ASSIGNMENT 1

Team – 7

(Rama Charan Pavan, Sri Charan, Prataprao)

1.

a. B Tree: It is a self-balancing tree data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time. The B-tree is a generalization of a binary search tree in that a node can have more than 2 children.
Space Complexity O(n), Average O(log n)

b. B + Tree: B+ tree is an n- ary tree with a variable but often large number of children per node. A B+ tree consists of a root, internal nodes and leaves. The root may be either a leaf or a node with two or more children
Space Complexity O(n), Average O(log n)

c. Binary Search Tree: A binary search tree (BST) is a binary tree where each node has a Comparable key (and an associated value) and satisfies the restriction that the key in any node is larger than the keys in all nodes in that node's left subtree and smaller than the keys in all nodes in that node's right subtree.
Complexity: Average Search O(log n)

d. KD Tree: k-dimentional tree is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key. *k*-d trees are a special case of binary search trees.
Complexity: Average Search O(log n)

e. R Tree: The R-tree is an object hierarchy which is applicable to arbitrary spatial objects which is formed by aggregating their minimum bounding boxes and storing the aggregates in a tree structure. There are two principal methods of determining how to fill each R-tree node. The most natural method is to take the space occupied by the objects into account when deciding which ones to aggregate. An alternative is to order the objects prior to performing the aggregation.
Complexity: Average Search O(log n)

f. Z Ordering: Z-order is an ordering of overlapping two-dimensional objects, such as windows in a graphical user interface (GUI), shapes in a vector graphics editor, or objects in a 3D application
Complexity: Average Complexity O(n)

2.

a. Submitted in github. B Tree Implemented using Java.
GitHub Link : https://github.com/jrcpavan7/Pa

```
175
176     public void postorder() {
177         postorder(root);
178     }
179
180     private void postorder(SortedBTree r) {
181         if (r != null) {
182             postorder(r.getLeft());
183             postorder(r.getRight());
184             System.out.print(r.getD() + " ");
185         }
186     }
187
188 }
189
190 public class Sort_BTree {
191     public static int N = 20;
192
193     public static void main(String args[]) {
194         Random random = new Random();
195         BTreeNodes bt = new BTreeNodes();
196
197         System.out.println("Sorting of randomly generated numbers using B TREE");
198
199         for (int i = 0; i < N; i++)
200             bt.insert(Math.abs(random.nextInt(100)));
201
202         System.out.println("The elements of the tree: ");
203         bt.preorder();
204
205         System.out.println("\nThe sorted sequence is: ");
206         bt.inorder();
207     }
208 }
```

```
133     }
134
135     private boolean search(SortedBTree r, int val) {
136         boolean found = false;
137         while ((r != null) && !found) {
138             int rval = r.getD();
139             if (val < rval)
140                 r = r.getLeft();
141             else if (val > rval)
142                 r = r.getRight();
143             else {
144                 found = true;
145                 break;
146             }
147             found = search(r, val);
148         }
149         return found;
150     }
151
152     public void inorder() {
153         inorder(root);
154     }
155
156     private void inorder(SortedBTree r) {
157         if (r != null) {
158             inorder(r.getLeft());
159             System.out.print(r.getD() + " ");
160             inorder(r.getRight());
161         }
162     }
163
164     public void preorder() {
165         preorder(root);
166     }
```

```
100                 while (p.getLeft() != null)
101                     p = p.getLeft();
102                 p.setLeft(lt);
103                 return p2;
104             }
105         }
106         if (k < root.getD()) {
107             n = delete(root.getLeft(), k);
108             root.setLeft(n);
109         } else {
110             n = delete(root.getRight(), k);
111             root.setRight(n);
112         }
113         return root;
114     }
115 
116     public int countNodes() {
117         return countNodes(root);
118     }
119 
120     private int countNodes(SortedBTree r) {
121         if (r == null)
122             return 0;
123         else {
124             int l = 1;
125             l += countNodes(r.getLeft());
126             l += countNodes(r.getRight());
127             return l;
128         }
129     }
130 
131     public boolean search(int val) {
132         return search(root, val);
133     }
```

```
67                  node.right = insert(node.right, d);
68              }
69              return node;
70          }
71
72          public void delete(int k) {
73              if (isEmpty())
74                  System.out.println("Tree is Empty");
75              else if (search(k) == false)
76                  System.out.println("Sorry" + k + "is not there");
77              else {
78                  root = delete(root, k);
79                  System.out.println(k + "deleted successfully from the tree");
80              }
81          }
82
83          private SortedBTree delete(SortedBTree root, int k) {
84              SortedBTree p, p2, n;
85              if (root.getD() == k) {
86                  SortedBTree lt, rt;
87                  lt = root.getLeft();
88                  rt = root.getRight();
89                  if (lt == null && rt == null)
90                      return null;
91                  else if (lt == null) {
92                      p = rt;
93                      return p;
94                  } else if (rt == null) {
95                      p = lt;
96                      return p;
97                  } else {
98                      p2 = rt;
99                      p = rt;
00                      while (p.getLeft() != null)
```

