Algorithmique
Correction Partiel n° 2

Info-Spé – Epita

4 mai 2010

Solution 1 **(Dénombrement et Graphes non orientés – *6 points*)**

1. Le nombre d'arêtes d'un graphe simple non orienté complet de n sommets est
   (a) pour un graphe n'admettant pas les arêtes réflexives de : $\frac{n(n-1)}{2}$
   Justification :
   Le nombre d'arêtes pour un graphe non orienté de n sommets admettant les boucles est de $(n-1)$ pour un sommet. Pour n sommets, cela fait donc $n(n-1)$ arêtes que l'on divise par 2 dans la mesure où les relations sont symétriques (a—b = b—a).

   (b) pour un graphe n'admettant pas les arêtes réflexives de : $\frac{n(n+1)}{2}$
   Justification : Même chose que pour le point précédent les n arêtes réflexives en moins. On y ajoute alors les n arêtes réflexives, ce qui donne $\frac{n(n-1)}{2} + n$ que l'on simplifie en $\frac{n(n+1)}{2}$

2. Construction d'un graphe non orienté simple :

   (a) NON. Si le graphe contient 4 sommets, le degré maximum de chacun d'eux est de 3. Soit une somme totale d'arêtes de 4x3=12. Cette somme étant égale au double du nombre d'arêtes (graphe non orienté), celui-ci ne peut excéder 6.

   (b) OUI. Si le graphe contient 5 sommets, le degré maximum de chacun d'eux est de 4. Soit une somme totale d'arêtes de 5x4=20. Cette somme étant égale au double du nombre d'arêtes (graphe non orienté), celui-ci ne peut excéder 10, donc 9 pas de problème.

   (c) NON. Si le graphe contient 10 sommets, le degré maximum de chacun d'eux est de 9. Soit une somme totale d'arêtes de 10x9=90. Cette somme étant égale au double du nombre d'arêtes (graphe non orienté), celui-ci ne peut excéder 45.

3. Si le graphe contient n sommets, le degré maximum de chacun d'eux est de $n-1$. Soit une somme totale d'arêtes de $n(n-1)$. Cette somme étant égale au double du nombre d'arêtes (graphe non orienté), celui-ci ne peut excéder $(n(n-1))/2$. Donc la constuction d'un graphe simple de n sommets et p arêtes est possible si et seulement si $p \leqslant (n(n-1))/2$.

Solution 2 **(ARM et autres... – *4 points*)**

1. Un arbre de recouvrement d'un graphe G non orienté est un graphe partiel de G qui est un arbre.

2. Il y a trois ARPM. Il faut mettre toutes les arêtes de poids 1. Pour celles de poids 2 il faut les prendre toutes, sauf l'une des 3 arêtes du cycle 2,4,5. Trois possibilités donc 3 arbres.

3. Un des trois ARM, par exemple le graphe de la figure 1.

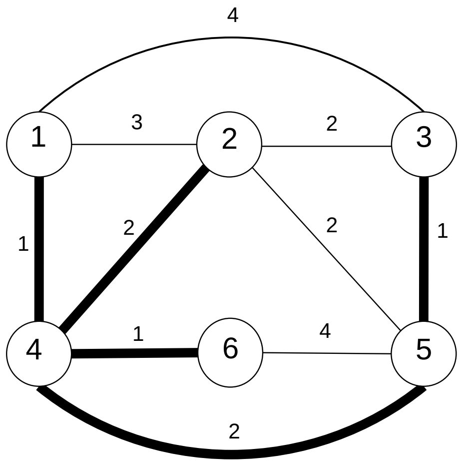

Fig. 1 – ARM sur graphe non orienté valué

4. Un arbre de recouvrement G' de G=<S,A,C> est défini par <S,A',C> avec A'⊆A. Comme A est fini, il y a un nombre fini de sous-ensembles A' ayant chacun pour coût C(A') correspondant aux coûts cumulés des arêtes constituant A'. Or tout ensemble fini admet un minorant, CQFD.

