# Algorithmique
# Correction Partiel n° 1

Info-Spé – Epita

*22 déc. 2009*

***Solution 1*** **(CC – *3 pts*)**

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

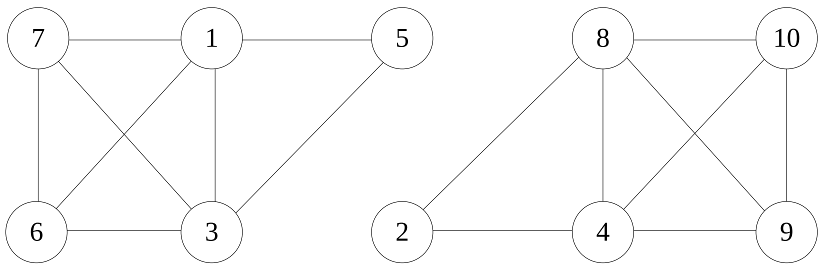

Fig. 1 – Graphe non orienté G.

2. Le tableau des degrés est le suivant : 

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| `Degré` | 4 | 2 | 4 | 4 | 2 | 3 | 3 | 4 | 3 | 3 |

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

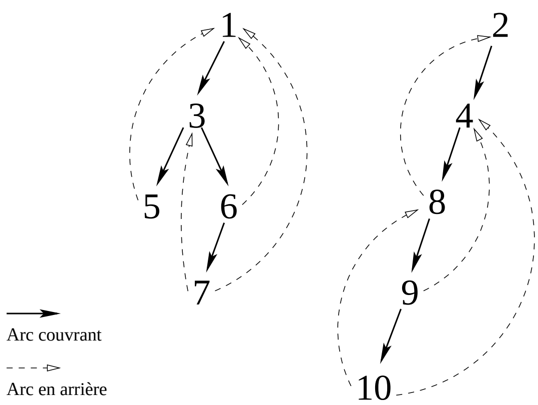


Fig. 2 – Forêt couvrante associée au parcours profondeur du Graphe non orienté G.

***Solution 2*** **(Largeur et principe – *6 pts*)**

1. Principe de l'algorithme :

   Le principe est simple, c'est celui d'un parcours en largeur d'un graphe orienté avec utilisation d'une file pour mémoriser hiérarchiquement les sommets du graphe. Une fois récupérer le sommet $s$, Le placement de l'affichage des arcs *(s,t)* ($t$ successeur de $s$) pouvant se faire n'importe où, en dehors du test de marquage (on veut les arcs, pas les sommets), dans l'itération basée sur le demi degré extérieur du sommet $s$.
   L'algorithme présenté affichera donc les arcs *(s,t)* dès la récupération du sommet $t$.

2. L'algorithme de parcours largeur d'un graphe orienté affichant les arcs rencontrés est :

```
algorithme procedure aff_arc_larg
parametres locaux
    entier      s
    graphe      g
parametres globaux
    t_vectNent  m
variables
    entier      i, t
    file        f
debut
    f ← filevide
    m[s] ← vrai
    f ← enfiler(s,f)

    tant que non(estvide(f)) faire
        s ← premier(f)
        f ← defiler(f)
        pour i ← 1 jusqu'a d°+(s,g) faire
            t ← i ème-succ-de s dans g
              ecrire ((s,t))  /* affichage de l'arc (s,t) */
            si non(m[t]) alors
                m[t] ← vrai
                f ← enfiler(t,f)
            fin si
        fin pour
    fin tant que

fin algorithme procedure aff_arc_larg
```

*Solution 3* **(Arbres AA – *5 pts*)**

**Spécifications :**

La fonction `insert_AA (t_element` $x$`, t_aAA` $A$`) : booleen` insère $x$ dans l'arbre $A$ sauf si celui-ci est déjà présent. Elle retourne un booléen indiquant si l'insertion a eu lieu.

```
algorithme fonction insert_AA : booleen
    parametres locaux
        t_element    x
    parametres globaux
        t_aAA   A
debut
    si A = NUL alors
        allouer (A)
        A↑.cle ← x
        A↑.niveau ← 1
        A↑.fg ← NUL
        A↑.fd ← NUL
        retourne (vrai)
    fin si
    si x = A↑.cle alors
            retourne (faux)
    fin si
    si x < A↑.cle alors
        si non (insert_AA (x, A↑.fg)) alors
            retourne (faux)
        sinon
            si A↑.fg↑.niveau = A↑.niveau alors
                skew (A)
                si (A↑.fd↑.fd <> NUL) et (A↑.fd↑.fd↑.niveau = A↑.niveau) alors
                    split (A)
                fin si
            fin si
            retourne (vrai)
        fin si
    sinon
        si non (insert_AA (x, A↑.fd)) alors
            retourne (faux)
        sinon
            si (A↑.fd↑.fd <> NUL) et (A↑.fd↑.fd↑.niveau = A↑.niveau) alors
                split (A)
            fin si
            retourne (vrai)
        fin si
    fin si
fin algorithme fonction insert_AA
```

***Solution 4* (Bipartite graph – *6 pts*)**

1. – Le graphe $G_3$ est biparti.
   – $S_1 = \{1, 4, 5, 9\}$ – $S_2 = \{2, 3, 6, 7, 8\}$

2. **Principe :**
   Pour tester si un graphe est biparti, il suffit de faire un parcours en largeur ou en profondeur en vérifiant que pour chaque arête empruntée, le sommet de départ n'est pas dans le même ensemble que le sommet d'arrivée. Il faut utiliser un système de marquage à deux valeurs (-1,1) afin de distinguer chaque ensemble.

   Le parcours est ici fait en profondeur. Dès qu'un sommet non marqué est trouvé, il prend la marque opposée de celle de son père. Si un sommet déjà marqué a la même marque que son prédécesseur, le graphe n'est pas biparti.
   Si aucune arête ne relie deux sommets de même marque (le parcours n'a pas été interrompu), le graphe est biparti.

   **Spécifications :**
   La fonction `test_rec (t_listsom` *ps*`, t_vect_entiers` *marque*`)` retourne un booléen indiquant si le sous-graphe parcouru à partir du sommet pointé par *ps* est biparti.

```
algorithme fonction test_rec : booleen
    parametres locaux
        t_listsom       ps
    parametres globaux
        t_vect_entiers    marque

    variables
        t_listadj    pa
        entier       s, sadj
debut
    s ← ps↑.som

    pa ← ps↑.succ
    tant que pa <> NUL faire
        sadj ← pa↑.vsom↑.som
        si marque[sadj] = 0 alors
            marque[sadj] ← - marque[s]
            si non test_rec (pa↑.vsom, marque) alors
                retourne faux
            fin si
        sinon
            si marque[sadj] = marque[s] alors
                retourne faux
            fin si
        fin si
        pa ← pa↑.suiv
    fin tant que

    retourne vrai
fin algorithme fonction test_rec
```