

# info404 : algorithmes sur les graphes

## TD 2 : parcours de graphes...

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

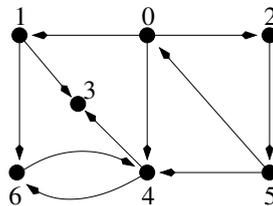
### Exercice 0 : classes de complexité

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 un graphe à  $n$  sommets. Pour chaque colonne, estimez le temps d'exécution de cet algorithme pour un graphe avec le nombre de sommets donné.

	5	10	20	40	100	500
$n$	$5 \times 10^{-6}$ s	$10^{-5}$ s	$2 \times 10^{-5}$ s	$4 \times 10^{-5}$ s	$10^{-4}$ s	$5 \times 10^{-4}$ s
$n^2$						
$n^3$						
$n^5$						
$2^n$						
$3^n$						

### Exercice 1 : 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 2 : le parcours en largeur, sans FIFO

on considère l'algorithme de la page suivante.

*Question 1* : appliquez cet algorithme sur le graphe de l'exercice précédent, en prenant les sommets dans l'ordre 6, 5, 4, 3, 2, 1, 0.

Essayez de comprendre les différences entre cet algorithme et celui donné en cours. Essayez d'expliquer pourquoi cet algorithme fonctionne correctement.

*Question 2* : à faire chez vous, implantez un de ces algorithmes en utilisant votre langage préféré. Testez-le...

```
pour tout sommet  $x$ 
faire
  etat[ $x$ ] := "non vu" ; pere[ $x$ ] := "NULL" ;
finfaire

 $i := 1$  ;  $j := 1$  ;
pour tout sommet  $x$ 
faire
  si etat[ $x$ ] == "non vu"
  alors
    a_traiter[ $j$ ] :=  $x$  ;  $j := j + 1$  ; etat[ $x$ ] := "vu" ;
    tant que  $i < j$ 
    faire
       $y := a\_traiter[i]$  ;  $i := i + 1$  ;
      pour tout  $z$  voisin de  $y$ 
      faire
        si etat[ $z$ ] == "non vu" ;
        alors
          etat[ $z$ ] := "vu" ; a_traiter[ $j$ ] :=  $z$  ;  $j := j + 1$  ; pere[ $z$ ] =  $y$  ;
        finsi
      finfaire
    finfaire
  finfaire
finsi
finfaire
```

un algorithme pour l'exercice 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.