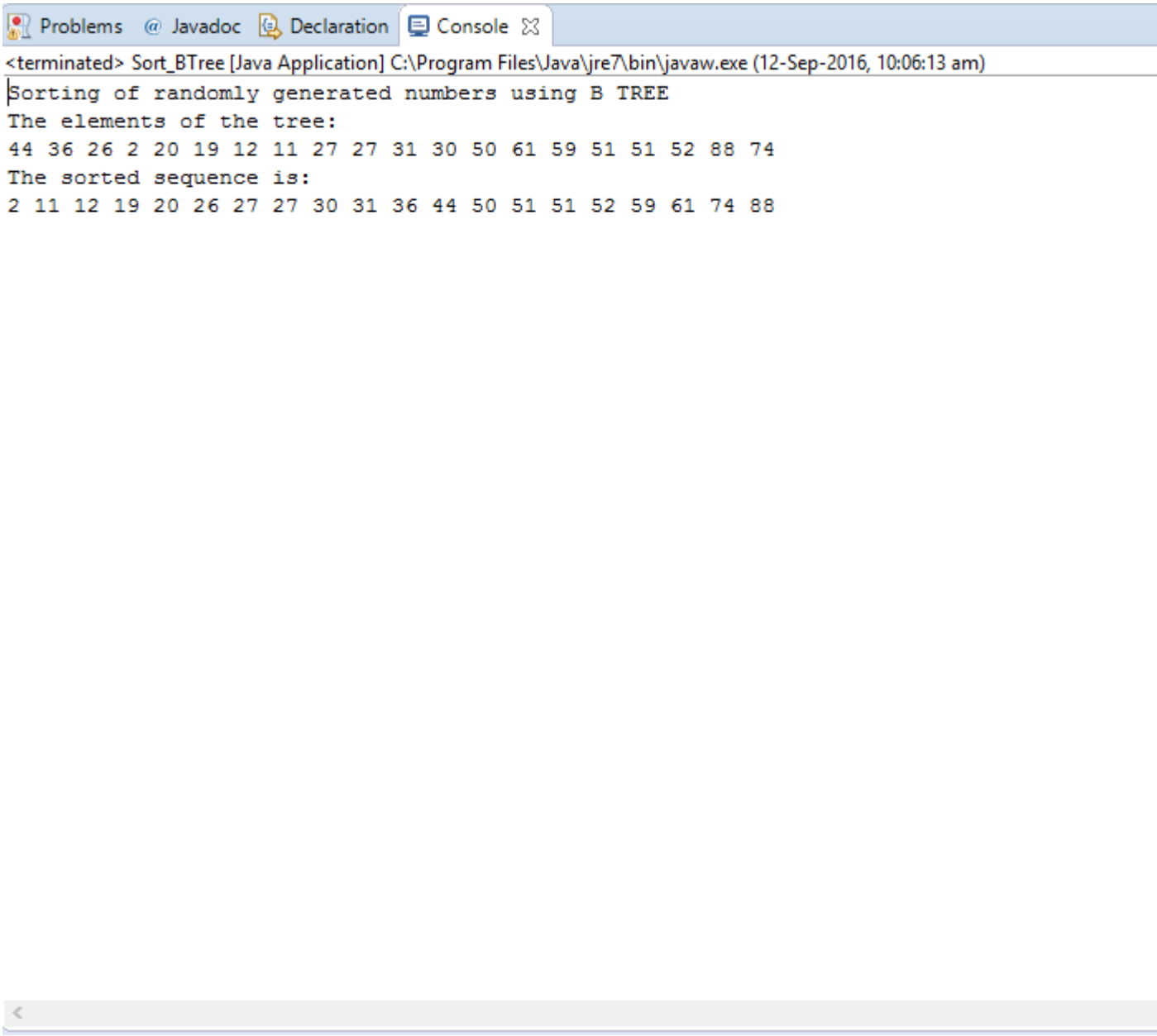
```
35      public int getD() {
36          return d;
37      }
38
39      public void setD(int d) {
40          this.d = d;
41      }
42
43  }
44
45  class BTreeNodes {
46      private SortedBTree root;
47
48      public BTreeNodes() {
49          root = null;
50      }
51
52      public boolean isEmpty() {
53          return root == null;
54      }
55
56      public void insert(int d) {
57          root = insert(root, d);
58      }
59
60      public SortedBTree insert(SortedBTree node, int d) {
61          if (node == null)
62              node = new SortedBTree(d);
63          else {
64              if (d <= node.getD())
65                  node.left = insert(node.left, d);
66              else
67                  node.right = insert(node.right, d);
68          }
```

OO

Sort_BTree.java | BPlusTree.java | SecondBPlusTree.java | InvertedIndex.java | Run.java

```java
import java.util.Random;

class SortedBTree {
    SortedBTree left, right;
    int d;

    public SortedBTree() {
        left = null;
        right = null;
        d = 0;
    }

    public SortedBTree(int n) {
        left = null;
        right = null;
        d = n;
    }

    public SortedBTree getLeft() {
        return left;
    }

    public void setLeft(SortedBTree left) {
        this.left = left;
    }

    public SortedBTree getRight() {
        return right;
    }

    public void setRight(SortedBTree right) {
        this.right = right;
    }
```

Output:



b. Algorithm:

1. Take the file which consists of random number list that is to be read into HDFS.
2. Read the file from local system to HDFS using val numList = sc.textFile("file.txt").
3. Write the mapper code to separate the list and sort them according to B Tree using split and flatMap commands
4. Write the reducer code to combine the output and Print them separately according to the sorted B Tree and give to the Output Text File using reduceByKey command.
5. Example Take the number List in the file like

25 25 10 9 6 1 12 13 16 97 27 79 30 69 56 71 92 82 80 86

6. We will get output in the sorted form like

1 6 9 10 12 13 16 25 25 27 30 56 69 71 79 80 82 86 92 97