TD Graphes : Parcours, distance et plus court chemin

I Graphes orientés

Q.1) - On considère un graphe orienté G ayant 6 sommets $(1, 2, 3, 4, 5, 6)$ et pour matrice d'incidence

$$M = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Représenter graphiquement ce graphe.

Q.2) - Représenter sous forme d'un dictionnaire ce graphe.

Q.3) - On considère l'algorithme suivant :

```
Donnée : deb : le sommet de départ
début
    initMarques()                    // initialise le tableau marque de booléens à FAUX,
                                     // le tableau marque étant un attribut du graphe.
    file ← initFile()                // initialise une file à vide
    file.ajouteALaFin(deb)
    marque.marquer(deb)
    tant que file.estVide()=FAUX faire
        s = file.premier()
        file.enlevePremier()
        pour chaque t successeur de s faire
            si marque.estMarque(t) =FAUX alors
                file.ajouteALaFin(t)
                marque.marquer(t)
fin
```

Algorithme 1 : Parcours(Sommet deb)

Dérouler cet algorithme sur le graphe précédent en partant du sommet 1 (on grisera au fur à mesure du parcours le graphe représenté graphiquement).

Q.4) - Dérouler ensuite cet algorithme en partant du sommet 5.

Q.5) - À quel type de parcours correspond cet algorithme ?

Q.6) - On dit qu'un sommet t est *accessible* à partir du sommet s s'il existe un chemin partant de s et arrivant à t. Dans ce cas, la *distance* de s à t, notée $d_s(t)$, est la longueur minimale d'un chemin partant de s et arrivant à t. (La longueur d'un chemin correspond ici au nombre d'arcs du chemin, c'est-à-dire au nombre de sommets moins un.)

En utilisant un tableau stockant les distances, compléter l'algorithme 1 afin qu'après le parcours on ait les distances du sommet de départ deb à ses sommets accessibles. (Pour un sommet non accessible, on dira qu'il est à distance infinie.)

Q.7) - Compléter votre algorithme par un tableau permettant de retrouver un chemin de longueur minimal du sommet de départ deb à ses sommets accessibles. (On stockera dans ce tableau les *pères* des sommets.)

II Graphes orientés valués

Dans la partie précédente, on a considéré que les longueurs des arcs étaient uniformément égales à 1. Il peut s'avérer utile de considérer des valeurs différentes suivant chaque arc. Un *graphe orienté valué* est un graphe orienté $G = [X, U]$ muni d'une fonction supplémentaire $w : U \to \mathbb{R}$ qui associe une valeur (coût, poids ou longueur) à chaque arc. On ne considérera ici que des graphes à valuation positive.

On peut représenter un graphe orienté valué par trois tableaux : un pour les origines, le second pour les extrémités et le dernier pour les longueurs des arcs.

Q.1) - Représenter graphiquement le graphe valué $G = [X, U, w]$ donné par les trois tableaux suivants :

$orig$	1	2	1	4	2	5	4
$extr$	2	3	4	2	5	3	5
w	4	1	1	2	1	3	2

La longueur d'un chemin dans un graphe valué est la somme des longueurs de ses arcs. Étant donné un sommet s source, le principe de l'algorithme de Ford est de calculer progressivement la longueur des plus courts chemins de s vers ses sommets accessibles. Pour cela,

- on initialise le tableau représentant la distance de s vers ses sommets accessibles par $d_s(s) = 0$ et $d_s(t) = \infty$ pour tout sommet $t \neq s$;
- on parcourt l'ensemble des arcs et pour chaque arc $(t, v) \in U$,
  - si $d_s(t) + w(t, v) < d_s(v)$, le chemin de s à v passant par t est plus court que celui trouvé précédemment et on pose alors $d_s(v) = d_s(t) + w(t, v)$,
  - sinon le chemin est plus long ;
- On réitère le parcours de l'ensemble des arcs jusqu'à ce qu'il n'y ait plus de chemins plus courts.

Q.2) - Dérouler cet algorithme sur le graphe G en partant du sommet source 1 et en parcourant l'ensemble des arcs dans l'ordre donné par les tableaux.

Q.3) - Qu'obtient-on après un, deux, trois parcours de l'ensemble des arcs ?

Q.4) - En général, combien faut-il au plus de parcours de l'ensemble des arcs ?

Q.5) - Quelle est la complexité de cet algorithme ?

Q.6) - Comment compléter cet algorithme pour mémoriser les plus courts chemins ?