TD 2

Introduction aux parcours

1 Parcours en profondeur

Une des premières choses qu'on veut faire sur les graphes est de les parcourir, c'est-à-dire de passer par tous les sommets en suivant les arêtes. Ceci peut servir à savoir s'il existe un chemin d'un sommet u à un autre sommet v, ou encore à afficher le graphe, etc.

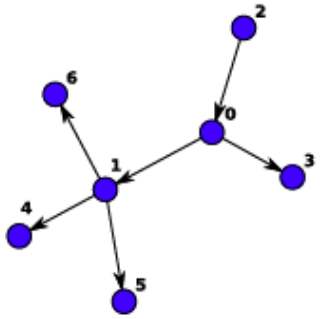
Commençons par un algorithme de parcours simplifié :

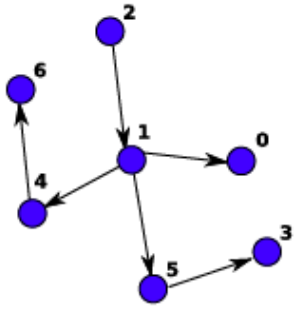
```
# Note : si pile est une liste, pile.pop() renvoie le
# dernier element de pile et l'en retire. Par exemple , si
# pile=[4,3,2,1] ; alors apres l'instruction x=pile.pop(), x
# vaut 1 et pile vaut [4,3,2]

def parcours(graphe,depart):
    pile=[depart]
    while(len(pile)>0):
        noeud_courant=pile.pop()
        print noeud_courant
        for i in graphe[noeud_courant]:
                pile.append(i)
```

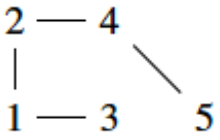
Exercice 1.

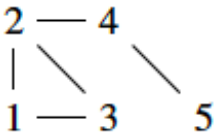
1. Que va afficher ce parcours pour les graphes suivants, en supposant qu'on a choisi le sommet 2 comme départ ?





2. Donnez le résultat d'un parcours en profondeur sur les graphes suivants. On prendra le sommet 1 comme sommet de départ :





2.1. Quel problèmes observe-t-on ?

2.2. Comment adapter l'algorithme précédent pour qu'il fonctionne sur les graphes ?

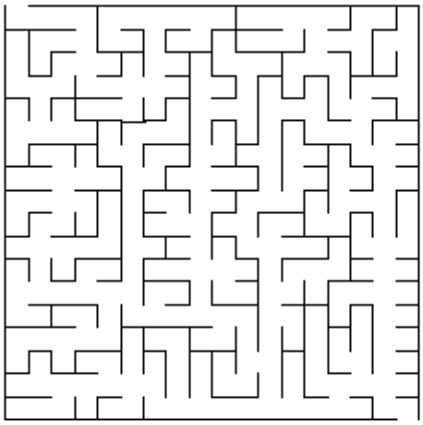
2.3. Proposez un programme python qui implémente le parcours en profondeur d'un graphe. Aidez vous des lignes python données dans le désordre et non indentées ci-dessous.

```
def parcours_profondeur(g,depart):
while(*******):
pile=[depart]
pile.append(i)
noeud_courant=pile.pop()
print noeud_courant
for i in g[noeud_courant]:
atteint={depart}
atteint.add(i)
if(i not in atteint):
```

2 Applications du parcours en profondeur

Exercice 2. Accessibilité

1. Sur le labyrinthe suivant, dessinez la trace que laisserait un personnage qui ferait une exploration en utilisant le parcours en profondeur en partant de l'entrée en haut à gauche.



2. Écrivez une fonction qui prend en entrée un graphe et deux sommets et renvoie vrai si on peut aller d'un sommet à l'autre et faux sinon.

3. Écrivez une fonction qui prend en entrée un graphe et un sommet du graphe et renvoie le nombre de sommets accessibles à partir de ce sommet.

Exercice 3. Cycles

1. Dans le labyrinthe ci-dessus, est-il possible de "tourner en rond" ?

2. En exécutant un parcours en profondeur à partir de l'entrée en bas à droite, observez qu'on peut se rendre compte qu'il existe un cycle dans le labyrinthe.

3. Écrivez une fonction qui prend en entrée un graphe et un sommet d'entrée, et renvoie vrai s'il existe un *cycle* dans la *composante connexe contenant ce sommet*. Pour cela, partez de l'algorithme de parcours en profondeur et rajoutez les instructions nécessaires.

Exercice 4. Composantes connexes

1. Écrivez une fonction qui prend en entrée un graphe et pour chaque sommet, effectue un parcours en profondeur du graphe à partir de ce sommet, s'il n'a pas déjà été visité auparavant.

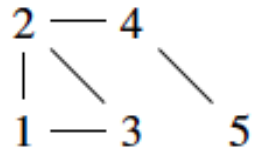
2. Écrivez une fonction qui prend en entrée un graphe et renvoie le nombre de composantes connexes du graphe.

3. Écrivez une fonction qui prend en entrée un graphe et renvoie la taille de la plus grande composante connexe (celle qui contient le plus grand nombre de sommets).

Exercice 5. Chemin

Dans un graphe, un chemin d'un sommet x vers un sommet y est une liste de sommets dont le premier élément est x et dont le dernier est y et tel que si deux éléments peuvent se suivre dans cette liste seulement si existe une arête dans le graphe entre ces deux éléments.

1. Proposez deux exemples de chemins entre le sommet 5 et le sommet 1 dans le graphe suivant :



2. On aimerait utiliser le parcours en profondeur pour générer un chemin entre le sommet 5 et le sommet 3 dans le graphe ci-dessus. Quel chemin cela donnerait-il ?

3. Modifiez l'algorithme de parcours en profondeur pour qu'à chaque sommet il associe un chemin depuis le point de départ jusqu'à ce sommet.