CONCOURS d'accès à l'ESI

Corrigé de l'épreuve d'algorithmique et programmation
Code : ALPRO

4 Juillet 2011

Question 1
Cette solution est une solution type. Toute autre solution proposée par le candidat et qui est cohérente sera retenue.

Nous pouvons représenter le graphe dans un tableau à deux dimensions. Bien entendu les cases vides contiennent en fait des zéros. Si nous observons notre matrice nous nous apercevons que la ligne indique le sommet d'un arc et la colonne l'extrémité de cet arc. On voit ainsi que le sommet A à 3 extrémités, donc on 3 arcs : AB, AC et AD.

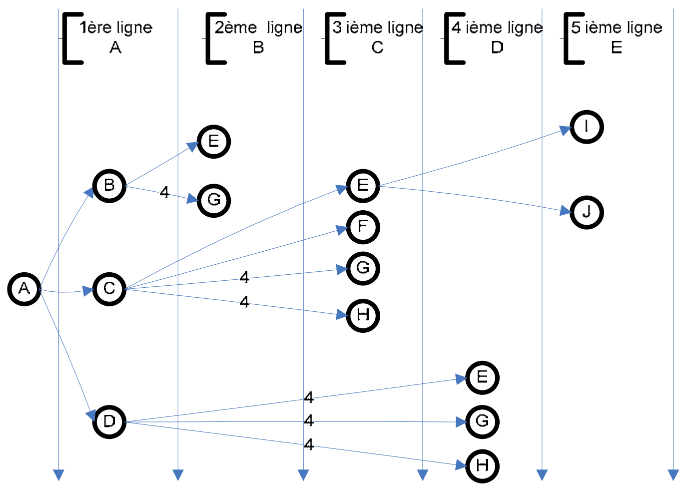
T

	A	B	C	D	E	F	G	H	I	J	K
A		8	5	7							
B					3		4				
C					5	4	6	4			
D					3		2	3			
E									4	5	
F									4	4	
G									3	2	
H										4	
I											5
J											7
K											

Type Indice = 'A'..'Z'
Tab = tableau [Indice, Indice] d'entier

Question 2

Avant de procéder au découpage, nous présentons l'idée générale de la solution.
Elle consiste à balayer tout le tableau (ligne par ligne) et construire progressivement tous les chemins possibles et en même temps calculer leurs coûts. Sur le schéma suivant nous observons d'abord que toutes les chaines commencent par A, puis en parcourant toutes les colonnes de la ligne A nous pouvons construire les segments : AB, AC et AD, puis en parcourant la ligne B nous pouvons construire les segments : ABE et ABG, puis en parcourant la ligne C nous pouvons construire les segments : ACE,ACF,ACG et ACH,…



A la fin du balayage de tout le tableau nous aurons tous les chemins possibles pour aller de A à K.

Nous allons donc conserver ces chemins et évaluer leurs coûts, progressivement, dans un tableau (CHEMINS) qui pourrait avoir la forme suivante :

chemin	coût
ABEIK	20
ABGIK	23
ACFIK	18
…..	….

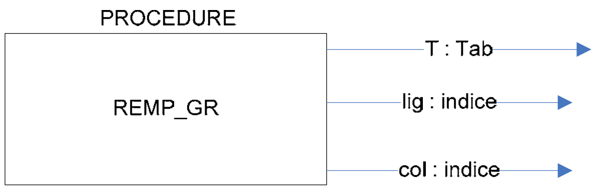
Ce tableau pourrait avoir la structure suivante :

Type C = enregistrement
Chemin : chaîne
Cout : entier
Fin

CHEMINS = tableau [1..1000] de C

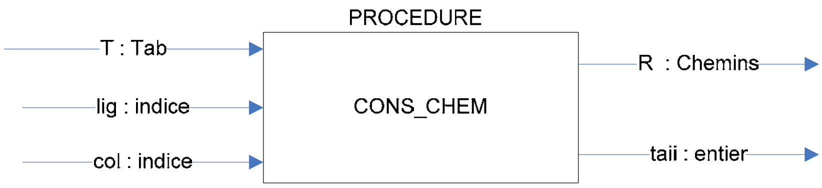
Il suffira ensuite d'écrire le contenu du tableau CHEMINS pour avoir tous les chemins, et la ligne qui contient le plus petit cout pour avoir le chemin ayant le meilleur cout de même que son coût.
Le découpage pourrait être le suivant :

