- Contrôle continu - SUJET A -

- CORRIGÉ -
Novembre 2015

- Exercice 1 - ACPM et voyageur de commerce *(4 points)* -

a. **(1.5 pts)** On obtient l'arbre d'arêtes $\{v_1v_2, v_2v_3, v_1v_4, v_4v_6, v_4v_5\}$ de poids total 21.

b. **(1 pt)** Voir cours. On obtient notamment une tournée de longueur totale mesurant au plus deux fois la longueur de la meilleure tournée possible.

c. **(1 pt)** Il y a de nombreuses possibilités. Une des possibilités donne la tournée : $v_1v_2v_3v_4v_5v_6$ de longueur totale $7 + 5 + 12 + 2 + 4 + 10 = 40$.

d. **(0.5 pts)** Une tournée moins longue est $v_1v_2v_3v_6v_5v_4$ de longueur 31. Ce n'est pas forcément la meilleure.. ?

- Exercice 2 - Parcours *(3.5 points)* -

1. **(1 pt)** Algo en $O(n)$ (on passe au plus une fois par sommet).

Algorithme : Niveau(x,r, pere)

```
début
    niv ⟵ 0;
    tant que x ≠ r faire
        x ⟵ pere(x);
        niv ⟵ niv + 1
    fin
    retourner niv;
fin
```

2. **(1 pt)** Algo en $O(n)$ (on passe au plus une fois par sommet).

Algorithme : Ancetre(x,y,r,pere)

```
début
    tant que x ≠ r ET x ≠ y faire
        x ⟵ pere(x);
    fin
    si x = y alors retourner VRAI;
    si x = r alors retourner FAUX;
fin
```

3. **(1.5 pt)** Algo en $O(mn)$: on teste si l'arbre est *normal*, c-à-d si pour toute arête xy du graphe, x est ancêtre de y ou y est ancêtre de x.

Algorithme : largeur ?(G,r,pere)
début
  pour $xy \in E(G)$ **faire**
    si *Ancetre(x,y,r,pere)=FAUX et Ancetre(y,x,r,pere)=FAUX* **alors**
      retourner *FAUX*;
    fin
  fin
  retourner *VRAI*;
fin

- Exercice 3 - Re-parcours *(1.5 point)* -

On obtient par exemple :
File des sommets à traiter : $AT = (e, a, b, h, d, c, f, g)$

`sommet`	a	b	c	d	e	f	g	h
`pere`	e	e	a	a	e	b	d	a
`ordre`	2	3	6	5	1	7	8	4
`niveau`	1	1	2	2	0	2	3	2

- Exercice 4 - Algo mystère... *(3.5 points)* -

a. **(1.5 pt)** Un exemple de déroulement : on traite d'abord les arêtes incidentes à a, on sélectionne a dans X, puis les arêtes restantes incidentes à b, on sélectionne b dans X, puis celles incidentes à d, on sélectionne d dans X, puis celles restantes incidentes à e et h, enfin celles restantes incidentes à f et g pour lesquels on sélectionne f et g dans X. On a finalement $X = \{a, b, d, f, g\}$.

b. **(1 pt)** La boucle ligne 3 prend un temps $O(n)$. La boucle ligne 3 prend un temps $O(m)$, le test et les affectations dans la boucle se faisant en temps constant. En tout `ALGO` s'exécute en temps $O(n + m)$.

c. **(0.5 pt)** A la fin de l'exécution de `ALGO`, toute arête du graphe est incidente à un sommet de X.

d. **(0.5 pt)** `ALGO` aurait pu retourner par exemple l'ensemble $\{a, b, c, d\}$.

- Exercice 5 - Graphes eulériens *(4.5 points)* -

a. **(0.5pt)** Un cycle par exemple est un graphe eulérien.

b. **(0.5pt)** Non, un arbre ayant au moins deux sommets contient au moins une (même deux) feuille qui est un sommet de degré 1. Un arbre ne peut donc pas être eulérien.

c. **(1pt)** Soit G un graphe eulérien et C un cycle de G. Tous les sommets de G ont un degré pair et les sommets $V(C)$ de C ont degré 2 dans C. Si on enlève les arêtes de C à G, les sommets de $V(G) \setminus V(C)$ ont un degré inchangé et les sommets de $V(C)$ perdent 2 à leur degré. Finalement les sommets de $G - C$ ont tous un degré pair et $G - C$ est eulérien.

d. **(2.5pts)** On prouve le résultat par récurrence sur le nombre d'arêtes du graphe considéré. Si le graphe a 0 arête est est connexe, il contient un seul sommet et admet donc une marche eulérienne. Sinon soit G un graphe eulérien connexe. Par la question b., G n'est pas un arbre, il contient donc au moins un cycle C. Le graphe $G - C$ est eulérien par la question c. Notons $G_1, \dots, G_c$ ses composantes connexes. Chaque graphe G_i est connexe et eulérien, donc admet une marche eulérienne M_i par hypothèse de récurrence. Pour i de 1 à c on note x_i un sommet de G_i appartenant à C ainsi que P_i le chemin de C de x_i à x_{i+1} (avec $x_{c+1} = x_1$). Quitte à modifier la marche M_i, on peut supposer que celle-ci commence et termine par le sommet x_i. Ainsi, G admet la marche eulérienne $M_1P_1M_2P_2 \dots P_{c-1}M_cP_c$.