DS Algorithmique - BioInformatique M1
Session 2

Université de Bordeaux - 22 juin 2016

Les documents ne sont pas autorisés.

Les différents exercices de ce devoir portera sur le graphe G de la figure 1.

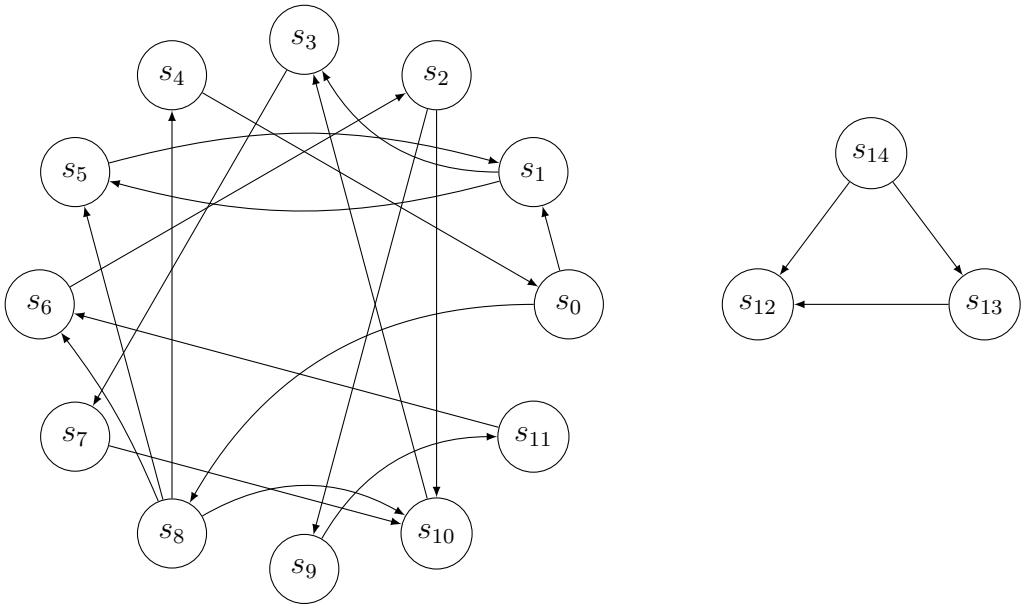

Figure 1: Un graphe orienté.

Exercice 1

1. Réalisez le parcours en largeur du graphe de la figure 1 en partant du sommet S_0. Vous donnerez les tables d, *pere*, *couleur*, ainsi que la pile F et vous réaliserez le parcours en utilisant l'ordre lexicographique.

2. Réalisez le parcours en profondeur du graphe de la figure 1. Vous donnerez les tables d, f, *pere* ainsi que l'arbre d'exécution de `visiter_pp`. Vous réaliserez le parcours en utilisant l'ordre lexicographique.

Exercice 2

1. Écrivez un algorithme (en pseudo-code), qui prend en paramètres un graphe G et deux sommets s_1 et s_2, et qui renvoie le chemin le plus court entre s_1 et s_2 codé sous forme d'une liste de sommets telle que le premier élément de la liste est s_1 et le dernier élément est s_2.

2. Appliquez votre algorithme sur le graphe de la figure 1 pour déterminer le plus court chemin entre les sommets S_0 et S_3.

3. Pour un graphe donné et deux sommet s_1, s_2 est-il possible d'avoir deux plus courts chemins différents ? Justifiez votre réponse.

Exercice 3

1. Donnez la définition de composantes connexes et de composantes fortement connexes d'un graphe.

2. Donnez les composantes connexes et les composantes fortement connexes du graphe de la figure 1.

3. Écrivez un algorithme en pseudo-code qui prend en paramètre un graphe G et qui renvoie l'ensemble des composantes connexes de G. Cette fonction renverra une liste $[c_1, c_2, \cdots]$ où c_i est la liste de tous les sommets d'une composante connexe.

Exercice 4

Dans la théorie des graphes, un arbre est un graphe connexe sans cycle.

1. Donnez des exemples d'arbres et des exemples de graphes qui ne sont pas des arbres.

2. Proposez un algorithme `est_un_arbre(G)` qui renvoie vraie si le graphe G est un arbre et qui renvoie faux sinon.

Exercice 5

Supposons que vous disposez uniquement des fonctions suivantes :

```
def fils( T, s ):
    '''
    Renvoie la liste des fils du sommet s dans l'arbre enraciné T
    '''

def racine( T ):
    '''
    Renvoie la racine de l'arbre T
    '''
```

Proposez un algorithme qui renvoie la liste des sommets situés à hauteur paire. On supposera que la racine est à hauteur 0.

Annexes

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

```
1 PP(G)
2   pour chaque sommet u de G faire
3       couleur[u] <- BLANC
4       pere[u] <- nil
5   temps <- 0
6   pour chaque sommet u de G faire
7       si couleur[u] = BLANC
8           alors Visiter_PP(u)
```

```
1 Visiter_PP(u)
2   couleur[u] <- GRIS
3   d[u] <- temps <- temps + 1
4   pour chaque sommet v adjacent à u faire
5       si couleur[v] = BLANC
6           alors pere[v] <- u
7                 Visiter_PP(v)
8   couleur[u] <- NOIR
9   f[u] <- temps <- temps + 1
```