Travaux Dirigés d'informatique n°9

Algorithmique

B-arbres de recherche et arbres rouge/noir.

Pour représenter un B-arbre, nous utiliserons la structure suivante:

```
#define ORDRE 2
typedef struct bnoeud
{
    int nbcle;
    int valeur[2*ORDRE-1];
    struct bnoeud (*fils)[2*ORDRE];
} Bnoeud, *Barbre;
```

où la constante `ORDRE` est l'ordre du B-arbre, l'entier `nbcle` est le nombre de clés contenus dans un nœud et la variable `fils` est un tableau de type `Barbre`.

Exercice 1 Recherche

Ecrire une fonction qui recherche la présence d'un entier dans un B-arbre. Cette fonction retournera l'adresse du nœud contenant cet entier ou `NULL` *sinon. On pourra supposer qu'il n'y a pas de doublons.*

Exercice 2 Insertion dans un B-arbre

Dans cet exercice, on traitera les éclatements des nœuds pleins à la descente.

1. *Dessiner les B-arbres d'ordre 2 et 3 obtenus après insertion, dans l'ordre, des entiers suivants: 37, 23, 50, 62, 58, 47, 83, 29, 36, 40, 46, 56, 33;*
2. *à quelle condition, lors de l'insertion, la hauteur de l'arbre est augmentée?*
3. *écrire une fonction qui permet d'insérer un entier dans un B-arbre. On n'oubliera pas de traiter les éclatements des nœuds pleins rencontrés avant de les traverser.*

Exercice 3 Suppression

Supprimer dans le B-arbre d'ordre 2 de l'exercice 2, les entiers 56, 23, 37, 58, 50. Vous utiliserez toujours le B-arbre de l'exercice 2 pour ces suppressions.

Exercice 4 Arbres rouge/noir

Construire l'arbre rouge noir pour la suite d'entiers de l'exercice 2.