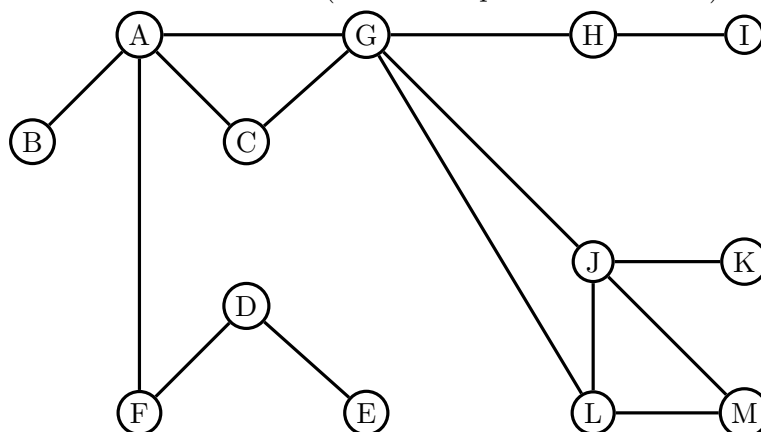


TD 2 - Parcours de graphes et applications.

Dans toute cette feuille de TD, on ne considère que des graphes **simples** non orientés.

Exercice 1.

- (a) Retrouvez dans votre cours les algorithmes réalisant respectivement l'exploration en profondeur et en largeur d'un graphe à partir d'un sommet donné.
- (b) Déroulez ces deux algorithmes sur le graphe suivant, en partant de A , puis en partant de M . Donnez les arbres correspondant aux différents parcours, et numérotez les sommets dans l'ordre où ils ont été découverts (sans les étapes intermédiaires).



- (c) Les algorithmes vus au cours renvoient une liste de sommets; modifiez-les afin qu'ils renvoient chacun l'arbre correspondant au parcours. Pour cette question, on supposera pour simplifier que le graphe à explorer est connexe.
- (d) Donnez un exemple de graphe G et un sommet v tels que l'exploration en largeur de G à partir de v ne parcourt pas tous les sommets de G .

Exercice 2.

- (a) Donnez un algorithme qui prend en entrée un graphe G et produit une forêt obtenue en parcourant le graphe avec une stratégie de parcours en largeur. (*Attention, on ne donne pas de point de départ pour ce parcours*)
- (b) Donnez un algorithme qui prend en entrée un graphe G et produit une forêt obtenue en parcourant le graphe avec une stratégie de parcours en profondeur.

Exercice 3.

Étant donné un graphe G et un sommet v de G , on note :

- $T_{L,v}$ l'arbre résultant de l'exploration en largeur de G à partir de v ;
- $T_{P,v}$ l'arbre résultant de l'exploration en profondeur de G à partir de v .

- (a) Soit G le graphe complet à 4 sommets numérotés de 0 à 3. Construisez $T_{P,0}$ et $T_{L,0}$.
- (b) Plus généralement, on appelle K_n le graphe complet à n sommets; définissez $T_{P,0}^n$ et $T_{L,0}^n$.
- (c) On veut démontrer que la propriété suivante est vraie pour tout graphe simple connexe :
 G est un arbre $\Leftrightarrow G$ est connexe et pour tout sommet v de G on a $T_{L,v} = T_{P,v}$.
 1. (\Rightarrow) Justifiez que si G est un arbre, alors pour tout sommet v de G on a $T_{L,v} = T_{P,v}$.
 2. (\Leftarrow) On va raisonner par contraposée. Reformulez la propriété à démontrer.
 3. Finissez la démonstration.

Exercice 4.

Reprenons l'algorithme d'exploration en profondeur d'un graphe dans sa version itérative.

- (a) Comment pourrait-on tester si un sommet a déjà été visité sans utiliser la structure `déjà_visités` ?
- (b) En utilisant cette idée, écrivez une fonction `parcours_profondeur_simple` qui effectue l'exploration en profondeur d'un graphe à partir d'un sommet sans construire la structure `déjà_visités`.
- (c) Quelle sera la complexité de l'algorithme résultant, en supposant qu'on l'exécute sur une liste d'adjacence ?
- (d) Quelle sera la complexité de l'algorithme résultant, en supposant qu'on l'exécute sur une matrice d'adjacence ?
- (e) Votre binôme de projet a pris l'initiative de "simplifier" l'algorithme d'exploration en profondeur en vous proposant le pseudocode ci-dessous. Ce pseudocode est-il correct ? Déroulez-le sur le graphe complet à quatre sommets. Expliquez où se trouve le problème.

Algorithme : PARCOURS_PROFONDEUR_DOUTEUX(G , départ)

```

1 résultat ← pile();
2 déjà_visités ← tableau( $G$ .nombre_sommets(), FAUX);
3 a_traiter ← pile();
4 a_traiter.empiler(départ);
5 tant que  $a\_traiter.pas\_vide()$  faire
6   | sommet = a_traiter.dépiler();
7   | résultat.empiler(sommet);
8   | déjà_visités[sommet] = VRAI;
9   | pour chaque  $v$  dans renverser( $G$ .voisins(sommet)) faire
10  |   | si  $\neg$  déjà_visités[ $v$ ] alors a_traiter.empiler(voisin) ;
11 renvoyer résultat;
```

Exercice 5.

Un étudiant a essayé d'implémenter les algorithmes de parcours en largeur, avec et sans reconstruction de l'arbre de parcours. Malheureusement, il a oublié de suivre la convention consistant à explorer les sommets dans l'ordre lexicographique. On veut pouvoir vérifier si ses résultats sont néanmoins corrects.

- (a) Écrivez un algorithme qui, étant donné un graphe connexe G , un arbre couvrant T pour G et un sommet source s , renvoie VRAI si T est un arbre de parcours en largeur pour G à partir de s et FAUX sinon. Seul l'arbre T est donné ; il n'est pas nécessaire de savoir dans quel ordre l'étudiant a numéroté ses sommets lors de son parcours.
- (b) Écrivez un algorithme qui, étant donné un graphe connexe G et un ordre L sur ses sommets, renvoie VRAI si L ordonne les sommets de G selon un parcours en largeur, et FAUX sinon.