TD 32 : Cycles et circuits dans un graphe

Le parcours en profondeur permet de détecter la présence d'un **circuit** dans un graphe orienté ou d'un **cycle** dans un graphe non orienté. Un graphe non orienté est traité en lui associant un graphe orienté dans les 2 sens : on travaille donc sur un graphe orienté, à la recherche d'un circuit.
Le marquage d'un sommet en 3 couleurs est fondamental dans ce contexte : on colorie justement les sommets pour éviter de tourner en rond dans un parcours en profondeur.
On rappelle que tous les sommets sont initialement **blancs**, puis **gris** en cours d'exploration et enfin **noirs** quand tous leurs voisins ont été explorés. Dans un graphe orienté, si l'on découvre un sommet déjà découvert au cours du parcours en profondeur :

- soit il est gris, et on en déduit la présence d'un circuit
- soit il est noir, et on vient seulement de découvrir un second chemin menant du sommet initial à ce sommet, mais cela ne correspond pas à un circuit.

Principe de déctection de circuit par un parcours en profondeur

Les voisins d'un sommet sont explorés après son coloriage en gris, mais avant son coloriage en noir. S'il existe un circuit bouclant sur le sommet `s`, on rencontre à nouveau le sommet `s` alors qu'il est gris. Comme tous les sommets ne sont pas forcément visités dans un parcours en profondeur, on relance la détection de circuit sur les sommets restés blancs, tant qu'il en existe.

Exercice 1 (Détection de circuit par un parcours en profondeur)

On suppose le graphe orienté représenté par une liste d'adjacence (tableau de liste des voisins). On adapte le parcours en profondeur pour créer une fonction de détection de circuit, qui retourne simplement un booléen indiquant la présence ou non d'un circuit.

1. Donner des exemples des deux situations évoquées ci-dessus et vérifier la conclusion annoncée. Deux graphes d'ordre 4 suffisent pour cette question. Expliquer pourquoi l'algorithme précédent fonctionne sur ces graphes.
2. Rappeler l'algorithme de parcours en profondeur d'un graphe, en le simplifiant pour le problème étudié (on n'a pas besoin des distances, des prédécesseurs ni des dates) dans sa version récursive.
3. Ecrire une fonction `circuit(g)` qui retourne un booléen indiquant si un circuit est détecté. On peut soit utiliser une fonction locale récursive, soit séparer le code en deux fonctions dont l'une est récursive. L'intégralité du graphe doit être explorée.
4. Testez votre fonction avec un petit graphe tel qu'en ajoutant / enlevant un arc, le circuit apparaisse / disparaisse. Trouver un exemple où tous les sommets ne sont pas explorés lors du premier parcours.
5. Ecrire une autre fonction `circuit2(g)` en implémentant le parcours en profondeur de manière impérative, à l'aide d'une pile.