Project 3 Report

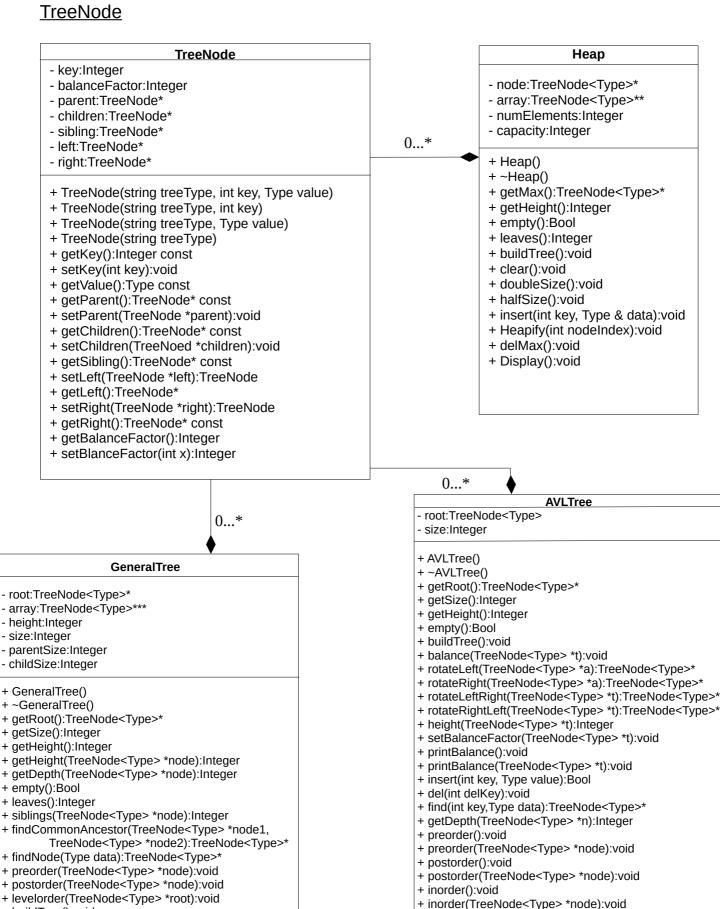
+ buildTree():void

+ insert(int head, int parent, int child, Type data):void

+ del(TreeNode<Type> *data):void

+ clear():void

+ display():void



+ clear():void + clear(TreeNode<Type> *d):void

+ siblings(TreeNode<Type> *node):Integer

+ leaves(TreeNode<type> *node):Integer

+ levelorder():void

HEAP

Heap

- node:TreeNode<Type>*
- array:TreeNode<Type>**
- numElements:Integer
- capacity:Integer
- + Heap()
- + ~Heap()
- + getMax():TreeNode<Type>*
- + getHeight():Integer
- + empty():Bool
- + leaves():Integer
- + buildTree():void
- + clear():void
- + doubleSize():void
- + halfSize():void
- + insert(int key, Type & data):void
- + Heapify(int nodeIndex):void
- + delMax():void
- + Display():void

Efficiency of getMax(): O(1)

Efficiency of getHeight(): O(1)

Efficiency of empty(): O(1)

Efficiency of getSize(): O(1)

Efficiency of leaves(): O(n)

Efficiency of buildTree(): O(n)

Efficiency of clear(): O(n)

Efficiency of doubleSize(): O(2n + n)

Efficiency of halfSize(): O(n + n/4)

Efficiency of insert(): O(nlogn (on average))

Efficiency of Heapify(): O(logn)

Efficiency of del(): O(nlogn)

Efficiency of Display(): O(n)

