

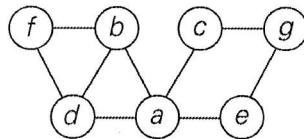
Algorithmique - ROB3

TD6 - Parcours de graphes

— Exercice 1 (Parcours en largeur et en profondeur)

Question 1.1 Soit $G = (S, A)$ un graphe non-orienté. Si G est un arbre, quelle est la complexité d'un parcours en profondeur de G ?

On considère dans les deux questions qui suivent le graphe non-orienté suivant :



Question 1.2 En démarrant du sommet a , et en départageant les ex-aequos par l'ordre alphabétique des sommets, écrire la séquence de visites des sommets dans un parcours en profondeur, et tracer la forêt couvrante associée.

Question 1.3 En démarrant du sommet a , et en départageant les ex-aequos par l'ordre alphabétique des sommets, écrire la séquence de visites des sommets dans un parcours en largeur, et tracer la forêt couvrante associée.

Question 1.4 Donner un graphe non orienté connexe à 5 sommets, et un sommet de départ, pour lesquels le parcours en profondeur et le parcours en largeur sont les mêmes.

— Exercice 2 (Détection de graphe biparti)

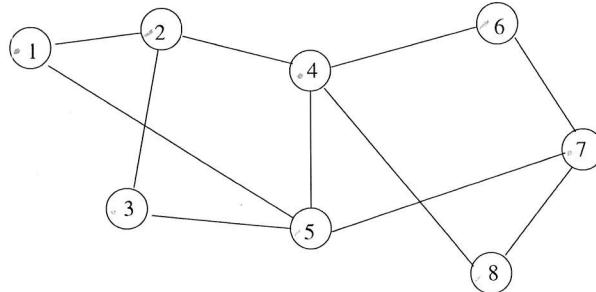
Soit $G = (S, A)$ un graphe non orienté connexe. L'objet de cet exercice est de concevoir un algorithme qui détermine si G est biparti. Un graphe est biparti si l'ensemble S peut être partitionné en deux sous-ensembles S_1 et S_2 (avec $S_1 \cup S_2 = S$ et $S_1 \cap S_2 = \emptyset$) tel qu'il n'existe pas d'arêtes entre sommets d'un même sous-ensemble (par exemple, si $x, y \in S_1^2$ alors il n'existe pas d'arête entre x et y).

Soit $L = (s_1, \dots, s_n)$ un parcours de G à partir d'un sommet quelconque de S et $F(L)$ la forêt couvrante associée à L . On colore (en rouge ou bleu) tous les sommets de S selon la règle de coloration suivante :

- le sommet s_1 est bleu ;
- pour $i = 2, \dots, n$, le sommet s_i est bleu (resp. rouge) si son père dans $F(L)$ est rouge (resp. bleu).

Le but de cet exercice est de montrer que G est biparti si et seulement si il n'existe pas d'arête de A entre deux sommets de même couleur.

Question 2.1 Faire un parcours (quelconque) du graphe de la page suivante à partir du sommet 1, en indiquant la liste L obtenue et une forêt couvrante associée $F(L)$. En utilisant la règle de coloration présentée plus haut, préciser si le graphe est biparti.



On considère désormais un graphe $G = (S, A)$ non-orienté connexe quelconque. L'objet des questions 2 à 4 est de prouver que G est biparti si et seulement si il n'existe pas d'arête de A entre deux sommets de même couleur.

Question 2.2 Montrer que s'il n'existe pas d'arête entre deux sommets de même couleur dans $A \setminus F(L)$, alors le graphe G est biparti.

Indication : on pourra noter S_r le sous-ensemble des sommets rouges et S_b le sous-ensemble des sommets bleus.

Question 2.3 Soit G un graphe biparti. Montrer que :

- a) s'il existe une chaîne de longueur paire (en nombre d'arêtes) dans G entre deux sommets x et y alors ces deux sommets x et y appartiennent nécessairement au même sous-ensemble (S_1 ou S_2) dans la partition des sommets;
- b) s'il existe une chaîne de longueur impaire dans G entre deux sommets x et y alors l'un appartient à S_1 et l'autre à S_2 .

Question 2.4 Soit L un parcours de G . Après coloration, on considère une arête $\{x, y\}$ de G entre deux sommets de même couleur. En considérant la chaîne qui relie x et y dans $F(L)$, montrer que le graphe n'est pas biparti.

On déduit naturellement des résultats précédents que l'on peut déterminer si un graphe G est biparti en réalisant un parcours quelconque avec coloration des sommets, puis en parcourant les arêtes de G afin de tester si il existe une arête dont les deux extrémités sont de même couleur. On souhaiterait toutefois déterminer si un graphe est biparti sans qu'il soit nécessaire de reparcourir les arêtes du graphe au terme du parcours.

