Algorithmique
Correction Partiel n° 1

Info-Spé – Epita

4 jan 2011 - 10 :00

Solution 1 **(Graphes : Court cours –** ***4 points*****)**

1. Il suffit dans la fonction d'appel de vérifier si tous les sommets ont été marqués. Cela peut se faire en contrôlant le marquage, ou à l'aide d'un compteur du nombre de sommets rencontrés qui, en retour de récursion, est comparé à l'ordre du graphe.

2. Non, parce que le fait qu'un sommet puisse atteindre tous les autres ne garantit pas le chemin inverse.

3. Ce sont des arcs en arrière ou des arcs croisés. Pour les différencier, il suffirait de comparer leur valeur suffixe. En effet, suff[x] est supérieur à suff[y] quand l'arc x→y est un arc croisé et inférieur quand l'arc x→y est un arc en arrière.

Solution 2 **(Graphes : dessiner c'est gagner –** ***4 points*****)**

1. Le graphe `G=<S,A>` non orienté correspondant à :
   `S={1,2,3,4,5,6,7,8,9,10}`
   `et A={(1,2),(1,6),(1,7),(2,3),(2,6),(3,1),(3,5),(4,3),(4,8),(4,9),(4,10),(5,1),(7,6),(8,5),(8,10),(10,9)}`
   est celui de la figure 1

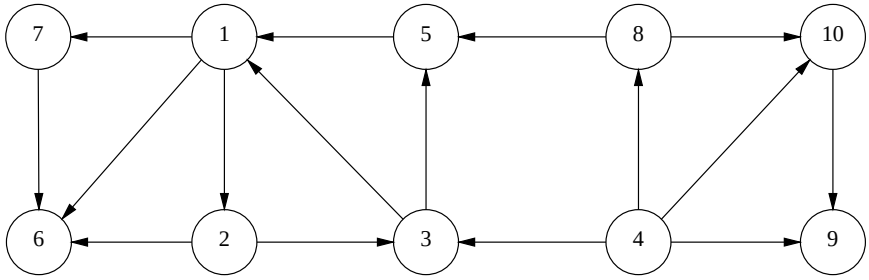

Fig. 1 – Graphe orienté.

2. Le tableau des degrés est le suivant :

	1	2	3	4	5	6	7	8	9	10
DemiDegréIntérieur	2	1	2	0	2	3	1	1	2	2

3. La forêt couvrante associée au parcours en profondeur du graphe `G` orienté est celui de la figure 2

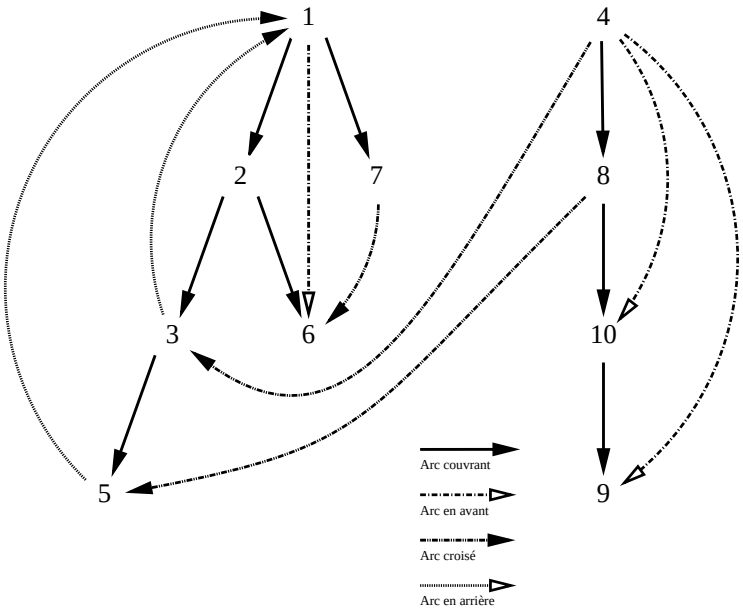

FIG. 2 – Graphe orienté G.

Solution 3 **(ARN : question d'équilibre – *5 points*)**

Spécifications : La fonction `test_arn (t_arn A)` retourne un entier négatif si A n'est pas équilibré, ou alors, 1 ou 0 selon que la racine de A est rouge ou pas.

```
algorithme fonction test_arn : entier
    parametres locaux
        t_arn    A
    parametres globaux
        entier    hauteur

    variables
        entier    hnoir_g, hnoir_d
        entier    rouge
debut
    si A = NUL alors
        hauteur ← -1
        retourne 0
    sinon
        rouge ← test_arn (A↑.fg, hnoir_g)
        si (rouge = -1) ou (A↑.rouge et (rouge = 1)) alors
            retourne (-1)
        fin si
        rouge ← test_arn (A↑.fd, hnoir_d)
        si (rouge =-1) ou ((A↑.rouge) et (rouge = 1)) ou (hnoir_g <> hnoir_d) alors
            retourne (-1)
        fin si
        si A↑.rouge alors
            hauteur ← hnoir_g
            retourne 1
        sinon
            hauteur ← hnoir_g + 1
            retourne 0
        fin si
    fin si
fin algorithme fonction test_arn
```

Solution 4 **Poids cumulé d'un arbre couvrant** – ***7 points***

1. **Principe :** Lors du parcours profondeur du sommet, on cumule, dans le vecteur poids, la somme des résultats de l'appel récursif pour chaque successeur non marqué et le coût des arcs couvrants utilisés.

```
algorithme fonction cumul : reel
  parametres locaux
    t_listsom                ps
  parametres globaux
    t_vect_entiers           pere
    t_vect_reel              poids
  variables
    t_listadj                pa
    entier                   s, sa
debut
  s ← ps↑.som
  pa ← ps↑.succ
  poids[s] ← 0
  tant que (pa <> NUL) faire
    sa ← pa↑.vsom↑.som
    si (pere[sa] = 0) alors
      pere[sa] ← s
      poids[s] ← ((poids[s] + pa↑.cout) + cumul(pa↑.vsom, pere, poids))
    fin si
    pa ← pa↑.suiv
  fin tant que
  retourne (poids[s])
fin algorithme fonction cumul
```

2.
```
algorithme fonction poids_cumul : reel
  parametres locaux
    entier                   s
    t_graphe_d               g
  parametres globaux
    t_vect_entiers           pere
    t_vect_reel              poids
  variables
    entier                   i
debut
  pour i ← 1 jusqu'a g.ordre faire
    pere[i] ← 0
    poids[i] ← ∞
  fin pour
  pere[s] ← - 1
  retourne(cumul(recherche(s, g), pere, poids))
fin algorithme fonction poids_cumul
```