DS - Algorithmique 2 - Master 1 Bioinformatique

23 mai 2015 - Durée : 1h30 minutes

Les documents, les ordinateurs et les téléphones ne sont pas autorisés.

Exercice 1

On suppose que vous disposez d'un module python implémentant les arbres binaires et qui contient les fonctions suivantes :

```
def racine(A)
    # renvoie la racine de l'arbre A
def pere(A,p)
    # renvoie le père du sommet p dans l'arbre A ou None
    # si c'est la racine
def fils(A, p)
    # renvoie la liste des fils du sommet p dans l'arbre A
def etiquette(A, p)
    # renvoie l'etiquette du sommet p dans l'arbre A
def modifier_etiquette(A, p, valeur )
    # remplace l'etiquette du sommet p de l'arbre A par
    # l'étiquette e passé en paramètre.
```

Soit T un arbre.

1. Proposez un algorithme qui étiquette les sommets de l'arbre par des numéros de tels sortes que, pour tout sommet s, l'étiquette de s est le nombre de sommets que contient le sous-arbre de racine s dans T. Vous illustrerez votre algorithme par un exemple en vous contentant de donner l'arbre et son étiquettage.
2. Pour tout arbre T, on appelle formule des équerres de T, notée $eq(T)$, l'entier défini de la façon suivante :

$$eq(T) = \frac{n!}{\prod_{s \text{ sommet de } T} \text{etiquette(T, s)}}.$$

On rappelle que $n! = 1 \times 2 \times 3 \times \cdots \times n$ et que

$$\prod_{s \text{ sommet de } T} \text{etiquette(T, s)}$$

est le produit de toutes les étiquettes de l'arbre.

Proposez un algorithme qui prends en paramètre un arbre et qui renvoie la formule des équerres de cet arbre. Vous illustrerez votre algorithme en donnant pour l'exemple de la question précédente sa formule des équerres.

Exercice 2

Supposons qu'après avoir appliqué l'algorithme du parcours en largeur `PL(G)` à un graphe non orienté simple ayant six sommets notés s_i avec $1 \leq i \leq 6$, nous ayons obtenu le tableau des distances `d[`s_i`]` suivant :

`d[`s_1`]`	`d[`s_2`]`	`d[`s_3`]`	`d[`s_4`]`	`d[`s_5`]`	`d[`s_6`]`
0	1	2	1	1	2

1. Dessiner un arbre T pouvant être un arbre couvrant obtenu par l'appel de `PL(G)` ayant produit le tableau `d[]` ci-dessus.
2. Dessiner un graphe (autre que T) ayant six sommets et pouvant avoir comme arbre couvrant en largeur l'arbre T définit précédemment.

Exercice 3

1. Rappeler la définition de cycles et de circuits d'un graphe.
2. Donner un algorithme qui détermine si le graphe possède un cycle.
3. À partir de maintenant et jusqu'à la fin de l'exercice, on supposera que les graphes sont tous connexes. Qu'est-ce qu'un arbre couvrant. Proposer un algorithme qui prends en paramètre un graphe et qui détermine l'arbre couvrant du graphe. On pourra coder l'arbre couvrant par une liste d'arêtes.
4. Le nombre cyclomatique d'un graphe est le nombre d'arêtes minimales qu'il faut enlever au graphe pour qu'il n'ai plus de cycles. Ce nombre se calcule facilement à partir d'un arbre couvrant T quelconque : c'est le nombre d'arêtes du graphe qui ne sont pas des arêtes de l'arbre T. Donnez 4 exemples de graphes, dont le nombre cyclomatique vaut 0, 1, 2 et 3.
5. Proposer un algorithme qui calcule le nombre cyclomatique d'un graphe.
6. Proposer un algorithme qui renvoie n cycles différents où n est le nombre cyclomatique du graphe.

Rappel :

```
0   PL(G,s)
1     pour chaque sommet u de X[G] \ {s} faire
2         couleur[u] <- BLANC
3         d[u] <- infini
4         pere[u] <- nil
5     couleur[s] <- GRIS
6     d[s] <- 0
7     pere[s] <- nil
8     Enfiler(F, s)
9     tant que non vide(F) faire
10        u <- tete(F)
11        pour chaque v de Adj(u) faire
12            si couleur[v] = BLANC
13                alors couleur[v] <- GRIS
14                      d[v] <- d[u] + 1
15                      pere[v] <- u
16                      Enfiler(F, v)
17        Defiler(F)
18        couleur[u] <- NOIR
```