COP 5536 Spring 2021 Programming Project Report

Name: Jingxi Su
UFID: 91410393
UF Email: jingxi.su@ufl.edu

Function Prototypes

Data node (Contains the keys and values of the variables)

```
class Bp_Datanode {
   protected float node_Val;
   protected int node_Key;

   public Bp_Datanode(int node_Key, float node_Value){
      this.node_Key = node_Key;
      this.node_Val = node_Value;
   }
}
```

Node (Node_Isleaf tells whether node is leaf or not\Node_issplit tells whether node is split or not)

```
class Bpnode {

   protected boolean Node_Isleaf = false;
   protected int Node_degrees;
   protected boolean Node_issplit = false;

}
```

Leaf node (extension of Bpnode)

```
class Node_leaf extends Bpnode {

   protected ArrayList<Bp_Datanode> data_Nodes = new
ArrayList<Bp_Datanode>();
   protected Node_leaf prev = null;
   protected Node_leaf next = null;
   protected Node_leaf() {
      this.Node_Isleaf = true;
   }

   /*Insert the datanode into the leaf node*/
```

```
   protected void inNode(Bp_Datanode data_Node){
      if(data_Nodes.isEmpty()){
         data_Nodes.add(data_Node);
      }
      else{ /*data node is inserted in a right path*/
         int i=0;
         while(i < data_Nodes.size()){
            if(data_Nodes.get(i).node_Key > data_Node.node_Key){
               data_Nodes.add(i, data_Node);
               return;
            }
            i++;
         }
         data_Nodes.add(data_Node);
      }
   }
}
```

Int node (extension of Bpnode)

```
class node_IntNode extends Bpnode {

   protected ArrayList<Integer> key_Number = new
ArrayList<Integer>();
   protected ArrayList<Bpnode> int_Ptrs = new ArrayList<Bpnode>();

   /*Insert in a sorted array*/
   protected void insert(int node_key, Bpnode ptr1, Bpnode ptr2){
      if(this.key_Number.isEmpty()){
         this.key_Number.add(node_key);
         this.int_Ptrs.add(ptr1);
         this.int_Ptrs.add(ptr2);
      }
      else{
         int i;
         for(i=0;i < this.key_Number.size();i++){
            if(this.key_Number.get(i) > node_key) {
            String value="null"; /*See how the flow changes*/
            while(value!="null")
            { if(value!="null")
               System.out.println("checking");
               else
               {
               System.out.println("good to go");
               }
            }
```

```
                this.key_Number.add(i, node_key);
                this.int_Ptrs.add(i + 1, ptr2);

                return;
            }

        }
        this.key_Number.add(node_key);
        this.int_Ptrs.add(ptr2);
      }
      return;
   }

}
```

bplus tree

```
class BPTree {

public static Bpnode root = null;
public static int node_Degrees;

public BPTree(int Node_Degrees){
   this.node_Degrees = Node_Degrees;
}

static BPTree delete (Map inputt,Integer key_Value){
   System.out.println(inputt);
   Iterator iterators=inputt.entrySet().iterator();
   BPTree bPlustree=new BPTree(node_Degrees);
   String aoi= new Integer(key_Value).toString();
   while(iterators.hasNext()){
      Map.Entry<String,String> mapElement=(Map.Entry)iterators.next();

      if(!mapElement.getKey().equals(aoi)){
         System.out.println(mapElement.getKey()+ " : "+
mapElement.getValue());

bPlustree.inNode(Integer.parseInt(mapElement.getKey()),Float.parseFlo
at(mapElement.getValue()));
      }
   }
   return bPlustree;
}
```

Structure of Program

1. public static void main(String[] args) throws Exception

This function will compile the code and create input and output files in the same directory.

```
public static void main(String[] args) throws Exception {
   Map<String,String> inputt=new HashMap<String,String>();

   /*Create input and output files, create buffered readers and
writers*/
   File inputFilename = new File(args[0]+".txt"); /*Enter input.txt
with cmd*/
   //File inputFilename = new File("input.txt"); /*Enter input.txt
with Idea*/
   BufferedReader infile = new BufferedReader(new
FileReader(inputFilename));

   File outputFilename = new File("output_file.txt");
   if(!outputFilename.exists()){
      outputFilename.createNewFile();

   }
   OutputStreamWriter filewriter = new
FileWriter(outputFilename.getAbsoluteFile());
   Writer printwriter = new PrintWriter(filewriter);
```

Then create an object and pass a parameter, which is Node_degrees

```
int Node_degrees =
Integer.parseInt(infile.readLine().split("Initialize\\(")[1].split("\
\)")[0]);
BPTree BPlustree = new BPTree(Node_degrees); /* Pass the degree to
creat a b+ tree*/
```

2. protected static Bpnode inNode(Bpnode root, Bp_Datanode data_Node)

This function inserts a Datanode into a tree with a given root.

First check to see if the root is a leaf. If the root is a leaf node, add the datanode to the root and use the insert function in leafnode. Otherwise, update the root until the root becomes leafNode and then move down the tree, and then use the insert function to insert.

3. protected void insert(int node_key, Bpnode ptr1, Bpnode ptr2)
protected static void inNode(int node_Key, float node_Value)

This function is the call function used to insert a node in insertNode().

4. protected static String node_search(int node_Key)

Used to implement search, search node_key to the leaf node where the data is located.

5. protected static String node_search(double key_One, double key_Two)

Perform a range search on key1 as in the node_search(int node_Key) function. This search operation will call a function and find the Key1 position. Since the nodes are interconnected with each other, it will directly display all node values until it encounters the position of key 2.

6. static BPTree delete (Map inputt,Integer key_Value)

This function will delete the node, otherwise it returns null.

Performance Analysis

Software:
Programming Language: Java
Hardware:
Operating Systems: Windows 10
RAM: 8GB
Hard Disk: 1TB

Sample Snapshots of Input and Output:

Initialize(3)	121.56
Insert(21,0.3534)	3.55,-3.95
Insert(108,31.907)	Null
Insert(56089,3.26)	-3.95
Insert(234,121.56)	
Insert(4325,-109.23)	
Delete(108)	
Search(234)	
Insert(102,39.56)	
Insert(65,-3.95)	
Delete(102)	
Delete(21)	
Insert(106,-3.91)	
Insert(23,3.55)	
Search(23,99)	
Insert(32,0.02)	
Insert(220,3.55)	
Search(33)	
Delete(234)	
Search(65)	

The results show that the requirements of the project can be achieved.