TELECOM Nancy Apprentissage 2A Mercredi 10 décembre 2014
Graphes et Recherche Opérationnelle

Examen partiel (2)

Durée : 1h30
Documents autorisés : feuille manuscrite A4 recto-verso contenant des éléments du cours et des TD.

Exercice 1. *Parcours en profondeur (5pts)*

Le but de cet exercice est d'écrire en MATLAB l'algorithme de parcours en profondeur d'un graphe. Le graphe est décrit en MATLAB par une structure `G` qui comporte les champs suivants :
- `G.n` qui donne le nombre de noeuds n du graphe ;
- `G.succ` est un tableau de "cells" de taille `G.n` où

  `G.succ{k}` : tableau d'entiers contenant les numéros des successeurs du noeud `k`

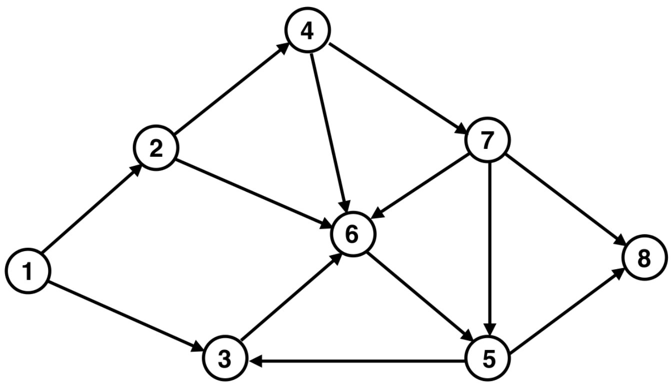
Le parcours profondeur utilise un tableau de marquage et une pile. Pour le marquage, on construit le tableau `marquage` de longueur n tel que `marquage(i)=1` si le sommet `i` est marqué, et `marquage(i)=0` sinon. Pour simuler une pile en MATLAB, on utilise un tableau `pile` de longueur n et un entier `top` représentant l'indice dans le tableau `pile` du haut de la pile.

```
pile = [ n1, n2, n3, .... , nk, ..... ]
                            ↑
                           top
```

Dans cet exemple l'indice `top` est égale à k.

Dans le parcours profondeur, on prend le sommet en haut de la pile et on empile le premier successeur non-marqué. Pour empiler un sommet, il faut d'abord mettre à jour l'indice `top` en l'incrémentant. Pour dépiler un sommet, il faut décrémenter l'indice `top`.

1. Indiquez l'ordre de parcours des sommets du graphe ci-dessous avec un parcours en profondeur en partant du sommet 1. Le premier successeur est celui qui a le plus petit numéro.



2. Compléter le script MATLAB en annexe pour le parcours profondeur d'un graphe `G` à partir d'un sommet `noeud_init`.

Exercice 2. *Programmation linéaire en variables binaires (5pts)*

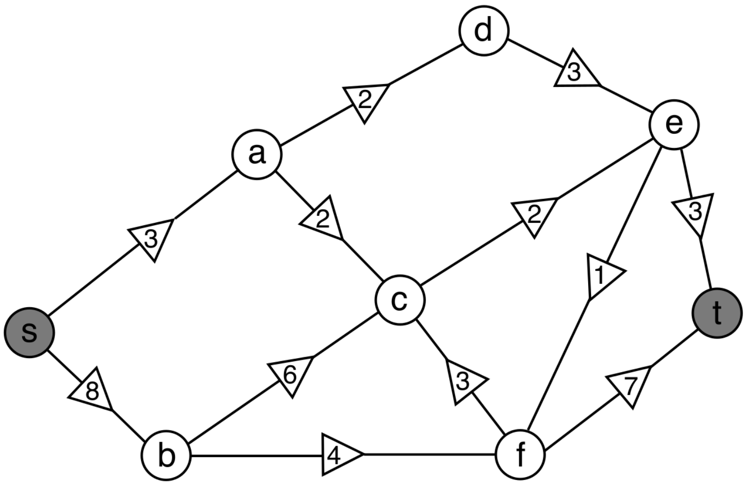
On considère le problème de programmation linéaire en variables $\{0, 1\}$ suivant (problème de "sac-à-dos") :

$$\begin{cases} \max\left[F(x) = 10x_1 + 8x_2 + 4x_3 + 5x_4\right] \\ 8x_1 + 3x_2 + 4x_3 + 2x_4 \leq 9 \\ 6x_1 + 4x_2 + 4x_3 + x_4 \leq 7 \\ x_1, x_2, x_3, x_4 \in \{0, 1\} \end{cases}$$

Résoudre ce problème par une procédure de séparation et évaluation ("Branch and bound"). Pour déterminer une solution réalisable initiale, vous examinerez les variables par ordre des coefficients **décroissants** dans F. Séparez toujours en premier le sous-ensemble qui a l'estimation principale la plus élevée.

Exercice 3. *Flot maximal dans un graphe (6pts)*

On considère le graphe valué suivant :



1. Déterminer le flot maximal à travers ce graphe de s à t en utilisant l'algorithme de Ford-Fulkerson. Utilisez un marquage en **file largeur** : à partir d'un sommet courant, marquer et mettre dans la file tous les sommets voisins qui sont accessibles et non encore marqués en respectant l'ordre **lexicographique**. On utilise toujours le sommet en tête de la file pour progresser. A chaque étape, vous construirez :
   – l'ensemble $\mathbb{E}$ des sommets de la pile (tous les sommets ajoutés à $\mathbb{E}$ sont marqués),
   – l'ensemble `origine` des successeurs/prédecesseurs des sommets de $\mathbb{E}$,
   – l'ensemble ε des améliorations possibles.

