Semaine 5 – Arbres avancés

Dnas ce TP, vous allez terminez l'implémentation des arbres binaires de recherche ainsi que le tri exploitant ces arbres.

1 Opération supplémentaires

Afin de terminer l'implémentation de nos arbres binaire de recherche, il nous reste à implémenter les fonctions de recherche et de suppression dans l'arbre.

— Commencer par impléenter une fonction de recherche d'une valeur spécifique qui renvois l'entier 1 si la valeur est présente dans l'arbre et 0 sinon :

```
int contiens(arbre *a, int val);
```

— Implémentez ensuite la fonction de suppression d'une valeur donnée :

```
void supprimer(arbre *a, int val);
```

Cette fonction est complexe et comporte beaucoup de cas particuliers, n'hésitez pas à définir des sous-fonctions afin de rendre son implémentation plus simple. *Pensez à bien tester les différents cas possibles, ils sont nombreux.*

2 Tri avec un arbre

Maintenant que notre implémentation des arbres binaires de recherche est complète, nous pouvons implémenter une fonction de tri avec.

— Implémentez une fonction de tri qui prend en paramètre un tableau et sa taille et en réalise le tri à l'aide d'un arbre :

```
void tri(int tab[], int n);
```

— *Approfondissement : Il est possible de réaliser le tri de manière plus efficace sans avoir à supprimer les élément un par un. Pour cela il conviens de parcourir l'arbre de manière récursive afin d'extraire les valeurs dans l'ordre. Il est toutefois complexe de garder la trace de la position ou un élément doit être placé dans le tableau. Implémenter une telle procédure de tri.*

3 Manipulation des arbres

De nombreuses opérations supplémentaires peuvent être implémentées sur les arbers en utilisant les techniques utilisées pour les fonction précédentes. Implémentez une fonction calculant la médiane d'un arbre binaire de recherche :

```
int mediane(arbre *a);
```