Rapport - Projet théorie des graphes

Younes Ben Yamna - Malek Zemni

20 avril 2016

1 Introduction

ici on parle de l'idée generale et comment on va faire pour le projet
Test avec accents : ça été à la télé, madère.

2 Structures

les structures utilsisee sont : la structure arc qui represente un arc (comme son nom l'indique)
la structure parcour qui contiend le poid de chaque sommet (par rapport au sommet de depart) son indice et une variable parcouru qui indique si il est parcouru ou non (-1 ou 0)
la structure antecedant qui contiend l'indice du sommet et son pere (le sommet par lequel on passe pour arrivera notre sommet le plus rapidement possible)

```
//definition du nombre de sommets et du nombre d'arcs du graphe
    orienté représentant la carte
#define V 45    //nombre de sommets
#define E 100   //nombre d'arcs

//Structure principale representant un arc du graphe
typedef struct
{
  char* nom;    //noms de l'arc de poids le plus faible entre le
    sommet de depart et le sommet d'arrivee
  char* depart; //nom du sommet de depart
  char* arrivee;  //nom du sommet d'arrivee
  int couleur;  //utilisee pour colorier l'arc lors de l'affichage
    graphique
  int poids;    //poids de l'arc de poids le plus faible entre le
    sommet de depart et le sommet d'arrivee
} Arc;

typedef struct
{
  int sommet;
  int poid;
  int parcouru;
} parcour;
```

```
typedef struct
{
  int sommet;
  int pere;
} antecedant;

Arc G[V][V];
```

src/definitions.h

3 Graphe

3.1 Sommets du graphe

Considération des sommets

On considère comme sommets les points où se croisent plusieurs pistes. La carte fournie a été simplifée, c'est à dire qu'on considère en fait comme sommets les zones (et non les points) où se croisent plusieurs pistes.
Les noms des sommets sont donc choisis par rapport au nom de la zone où se trouve le sommet. Si la zone ne porte pas de nom, le nom du sommet sera choisi par rapport au nom de la piste ou de la remontée la plus proche.

Indices des sommets

Chaque sommet est repéré par un indice allant de 0 à 44 : on a donc 45 sommets au total.
Voici la liste complète des sommets, ainsi que leurs indices respectifs :

0 PIC BLANC
1 GROTTE DE GLACE
2 SOMMET 3060
3 SARENNE BASSE
4 CLOCHER DE MACLE
5 LAC BLANC
6 LIEVRE BLANC
7 PLAT DES MARMOTTES
8 MINE DE L'HERPIE
9 SOMMET 2100
10 SOMMET DES VACHETTES
11 SIGNAL DE L'HOMME
12 L'ALPETTE
13 COL DU COUARD
14 CASCADE
15 CLOS GIRAUD
16 MONFRAIS
17 SIGNAL
18 RIFNEL EXPRESS
19 CHALVET
20 AURIS EXPRESS
21 FONTFROIDE
22 LOUVETS
23 POUTRAN
24 CHAMP CLOTURE

```
25 STADE
26 SCHUSS
27 ALPE D'HUEZ
28 GRANDE SURE
29 ECLOSE
30 SURES
31 COL
32 AURIS EN OISANS
33 LA VILLETTE
34 VAUJANY
35 L'EVERSIN D'OZ
36 OZ EN OISANS
37 PETIT PRINCE
38 VILLAGE
39 MARONNE
40 VILLARD RECULAS
41 HUEZ
42 DOME DES PETITES ROUSSES
43 ALPAURIS
44 L'ALPETTE BASSE
```

data/sommets.txt

3.2 Arcs du graphe

Considération des arcs

voici tout les arcs qu'on a consideré dans notre travail

NB : si deux arcs relient deux sommet dans le meme sens on garde celui avec le poid le plus petit

```
0 descente SARENNE HAUTE / CHATEAU NOIR
1 descente GLACIER
2 descente BRECHE
3 descente TUNNEL
4 descente CRISTAILLERE
5 descente SARENNE BASSE
6 descente DOME
7 descente LAC BLANC
8 descente PROMONTOIRE
9 descente COMBE CHARBONNIERE
10 descente ROUSSES
11 descente CHAMOIS
12 descente ANCOLIES
13 descente CANYON
14 descente CAMPANULES
15 descente COL DE CLUY
16 descente VERNETTES
17 descente CHALVET-ALPAURIS
18 descente ETERLOUS
19 descente FONTFROIDE
20 descente PRE-ROND
21 descente COL 1
22 descente LES FARCIS
23 descente GENTIANES
24 descente COL 2
25 descente CORNICHE
```

