```
# -*- coding: utf-8 -*-

# ========================================================== #
# ========================================================== #
# Programme mettant en oeuvre l'algorithme de WELSH-POWELL. #
# DECEMBRE 2016                                              #
# ========================================================== #
# ========================================================== #

## IMPORTATIONS
# N'importons de numpy que ce dont nous avons réellement besoin !
from numpy import zeros

## CONSTRUCTION DE LA MATRICE D'ADJACENCE : FONCTION MatAdj
# Les arguments sont :
# (*) n : ordre du graphe
# (*) LA : liste des arêtes
def fMatAdj(n,LA):
    # n est le nombre de sommets du graphe
    # LA est une liste dont chaque élément est une liste comportant deux entiers
    # i et j si l'arête [S(i)S(j)] existe.
    Adj = zeros((n,n),dtype=int16)
    # Construction de la matrice d'adjacence Adj. La validité de la liste LA est
    # vérifiée au fur et à mesure.
    for k in range(len(LA)):
        i, j = LA[k][0], LA[k][1]
        if len(LA[k]) > 2 or i == j or Adj[i,j] != 0:
            return ('Liste non valide !')
        else:
            Adj[i,j], Adj[j,i] = 1, 1
    return Adj

## ALGORITHME DE WELSH-POWELL : FONCTION WP
# Arguments :
# (*) n  : ordre du graphe
# (*) LA : liste des arêtes
def WP(n,LA):
    # CONSTRUCTION DE LA MATRICE D'ADJACENCE
    M = fMatAdj(n,LA)

    # INITIALISATION DE LA LISTE DES DEGRES
    D = []
    # CONSTRUCTION DE LA LISTE DES DEGRES
    for i in range(n):
        d = 0
        # On balaie chaque ligne de la matrice d'adjacence
        for j in range(n):
            # Si un coefficient de la ligne est non nul, on incrémente d
            if M[i,j] != 0:
                d += 1
        D.append([i,d])

    # TRI DE LA LISTE DES DEGRES (DANS L'ORDRE DECROISSANT)
    D.sort(key=lambda degre: degre[1])
    D.reverse()

    # COLORATION
    # Initialisation de l'indice des couleurs
    C = 0
    # Initialisation du nombre de sommets coloriés
    ColoredVertices = 0
    # Boucle principale : on balaie D tant qu'il reste au moins un sommet à...
    # colorier !
    while ColoredVertices < len(D):
        for i in range(len(D)):
            # On ne s'intéresse qu'aux sommets non encore coloriés
            if len(D[i]) == 2:
```

```
                # Le sommet est potentiellement coloriable dans la couleur
                # courante
                ColPoss = True
                # Pour tous les sommets précédant le sommet courant dans la
                # liste D...
                for j in range(i):
                    # Si le sommet d'indice j<i dans D est déjà colorié avec la
                    # couleur C et adjacent au sommet d'indice i dans D alors
                    # le sommet d'indice i dans D ne peut être colorié avec C et
                    # on va passer au sommet suivant dans D sans rien faire.
                    if len(D[j]) == 3 and D[j][2] == C and M[D[i][0],D[j][0]] == 1:
                        ColPoss = False
                        break
                # Si on est dans une situation favorable, on colorie le sommet
                # d'indice i dans D avec la couleur courante.
                if ColPoss:
                    D[i].append(C)
                    ColoredVertices += 1
        # La liste D a été balayée, on passe à la couleur suivante...
        C +=1
    # Affichage des résultats
    print("Nombre de couleurs utilsées :",C)
    print("Sommets ayant la même couleur :")
    for i in range(C):
        s = "couleur " + str(i) + " : "
        for e in D:
            if e[2] == i:
                s += str(e[0]) + " "
        print(s)

## TESTS

# Définition du graphe par ses arêtes.
# 1ère situation : on travaille ici avec un petit graphe d'ordre 5.
# n = 5
# LA = [[0,2],[2,3],[3,1],[0,4],[2,4],[1,2],[1,4]]

# 2ème situation : le premier graphe de l'exercice N°3.
n = 6
LA = [[0,3],[0,5],[1,2],[1,4],[2,5],[3,4]]

# 3ème situation : le deuxième graphe de l'exercice N°3.
# n = 6
# LA = [[0,4],[0,5],[3,1],[3,2],[1,5],[4,2]]

# Coploration du graphe
WP(n,LA)
```