5. Minimiser le coût d'installation de lignes téléphoniques, Minimiser le coût de gestion d'alimentation électrique de plusieurs villes, etc.

Solution 3 **(Plus court chemin et parcours profondeur – *17 points*)**

1. $s \rightarrow sa$ est un arc en arrière si : $os[s] < os[sa]$

2.
```
algorithme procedure pprof_rec
  parametres locaux
    t_listsom         ps
  parametres globaux
    entier            cpt
    t_vect_entiers    op, os
    t_pile            p
  variables
    t_listadj         pa
    entier            s, sa
debut
  s ← ps↑.som
  cpt ← (cpt + 1)
  op[s] ← cpt
  pa ← ps↑.succ
  tant que (pa <> NUL) faire
    sa ← pa↑.vsom↑.som
    si (op[sa] = 0) alors
      pprof_rec(pa↑.vsom, cpt, op, os, p)
    fin si
    pa ← pa↑.suiv
  fin tant que
  cpt ← (cpt + 1)
  os[s] ← cpt
  p ← empiler(ps, p)
fin algorithme procedure pprof_rec
```

3.
```
algorithme procedure bellman_prof
  parametres locaux
    t_graphe_d        g
    entier            src
    t_vect_entiers    op, os
  parametres globaux
    t_pile            p
    t_vect_entiers    pere
    t_vect_reels      dist
  variables
    t_listsom         ps
    t_listadj         pa
    entier            s, sa
debut
  pour s ← 1 jusqu'a g.ordre faire
    pere[s] ← 0
    dist[s] ← ∞
  fin pour
  pere[src] ← src
  dist[src] ← 0
  tant que non est_vide(p) faire
    ps ← depiler(p)
    s ← ps↑.som
    pa ← ps↑.succ
    tant que (pa <> NUL) faire
      sa ← pa↑.vsom↑.som
      si non (os[s] < os[sa]) alors
        si (dist[sa] > (dist[s] + pa↑.cout)) alors
          dist[sa] ← (dist[s] + pa↑.cout)
          pere[sa] ← s
        fin si
      fin si
      pa ← pa↑.suiv
    fin tant que
  fin tant que
fin algorithme procedure bellman_prof
```

4.

	1	2	3	4	5	6	7	8
op	1	2	14	3	4	5	6	7
os	16	13	15	12	11	10	9	8

p	1	3	2	4	5	6	7	8

5. Arcs ignorés : $6 \rightarrow 2$ et $8 \rightarrow 7$

6.

	1	2	3	4	5	6	7	8
dist	0	1	5	2	1	-2	6	-9
pere	1	1	1	2	4	5	5	2

7. $1 \rightarrow 2$ (1)

8. **OUI** : $1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 2$ (-11)

Solution 4 (Graphe réduit – *3 points*)

Spécifications :

La procédure `graphe_reduit (t_graphe_s` G, `t_vect_entiers` cfc, `entier` nb_cfc, `t_graphe_s` Gr`)` construit Gr graphe réduit de G à partir du vecteur des composantes fortement connexes de G, cfc, ainsi que leur nombre nb_cfc.

```
algorithme procedure graphe_reduit
    parametres locaux
        t_graphe_s      G
        t_vect_entiers  cfc        /* le vecteur des composantes fortement connexes */
        entier          nb_cfc      /* le nombre de composantes */
    parametres globaux
        t_graphe_s      Gr

    variables
        entier    x, y
  debut
    Gr.orient ← vrai
    Gr.ordre ← nb_cfc
    pour x ←  jusqu'a Gr.ordre faire
        pour y ← Gr.ordre faire
            Gr.adj[x,y] ← 0
        fin pour
    fin pour

    pour x ← 1 jusqu'a G.ordre faire
        pour y ← 1 jusqu'a G.ordre faire
            si G.adj[x,y] <> 0 alors
                si cfc[x] <> cfc [y] alors
                    Gr.adj[cfc[x], cfc[y]] ← 1
                fin si
            fin si
        fin pour
    fin pour
  fin algorithme procedure graphe_reduit
```