1. <u>il faut remplir la matrice, qui est carrée, en principe</u>



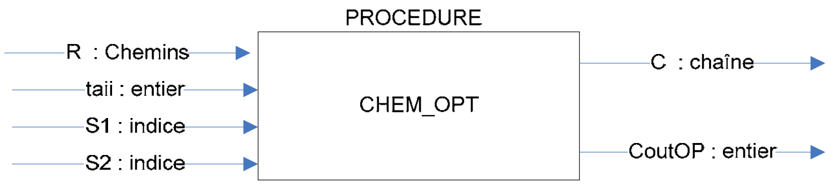
<u>Rôle</u> : Lit un tableau qui représente un graphe (Cf RO) dont les éléments sont des entiers. (Les indices du tableau étant des caractères compris entre A et Z)

2. <u>construire le tableau CHEMINS, qui contient tous les chemins et leurs coûts</u>



<u>Rôle</u> : Fourni un tableau de type CHEMINS , Qui contient tous les chemins possibles, et leurs coûts, contenus dans le tableau T qui contient un graphe (dont les sommets ,donc les indices, sont des lettres comprises entre A er Z (Cf RO))

3. <u>trouver le chemin optimum (ayant le plus bas coût) et le coût lui-même</u>



<u>Rôle</u> : Fourni la chaine C, commençant par le sommet S1 et se terminant par le sommet S2, se trouvant dans le tableau R (de taille tai), ayant le plus petit coût de même que son coût

<u>Question 3</u>
Construction du module LECT2D
Le module LECT2D étant très facile nous ne donnons pas son corrigé.

Construction du module CONS_CHEM

Analyse :

Nous allons détailler un peu plus l'idée vue dans la question 2.
Nous n'utilisons pas la récursivité dans notre solution.
Donc :

- on fait varier i = 'A ' , 'B',….,'K' *(On parcourt toutes les lignes)*
    - on fait varier j = 'A ' , 'B',….,'K' et à chaque fois *(On parcourt toutes les colonnes)*
        - si T[i, j] <> 0
            - on parcourt le tableau CHEMIN
                - si le dernier élément de Chemin = i
                    - *et on crée un nouvel élément dans CHEMINS et on y met*
                        - la concaténation de Chemin avec j
                        - et le cumul du cout à Cout

on aura ainsi :

Au départ	chemin :	A	Cout =0
Ensuite		AB	8
		AC	5
		AD	7
		ABE	11
		ABG	12
		ACE	10
		ACF	9
		ACG	11
		ACH	9
		………	

Ainsi on aura tous les chemins possibles, et il suffira de donner le sommet de départ et celui de fin d'un chemin pour avoir son coût.

Algorithme

```
Procédure CONS_CHEMIN (T : tab ; lig, col :entier ; R : Chemins ;tai : entier)
Variables i, j, k , ct: entier
          Ch : chaîne
DEBUT
R[1].chemin ← 'A'
R[1].cout ← 0
Tai ← 1
Pour  i allant de  'A '  à 'K'  faire
      Pour  j allant de  'A ' à 'K' faire
            si T[i, j ] <> 0 alors
                  Pour k allant de 1 à tai faire
                        Dpour
                        | Ch ← R1[k].chemin
                        | Ct ←  R1[k].cout
                        | Si ch(length (ch)) = i alors
                        |          | Dsi
                        |          | Tai ← tai +1
                        |          | R[tai].chemin ← ch + j
                        |          | R[tai].cout ← ct + t[i, j]
                        |          | FSI
                        | Fpour
FIN
```

Construction du module CHEM_OPT

Analyse

- on parcourt le tableau jusqu'à trouver un chemin qui commence par le sommet S1 et se termine par le sommet S2, on considère que son coût est le plut petit cout (PETIT)
- on parcourt le reste du tableau
    - si le chemin commence par le sommet S1 et se termine par le sommet S2 alors
        - si son coût est plus petit que PETIT (on vient de trouver un chemin moins coûteux alors
            - on garde ce chemin et son cout

Nota : dans le cas ou ce chemin n'existe pas , en sortie on aura une chaîne vide et un CoutOp égal à 0.

