre son déroulement et déceler les éventuelles erreurs.

Compilation : **“make testDijkstra”**

A) Résultat du dijkstra sur graphe1.txt

```
Sommets a visiter:
7, 6,

valeur de compteur:0
Noeud: 7  nom: Haa
PCC[7]: 52

min = 52 tampon =0
valeur de compteur:1
Noeud: 6  nom: Gaa
PCC[6]: 2147483647

tampon=0
numero du noeud retourné :7

Sommets visités:
7, 3, 5, 2, 4, 1, 0,

+++++++boucle++++++++
valeur de C:6,

valeurs de p6,

Sommets a visiter:
6,

tampon=0
numero du noeud retourné :6

Sommets visités:
6, 7, 3, 5, 2, 4, 1, 0,

target.numero=6

+++destination innaccessible+++

sommet a parcourir


kocevt@kocevt-XPS-15-9530:~/Documents/Tdinfo/projet$
```

On remarque que notre algorithme permet la détection et l'affichage d'une alerte en cas d'impasse sur le chemin choisi.

B) Mise en évidence d’un problème :
Lors de l’ouverture d’un fichier volumineux notre programme échoue.
L’erreur se trouve lors de l’allocation d’un tableau “pere” de T_SOMMET. Le problème c’est qu’il semble être trop volumineux en mémoire résultant une segmentation fault.

Sans doute qu'un adressage dynamique permettra de contourner le problème mais nous n'avons pas eu le temps de le tester.

```
264341 262982 3920.000000
263414 264342 276.000000
264342 263414 276.000000
245997 264343 969.000000
264343 245997 969.000000
244450 244447 3451.000000
244447 244450 3451.000000
244449 244450 1663.000000
244450 244449 1663.000000
260204 264344 1158.000000
264344 260204 1158.000000
236218 264345 323.000000
264345 236218 323.000000
264345 236008 368.000000
236008 264345 368.000000
237339 237340 1873.000000
237340 237339 1873.000000
261276 239953 156.000000
239953 261276 156.000000
260311 263806 730.000000
263806 260311 730.000000
239391 239389 1557.000000
239389 239391 1557.000000
263465 263812 106.000000
263812 263465 106.000000
261227 259706 389.000000
259706 261227 389.000000
fin MAJ
1. lecture fichier effectuée

entrez le numero du point de départ
5
entrez le numero du point d'arrivée
84
Segmentation fault (core dumped)
kocevt@kocevt-XPS-15-9530:~/Documents/Tdinfo/proj
```

segmentation fault dans des grands fichiers

1.2.3 Test de la demande à l'utilisateur et affichage plus court chemin à suivre (testé avec le fichier graphe2.txt)

Compilation: **make "reseauroutieramericain"**

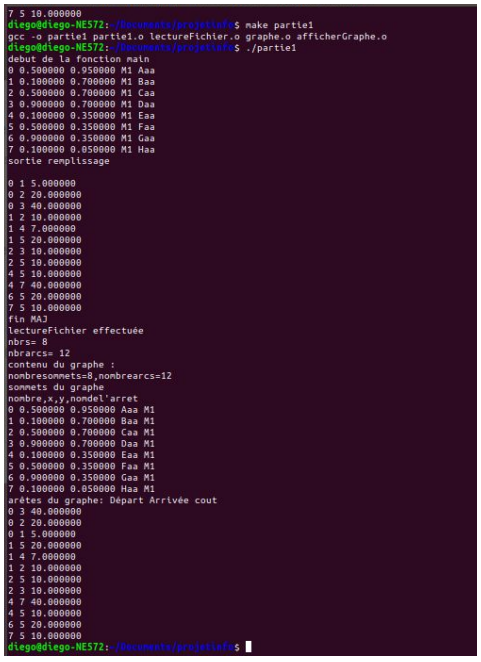
```c
#include"graphe.h"
#include<stdio.h>
#include<stdlib.h>
#include"liste.h"
#include"dijkstra.h"
int main()
{

GRAPH G;
T_SOMMET a;
T_SOMMET b;
Liste l1;
int depart,arrivee;
int nbrs,nbarc;
G=lectureFichier("graphe2.txt", &nbrs,&nbarc);
printf("fin de lecture\n");
printf("Rentrez le numéro de sommet de depart\n");
printf("depart:\n");
scanf("%d",&depart);
printf("arrivée:\n");
scanf("%d",&arrivee);
printf("++++debutequivalence\n++++");
equivalentnumero(G,&a,&b,depart,arrivee,nbrs); /*fonction qui renvoie les sommets qui correspondent aux indices depart et arrivee*/
printf("equivalenceréalisée");
printf("Debut du Dijkstra\n");
l1=dijkstra(G,nbrs,a,b); /*liste de T-SOMMETS*/
printf("fin du Dijkstra\n");
AffichageDijkstragraphe2(l1,b);
return 0;
}
```

