Diplôme Inter-Universitaire
ENSEIGNER L'INFORMATIQUE AU LYCÉE

Bloc 5 : TP 3

Parcours de graphes

Année 2019-20

1 Parcours en profondeur

On se donne l'implémentation récursive suivante du parcours en profondeur à partir d'un sommet s.

```
def parcours_profondeur(G, s) :
    """ Renvoie la liste du parcours en profondeur du graphe G
    représenté par sa liste d'adjacence (dictionnaire),
    à partir du sommet s : exploration restreinte à la partie
    accessible à partir de s """
    sommets_visites = []
    def parcours_sommet(u):
        if u not in sommets_visites:
            sommets_visites.append(u)
            for x in G[u]:
                parcours_sommet(x)
    parcours_sommet(s)
    return sommets_visites
```

Question 1 Appliquer l'algorithme à un graphe non connexe (tel qu'il existe deux sommets non reliés par un chemin), à partir de différents sommets, pour vérifier les listes de parcours renvoyées.

Question 2 Justifier qu'à tout moment de l'exécution de l'algorithme de parcours en profondeur, la pile d'appels des arguments de la fonction `parcours_sommet` forme un chemin de s au sommet x, le dernier sommet sur lequel on a appelé la fonction récursivement.

Question 3 Quelle est la complexité dans le pire des cas de l'algorithme de parcours en profondeur précédent ?

Question 4 Comment modifier l'algorithme afin qu'il ait une complexité linéaire en le nombre d'arcs et le nombre de sommets du graphe ?

Question 5 Comment modifier à nouveau le programme afin qu'il parcourt la totalité du graphe, même si celui-ci est non connexe ? En particulier, on cherche à écrire une nouvelle fonction `parcours_profondeur` qui renvoie une liste de sommets L qui vérifie les propriétés suivantes :

- chaque sommet de S apparaît une fois et une seule dans L ;
- chaque sommet de la liste (sauf le premier) appartient à la frontière du sous-ensemble de sommets placés avant lui dans la liste, si toutefois cette frontière est non vide. (On rappelle que la frontière d'un ensemble de sommets $T \subseteq S$ est l'ensemble des sommets $s \in S \setminus T$ qui sont adjacents à au moins un sommet de T)

L'algorithme ainsi modifié porte souvent le nom de *parcours itéré*...

2 Tri topologique

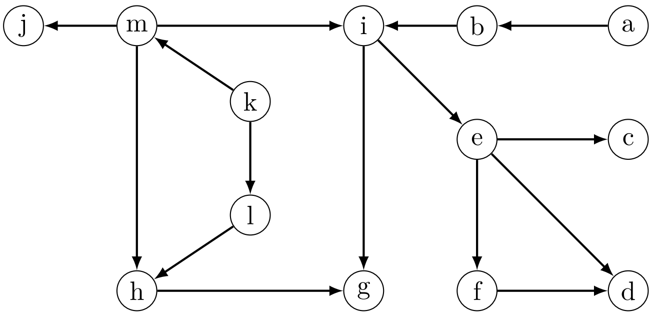
Étant donné un graphe orienté acyclique G (c'est-à-dire qui ne possède pas de cycle), on veut trouver un ordre linéaire total des sommets de G (autrement dit, on veut numéroter les sommets de 1 à n, ce qui définit une relation d'ordre total, $u < v$ si l'indice de u est plus petit que l'indice de v), tel que cet ordre vérifie la propriété :

$$\text{pour tout arc } (u, v) \text{ de } G,\ u < v$$

Ce tri permet d'ordonnancer des tâches à effectuer en présence de contrainte de précédence, du type la tâche A doit être faite avant la tâche B. C'est typiquement le cas lors de la compilation de projets avec plusieurs fichiers qui dépendent les uns des autres, `make` par exemple utilise un tri topologique pour décider de l'ordre de compilation.

Question 6 Montrer que si le graphe comporte un cycle, un tel ordre n'existe pas. Ceci explique pourquoi une compilation peut échouer avec un message d'erreur contenant les termes *dépendance cyclique*. Ce message indique la présence d'un cycle dans le graphe de dépendance.

Question 7 Ordonner le graphe suivant.



Question 8 Expliquer pourquoi un graphe G possède un cycle si et seulement si à un moment de l'exécution de l'algorithme de parcours itéré en profondeur, le chemin d'appel possède un cycle.

Question 9 En déduire un algorithme produisant un tri topologique s'il en existe un.

Question 10 Quelle est la complexité de cet algorithme ?

Question 11 Que peut-on dire de l'ordre topologique, par rapport à l'arborescence de parcours en profondeur ?

Question 12 Soit C le graphe orienté de sommets $\{a, b, c, d, e\}$ et d'arcs $\{(a, b), (a, c), (b, e)), (c, d), (d, e)\}$. Enumérer tous les tris topologiques de ce graphe.

3 Parcours en largeur

On cherche à implémenter un parcours en largeur d'un graphe. Pour cela, on se rappelle qu'il s'agit d'un parcours générique dans lequel on choisit le sommet de la bordure à l'aide d'une file.

Question 13 Utiliser une classe implémentant une file pour écrire une fonction Python réalisant un parcours en largeur d'un graphe à partir d'un de ses sommets, en se limitant aux sommets accessibles depuis ce sommet initial.

Question 14 Sachant qu'un parcours en profondeur peut être implémenté à partir du parcours générique en choisissant le sommet de la frontière à l'aide d'une pile, en déduire une implémentation non récursive du parcours en profondeur d'un graphe à partir d'un sommet.