Algorithme

```
Procédure CHEM_OPT ( T :tab ; tai : entier ; S1,S2 : Indice; c :chaine, coutOp : entier )
Variables i : entier
          Ch : chaine

DEBUT
I ← 0
Répéter
 | I ← I + 1
Jusqu'à (R[i].chemin[1] = S1) et (R[i].chemin[length(R[i].chemin)] = S2)
Petit ← R[i].cout
Ch ← R[i].chemin
Répéter
 | I ← I + 1
 | Si (R[i].chemin[1] = S1) et (R[i].chemin[length(R[i].chemin)] = S2) alors
 |         Si R[i].cout < Petit alors
 |                                  | Dsi
 |                                  | Petit ← R[i].cout
 |                                  | Ch ← R[i].chemin
 |                                  | Fsi
Jusqu'à i = tai
CoutOp ← R[i].cout
C ← R[i].chemin
FIN
```

Construction de l'algorithme principal

Il faut :

- remplir le tableau historique
- construire tous les chemins possibles
- trouver le plus court chemin et son cout
- et bien sur les imprimer

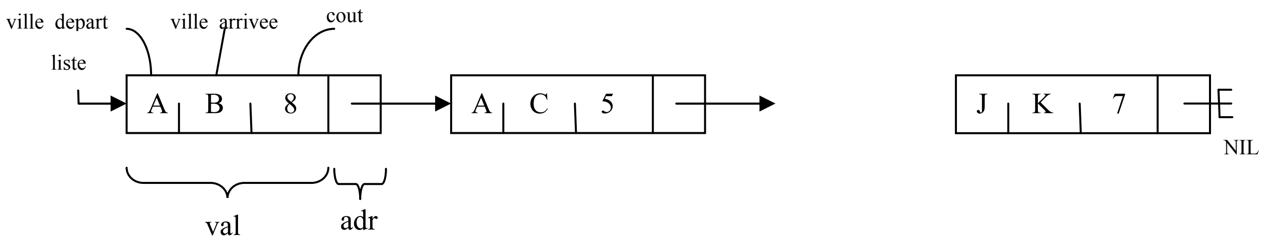
```
Algorithme CONC_CPI
    Type    Indice = ’A’..’Z’
            Tab = tableau [Indice, Indice] d’entier
            C =      enregistrement
                   | Chemin : chaîne
                   | Cout : entier
                   | Fin
            CHEMINS = tableau [1..1000] de C
    Variables   X : tab
                L, c, Sommet1, sommet2  : indice
                Chem :Chemins
                T , Cout_Opt :entier
                Ch_Opt : chaine

    DEBUT
    REMP_GR (X, l, c)
    CONS_CHEM ( X, l, c, Chem, t)
    CHEM_OPT ( Chem, t, Sommet1,Sommet2, Ch_Opt, cout_Opt)
    Ecrire ( ‘Le meilleur chemin est : ‘ , Chem_opt, ‘ et son cout est : ‘, cout_opt)
    FIN
```

Question 5

Cette solution est une solution type. Toute autre solution proposée par le candidat et qui est cohérente sera retenue.

5.1) Supposons que le nombre des points intermédiaires reliant deux villes n'est pas défini préalablement, la structure appropriée est une liste linéaire chainée.

