Institut National Universitaire Champollion

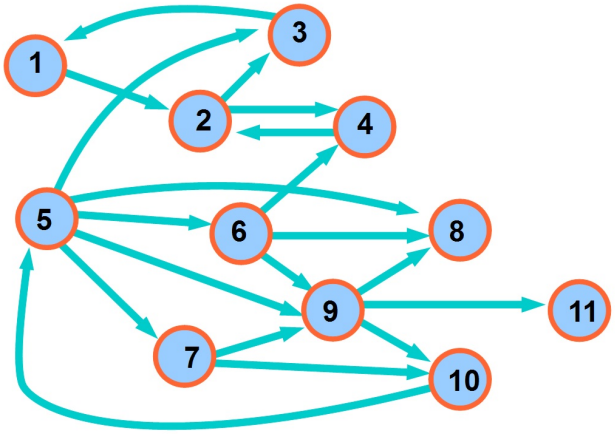
Contrôle de TP n°1

Théorie et Algorithmique des Graphes
L3 INFO - Semestre 6

1h30 - Aucun document n'est autorisé

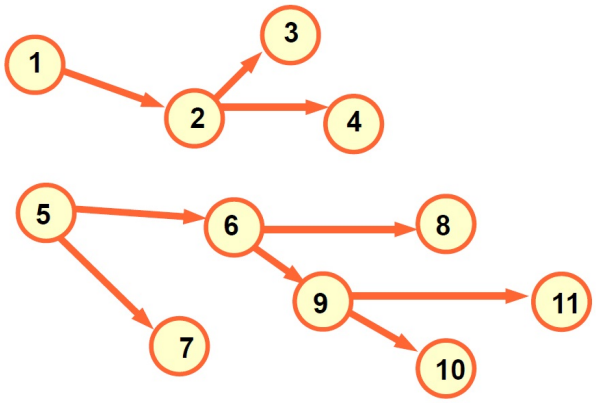
L'objectif de ce problème est d'adapter l'algorithme de parcours en profondeur d'un graphe simple orienté à la classification de ses arcs.

La première partie est consacrée à quelques fonctions élémentaires sur les arborescences, la deuxième traite du parcours en profondeur généralisé et la troisième à son adaptation à la classification des arcs. Les deux premières parties sont indépendantes mais il va de soi que la troisième partie s'appuie largement sur les deux premières. Toutes vos fonctions pourront être testés sur le graphe orienté G ci-dessous :



1. Sur les arborescences

On rappelle qu'on peut représenter une forêt (d'arborescences) par son tableau des pères `P`. Dans cette représentation une forêt à n sommets sera représentée par un vecteur `P` de taille $n+1$ tel que $P[i] = i$ si i est la racine d'une des arborescences de la forêt et $P[i]$ est le numéro du père du sommet i sinon.
Par exemple, la forêt :



est représenté par le tableau :

P = [0,1,1,2,2,5,5,5,6,6,9,9]

Dans toute la suite, P est le tableau des pères d'une forêt d'arborescence F.
Une forêt étant un graphe orienté particulier, nous pouvons aussi représenter une forêt par sa liste d'adjacences.

1. Écrire une fonction de conversion `tabPereToListe(P)` qui renvoie la représentation par liste d'adjacences de la forêt F (donnée par son tableau des pères P).
2. Écrire une fonction `ancetres(i,P)` qui renvoie la liste des ancêtres d'un sommet s de la forêt F. Par exemple :

   ```
   print("Les ancêtres du sommet 9 sont  : " , ancetres(9,P))
   print("Ceux du sommet 11 sont  : ", ancetres(11,P))
   print("Ceux de 1 sont  : ", ancetres(1,P))
   ```

   doit renvoyer

   ```
   Les ancêtres du sommet 9 sont  :  [6, 5]
   Ceux du sommet 11 sont  :  [9, 6, 5]
   Ceux de 1 sont  :  []
   ```

3. Écrire une fonction booléenne `ancetre(x,y,P)` valant vrai si x est un ancêtre du sommet y dans la forêt F.

2. Parcours en profondeur

G étant un graphe simple orienté représenté par ses listes d'adjacence, on souhaite écrire un parcours en profondeur généralisé qui renvoie la forêt de parcours sous forme d'un tableau des pères P.
Écrire les fonctions suivantes :

1. `profRec(G,i,Visite,P)` : fonction auxiliaire récursive qui provoque un parcours en profondeur du graphe à partir du sommet i. Cette fonction ne retourne aucun résultat et se contente de mettre à jour les paramètres `Visite` et `P`.
2. `profondG(G)` : effectue le parcours en profondeur généralisé à tout le graphe G, la fonction doit renvoyer le tableau des pères `P` représentant la forêt de parcours.

3. Application à la classification des arcs

On rappelle que le parcours en profondeur d'un graphe orienté peut conduire à quatre types d'arcs : les arcs couvrants, en avant, en arrière et transverses.

1. Adapter une copie des fonctions de parcours de la section précédente, pour qu'elles précisent les arcs couvrants.
2. Rappeler en commentaire comment le vecteur de visite peut être utilisé pour déterminer les arcs en arrière. Adapter une copie des fonctions précédentes pour qu'elles précisent les arcs couvrants et les arcs en arrières.
3. Comment peut-on, à l'aide des structures de données existantes et d'une fonction précédemment écrite, déterminer qu'un arc est en avant ? (répondre dans un commentaire).
4. Adapter une copie des fonctions précédentes pour qu'elles précisent la nature de tous les arcs du graphe.