Complexité & graphes : TD3

Université de Tours

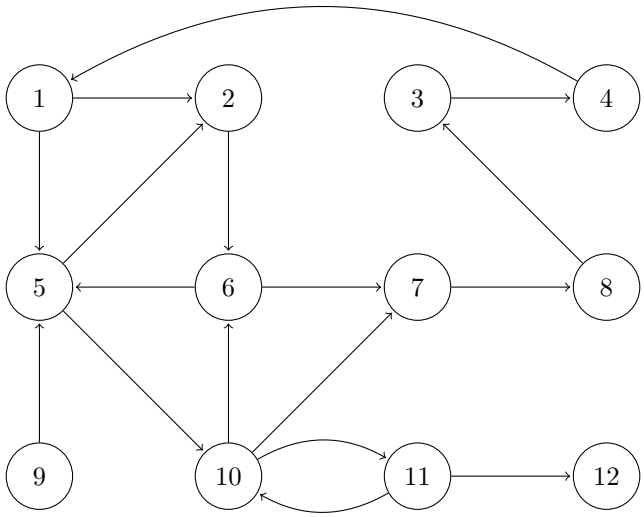
`Département informatique de Blois`

Algorithmes sur les graphes

*
* *

Problème 1

Soit le graphe G suivant :



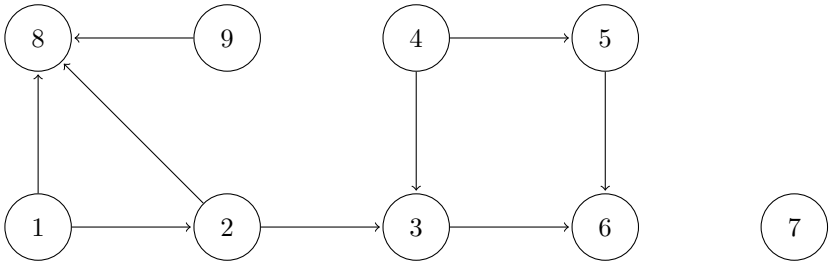
1. Appliquer un parcours en largeur de G en partant du sommet 1. En cas de choix multiples possible, on explorera en priorité le sommet avec l'identifiant le plus petit.
2. Dans un parcours en profondeur, il est possible de numéroter les sommets (de 1 en 1 et en commençant par 1).
   - Soit en ordre préfixe : dans ce cas, un sommet est numéroté dès qu'il est exploré,
   - Soit en ordre suffixe : dans ce cas, un sommet est numéroté seulement quand tous ses successeurs ont été explorés.

   Déterminer ce que seront les deux numérotations des sommets du graphe G (soit en ordre préfixe, soit en ordre suffixe) dans les conditions suivantes :
   - Commencer l'exploration par le sommet 1.
   - En cas de choix multiples possibles (un sommet a encore plusieurs successeurs non explorés), commencer par explorer le somment avec l'identifiant le plus petit.
3. Calculs de composantes fortement connexes
   (a) Calculer les composantes fortement connexes de G.
   (b) Quel(s) arc(s) peut-on ajouter pour que G soit un graphe fortement connexe ?

Problème 2

Un *tri topologique* d'un graphe orienté acyclique $G = (S, A)$ est un ordre linéaire des sommets de G tel que si G contient l'arc (u, v), u apparaît avant v. Le tri topologique d'un graphe peut être vu comme un alignement de ses sommets dans une liste tel que tous les arcs soient orientés de gauche à droite.

1. Calculer le tri topologique du graphe G ci-dessous.



2. Modifiez l'algorithme de parcours en profondeur vu en cours pour qu'il calcule pour chaque nœud u sa date de fin de traitement $end[u]$ (la date à laquelle lui et ses fils ont été traités).

3. Quel lien pouvez-vous faire entre les dates de fin de traitement et un tri topologique ?

4. En vous appuyant sur l'algorithme du parcours en profondeur, proposer un algorithme de tri topologique et donner sa complexité.

Problème 3 (Examen 2019)

Un robot se promène sur le graphe non orienté $G = (S, A)$ représenté par la matrice d'adjacence ci-dessous :

$$G = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & s_7 & s_8 & s_9 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

1. Montrer que le graphe ci-dessus est planaire. C'est-à-dire qu'il peut être dessiné sans que ses arcs ne se croisent.

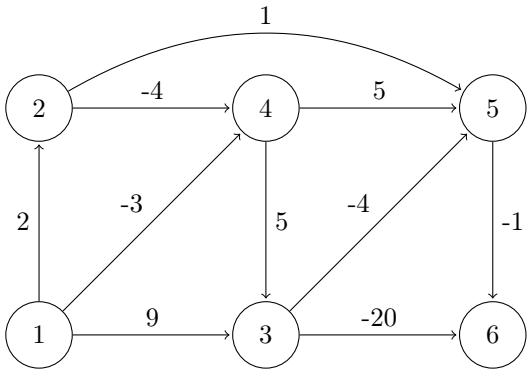
2. Partant d'un sommet quelconque s, appelé sommet de stockage, le robot est chargé de déposer un cube sur chacun des autres sommets. Il possède suffisamment de cubes sur le sommet de stockage, mais ne peut transporter qu'un seul cube à la fois (il doit donc repasser par le sommet de stockage avant de livrer un autre cube).

   On admet que le robot met 1 seconde pour passer d'un sommet s_i à un sommet s_j si $(s_i, s_j) \in A$.

