ISM MPSI/PCSI, Informatique
Année 2023/2024

TP Informatique 31

On complètera le fichier `TP31_GRAPHES1_std.py` en ligne sur le site de la classe.

Exercice 1

Le principe du *parcours en profondeur* pour un graphe orienté ou non orienté est le suivant :
- on part d'un sommet donné que l'on considère comme « en cours de visite » ;
- s'il admet des sommets adjacents « non visités » (des successeurs), on appelle récursivement le parcours en profondeur depuis chacun d'eux ;
- s'il n'y a pas de successeur, le sommet est considéré comme « visité ».

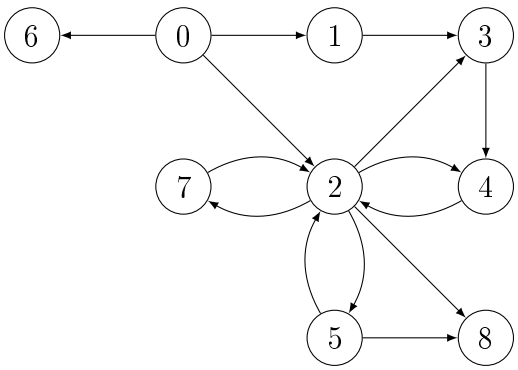
Les états d'un sommet seront décrits par les chaînes de caractères :
- `"blanc"` pour non visité ;
- `"gris"` pour en cours de visite ;
- `"noir"` pour visité.

On utilisera les dictionnaires suivants :
- `couleur` : pour `s` un sommet, `couleur[s]` est sa couleur ;
- `deb` : pour `s` un sommet, `deb[s]` est le temps de début de visite ;
- `fin` : pour `s` un sommet, `fin[s]` est le temps de fin de visite ;
- `predec` : pour `s` un sommet, `predec[s]` est son prédécesseur dans le parcours en profondeur.

On utilisera une liste `tps` initialisée à `[0]` pour mesurer le temps écoulé lors du parcours en profondeur.

Les listes et les dictionnaires sont mutables et sont adaptés pour la transmission d'arguments dans l'écriture récursive du parcours en profondeur.

1. Écrire une fonction `dfs_rec(S,A,sorg,tps,couleur,deb,fin,predec)` d'arguments `S` la liste des sommets, `A` la liste des arêtes, `sorg` le sommet d'origine et `tps`, `couleur`, `deb`, `fin`, `predec` les variables décrites préalablement et qui effectue le parcours en profondeur du graphe $G = (S, A)$ depuis le sommet `sorg`. La fonction `dfs_rec` ne renvoie rien mais modifie directement les variables mutables qui lui sont transmises en arguments.
2. Écrire une fonction `dfs_init(S,A,s0)` d'arguments `S` la liste des sommets, `A` la liste des arêtes, `s0` un sommet et qui renvoie les dictionnaires `deb`, `fin`, `predec` et `couleur` à l'issue du parcours en profondeur du graphe $G = (S, A)$ depuis le sommet s_0.
3. Tester la fonction `dfs_init` sur le graphe fourni en exemple depuis le sommet 4



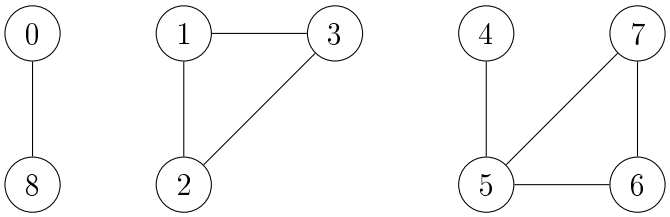
Exercice 2

Le parcours en profondeur d'un graphe non orienté depuis un sommet permet de parcourir toute la composante connexe de ce sommet. On peut donc déterminer le nombre de composantes connexes en suivant le principe suivant :

- on initialise un compteur ;
- Tant qu'il y a des sommets non visités :
  - on lance le parcours en profondeur depuis un sommet non visité ;
  - on incrémente le compteur ;

L'état final du compteur est le nombre de composantes connexes du graphe non orienté.

1. Écrire une fonction `nb_cc(S,A)` d'arguments `S` la liste des sommets, `A` le dictionnaire des arêtes et qui renvoie le nombre de composantes connexes du graphe non orienté $G = (S, A)$.
2. Tester la fonction sur le graphe fourni en exemple



Exercice 3

Lors d'un parcours en profondeur d'un graphe orienté, si l'on rencontre un sommet « en cours de visite » depuis un autre sommet lui-même « en cours de visite », alors il existe un circuit entre ces deux sommets.

1. Copier-coller les fonctions `dfs_rec`, `dfs_init` que l'on renommera respectivement en `circuit_rec`, `circuit` et les adapter afin que la fonction `circuit_deb(S,A,s0)` affiche les couples de sommets entre lesquels passent un circuit, couples identifiés lors du parcours en profondeur depuis le sommet s_0.
2. Tester la fonction `circuit_deb` sur le graphe de l'exercice 1 depuis un sommet au choix.
3. Adapter la fonction `circuit_deb` en une fonction `circuit(S,A)` qui affiche les couples de sommets de tous les circuits d'un graphe orienté.
4. Adapter la fonction `circuit` en une fonction `cycle(S,A)` qui affiche les couples de sommets de tous les cycles d'un graphe non orienté.