UNIVERSITÉ PARIS 13 | USPC Université Sorbonne Paris Cité

Structures de Données Avancées

Rapport TP4

Arbre B et AVL

Réalisé par :

Aoudjehane Sarah

KOUACHI Abdeldjalil

1) **implémentation de B arbre**

Notre structure comporte 3 classes: classe ArbreNode,BNode et la classe main.Cette structure permet de réaliser toutes les fonctions demandés:création de l'arbre,insertion,suppression,affichage de l'arbre et rechercher un élément

.

```
public BNode search(BNode root, int key)
{
    int i = 0;//we always want to start searching the 0th index of node.

    while(i < root.count && key > root.key[i])//keep incrementing
                                              //in node while key >
                                              //current value.
    {
        i++;
    }

    if(i <= root.count && key == root.key[i])//obviously if key is in node
                                             //we went to return node.
```

```
public void deleteKey(ArbreB t, int key)
{

 BNode temp = new BNode(order,null);//temp Bnode

 temp = search(t.root,key);//call of search method on tree for key

 if(temp.leaf && temp.count > order - 1)
 {
     int i = 0;

     while( key > temp.getValue(i))
     {
```

```
public void insert(ArbreB t, int key)
{
    BNode r = t.root;//this method finds the node to be inserted as
             //it goes through this starting at root node.
    if(r.count == 2*order - 1)//if is full
    {
        BNode s = new BNode(order,null);//new node

        t.root = s;     //\
                        // \
        s.leaf = false;//  \
                        //   > this is to initialize node.
        s.count = 0;    //  /
                        // /
```

l'intégralité du code se trouve dans le fichier Tp4

2) La classe BNode comporte les propriétés suivante

```
    public BNode(int t, BNode parent)
    {
        this.t = t;  //assign size

        this.parent = parent; //assign parent

        key = new int[2*t - 1];  // array of proper size

        child = new BNode[2*t]; // array of refs proper size

        leaf = true; // everynode is leaf at first;

        count = 0; //until we add keys later.
    }
public class BNode
{
    static int t;  //variable to determine order of tree

    int count; // number of keys in node

    int key[];  // array of key values

    BNode child[]; //array of references

    boolean leaf; //is node a leaf or not

    BNode parent;  //parent of current node.

// ------------------------------------------------
```

Un degré t,un nombre de clef,un tableau pour stocker les clés de chaque noeud , un tableau children pour stocker les fils de chaque noeud,un noeud parent et un boolean pour tester si le noeud courant est une feuille ou non. Le nombre de clef est fixe (t-1,2*t - 1) et nombre des fils aussi 2*t c'est pour cela on utilise un tableau fixe pour stocker leurs valeurs.

3) la structure de AVL

Il y a 3 classes AvlTree ,Node et main, cette structure permet d'insérer les éléments ,chercher un élément , supprimer une clef données et rééquilibrage à droite et gauche

```
1  public class AVLTree {
2      private Node root;
3
4      public AVLTree() {
5          root = null;
6      }
7
8      private void printHelper(Node currPtr, String indent, boole
9          // print the tree structure on the screen
0          if (currPtr != null) {
1             System.out.print(indent);
2             if (last) {
3                System.out.print("R----");
4                indent += "     ";
5             } else {
```

```
onsole
(4) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (30 déc. 2019 à 11:16:28)
```

```
    // delete the node from the tree
    Node deleteNode(int data) {
        return deleteNodeHelper(this.root, data);
    }

    // print the tree structure on the screen
    public void prettyPrint() {
        printHelper(this.root, "", true);
    }
}
```

```
    // insert the key to the tree in its appropriate
    public void insert(int key) {
        // PART 1: Ordinary BST insert
        Node node = new Node(key);
        Node y = null;
        Node x = this.root;

        while (x != null) {
            y = x;
            if (node.data < x.data) {
                x = x.left;
            } else {
                x = x.right;
            }
```

L’intégralité de code se trouve dans le fichier Tp4