TD 2

Exercice 1

Question 1

Graphe 1

| Graphic 1 | |
|-----------|--------------|
| Pile | Nœud Courant |
| [2] | 2 |
| [0] | 0 |
| [1,3] | 3 |
| [1] | 1 |
| [4,5,6] | 6 |
| [4,5] | 5 |
| [4] | 4 |

Graphe 2

| Pile | Nœud Courant |
|---------|--------------|
| [2] | 2 |
| [1] | 1 |
| [0,4,5] | 5 |
| [0,4,3] | 3 |
| [0,4] | 4 |
| [0,6] | 6 |
| [0] | 0 |

Question 2

Graphe 1

| Pile | Nœud Courant |
|-------|--------------|
| [1] | 1 |
| [2,3] | 3 |
| [2,1] | 1 |
| [2,3] | 3 |
| [2,1] | 1 |

Ça boucle ...

Graphe 2

| Pile | Nœud Courant |
|---------|--------------|
| [1] | 1 |
| [2,3] | 3 |
| [2,2,1] | 1 |

Ça boucle ...

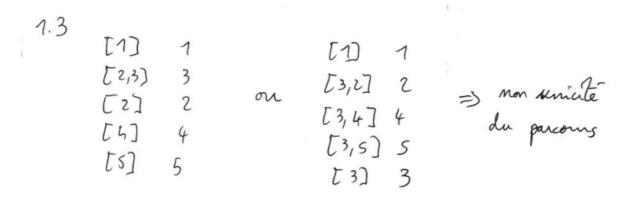
Question 2.1

On peut observer que l'on boucle au niveau des sommets car on a un graphe non orienté.

Question 2.2

Il faut modifier l'algo de sorte à ce que les graphes déjà visités ne se rajoute pas à la pile. Voir TD.py

Exécution du premier graphe de la question 2 avec l'algo parcours profondeur



Exercice 2

Question 2

```
def accessible (graphe, sommet1, sommet2) :
    return sommet2 in parcours_profondeur(graphe, sommet1)
```

Question 3

```
def nb_sommets_accessibles(graphe, sommet):
    return len(parcours_profondeur(graphe, sommet))
```

Exercice 3

Question 1

Pas si on part du début.

Question 2

```
Oui je l'ai vu
Question 3
def cycle (g, depart) :
    pile = [depart]
    atteint = {depart}
    while (len(pile) > 0):
        noeud_courant = pile.pop()
        print noeud_courant
        if (accessible(g, noeud_courant, depart)) :
            return True
        for i in g[noeud courant] :
            if (i not in atteint) :
                pile.append(i)
                atteint.add(i)
    return False
Exercice 4
Question 1
def parcours_prof_chaque_sommet (graphe):
    pile = graphe.nodes()
    atteint = set()
    while (len(pile) > 0):
        noeud_courant = pile.pop()
        if noeud_courant not in atteint :
```

atteint.add(i)

return atteint

```
Question 2
def nb_composante_connexe (graphe):
    pile = graphe.nodes()
    atteint = set()
    nb_comp = 0
    while (len(pile) > 0):
        noeud_courant = pile.pop()
        if noeud_courant not in atteint :
            nb_comp += 1
            for i in parcours_profondeur(graphe, noeud_courant) :
                 atteint.add(i)
    return nb_comp
```

for i in parcours_profondeur(graphe, noeud_courant) :

```
Question 3
```

Exercice 5

Question 1

1-2-4-5

1-3-2-4-5

Question 2

5-4-2-3

Question 3

```
def chemin (graphe, u, v):
    pile = [(u,[u])]
    atteint = {u}
    while pile :
        courant, chem = pile.pop()
        for vois in graphe[courant] :
            if not vois in atteint :
                atteint.add(vois)
                pile.append((vois, chem[vois]))
                if vois == v :
                     return chem+[vois]
```