Université de Paris X Nanterre — lundi 7 juin 2004
licence IUP MIAGE — de 14h à 16h

contrôle d'algorithmique de graphes

I graphe orienté

Appliquer la procédure de recherche de composantes fortement connexes vue en cours au graphe suivant, et donner l'ordre inverse de la fin de la visite lors du premier parcours en profondeur d'abord. Donner aussi les numéros de composantes fortement connexes attribués à chacun des sommets. On prendra les sommets dans l'ordre croissant pour le premier parcours et pour l'ensemble des prédécesseurs ou des successeurs d'un sommet.

Il y a 11 sommets numérotés de 0 à 10. (Les sommets 0 et 7 sont dessinés deux fois, mais ils n'existent qu'en un seul exemplaire.) Les 20 arcs sont 0→3 1→4 0→7 2→5 2→6 0→8 8→1 6→3 7→3 9→1 9→2 7→4 4→8 10→2 3→10 8→5 9→5 9→6 6→10 7→10

II arbres binaires de recherche

En partant d'un arbre vide dessiner les arbres obtenus en ajoutant un par un les 20 nombres 159 22 29 63 34 86 48 55 89 60 13 74 81 16 87 44 26 35 78 94 puis en les enlevant un par un dans le même ordre.

III graphe non orienté

Ecrire une fonction ayant pour prototype

```
int dist(int n,int x1,int y1,int x2,int y2);
```

qui rend le nombre minimal de déplacements de cavalier nécessaires pour aller de la case (x_1, y_1) à la case (x_2, y_2) sur un échiquier de n cases sur n. (Pour un échiquier normal $n = 8$). La fonction rendra -1 s'il n'est pas possible d'aller d'une case à l'autre, 0 si les deux cases sont identiques, 1 si $(x_1 - x_2)^2 + (y_1 - y_2)^2 = 5$ etc..

On pourra fabriquer un tableau représentant le graphe des déplacements élémentaires du cavalier, puis appeler une ou plusieurs fonctions vues en cours, ou réécrire en la modifiant une des fonctions vues en cours.

La fonction devra libérer proprement la place mémoire qu'elle utilise (par des `free` s'il y a des `malloc`).

Elle doit avoir un temps de calcul en $O(n^2)$.

Tous les documents sont autorisés.

Barème 7.333+7.8+6=21.133 points

I graphes

22 nombres $\times (1/3) = 7.333$ points.

II arbres binaires de recherche

insertion : 20 nombres $\times 0.2 = 4$ points.
délétion : 19 arbres $\times 0.2 = 3.8$ points.

III graphe non orienté 1+2+0.5+0.5+1+1=6 points

1pt. déclaration du tableau des arètes
- 0.5pt. rangement d'au moins une arète
- 1pt. rangement de plusieurs arètes
- 2pt. rangement de toutes les arètes

0.5pt nombre de sommets correct
0.5pt nombre d'arètes correct
1pt appel de `setdist` avec case de départ
1pt valeur de la fonction prise dans `dist[` case d'arrivée `]`