5.2) Représentation graphique



Déclarations

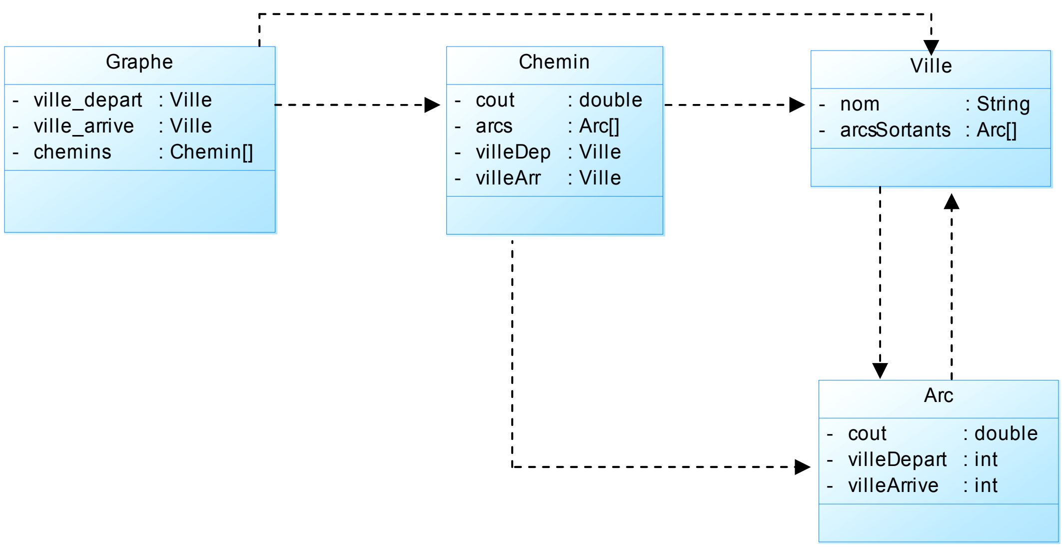
```
type element = structure
    ville_depart: char
    ville_arrivee: char
    cout: reel
fin
type maillon = structure
    val: element
    adr: pointeur(maillon)
fin
var liste: pointeur(maillon)
```

5.3) On suppose que notre structure est construite, nous voulons la triée d'après le coût croissant de réalisation. L'algorithme de tri proposé est le tri par Bulles. L'étudiant a libre choix de choisir de développer le tri de son choix.

```
Procedure Tri_Liste (var liste: pointeur(maillon))
    // la liste est en entrée et en sortie de la procédure

    // Déclaration des variables
    var     P, Q : pointeur(maillon)
            permut : booleen
            temp : element

    Debut
    |       permut ← vrai
    |
    |       TANTQUE permut FAIRE
    |           Debut
    |           |   P ← liste
    |           |   permut ← faux
    |           |   Q ← suivant(p)
    |           |
    |           |   TANTQUE Q <> nil FAIRE
    |           |       Debut
    |           |       |
    |           |       |   SI P.val.cout >= Q.val.cout ALORS
    |           |       |       Debut
    |           |       |       |   temp ← valeur(P)
    |           |       |       |   valeur(P) ← valeur (Q)
    |           |       |       |   valeur(Q) ← temp
    |           |       |       |   permut ← vrai
    |           |       |       Fin
    |           |       |
    |           |       |   P ← suivant(P)
    |           |       |   Q ← suivant(Q)
    |           |       Fin
    |           Fin
    Fin
```

Question 6

6.1) Modélisation Orienté Objet :



6.2) Implémentation en java de la méthode cheminsAutoroute de la classe Graphe

```
public void cheminAutoroute(){

      cheminsValides=new Chemin[cheminsPossibles.length];
      int j=0;
      for (int i=0; i<cheminsPossibles.length; i++){
         if
((cheminsPossibles[i].getVilleDep().getNom().compareTo(this.ville_depart.getNom())==0)
&&(cheminsPossibles[i].getVilleArr().getNom().compareTo(this.ville_arrive.getNom())==0))
              {
                    cheminsValides[j]=cheminsPossibles[i];
                    j++;
              }
      }


   }
```

6.3) Implémentation en java de la méthode cheminOptimal de la classe Graphe

Solution 1 :

```
public Chemin cheminOptimal(){

      cheminAutoroute();
      Arrays.sort(cheminsValides);
      return cheminsValides[0];

   }
```

Avec :

```
public class Chemin implements Comparable<Chemin> {
   …
public int compareTo(Chemin arg0) {
      if (this.cout==arg0.cout) return 0;
      else if (this.cout>arg0.cout) return 1;
      else return -1;
   }
…
}
```

Solution 2 : implémenter une méthode de tri

```
public Chemin cheminOptimale(){
   cheminAutoroute();
   trieTableau(cheminsValides); //méthode à implémenter avec un algorithme de tri
   return cheminsValides[0];
}
```

6.4) Modification de la modélisation pour prendre que l'autoroute passent que par des arcs carrossables à deux chaussées (deux directions) :