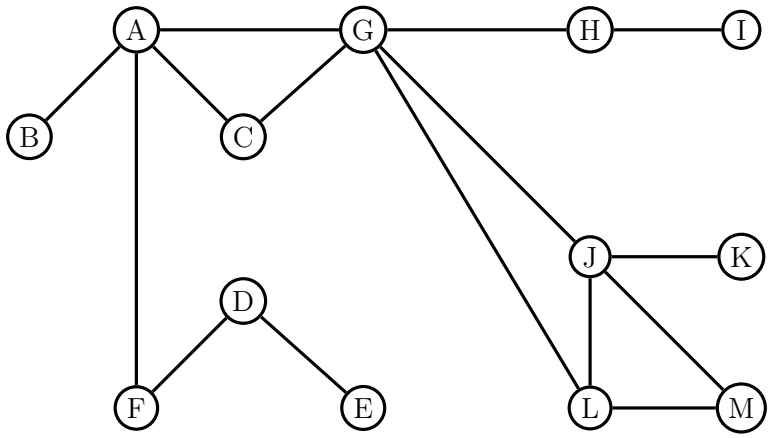
TD 2 - Parcours de graphes et applications.

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

Exercice 1.

1. Retrouvez dans votre cours les algorithmes réalisant respectivement l'exploration en profondeur et en largeur d'un graphe à partir d'un sommet donné.
2. 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).



3. Les algorithmes vus en 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.
4. 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.

1. 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)*
2. 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 donnés 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.

1. Soit G le graphe complet à 4 sommets numérotés de 0 à 3. Construisez $T_{P,0}$ et $T_{L,0}$.
2. Plus généralement, on appelle K_n le graphe complet à n sommets ; définissez $T^n_{P,0}$ et $T^n_{L,0}$.
3. On veut démontrer que la propriété suivante est vraie pour tout graphe simple connexe :

$$G \text{ est un arbre} \Leftrightarrow G \text{ est connexe et pour tout sommet } v \text{ de } G \text{ on a } T_{L,v} = T_{P,v}.$$

   (a) ($\Rightarrow$) Justifiez que si G est un arbre, alors pour tout sommet v de G on a $T_{L,v} = T_{P,v}$.

   (b) ($\Leftarrow$) On va raisonner par contraposée. Reformulez la propriété à démontrer.

(c) Finissez la démonstration.

Exercice 4.

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

1. Comment pourrait-on tester si un sommet a déjà été visité sans utiliser la structure `déjà_visités` ?
2. 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`.
3. Quelle sera la complexité de l'algorithme résultant, en supposant qu'on l'exécute sur une liste d'adjacence ?
4. Quelle sera la complexité de l'algorithme résultant, en supposant qu'on l'exécute sur une matrice d'adjacence ?
5. 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 : PARCOURSPROFONDEURDOUTEUX(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 ∈ renverser(G.voisins(sommet)) faire
10 |  |  si ¬ 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.

1. 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.
2. 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.