Mini épreuve de Graphes
30 juin 2021 - DIU Bloc 5 V2
Durée 40 Min

Épreuve proposée par Laure Gonnord

D'après l'écrit de CAPES 2021, à réaliser sur feuille

SOLUTION. Des éléments de correction sont donnés dans les cadres bleus.

Sujet papier, vous pouvez répondre dans un fichier texte ou un document traitement de texte **exporté en pdf** *ou encore sur véritable papier scanné* **en pdf** *ou encore pris en photo avec votre téléphone et* **exporté en pdf** *(par exemple avec l'appli scancam)*

À moins de les redéfinir explicitement, l'utilisation des fonctions autres que `len`, `append` et les accès aux éléments avec `li[...]` sont interdites. On rappelle qu'une fonction qui s'arrête sans avoir rencontré l'instruction `return` renvoie `None`.

L'objectif ici de déterminer les composantes connexes d'un graphe $G = (V, E)$ avec n sommets et m arêtes. On supposera que l'ensemble V est constitué de sommets numérotés par des entiers consécutifs commençant à 0, c'est-à-dire que $V = \{0, 1, ..., n-1\}$.

On vous fournit un graphe sous la forme d'une liste de tuples à deux valeurs qui décrivent les arêtes. On souhaite avoir une représentation par listes d'adjacence, ie une liste de n sous listes (attention, ce n'est pas la représentation par dictionnaire donnée dans le cours)

On illustre ces différentes représentations avec le graphe G de la figure 1 (à gauche).

Le graphe est donné par `g_ex_a = [(0, 1), (0, 2), (0, 3), (1, 0), (1, 4), (1, 8),(2, 0), (2, 3), (3, 0), (3, 2), (3, 6),(4, 1), (5, 6), (6, 3), (6, 5),(7, 9), (8, 1), (9, 7)]`

On veut la représentation : `g_ex_b = [ [1, 2, 3], [0, 4, 8], [0, 3],[0, 2, 6], [1], [6], [3, 5] [9], [1], [7]]`

1. Écrire une fonction `adjacences(n, li)` qui prend en argument , le nombre de sommets du graphes et `li` une liste de couples correspondant à un ensemble E (comme par exemple `g_ex_a` dans un ordre quelconque, et renvoyant la représentation du graphe $G = (V, E)$ sous forme de listes d'adjacences (comme par exemple `g_ex_b`).

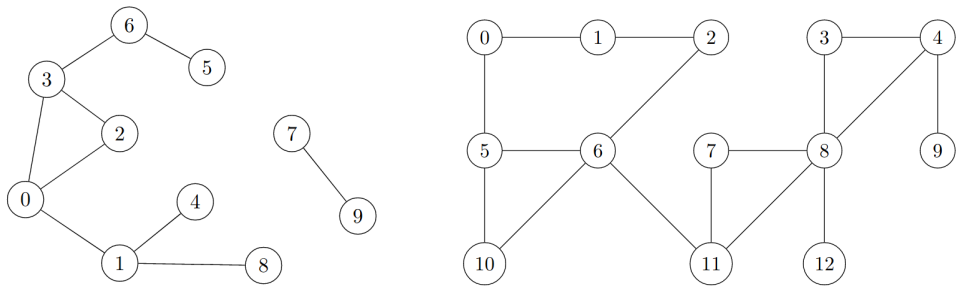


FIGURE 1 – À gauche, le graphe G_{ex}, à droite le graphe G'_{ex}

SOLUTION. Il s'agit d'ajouter un à un les sommets du graphe en les mettant au bon indice dans la liste résultat :

```
def adjacences(n, li):
    """ transforme une liste d'arêtes en listes d'adjacences"""
    res = [[] for i in range(n)]
    for (v1, v2) in li:
            res[v1].append(v2) # ajoute la nouvelle arête
    return res
```

Notes. Un unique parcours suffit. Les fonctions quadratiques sont refusées (erreur à comprendre, trop souvent faite).

On se donne le programme de la figure 2 (attention, la fonction `parcours` comporte une sous-fonction `explorer`, et les dernières 6 lignes font bien partie de la fonction `parcours`).

2 Quel est le type de la valeur renvoyée par la fonction parcours ? Appliquer à la main cette fonction sur le graphe exemple G_{ex}. Quel est le nom de ce parcours ?

SOLUTION. Le type renvoyé par la fonction explorer est un arbre, donc le type de la valeur renvoyée par la fonction `parcours` est une liste d'arbres. Il s'agit d'un parcours récursif (en profondeur) à partir du noeud 0.

