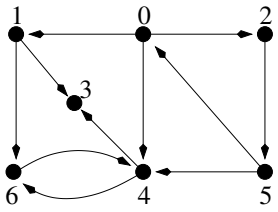
info505 : graphes et algorithmes
TD 2 : profondeur, algorithme de Prim...

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

Exercice 1 : parcours en profondeur

On va utiliser le graphe suivant



Question 1. Utilisez l'algorithme décrit en cours pour faire un parcours en profondeur du graphe.

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 retrouver la forêt en profondeur à partir de cette représentation ?

Question 4. 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 : 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 satisfaire 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'ordonnancement.

On considère l'algorithme suivant

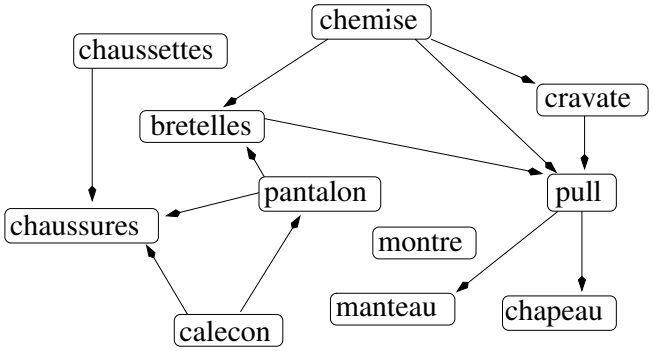
```
tri_topologique(G)
  liste := NULL                    -- une liste chaînée

  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...

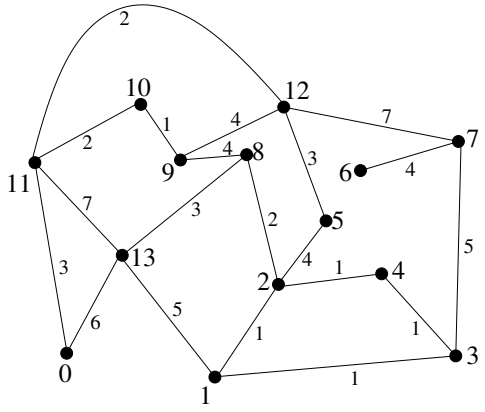
Question 1. Appliquez l'algorithme précédent sur le graphe



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

Exercice 3 : algorithme de Prim

Question 1. Appliquez l'algorithme de Prim sur le graphe suivant



Question 2. Un tas binaire est une structure de données en arbre binaire, où le père est toujours plus petit que ces deux fils. De plus, on impose que l'arbre soit bien balancé (les branches maximales sont de taille t ou $t+1$). Expliquez comment améliorer la complexité de l'algorithme du cours en utilisant des tas binaires.

À quelle complexité estimez-vous le nouvel algorithme ?

Exercice 4 : labyrinthes

Question 1. Essayez d'adapter l'algorithmes de Prim à la génération de labyrinthes sur des quadrillages.

Question 2. Quelle méthode proposez-vous pour résoudre un labyrinthe ?