Algorithmique et Structures de données

L2 2021-2022

Travaux Pratiques 7 - Arbre binaire de recherche -

`gcc mon_fichier.c -std=c11 -Wall -Wextra -o mon_programme`
Les exercices marqués d'un @ sont optionnels et à faire dans un second temps.

On utilisera les définitions de types suivantes pour les arbres binaires de recherche :

```
typedef struct noeud
{
   int val;
   struct noeud* g;
   struct noeud* d;
} Noeud;
typedef Noeud* Arbre;
```

Exercice 1. *Ajout*
Écrire la fonction `void ajouter(Arbre* a,int x)` qui ajoute l'entier `x` à l'arbre `*a`. Si l'entier existe déjà on ne fait rien.

Exercice 2. *Vérif*
Écrire la fonction `int estABR(Arbre a)` qui renvoie 1 si `a` est bien un ABR, 0 sinon.

Exercice 3. *Recherche*
Écrire la fonction `Noeud* rechercher(Arbre a,int x)` qui renvoie l'adresse du noeud si l'entier `x` est présent dans l'arbre `a`. Si `x` n'est pas dans `a` renvoie `NULL`.

Exercice 4. *Extraire max*
Écrire la fonction `Noeud* extraireMax(Arbre* a)` qui retire de l'arbre `*a` le noeud ayant la plus grande étiquette. Renvoie l'adresse du noeud extrait.

Exercice 5. *@Extraire min*
Écrire la fonction `extraireMin`.

Exercice 6. *Supprime*
Écrire la fonction `Noeud* extraire(Arbre* a,int x)` qui extrait le noeud ayant pour valeur `x` dans l'arbre et renvoie l'adresse du noeud. Renvoie `NULL` si `x` n'est pas présent dans l'arbre.

Exercice 7. *Hauteur*
Écrire la fonction `int hauteur(Arbre a)` qui renvoie la hauteur de l'arbre.

Exercice 8. *@Hauteur intégré*
On veut conserver la hauteur d'un arbre dans un champs supplémentaire. On utilise pour cela la structure.

```
typedef struct noeudh
{
   int val;
   int haut; // pour la hauteur
   struct noeudh* g;
   struct noeudh* d;
} Noeud;
typedef Noeudh* ArbreH;
```

Réécrire les fonctions précédentes en mettant à jour le champs `hauteur` à chaque opération modifiant l'arbre.