   Indiquez clairement la progression de la tête de la file.

   (*indication : convergence de l'algorithme en 5 étapes (pile vide)*).
2. Déterminer la coupe minimale correspondante.

Exercice 4. *Affectation multiple (4pts)*

Le tableau ci-dessous indique les affectations possibles pour un problème d'affectation multiple. Pour chaque offre L_i disposée en ligne, le maximum d'affectations possibles est indiqué par la valeur a_i. De même, pour chaque demande C_j en colonne, le maximum d'affectations possibles est b_j.

	C_1	C_2	C_3	C_4	C_5	a_i
L_1						19
L_2						28
L_3						9
L_4						5
b_j	5	10	10	14	10	

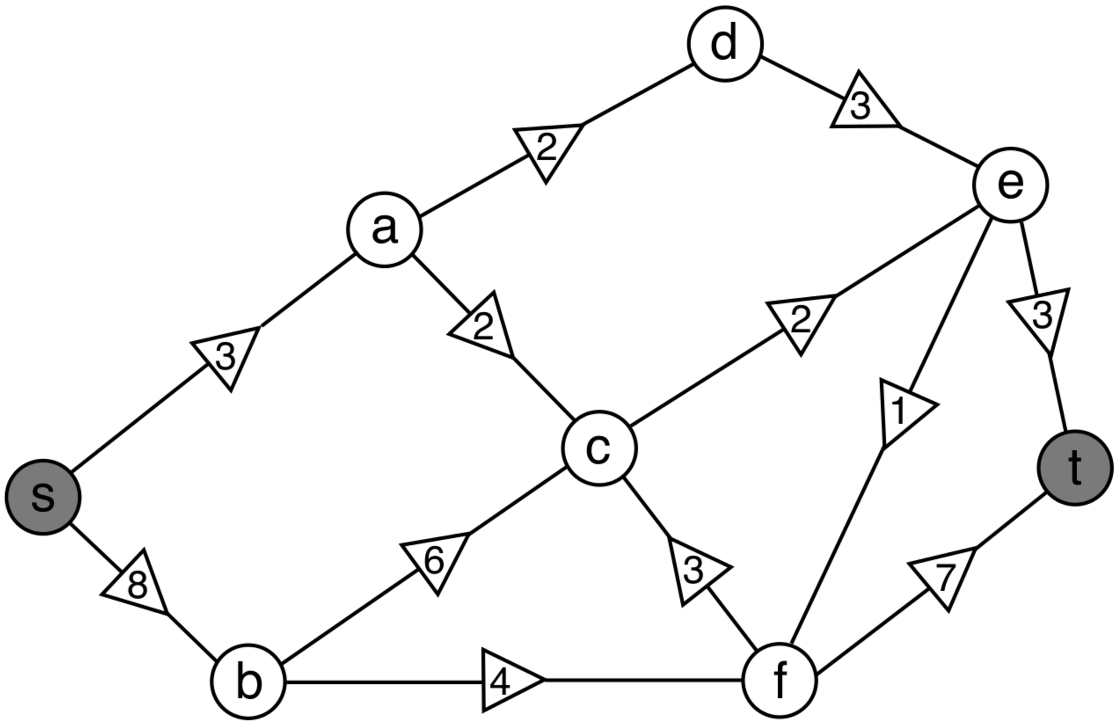
1. Trouver une initialisation des affectations par la méthode du coin "nord-ouest".
2. Donner le graphe biparti complété associé à ce problème d'affectation. Indiquer sur le graphe les flots correspondants à l'initialisation trouvée par la méthode du coin "nord-ouest".
3. Résoudre le problème d'affectation maximale par l'algorithme de Ford-Fulkerson avec un marquage en **pile profondeur**. Donner le tableau des affectations maximales.

Nom :

Exercice 1.

```
function parcours_profondeur(G,noeud_init)
   marque = zeros(1,G.n);                      % tableau pour marquage
   marque(noeud_init)=1;                       % on marque le noeud initial
   pile = zeros(1,G.n);                        % tableau pour la pile
   top = 1;                                    % initialisation de l'indice du haut de la pile
   pile(top) = noeud_init;                     % initialisation de la pile

   fprintf('\n noeuds successifs visités : %d', noeud_init)

   while top>0       % tant que la pile n'est pas vide

      % on prend le sommet Q en haut de la pile


      % on détermine le premier successeur P non marqué de Q :
      %   Si P existe alors nsucc=nn le numero de P et on affiche nsucc,
      %   Sinon nsucc=0
      nsucc = 0;










      if nsucc ~= 0   % si on a trouvé un successeur P non marqué, alors
          % maj de l'indice top du haut de la pile


          % on met le sommet P en haut de la pile


      else  % sinon on "supprime" Q de la pile
          % maj de l'indice top du haut de la pile


      end

   end   % Fin tant que
   fprintf('\n')
```

Exercice 3.



Exercice 4.

⋆ Initialisation des affectations par l'heuristique "coin nord-ouest" :

	C_1	C_2	C_3	C_4	C_5	a_i
L_1						19
L_2						28
L_3						9
L_4						5
b_j	5	10	10	14	10	

⋆ Graphe biparti complété :

⋆ Affectation maximale :

	C_1	C_2	C_3	C_4	C_5	a_i
L_1						19
L_2						28
L_3						9
L_4						5
b_j	5	10	10	14	10	