General Tree

GeneralTree

- root:TreeNode<Type>*
- array:TreeNode<Type>***
- height:Integer
- size:Integer
- parentSize:Integer
- childSize:Integer
- + GeneralTree()
- + ~GeneralTree()
- + getRoot():TreeNode<Type>*
- + getSize():Integer
- + getHeight():Integer
- + getHeight(TreeNode<Type> *node):Integer
- + getDepth(TreeNode<Type> *node):Integer
- + empty():Bool
- + leaves():Integer
- + siblings(TreeNode<Type> *node):Integer
- + findCommonAncestor(TreeNode<Type> *node1, TreeNode<Type> *node2):TreeNode<Type>*
- + findNode(Type data):TreeNode<Type>*
- + preorder(TreeNode<Type> *node):void
- + postorder(TreeNode<Type> *node):void
- + levelorder(TreeNode<Type> *root):void
- + buildTree():void
- + clear():void
- + display(TreeNode<Type> *node):void
- + insertTreeNode<Type> *data, int parentNodeKey):void
- + del(TreeNode<Type> *data):void

Efficiency of getRoot: O(1)

Efficiency of getSize(): O(1)

Efficiency of getHeight(): O(1)

Efficiency of empty(): O(1)

Efficiency of buildTree(): O(n)

Efficiency of insert(): O(logn)

Efficiency of del(): O(logn)

Efficiency of findCommonAncestor():

Efficiency of findNode(): O(logn)

Efficiency of getDepth(): O(logn)

Efficiency of preorder(): O(n)

Efficiency of postorder(): O(n)

Efficiency of levelorder(): O(n)

Efficiency of siblings(): O(1)

Efficiency of leaves(): O(n)

Efficiency of clear(): O(n)

Efficiency of display():O(n)

AVL Tree

Efficiency of getRoot: O(1)

Efficiency of getSize(): O(1)

Efficiency of getHeight(): O(1)

Efficiency of empty(): O(1)

Efficiency of buildTree(): O(n)

Efficiency of balance(): O(logn)

Efficiency of height(): O(logn)

Efficiency of setBalance(): O(1)

Efficiency of printBalance(): O(1)

Efficiency of insert(): O(logn)

Efficiency of del(): O(logn)

Efficiency of find(): O(logn)

Efficiency of getDepth(): O(logn)

Efficiency of preorder(): O(n)

Efficiency of postorder(): O(n)

Efficiency of inorder(): O(n)

Efficiency of levelorder(): O(n)

Efficiency of siblings(): O(1)

Efficiency of leaves(): O(n)

Efficiency of clear(): O(1)

Efficiency of rotateLeft(): O(logn)

Efficiency of rotateRight(): O(logn)

Efficiency of rotateLeftRight(): O(logn)

Efficiency of rotateRightLeft(): O(logn)

AVLTree

- root:TreeNode<Type>
- size:Integer
- + AVLTree()
- + ~AVLTree()
- + getRoot():TreeNode<Type>*
- + getSize():Integer
- + getHeight():Integer
- + empty():Bool
- + buildTree():void
- + balance(TreeNode<Type> *t):void
- + rotateLeft(TreeNode<Type> *a):TreeNode<Type>*
- + rotateRight(TreeNode<Type> *a):TreeNode<Type>*
- + rotateLeftRight(TreeNode<Type> *t):TreeNode<Type>*
- + rotateRightLeft(TreeNode<Type> *t):TreeNode<Type>*
- + height(TreeNode<Type> *t):Integer
- + setBalanceFactor(TreeNode<Type> *t):void
- + printBalance():void
- + printBalance(TreeNode<Type> *t):void
- + insert(int key, Type value):Bool
- + del(int delKey):void
- + find(int key, Type data): TreeNode < Type >*
- + getDepth(TreeNode<Type> *n):Integer
- + preorder():void
- + preorder(TreeNode<Type> *node):void
- + postorder():void
- + postorder(TreeNode<Type> *node):void
- + inorder():void
- + inorder(TreeNode<Type> *node):void
- + levelorder():void
- + siblings(TreeNode<Type> *node):Integer
- + leaves(TreeNode<type> *node):Integer
- + clear():void
- + clear(TreeNode<Type> *d):void