

REPORT – Bplus Tree Implementation

Name: Santhi Sushma Katragadda

UFID: 1748 – 9431

UF Email: skatragadda@ufl.edu

Main logic

Main class is “treesearch” inside which Mainnode class is defined which is a parent class for leaf and indexnode class. While insertion logic and variables are different for leaf node and index node, different classes are defined and functions are defined separately for each of them. Going forward list of classes and functions that were used are given. Later the classes are described briefly with all the functions inside them, the logic and variables.

List of Classes

- 1) Treesearch
- 2) Mainnode
- 3) Leaf
- 4) IndexNode
- 5) Splitvalue

List of Functions:

- 1) Insert(key,value)
- 2) Search(key)
- 3) Search(key1,key2)
- 4) getLocation(key)
- 5) Insert(key,value) – leaf node
- 6) Insert(key,value) – Index node
- 7) Insertwhennotfull(key,value,index)

Classes – Variables, Functions inside the class

1) Class - treesearch

```
public class treesearch<Double extends Comparable<? super Double>, Value>
```

a) Variables defined in the class

root node which could be index node or leaf node.

```
private Mainnode root;
```

degree – includes leaf node and index node degree

b) Functions inside this class

a) Insert (Double key, Value value) – Inserts a key value pair with key and value into the tree.

Logic: This calls the insert function in mainnode which returns splitvalue. If the splitvalue is null, nothing is done and element is inserted in the node. But if the split value is not null, a new root is created with left and right nodes being child nodes.

```
public void insert (Double key, Value value)
```

b) Search(key) – Returns array list of values with key value ‘key’

Tree is traversed from root node till end until leaf node is reached for given key. Now since leaf nodes are connected as doubly linked list, leaf nodes are traversed from the obtained leaf node in front and back ward direction until the leaf node has given key element or until the leaf node is null.

```
public ArrayList<String> search (Double key)
```

c) Search (key1, key2) – Returns array list of all key value pairs with keys in the range of key1 and key2 inclusive. Tree is traversed from root node till end until leaf node is reached for left most key of given range(key1). Now since leaf nodes are connected as doubly linked list, leaf nodes are traversed from the obtained leaf node in front and back ward direction until the leaf node has given range of key element or until the leaf node is null.

```
public ArrayList<String> search (Double key1, Double key2)
```

2) Class - Mainnode

`abstract class Mainnode`

a) Variables defined in the class

Keys – Leaf node and Index node inherits the same variables

`protected Double[] keys;`

b) Functions inside this class

a) **getLocation(Double key)** – To get the location of Node with a given key value.

`abstract public int getLocation(Double key);`

b) **Splitvalue insert(Double key, Value value)** – This is basic function called for a node when insert is called from tree search. This function is different for leaf node and index node. Logic of this is explained separately in leaf node class and index node class.

If split is not required, insert non full is called for a different split.

`abstract public Splitvalue insert(Double key, Value value);`

3) Class - leaf – This class inherits Node with variables defined in Node and adding few more specific variables and functions as leaf node insert is different from index node. (Reason being, insert in index node works similar to B tree and not B plus tree)

a) Variables defined in the class

Next and prev Nodes as all leaf nodes are connected through doubly linked list.

leaf `next = null;`

leaf `prev = null;`

key value pairs stored in the form of array and values being sorted with priority queue

`final Value[] values = (Value[]) new Object[M-1];`

`{ keys = (Double[]) new Comparable[M-1]; }`

b) Functions inside this class

a) **getLocation(Double key)** – To get the location of Node with a given key value. If the element does not exist it returns the length of node.

`public int getLocation(Double key);`

b) **Splitvalue insert(Double key, Value value)** – Split function called for leaf node. Checks for the condition if node length is full, if it is not full, it calls Insert with not full function. Insertion also occurs as doubly

linked list insertion as leaf nodes are connected as doubly linked list nodes. Normal array copy is called as a new element is inserted to allot new array size for extra element.

```
public Splitvalue insert (Double key, Value value);
```

- c) **Insertwhennotfull (Double key, Value value)** – This function is called when split is not required and a normal insertion. Normal array copy is called as a new element is inserted to allot new array size for extra element.

```
private void Insertwhennotfull(Double key, Value value, int idx)
```

- 4) **Class - IndexNode** – This class inherits Node with variables defined in Node and adding few more specific variables and functions as Index node insert is different from leaf node. (Reason being, insert in index node works similar to B tree and not B plus tree)

a) **Variables defined in the class**

Child node is defined as index node has children whereas leaf node does not.

```
final Mainnode[] child = new treeearch.Mainnode[N];
```

key value pairs stored in the form of array and values being sorted with priority queue

```
{ keys = (Double[]) new Comparable[N-1]; }
```

b) **Functions inside this class**

- a) **getLocation(Double key)** – To get the location of Node with a given key value. If the element does not exist it returns the length of node. Linear search is implemented and index is returned when an element is found.

```
public int getLocation(Double keyi);
```

- d) **Splitvalue insert(Double key, Value value)** – Split function called for Index node. Checks for the condition if node length is full, If it is not full, it calls Insert with not full function. Normal array copy is called as a new element is inserted to allot new array size for extra element.

```
public Splitvalue insert (Double key, Value value);
```

- e) **Insertwhennotfull (Double key, Value value)** – This function is called when split is not required and a normal insertion. Normal array copy is

called as a new element is inserted to allot new array size for extra element.

```
private void Insertwhennotfull(Double key, Value value, int idx)
```

5) Splitvalue :

This defines the partition where split occurred with left and right nodes defined.

a) **Variables in this class:** key, leftnode, rightnode

6) Main

Reads input file which is passed as an argument and calls specific functions defined and writes output to the output file.

a) **Variables:**

in: Buffered reader to read the input from input file where location is passed as an argument.

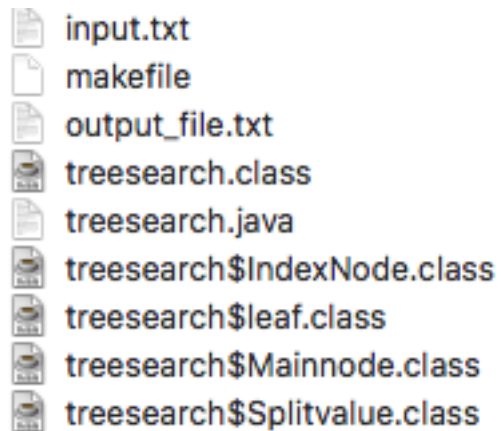
Output: Buffered writer to write output to outputfile.

Degree: degree of the B+ tree that can be parsed from input file first line. File is read till the end and appropriate function is called based on the input command with output being written to output file.

List of Imported Libraries:

```
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.InputStreamReader;  
import java.text.DecimalFormat;  
import java.text.ParseException;
```

List of Classes screenshot:



- input.txt
- makefile
- output_file.txt
- treesearch.class
- treesearch.java
- treesearch\$IndexNode.class
- treesearch\$leaf.class
- treesearch\$Mainnode.class
- treesearch\$Splitvalue.class