Question 2.5 Compléter la procédure de la page suivante pour qu'elle permette de détecter si un graphe est biparti en $O(n + m)$. La variable globale `est_biparti` est initialisée à vrai. La valeur de cette variable à l'issue de la procédure doit être vrai si le graphe est biparti, et faux sinon. Le tableau C indexé sur les sommets indique la couleur des sommets. Il est initialisé de la façon suivante : pour tout x , $C[x] = \text{incolore}$. Etant donné un sommet s_1 de départ, le premier appel de la procédure est `Parcours_Rec(G, s1, bleu)`.

```

procedure Parcours_Rec( $G : \text{Graphe}$ ,  $s : \text{Sommet}$ ,  $c : \text{Couleur}$ )
variable locale  $x : \text{Sommet}$ 
 $C[s] \leftarrow c$ 
pour tout voisin  $x$  de  $s$  faire
    si  $alors est\_biparti \leftarrow \text{faux}$ 
    fin si
    si  $C[x] = \text{incolore}$  alors
        fin si
    fin pour

```

— Exercice 3 (Stable maximum)

Soit un graphe non-orienté $G = (V, E)$ avec $V = \{1, \dots, n\}$. Un sous-ensemble de sommets est dit stable s'il n'existe pas d'arête entre eux. Le problème du stable maximum vise à déterminer un sous-ensemble stable $S \subseteq \{1, \dots, n\}$ comportant un nombre maximum de sommets. Trouver un stable maximum dans un graphe est un problème difficile. Par contre, si le graphe est une forêt (un ensemble d'arbres), c'est un cas particulier plus facile. Nous nous intéressons précisément dans cet exercice au cas particulier où G est une forêt.

Question 3.1 Montrer que si v est une feuille d'un arbre de la forêt, il existe un stable maximum qui contient v .

On considère dans la suite l'algorithme glouton ci-dessous, où $\Gamma(i)$ désigne l'ensemble des voisins du sommet i dans G et $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ est une fonction bijective qu'il va vous falloir identifier.

```

 $S \leftarrow \emptyset ; I \leftarrow \emptyset ; /* S est le stable en construction, I est l'ensemble des voisins des sommets de S */$ 
déterminer  $\pi /*$  voir question 3.2 */
for  $i = 1$  to  $n$  do
    if  $\pi(i) \notin I$  then
         $S \leftarrow S \cup \{\pi(i)\}$ 
         $I \leftarrow I \cup \Gamma(\pi(i))$ 
    end if
end for
return  $S$ 

```

Question 3.2 D'après la question précédente, pour que l'algorithme glouton soit valide (autrement dit, retourne un stable maximum), il est suffisant que, lors de chaque itération i de

la boucle **pour**, on vérifie : $\pi(i) \in I$ ou $\pi(i)$ est une feuille dans le sous-graphe induit par $V \setminus (S \cup I)$. A l'aide d'un parcours en profondeur **depuis le sommet 1**, déterminer une fonction π vérifiant cette condition pour le graphe G représenté sur la figure 1. Au cours du parcours en profondeur, à chaque itération on visitera en priorité le sommet de plus petit indice parmi les voisins du dernier sommet visité ouvert. On donnera l'ordre de visite des sommets lors du parcours en profondeur, ainsi que les numérotations $\text{pre}[v]$ et $\text{post}[v]$ ($v \in \{1, \dots, n\}$) telles que définies en cours. A partir de la numérotation $\text{post}[v]$, on indiquera comment déterminer une fonction π vérifiant la condition énoncée plus haut, et on donnera cette fonction pour G .

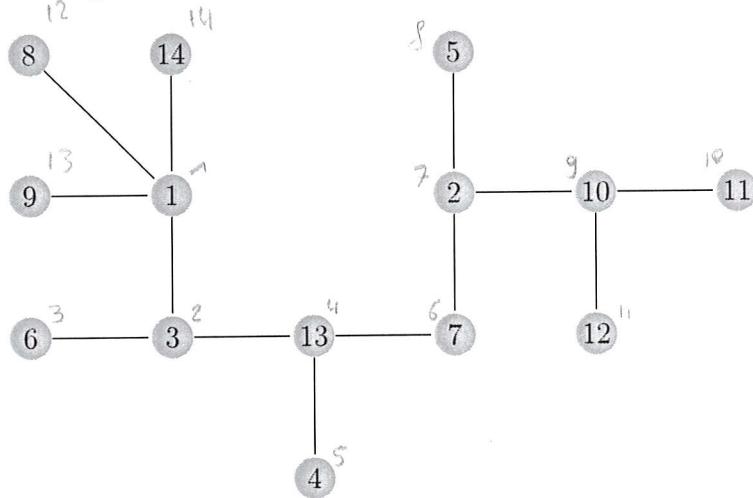


FIGURE 1 – Graphe G .

Question 3.3 En déduire un sous-ensemble stable maximum de sommets dans le graphe précédent.

Question 3.4 Quelle est la complexité de cet algorithme glouton ? On n'oubliera pas de prendre en compte le temps passé à déterminer π , et de tenir compte également que le graphe est une forêt. On précisera de plus l'hypothèse faite concernant la représentation en machine du graphe et des ensembles S et I pour obtenir cette complexité.