26 descente FONTFROIDE–LOUVETS
27 descente BARTAVELLES
28 descente POUTRAN
29 descente JEUX
30 descente AGNEAUX
31 descente LES BERGES
32 descente LOUP BLANC
33 descente POUSSINS
34 descente ANEMONES
35 descente SIGNAL–STADE
36 descente SIGNAL
37 descente SCHUSS–ECLOSE
38 descente SCHUSS–GRANDE SURE
39 descente ECLOSE–GRANDE SURE
40 descente VILLAGE 1
41 descente HUEZ
42 descente VILLAGE 2
43 descente PETIT PRINCE
44 descente LA FORET
45 descente L'OLMET
46 descente CHAMPCLOTURY
47 descente ALPETTE
48 descente LUTINS
49 descente CARRELET
50 descente CHALETS
51 descente LA FARE
52 descente CASCADE
53 descente ETOURNEAUX
54 descente EDELWEISS
55 descente VAUJANIATE
56 descente VILLARD

Liste des remontées :

100 TELEPHERIQUE PIC BLANC
101 TELESIEGE GLACIER
102 TELESIEGE HERPIE
103 FUNITEL MARMOTTES III
104 TELESIEGE LAC BLANC
105 TELEPHERIQUE ALPETTE–ROUSSES
106 DMC 2EME TRONCON
107 TELESIEGE LIEVRE BLANC
108 TELECABINE MARMOTTES II
109 TELECABINE POUTRAN II
110 DMC 1ER TRONCON
111 TELESIEGE ROMAINS
112 TELESIEGEBULLE MARMOTTES I
113 TELEMIXTE RIFNEL EXPRESS
114 TELESIEGE SIGNAL
115 TELESKI STADE
116 TELESKI ECLOSE–SCHUSS
117 TELESIEGE GRANDE SURE
118 TELECABINE TELEVILLAGE
119 TELESIEGE BERGERS
120 TELESKI PETIT PRINCE
121 TELESIEGE TSD LE VILLARAIS
122 TELESIEGE ALPAURIS–ALPE D'HUEZ
123 TELESIEGE CHALVET
124 TELESIEGE FONTFROIDE

```
125 TELESIEGE LOUVETS
126 TELESIEGE ALPAURIS
127 TELESIEGE AURIS EXPRESS
128 TELESKI COL
129 TELESIEGE SURES
130 TELESIEGE MARONNE
131 TELECABINE L'ALPETTE
132 TELESKI L'ALPETTE
133 TELECABINE POUTRAN I
134 TELESKI HAMP CLOTURE
135 TELEPHERIQUE VAUJANY–ALPETTE
136 TELESIEGE CLOS GIRAUD
137 TELESKI MONTFRAIS
138 TELESIEGE MONTFRAIS
139 TELESIEGE VALLONNET
140 TELECABINE VILLETTE–MONTFRAIS
141 TELECABINE VAUJANY–VILLETTE
142 TELECABINE VAUJANY–ENVERSIN
```

data/arcs.txt

3.3 Fichier du graphe

le premier entier est le nom du sommet de depart (enfin l'indice qui represente ce nom)
le deuxieme entier est le nom du sommet d'arrivee(enfin l'indice qui represente ce nom)
le troisieme entier est le nom de l'arc (enfin l'indice qui represente ce nom)
le quatrieme entier est le poid de cet arc (sans considerer la couleur)
le cinqieme entier est la couleur de cet arc

```
42 1 6 2 3
1 5 7 3 3
4 6 8 2 3
4 8 9 3 10
1 12 10 2 10
1 9 11 2 6
7 9 12 1 6
7 27 13 2 8
8 27 14 2 6
```

data/graphe.txt

3.4 lecture du graphe

la fonction utilisee est elle fait ...

```
void lectureGraphe(char* nomFichier, Arc G[V][V])
{ //Lit le graphe à partir du fichier fourni
  FILE* F = fopen(nomFichier,"r");  //doit etre deja present, sinon
     NULL
```

```
  if (F == NULL){
    printf("fichier du graphe introuvable\n");
    return;
  }

  int k, temps, experience = getExperience();

  initialise(G);

  for (k = 0; k < E; k++)   //boucle qui parcourt les lignes du
    fichier : E lignes <=> E arcs
  {
    int i, j, indiceArc, couleur;
    //i : indiceSommetDepart, j : indiceSommetArrivee

    fscanf(F,"%d %d %d %d %d",&i, &j, &indiceArc, &couleur, &temps)
    ;

    G[i][j].nom = nomArc(indiceArc);
    G[i][j].depart = nomSommet(i);
    G[i][j].arrivee = nomSommet(j);
    G[i][j].couleur = couleur;
    G[i][j].poids = calculPoids(G[i][j].nom, couleur,temps,
    experience);
  }
```

src/graphe.c

4 les fonctions

4.1 get experience

c'est la fontion qui

```
  char c;
  while(1){
    printf("Etes vous debutant o/n ?\n");
    scanf("%c",&c);
    if (c == 'o')
      return 1;
    if (c == 'n')
      return 0;
    scanf("%c",&c); //libere le buffer
  }
}

int calculPoids(char* nomArc, int couleur, int temps, int
    experience)
```

