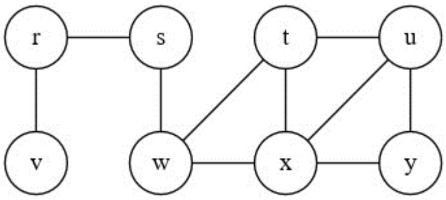
Exercices 04 : Graphes-2

Exercice 1. Parcours en largeur

On considère le graphe non orienté suivant :



dont on donne la représentation en machine suivante :

```
dAdj01 = {'r': ['s', 'v'], 's': ['r', 'w'], 't': ['u', 'w', 'x'],
          'u': ['t', 'x', 'y'], 'v': ['r'], 'w': ['s', 't', 'x'],
          'x': ['t', 'u', 'w', 'y'], 'y': ['u', 'x']}
```

On donne l'implémentation suivante du parcours en largeur

```
from collections import deque

def BFS(dA, s0):
    atteints = {s: False for s in dA.keys()}
    file = deque()
    atteints[s0] = True
    file.appendleft(s0)
    while len(file) > 0:
        s = file.pop()
        for succ in dA[s]:
            if not atteints[succ]:
                atteints[succ] = True
                file.appendleft(succ)
```

Q1. Déterminer pour le graphe en exemple, la distance de chaque sommet au sommet s, à l'aide d'un parcours en largeur.

Q2. Modifier le code de la fonction `BFS` afin qu'elle renvoie la liste des sommets accessibles depuis la source `s0`.

Q3. Modifier le code de la fonction `BFS` afin de définir trois « couleurs » pour les sommets :
- « blanc » pour les jamais encore découverts ;
- « gris » pour les sommets qui ont été découverts et mis dans la file ;
- « noir » pour les sommets dont l'exploration du voisinage est terminée.

On remplacera le dictionnaire `atteints` par un dictionnaire `couleurs`, afin de définir la couleur de chaque sommet. Les emplois qui sont fait du dictionnaire `atteints` seront remplacés par une utilisation du dictionnaire `couleurs`.

Tous les sommets accessibles verront ainsi leur couleur passer successivement de blanc à gris puis à noir.

Q4. Définir une fonction `BFS4(dA, s0)`, implémentant, en plus, un dictionnaire `distances` et un dictionnaire `predecesseurs`, ayant pour clés les noms des sommets, et pour valeurs associées, respectivement, la distance (en nombre d'arcs ou d'arêtes) entre s_0 et le sommet-clé, le long du chemin trouvé, et l'avant dernier sommet le long de ce chemin.

Pour le sommet source, `s0`, on donnera la valeur `None` à son prédécesseur (qui n'existe pas).

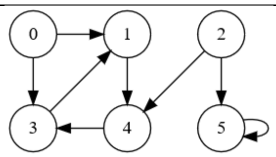
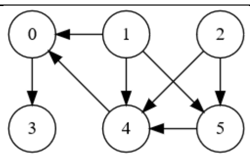
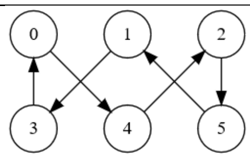
La fonction `BFS4` renverra, non plus la liste des sommets accessibles depuis `s0`, mais les deux dictionnaires, `distances` et `predecesseurs`.

Q5. Vérifier les résultats de la question **Q1**, par un appel à la fonction `BFS4`.

Exercice 2. Existence de circuits - 1

On considère ici des graphes orientés sans arcs multiples, représentés par leur matrice d'adjacence. Pour faciliter les démonstrations, les lignes et les colonnes des matrices seront numérotées à partir de zéro, de même que les sommets des graphes considérés, que l'on nommera indifféremment s_i ou i.

On considèrera à titre d'exemple les graphes suivants :

Graphe G_1	**Graphe G_2**	**Graphe G_3**
		

Un chemin de longueur k ($k \geq 1$) dans un graphe orienté dont les sommets sont $s_0, s_2, \dots, s_{n-1}$, est une suite de k sommets $s_{i_1}, s_{i_2}, \dots, s_{i_k}$ dont les sommets consécutifs sont reliés par des arcs.
On considère la matrice d'adjacence décrivant un tel graphe, notée $M = (m_{ij})_{0 \leq i,j \leq n-1}$, telle que $M_{i,j} = m_{ij}$ vaut 1 s'il existe un arc de s_i vers s_j et 0 sinon.

Q1. Démontrer par récurrence que, pour tout $p \geq 1$, M^p est telle que $M^p_{i,j}$ dénombre les chemins de longueur p reliant le sommet numéroté i au sommet numéroté j.

Q2. Écrire une fonction `puissance`, prenant en argument une matrice d'adjacence, M, et un entier $p \geq 1$ et renvoyant la puissance $p^{ème}$ de M.
Quelle est la complexité de l'algorithme utilisé en fonction de p et de n, l'ordre du graphe ?

Q3. Existe-t-il des circuits dans les graphes donnés en exemple ?