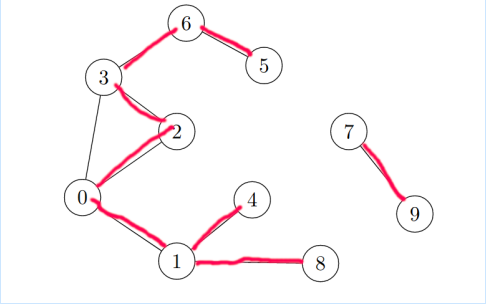
Sur le graphe exemple, l'appel `parcours (g_ex_b)` réalise les opérations suivantes :

- n reçoit 10 (taille de la liste d'adjacences)
- liste résultat initialisée à 0.
- Pour $i = 0$:
  - Comme i n'est pas déjà vu, il est noté vu, et on fait un appel à explorer(0) :
    - On construit un arbre enraciné à 0.
    - On récupère les voisins de i, c'est à dire $1, 2, 3$
    - On marque 1 comme déjà vu et on explore à partir de 1, ...
    - (il se passe une exploration à partir de 1)
    - quand l'exploration à partir de 1 est finie, cela donne un arbre de parcours de racine 1 qui est ajouté à l'arbre de racine 0 précédent. (les noeuds 4 et 8 sont parcourus)
    - on continue...

```
    class Arbre():
    """ Classe Arbre du sujet CAPES NSI -1 -2021 """

    def __init__(self,sommet):
        self.sommet = sommet
        self.children = []

    def add_child(self,child):
        self.children.append(child)

def parcours(listes_adjacences):
    n = len(listes_adjacences)
    deja_vu = [False] *n

    def explorer(i):
        """ fonction auxiliaire qui explore àpartir de i"""
        arbre = Arbre(i)
        voisins = listes_adjacences[i]
        for s in voisins:
            if not deja_vu[s]:
                deja_vu[s] = True
                arbre.add_child(explorer(s))
                return arbre

    res=[]
    for i in range(n):
        if not deja_vu[i]:
            deja_vu[i] = True
            res.append(explorer(i))
    return res
```

FIGURE 2 – Programme donné

On trouve alors l'arbre (couvrant) dessiné en rouge ci-dessous :



3 Rappeler la/une définition de connexité d'un graphe.

SOLUTION. Un graphe est connexe s'il ne comporte qu'une seule composante connexe. Une composante connexe d'un graphe est un ensemble maximal de noeuds qui forme un graphe connexe (ie toute paire de noeuds est connectée par un chemin).

4 Écrire une fonction `composantes_connexes(p_graphe)` prenant en argument le graphe obtenu avec la fonction parcours et renvoie les composantes connexes sous forme de liste de listes de sommets.

SOLUTION. Il s'agit de reparcourir la liste d'arbres pour traquer les sommets de ces arbres, il aurait mieux fallu stocker au fur et à mesure dans des listes. Les solutions à base de parcours d'arbre sont évidemment ok.

On suppose maintenant que cette partie que G est un graphe connexe à n sommets. Si $\forall i \in [0;k], x_i \in V$, on appelle chaine une suite finie $(x_0, x_1, ..., x_k)$ telle que pour tout i, on ait $(x_i, x_{i+1}) \in E$. Cette chaine est un cycle lorsque $x_0 = x_k$, et c'est de plus un cycle élémentaire si tous les sommets $x_0, ..., x_{k-1}$ sont distincts deux à deux. On dit que $G = (V, E)$ est biconnexe lorsque :

— $n = |V| = 1$;
— ou $n = |V| = 2, V = a, b$ et $(a, b) \in E$
— ou $n = |V| \geq 3$et pour toute paire (x, y) de sommets, il existe un cycle élémentaire contenant x et y.

Notes. Attention, la définition n'est PAS équivalente à "il existe un cycle reliant tous les sommets".

5 Montrer qu'un graphe biconnexe est également connexe.

SOLUTION. pour n=1 ou 2, le graphe est connexe. Pour $n = 3$, pour toute paire de sommet il existe un cycle élémentaire contenant x et y. On déduit de ce cycle un chemin entre x et y et donc le graphe est connexe.

6 Donner un exemple de graphe connexe mais pas biconnexe.

SOLUTION. $G'ex$ est connexe mais pas biconnexe car il n'existe pas de cycle élémentaire entre 7 et 10 : en effet, un tel cycle passe forcément par 6 et 11, et il n'existe qu'une unique arête entre 6 et 11,...

On dit qu'un sommet x de G est un point d'articulation lorsque le graphe G privé du sommet x (et des arêtes issues de x) n'est plus connexe.

7 Donner les points d'articulation de G'_{ex} (graphe de droite de la figure 1).

SOLUTION. 4, 6, 8 et 11 sont les points d'articulations. ...

8 Soit $G = (V, E)$ possédant un point d'articulation. Montrer que G n'est pas biconnexe.

SOLUTION. Prenons un point d'articulation x. Par définition $G' = G \setminus \{x\}$ n'est plus connexe. Il existe donc y, z deux points déconnectés dans G'. Observons G. Un cycle comportant x, y passe forcément par x deux fois, donc il ne peut être élémentaire, donc G n'est pas biconnexe.

9 Expliquer comment on peut déterminer si un sommet particulier x est un point d'articulation à l'aide d'un parcours en profondeur.

SOLUTION. Lors d'un parcours en profondeur, on peut distinguer les descendants et les ancêtres de x. La question est alors de savoir s'il existe une chaîne entre ceux-ci dans $G \setminus \{x\}$. Dans le parcours en profondeur, cela signifie que si y est plus bas que x (ie avec une numerotation > à celle de x dans le parcours) alors il n'existe pas d'arc retour au dessus de x (ie pas d'ancêtre de x avec une numérotation plus faible). Faire un dessin :-)