src/fonctions.c

4.2 calcul poid

c'est la fontion qui

```
  //0Vert 1Bleu 2Rouge 3Noir
  double typeRemontee = 1;
  //typeRemontee est le nombre par lequel on multiplie (ou plutot
    divise) le poids de la remontee en fonction de son type

  if (couleur==0)
  { //Les remontees sont des arcs de couleur 0
    //Plus ce type de remontée est rapide plus son poids va
    diminuer
    if (strstr(nomArc, "TELEPHERIQUE") != NULL)
      typeRemontee = 0.3;
    if (strstr(nomArc, "FUNITEL") != NULL)
      typeRemontee = 0.5;
    if (strstr(nomArc, "DMC") != NULL)
      typeRemontee = 0.6;
    if (strstr(nomArc, "TELECABINE") != NULL)
      typeRemontee = 0.7;
    if (strstr(nomArc, "TELEMIXSTE") != NULL)
      typeRemontee = 0.8;
    if (strstr(nomArc, "TELESIEGEBULLE") != NULL)
      typeRemontee = 0.85;
    if (strstr(nomArc, "TELESIEGE") != NULL)
      typeRemontee = 0.9;
    return (int)(temps*typeRemontee);
  }
  if (couleur==1)
    return (experience*temps + temps);
  if (couleur==2)
    return (experience*2*temps + temps);
  if (couleur==3)
    return (experience*3*temps + temps);
  return 1000;
}
```

src/fonctions.c

5 Algorithme de Dijkstra

5.1 intro

la on parle du principe comment et pourquoi

5.2 initialisation des tableaux

```
  int i;
  for (i = 0; i < V; i++)
  {
    a[i].sommet=i;
    a[i].pere=-1;
  }
}

void initialise_tableau_parcour(parcour p[],int depart)
{ //initialise le tableau parcour
```

```
  int i;
  for (i = 0; i < V; i++)
  {
    p[i].sommet=i;
    p[i].parcouru= 0;
    p[i].poid=G[depart][i].poids;
  }
}

int recherche_pere(parcour p[])
```

src/dijkstra.c

5.3 recherche des peres et des fils

```
  int indice_pere=0;
  while((p[indice_pere].poid<1)&&(indice_pere<(V-1))) indice_pere
    ++;
  int i;
  for (i = (indice_pere+1); i < V; i++)
  {
    if( ( p[i].poid < p[indice_pere].poid ) && ( p[i].poid > 0 ) &&
     (p[i].parcouru!=1))
      indice_pere = i;
  }
  if(( p[indice_pere].poid < 1 ) || (p[indice_pere].parcouru == 1 )
    )
    return -1;
  return indice_pere;
}

int recherche_fils(int pere,int f[])
{//recherche tout les fils de pere
  int nombre_fils=0;
  int i;
  for (i = 0; i < V; i++)
  {
    if ( G[pere][i].poids < 1000 ){
      f[nombre_fils] = i;
      nombre_fils++;
    }
  }
  return nombre_fils;
}

void dijkstra(int depart,int arrivee,antecedant a[],parcour p[])
```

src/dijkstra.c

5.4 l'algo de dijkstra

```
  int pere=depart;
  int i,j,fils;
  int f[10];
  if (depart==arrivee){
     printf("Vous etes deja a destination '%s'\n\n\n",nomSommet(
    depart));
```

```
    return;}
  while(pere!=-1)
  {
      printf("le pere est %d\n",pere);
    int nbf=recherche_fils(pere,f);
    for (i = 0; i < nbf; i++)
    {
      fils=f[i];
      printf("le fils est %d\n",fils);
      if((p[fils].parcouru==0)&&((p[fils].poid==-1)||(p[pere].poid+
    G[pere][fils].poids<p[fils].poid)))
      {
        p[fils].poid=p[pere].poid+G[pere][fils].poids;
        a[fils].pere=pere;
      }
    }
    p[pere].parcouru=1;
    pere=recherche_pere(p);
  }
  //##############affichage####################
  int chemin[50];
  i=0;
  while(-1!=arrivee)
  {
    chemin[i]=arrivee;
    i++;
    arrivee=a[arrivee].pere;
  }
  chemin[i]=depart;
  i++;
  for (j = 0; j < i-1 ; j++)
  {
    printf("%d ) vous devez partir du sommet '%s' et prendre '%s'
    jusqu'au sommet '%s'\n\n",j,nomSommet(chemin[i-j-1]),G[chemin[i
    -j-1]][chemin[i-j-2]].nom,nomSommet(chemin[i-j-2]));
  }
  printf("%d ) vous etes arrivé a votre destination '%s' en %d
    secondes\n\n\n",j,nomSommet(chemin[0]),p[chemin[0]].poid);
}
```

src/dijkstra.c

6 Conclusion et difficultées