info505 : graphes et algorithmes
TD 1 : représentations et parcours en largeur

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 0 : complexité

Question 1. À l'aide d'une calculatrice ou d'un ordinateur, remplissez le tableau suivant. La première colonne indique la complexité en microseconde d'un algorithme pour une entrée de taille n. Pour chaque colonne, estimez le temps d'exécution de cet algorithme pour une entrée de la taille donnée.

	5	10	50	100	1000	10000
n	5×10^{-6} s	10^{-5} s	5×10^{-5} s	10^{-4} s	10^{-3} s	10^{-2} s
$n\log(n)$						
n^2						
n^3						
n^5						
2^n						
n^n						
2^{2^n}						

Question 2. Quelles sont les classes de complexité "utilisables" ?

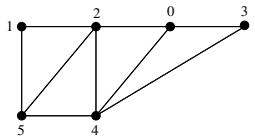
Si on estime que la loi de Moore est valide (la puissance de calcul double tous les deux ans), quelles sont les classes de complexité qui peuvent basculer du "infaisable" dans le "raisonnable" ?

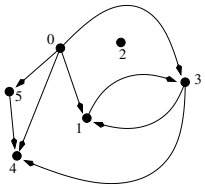
Exercice 1 : représentations des graphes

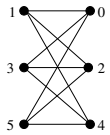
Question 1. Pour chacun des graphes suivants sur $S = \{0, 1, 2, 3, 4, 5\}$, donnez la représentation sous les différentes formes données en cours :

- sous-ensembles de $\mathcal{P}_2(S)$,
- relation
- liste d'adjacence,
- matrice d'adjacence.

Une arête non-orientée peut être vue comme deux arêtes orientées : une dans chaque direction...







Question 2. Donnez la représentation sous forme de matrice d'adjacence des graphes sur $\{0, 1, 2, 3, 4\}$ représentés par les listes d'adjacences suivantes :

$$\begin{bmatrix} - \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 1, 2, 3, 4 \\ 0, 2, 4 \\ 0, 1, 3 \\ 0, 2, 4 \\ 0, 1, 3 \end{bmatrix} \qquad \begin{bmatrix} 1, 2, 3, 4 \\ 0, 2, 3, 4 \\ 0, 1, 3, 4 \\ 0, 1, 2, 4 \\ 0, 1, 2, 3 \end{bmatrix}$$

Dessinez les graphes correspondants et comparez vos résultats...

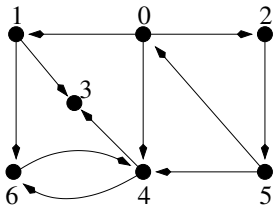
Question 3. Écrivez un algorithme qui permettra de passer de la représentation d'un graphe sous forme de liste d'adjacence à une représentation en matrice d'adjacence.

Faites un autre algorithme pour la transformation inverse.

Que se passe-t'il si on applique le premier algorithme puis le second ?

Exercice 2 : parcours en largeur

On va utiliser le graphe suivant



Question 1. Donner une représentation sous forme de liste d'adjacence de ce graphe.

Question 2. Utilisez l'algorithme décrit en cours pour faire un parcours en largeur du graphe. Décomposez l'algorithme pour être sûr de comprendre ce qui se passe.

Question 3. Dessinez la forêt obtenue par l'algorithme (dans le tableau `pere[]`).

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

Justifiez attentivement votre réponse.

Question 5. Ré-appliquez l'algorithme du cours en prenant, pour la première boucle ("`pour tous les sommets` u") les sommets dans l'ordre suivant : 3, 4, 5, 6, 0, 1, 2.

Dessinez la forêt correspondant à cette exécution. Qu'en pensez-vous ?

Question 6. Calculez, en utilisant l'algorithme du cours, le tableau des distances au sommet 2.

Exercice 3 : deux applications

Question 1. Le *diamètre* d'un graphe est le maximum des distances possibles entre deux sommets. Donnez un algorithme efficace permettant de calculer le diamètre d'un *arbre* non orienté et analysez sa complexité.

Question 2. Un graphe non orienté est *biparti* si on peut séparer ses sommets en deux sous-ensembles S_1 et S_2 qui ont la propriété suivante :

> *il n'y a aucune arête entre deux sommet de S_1 et il n'y a aucune arête entre deux sommets de S_2. (Autrement dit, toutes les arêtes sont incidentes à un sommet de S_1 et un sommet de S_2.)*

Décrivez un algorithme efficace pour tester si un graphe (non-orienté connexe) est biparti. Donnez la complexité de cet algorithme.