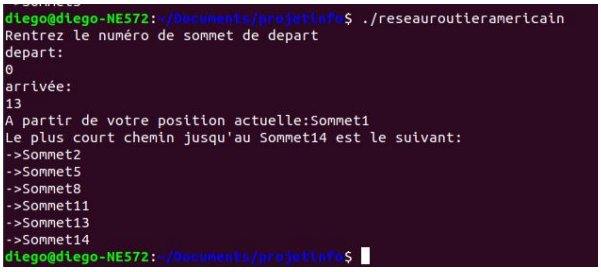
main pour le test de

1.3.Exemple d'exécution

1.3.1. Exécution lecturefichier()/afficheGraphe()

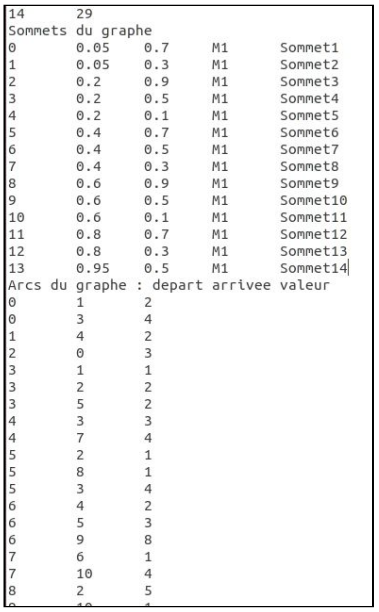


3.2. Exécution de la demande à l'utilisateur et de l'affichage plus court chemin à suivre avec le fichier graphe2.txt



Lors de l'exécution il y a un décalage des indices qui vient du fichier graphe2.txt. Dans ce fichier, le nom du fichier est décalé de +1 par rapport au numéro du sommet.

Fichier graphe2.txt



1.4.Les optimisations et les extensions réalisées

La complexité de la recherche du minimum avec l'algorithme de Dijkstra est de o(n).L'optimisation qu'il faut mettre en place pour diminuer cette complexité c'était d'implémenter un tas. L'élément en haut de ce tas sera le minimum. La complexité est de o(ln(n)) car le fait de garder le min en haut du tas oblige le rangement de tous les éléments en bas.
Pour la recherche des stations par leur nom, une table de hachage avec des listes chaînées semblait la meilleure option pour gérer les lignes de métro qui avaient des arrêts en commun avec le même nom.

2/Suivi

1.Problèmes rencontrés

Organisation du travail

Malgré nos efforts avant les premières séances lors d'une réunion préalable de 4h, durant laquelle nous avons tenté d'établir la répartition des tâches et l'établissement des prototypes de fonctions, on s'est retrouvé à modifier constamment nos paramètres dans la pratique.
Il a été très difficile de s'y retrouver surtout si le travail se fait à distance, on a utilisé dès le début un google drive mais cet outils est limité pour ce qui est du développement informatique. Il est facile d'effacer le travail de l'autre par mégarde, ou créer des quiproquos. Sans doute qu'il existe des plateforme plus adaptées (je pense à Github) ou il est possible de travailler sur plusieurs versions sans les perdre etc…
Manque de temps

2.Planning effectif

Nous avons répartis les tâches de la façon suivante:

Timothee:

1. Fonction lecture fichier
2. Dijkstra

Diego:

1. Fonctions des opérations dans les graphe (Voisins,ajout_arc, ajout_noeud, etc)
2. Fonction affichage du graphe construit
3. Fonction affichage du plus court chemin après réalisation du Dijkstra

Nous avons travaillé de manière régulière en revanche la révisions de partiel nous a imposé une grosse pause de deux semaines.

3.Qu'avons nous appris et que faudrait il de plus ?

Timothee

Dlego
Personnellement, ce projet informatique m'a permis de comprendre quels sont les principaux problèmes rencontrés lorsqu'on travaille en binôme sur une même fonction et quels sont les précautions à prendre lorsqu'on réalise la répartition du travail pour que le projet avance de façon efficace.Ce projet m'a permi aussi d'approfondir des notions de cours et de savoir les exploiter si besoin.

4.Suggestion d'améliorations du projet

3/Conclusion

Lors de ce projet nous avons pu appliquer et revoir un bon nombre de notions de ce semestre. Les listes, les tas et les arbres etc..
Nous avons pu également tester le travail collaboratif en informatique et les pépins logistiques qui en découle.