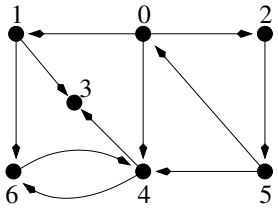
info602 : théorie des graphes et algorithmes sur les graphes

TD 3 : parcours de graphes, bis...

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22
téléphone : 04 79 75 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Exercice 1 : parcours en profondeur

On va utiliser le graphe suivant (le même que pour le TD 2)



Question 1 : utilisez l'algorithme décrit en cours pour faire un parcours en profondeur du graphe. Décomposez l'algorithme pour être sûr de comprendre ce qui se passe.
Pour se faire, utilisez les sommets dans l'ordre croissant 0, 1, 2, 3, 4, 5 et 6.

Question 2 : dessinez la forêt obtenue par l'algorithme (dans le tableau `pere[]`). Rajoutez les arcs de transitivité, de retours et les arcs couvrants.

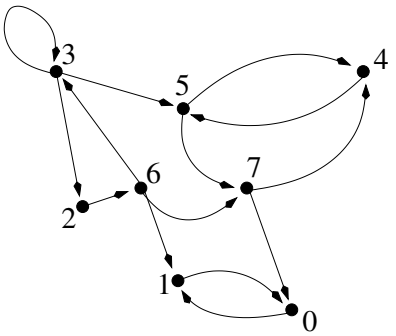
Question 3 : donnez une représentation temporelle des tableaux `debut[]` et `fin[]` créés par l'algorithme. Peut-on retouver la forêt en profondeur à partir de cette représentation ?

Question 4 : est-ce que cette forêt est entièrement déterminée par le graphe ? Par sa représentation ?

Question 5 : comptez *attentivement* le nombre d'opérations effectuées par l'algorithme sur un graphe avec n sommets et m arêtes. Donnez une approximation asymptotique de sa complexité en fonction de n et m.

Exercice 2 : recherche de composantes fortements connexes

On s'intéresse au graphe suivant :



Question 1 : en décomposant l'algorithme du cours ; calculez les composantes fortements connexes de ce graphe.

Question 2 : est ce que le résultat est déterminé par le graphe ? Par sa représentation ?

Exercice 3 : le tri topologique (application du parcours en profondeur)

Un ensemble de contraintes (ou de dépendances) peut être vu comme un graphe orienté sans circuit : une arête de x vers y représente la contrainte "y dépend x". La non présence de cycle indique qu'il n'y a pas de contraintes cycliques, impossibles à satisfaires séquentiellement.

On va regarder un algorithme qui permet de "linéariser" un tels graphe de contraintes : on veut trouver un ordre linéaire "$\prec$" sur les sommets qui vérifie la propriété suivante

$$\text{il y a un arc } (x, y) \text{ dans } G \quad \Rightarrow \quad x \prec y$$

Autrement dit ; si un évènement y dépend d'un évènement x, on veut mettre y après x. Ce genre d'algorithme est très utile quand on veut faire de l'ordonancement.

On considère l'algorithme suivant

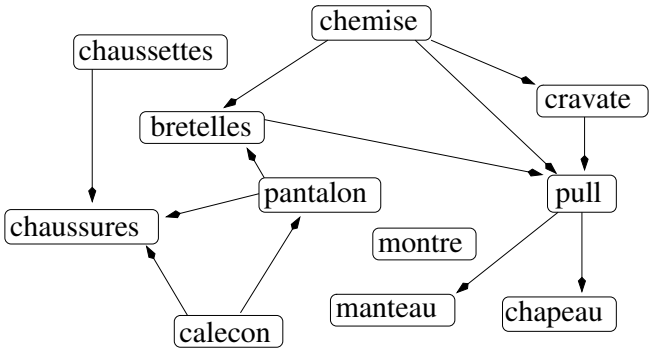
```
tri_topologique(G)
   liste := NULL                    -- une liste chaînée

   parcours_profondeur_bis(G)       -- comme parcours_profondeur[], mais
                                    -- chaque fois qu'un sommet devient
                                    -- "examiné", on le place en tête de
                                    -- "liste"

   renvoie(liste)
```

un algorithme pour l'exercice 3...

Question 1 : appliquez l'algorithme précédent sur le graphe



Question 2 : est que le résultat est déterminé par le graphe ? Par sa représentation ?

Question 3 : montrez la propriété suivante

Propriété. *Un graphe orienté est sans-circuit si et seulement si un parcours en profondeur ne génère aucun arc retour.*

Question 4 : en déduire que l'algorithme si dessus donne bien un tri topologique du graphe G.