

**- Examen du 7/3/2007. -**

**- Durée 2h. Aucun document autorisé. -**

*- Une large part de la notation prendra en compte la clarté de la rédaction et la rigueur des justifications. -*

**- Exercice 1 -** On se donne un graphe orienté sur l'ensemble de sommets  $\{1, \dots, n\}$  dont les arcs ont une longueur donnée. Ceci est codé par une matrice  $L$  de taille  $n \times n$  dont les entrées  $l_{ij}$  vérifient :

- $l_{ii} = 0$  pour tout  $i = 1, \dots, n$ .
- s'il n'y a pas d'arc de  $i$  vers  $j$ , on pose  $l_{ij} = \infty$ .
- s'il y a un arc de  $i$  vers  $j$ ,  $l_{ij}$  est sa longueur.

La *distance* de  $i$  à  $j$ , notée  $d_{ij}$ , est la plus petite somme des longueurs des arcs d'un chemin orienté de  $i$  à  $j$ . Par convention,  $d_{ij} = \infty$  lorsqu'un tel chemin n'existe pas.

- a. Ecrire un algorithme `distance(L)` prenant en entrée la matrice  $L$  et retournant la matrice  $D$  de taille  $n \times n$  dont les entrées sont les  $d_{ij}$ .
- b. Expliquer son fonctionnement, et calculer sa complexité en fonction de  $n$ .
- c. Ecrire un algorithme `chemins(L)` afin qu'il retourne une matrice  $P$  de taille  $n \times n$  dont les entrées  $p_{ij}$  vérifient :
  - $p_{ii} = NULL$  pour tout  $i$ .
  - $p_{ij} = k$  s'il existe un chemin orienté de longueur minimum de  $i$  à  $j$  commençant par l'arc  $ik$ .
  - $p_{ij} = \infty$  s'il n'existe pas de chemin de  $i$  à  $j$ .
- d. Ecrire un algorithme `itineraire(P, i, j)` prenant en entrée la matrice  $P$  et deux sommets  $i, j$  et qui retourne un chemin orienté de longueur minimale de  $i$  à  $j$  s'il en existe un, et FAUX sinon.

**- Exercice 2 -** On garde les mêmes hypothèses que l'exercice précédent, la matrice  $L$  codant les longueurs des arcs. On suppose de plus que le graphe considéré est sans circuit orienté et que  $1, 2, \dots, n$  est un tri-topologique des sommets du graphe.

- a. Rappeler la définition de tri-topologique.
- b. Montrer que tout graphe sans circuit admet un tri-topologique.
- c. Ecrire un algorithme `distancesanscircuits(L)` prenant en entrée la matrice  $L$  et retournant la matrice  $D$  de taille  $n \times n$  dont les entrées sont les  $d_{ij}$ . La complexité de votre algorithme, que vous calculerez, devra être meilleure que celle de l'algorithme `distance(L)`.

**- Exercice 3 -**

On se donne un graphe non orienté sur l'ensemble de sommets  $\{1, \dots, n\}$  dont les arêtes ont une longueur donnée. Ceci est codé par une matrice  $L$  de taille  $n \times n$  dont les entrées  $l_{ij}$  vérifient :

- $l_{ii} = 0$  pour tout  $i = 1, \dots, n$ .
- s'il n'y a pas d'arête entre  $i$  et  $j$ , on pose  $l_{ij} = \infty$ .
- s'il y a une arête entre  $i$  et  $j$ ,  $l_{ij}$  est sa longueur.
- $l_{ij} = l_{ji}$ .

- a. Définir précisément la notion d'arbre, puis celle d'arbre couvrant d'un graphe.
- b. Ecrire, ou au pire décrire, un algorithme **arbrecouvrantminimum**( $L$ ) prenant en entrée la matrice  $L$  et retournant la somme minimale des longueurs des arêtes d'un arbre couvrant du graphe et FAUX si le graphe n'admet pas d'arbre couvrant.
- c. Dérouler votre algorithme sur l'exemple suivant.

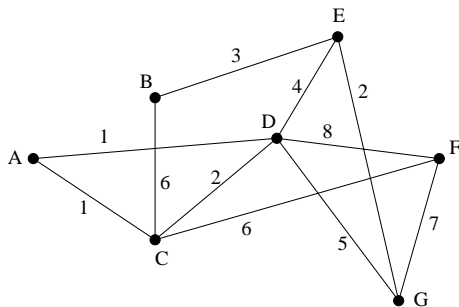


FIG. 1 –

- d. Calculer la complexité de votre algorithme en fonction de  $n$ .
- e. (Hors barème) Montrer que si toutes les longueurs des arêtes sont distinctes et que le graphe admet un arbre couvrant, alors il existe un unique arbre couvrant de longueur minimale.

Barème approximatif :

Exercice 1 : 2+2+2+1

Exercice 2 : 1+2+3

Exercice 3 : 2+3+1+1 (+3)