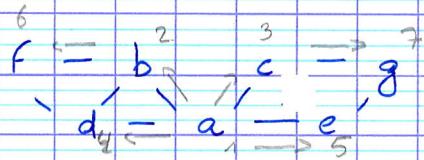
TD 6

Exercice 1

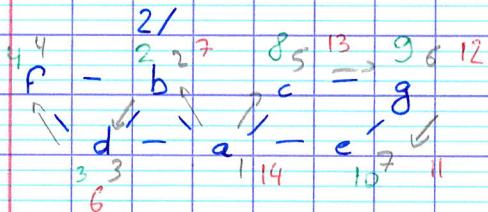
1)

$O(n)$ car $m = n \rightarrow 1$

2/



(a, b, c, d, e, F, g) : parcours en largeur.



(a, b, d, F, c, g, e) : parcours en profondeur.

Numéro de prédécesseur (visite).

Numéro de post visite ("rebrousse chemin") quand on a pas de voisins

Remarque :

[pres, ponto] [pres', ponto']

$\{ \begin{array}{l} \text{pres}' > \text{pres} \\ \text{ponto}' < \text{ponto} \end{array} \}$: Arc de la forêt couvrante

$\Rightarrow [\text{pres}', \text{ponto}'] \subset [\text{pres}, \text{ponto}]$

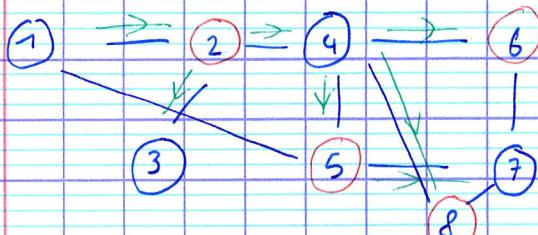
Si (s, s') est un arc de la forêt couvrante du parcours.

4/

① ② ③ ④ ⑤ Parcours identiques (en largeur et en profondeur) si on part du sommet 1.

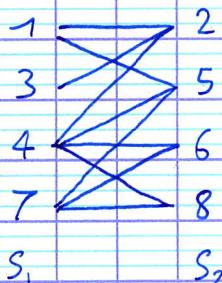
Exercice 2

1/



Parcours géminique: $(1, 2, 4, 3, 5, 7, 6, 8)$

Le graphe est bipartie entre les sommets rouges car il n'y a aucune arête entre sommets de même couleur.



2/

Par construction, il n'existe pas d'arête entre deux sommets du même couleur dans $F(L)$. Si il n'y a pas non plus d'arête monocolore dans $A \setminus F(L)$, alors, il n'y a pas d'arête monocolore dans A et le graphe est bipartie entre sommets bleus et sommets rouges.

3/

Soit $\mu = (x_1 = x, \dots, y = x_p)$ une chaîne entre x et y dans G . Sans perte de généralité, supposons que $x_i \in S_1$. On a nécessairement $x_2 \in S_2$ (car pas d'arête entre deux sommets de S_1), $x_3 \in S_1$, etc.

- o Preuve du a): Si la chaîne μ est de longueur paire, alors, p est impaire et x_1 et x_p appartiennent tous deux à S_1 .

- Preuve du b) : Si μ est de longueur impaire, alors μ est pair et x_μ appartient à S_2 .

4)

Sait $\{x, y\}$ une arête monocolore au terme du pancrus. L'arête $\{x, y\}$ est une chaîne de longueur 1, donc impaire, donc x et y appartiennent l'un à S_1 , l'autre à S_2 (cf. 2.3.a/b). Si maintenant on considère la chaîne qui relie x et y dans $F(L)$, elle de longueur pair car ses 2 extrémités sont du même couleur et que chaque arête est bicolorée. Donc x et y devraient appartenir tous les 2 à S_1 ou tous les 2 à S_2 .

Les 2 assertions sont impossibles à satisfaire simultanément, donc le graphe n'est pas bipartit.

5)

$$([x] = c)$$

Si $c = \text{rouge}$ alors

Pancrus-Rec(G, r , bleu);

Sinon

Pancrus-Rec(G, r , rouge);

Exercice 3

1)

Sait S^* un stable maximum, v une feuille et u son voisin.

Plusieurs cas :

- Si $v \in S^*$ alors S^* n'est pas un stable max qui $\in v$.
- $v \notin S^*$ et $u \notin S^*$: Impossible car $S^* \cup \{v\}$ serait stable de cardinal \geq card(S^*) qui était supposé le card max.

- o $v \notin S^*$ et $u \in S^*$ alors $S^* \cup \{v\} \setminus \{u\}$ est un stable du même card(S^*) donc un cardinal maximum.

2)

En examinant par ordre croissant du numéro de portée dans l'algorithme glouton, on s'assure qu'à chaque fois qu'on examine un sommet, soit c'est une feuille du sous graphe induit par les sommets non-examinés, soit c'est un sommet intérieur.

4)

Complexité :

- o Parcours en profondeur : $O(m+m) \rightarrow O(m)$ car $m = m - 1$
- o Algo glouton : $O(nt_m) \rightarrow O(m)$
 $\rightarrow O(n)$.