L2 Algorithmique des arbres - Session 2

Durée 2 heures.

Notes de cours, de TD et de TP autorisées.

- Les différentes parties (5 exercices) sont indépendantes et peuvent être traitées dans n'importe quel ordre.
- Chaque fonction sera écrite en C. On veillera à minimiser la complexité des fonctions.
- On pourra écrire des fonctions annexes, si l'utilisation de paramètres supplémentaires est nécessaire.

▸ **Exercice 1. Arbres binaires.** (6 points)
On utilise la structure :

```
typedef struct noeud{
    int val;
    struct noeud *fg, *fd;
} Noeud, *Arbre;
```

1. Écrire une fonction `int nb_fils_uniques(Arbre a)` qui renvoie le nombre de nœuds internes ayant un enfant unique.
2. Écrire une fonction `int hauteur(Arbre a)` qui renvoie la hauteur de l'arbre a.
3. Écrire une fonction `int egal(Arbre un, Arbre deux)` qui renvoie 1 si les deux arbres passés en paramètres sont égaux, 0 sinon.
4. On rappelle qu'un arbre est complet s'il est vide ou s'il vérifie :
   — toutes ses feuilles sont à la même hauteur ;
   — tous les nœuds internes ont deux enfants.

   Écrire une fonction `int complet(Arbre a)` qui renvoie 1 si l'arbre a est complet, 0 sinon.

▸ **Exercice 2. QuadTree** (6,5 points)
On souhaite représenter des images carrées, coloriées en 256 niveaux de gris, sous forme d'un arbre quaternaire (*quadTree* en anglais).

Une couleur (ou niveau de gris) c est représentée par un `unsigned char`, *i.e.* $0 \leq c < 256$. Une image est mémorisée par un tableau `unsigned char image[MAX][MAX]`, où `MAX` est une puissance de 2 prédéfinie par une directive `#define` (par exemple : `#define MAX 512`).

On utilisera donc la structure :

```
typedef struct Qneu{
    unsigned char couleur;
    struct Qneu *fils[4];
} Qnoeud, * Qarbre;
```

Un nœud d'un arbre quaternaire qui n'est pas une feuille contiendra comme couleur la moyenne des couleurs de ses quatre enfants. Les enfants seront toujours traités dans l'ordre suivant :

0	1
3	2

On suppose que l'on dispose des fonctions suivantes :

- `Qnoeud * alloue_noeud(unsigned char c)` qui renvoie l'adresse d'un `Qnoeud` de couleur `c` dont les quatre enfants sont vides ;
- `int est_monochrome(unsigned char image[][MAX], int x, int y, int taille)` qui renvoie 1 si le carré de coté `taille` dont le coin supérieur gauche a pour coordonnées (x, y) est unifforme en couleur, *i.e.* est formé de pixels de la même couleur, et 0 sinon ;
- `void colorie(unsigned char image[][MAX], int x, int y, int taille, unsigned char couleur)` qui remplit avec `couleur` les cases du tableau correspondant au carré de coté `taille` dont le coin supérieur gauche a pour coordonnées (x, y).

1. Comment distingue t-on un nœud interne d'une feuille dans un arbre quaternaire ?
2. Écrire la fonction `void arbre_2_img(Qarbre a, unsigned char image[][MAX])` qui reçoit un arbre quaternaire décrivant une image carrée de taille `MAX` et remplit le tableau `image` avec les couleurs correspondantes.
3. Écrire la fonction `int img_2_arbre(unsigned char image[][MAX], Qarbre *a)` qui reçoit une image et construit un arbre quaternaire la représentant.

   L'arbre reçu sera supposé vide au premier appel. L'entier renvoyé indiquera si l'arbre a bien été construit.

▸ Exercice 3. Arbre binaire de recherche (3 points)

1. Dessiner l'arbre binaire de recherche obtenu après insertion dans un arbre initialement vide des nombres 6, 4, 9, 1, 7, 8, 3, 10, 5 et 2.
2. Dessiner chaque arbre binaire de recherche obtenu après suppression des nombres 10, 1, 6 et 9 dans l'arbre précédent

   Lors d'une suppression dans un arbre binaire de recherche, on remontera lorsqu'il y aura le choix la valeur minimale du sous-arbre de droite.

▸ Exercice 4. Union-Find (3,5 points)

On considère les ensembles disjoints représentés par la table des pères suivantes :

0	1	2	3	4	5	6	7	8
0	0	0	4	4	3	0	2	5

On va réaliser dessus les opérations suivantes :

```
x = Trouver(7)
y = Trouver(6)
Fusion(x, y)
```

1. Représenter la forêt d'ensembles disjoints initiale représentée par la table des pères.

2. Quelle est la forêt d'ensembles disjoints obtenues après la réalisation des trois opérations sans fusion par rang, ni compression des chemins ?
3. En supposant que le rang des arbres est exactement leur hauteur et en utilisant uniquement la fusion par rang, quelle est la forêt d'ensembles disjoints obtenues ?

   On précisera la valeur des rangs après la fusion.
4. En utilisant uniquement la compression des chemins, quelle est la forêt d'ensembles disjoints ?

Remarque : Lors d'une fusion, sans précision ou en cas d'égalité de rang, la nouvelle racine sera choisie comme le plus petit des représentants des éléments fusionnés.

▸ **Exercice 5. AVL** (4 points)

1. Quel est l'arbre de Fibonacci $\mathbf{F}_4$?
2. Donner un ordre d'insertion des entiers 1, 2, $\cdots$, 7 dans un AVL initialement vide qui permette d'obtenir $\mathbf{F}_4$ sans qu'une rotation soit nécessaire pour équilibrer l'arbre.

   La réponse devra être justifiée.
3. Justifier que si l'on obtient l'arbre $\mathbf{F}_4$ après insertion des entiers 1, 2, $\cdots$, 7 dans un certain ordre dans un AVL initialement vide, alors 1 est le dernier entier a avoir été inséré.
4. Donner un ordre d'insertion des entiers 1, 2, $\cdots$, 7 dans un AVL initialement vide qui permette d'obtenir $\mathbf{F}_4$ en ayant besoin de réaliser exclusivement une rotation gauche pour équilibrer l'arbre.
5. Donner un ordre d'insertion des entiers 1, 2, $\cdots$, 7 dans un AVL initialement vide qui permette d'obtenir $\mathbf{F}_4$ en ayant besoin de réaliser une rotation droite-gauche pour équilibrer l'arbre.