   (a)) En utilisant un des algorithmes du cours que l'on précisera, calculer le temps de trajet minimal effectué par le robot depuis le sommet de stockage s_7 pour livrer un bloc à tous les autres sommets de G. On veillera à détailler les étapes de calcul.

(b) Quel est le meilleur sommet de stockage ? Est-il unique ? Expliquer et justifier votre raisonnement sans détailler ici les calculs intermédiaires de plus court chemin.

3. On considère désormais que le robot évolue sur le graphe orienté valué $G' = (S', A', \omega)$ donné ci-dessous :



On donne ici la signification des poids suivante :

Si $\omega(s_i, s_j) < 0$, cela signifie que le robot dépense de l'énergie pour aller du sommet s_i au sommet s_j. À l'inverse si $\omega(s_i, s_j) > 0$, alors le robot récupère de l'énergie entre les deux sommets.

En utilisant un des algorithmes du cours que l'on précisera, calculer le pire chemin (c'est-à-dire le chemin qui minimise l'énergie du robot à son arrivée) du sommet s_1 vers tous les autres sommets du graphe du graphe G'. On veillera à détailler les étapes de calcul.

4. Notre but est désormais de déterminer le chemin optimal entre tous les couples de sommets, c'est-à-dire l'ensemble des chemins permettant de maximiser la quantité d'énergie reçue par le robot à la fin de son trajet entre deux sommets quelconques. Pour se faire, on propose d'utiliser une variante de l'algorithme de Floyd que vous trouverez en annexe.

   On note ici $D^{(k)}$ et $P^{(k)}$ les matrices D et P de l'algorithme de Floyd modifié une fois l'itération k effectuée.

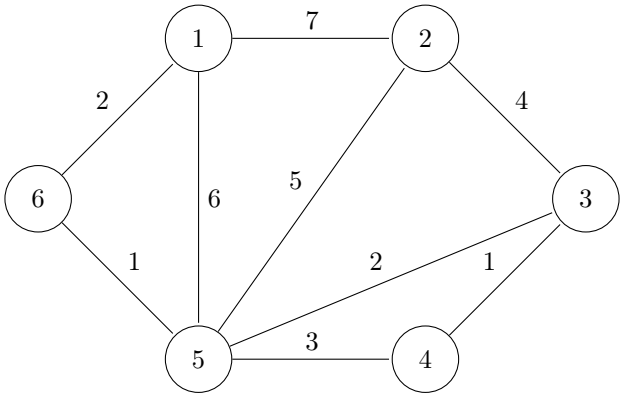
   (a) Écrire les matrices $D^{(0)}$ et $P^{(0)}$ pour l'initialisation, puis à l'aide des matrices $D^{(4)}$et $P^{(4)}$ données en annexe, calculer $D^{(5)}$et $P^{(5)}$.

   (b) Expliquer pourquoi on sait, sans faire de calcul, que $D^{(5)} = D^{(6)}$ et $P^{(5)} = P^{(6)}$. Quel est le chemin optimal de s_1 à s_6 ?

5. Un algorithme est dit correct s'il s'arrête et qu'il implémente sa spécification. Sous quelle(s) condition(s) l'algorithme de Floyd modifié est correct ?

Problème 4

Soit $G = (N, A, \omega)$, un graphe non dirigé, connexe et valué. Un arbre de recouvrement optimal $T = (N, A_T, \omega)$ est un arbre tel que $A_T \subseteq A$ et $\sum_{a \in A_T} \omega(a)$ est minimal.
Autrement dit, on cherche un arbre T de G, tel que la somme des poids des arcs de T est minimale.
Soit le graphe G suivant :



1. Appliquer l'algorithme de Prim pour trouver un arbre couvrant de poids minimal sur G. On commencera à construire l'arbre en partant du nœud 1.

2. Appliquer l'algorithme de Kruskal sur le même graphe, en ordonnant les arcs (x, y) par coût croissant, puis par indice x croissant, et enfin par indice y croissant. Que constatez-vous sur cet exemple ?

3. Dans quel cas peut-on avoir, pour un même graphe donné, plusieurs arbres de recouvrement de même coût minimal ? Expliquer pourquoi les algorithmes de Prim et Kruskal peuvent donner des résultats différents.

Annexe

$k = 4$

$D^{(4)}$

i	1	2	3	4	5	6
1	0	-2	-9	2	-5	11
2	∞	0	-1	4	-1	19
3	∞	∞	0	∞	4	20
4	∞	∞	-5	0	-5	15
5	∞	∞	∞	∞	0	1
6	∞	∞	∞	∞	∞	0

$P^{(4)}$

i	1	2	3	4	5	6
1	1	1	1	2	3	3
2	2	2	4	2	2	4
3	3	3	3	3	3	3
4	4	4	4	4	4	3
5	5	5	5	5	5	5
6	6	6	6	6	6	6

Algorithme 1 : Algorithme de Floyd modifié pour plus longs chemins

Data : $G = (S, A, \omega)$ `tel que` $S = \{1, ..., n\}$
Result : `Les plus longs chemins entre toutes paires de nœuds`
begin
  for $i : 1 \to n$ **do**
    for $j : 1 \to n$ **do**
      $D[i][j] \leftarrow -\omega(i, j)$ `// Opposé du poids entre` i `et` j
      $P[i][j] \leftarrow i$
  for $k : 1 \to n$ **do**
    for $i : 1 \to n$ **do**
      for $j : 1 \to n$ **do**
        if $D[i][k] + D[k][j] < D[i][j]$ **then**
          $D[i][j] \leftarrow D[i][k] + D[k][j]$
          $P[i][k] \leftarrow k$