ADVANCED DATA STRUCTURES (COP5536)

Spring 2016

Programming Project Report

Name : Sagar Rajani
UFID : 3596-9932
E-Mail ID : srajani@ufl.edu

I have implemented an event counter using red-black tree. Each event has two fields: ID and count, where count is the number of active events with the given ID. The event counter stores only those ID's whose count is > 0. Once a count drops below 1, that ID is removed. Initially, your program must build red-black tree from a sorted list of n events (i.e., n pairs (ID, count) in ascending order of ID) in O(n) time. Your counter should support the following operations in the specified time complexity.

1. Working Environment

- Hardware Requirements

  - Hard Disk space: Minimum 5 GB

  - RAM: Minimum 2 GB

  - CPU: X86 or running 32 bit on X64

- Operating System

  - Mac OSX

- Compiler

  - javac

2. Compiling Instructions

The project has been compiled and tested under

Thunder (thunder.cise.ufl.edu)

Eclipse

Mac OSX

Executed the program using following commands

a). open server using -> ssh srajani@thunder.cise.ufl.edu

b). Compile the java code using

```
javac bbst.java
```

c). Run the file using

```
java bbst test_100.txt < commands.txt > output.txt
```

3. Function Prototypes and Program Structure

class RBNode
It is a class that initializes a node.

Data Members
RBNode left
RBNode right
Int count
Int id
Int color

Constructor: RBNode(int id,int count)
Arguments: id and count of a node.
Description: initializes a node with id and count.

Constructor: RBNode(int id,int count,RBNode l,RBNode r)
Arguments: id, count, RBNode l, RBNode r.
Description: initializes a node with id, count, left, right and Black color.

Class RBTree
It is a class that implements event counter using red black tree.

Data Members
RBNode node
RBNode parent
RBNode parent2
RBNode parent3
RBNode head
RBNode nullnode

Constructor: RBTree(int id,int count)
Arguments: id, count
Description: Takes in id and count for root node with left and right equal to null.

Function: insert(int id,int count)
Arguments: Takes in id and count for nodes of red black tree.
Returns: void
Description: Inserts the node in the tree with id and count.

Function: reStructure(int id)
Arguments: takes in the id.
Returns: void
Description: Restructures the tree.

Function: rotate(int id, RBNode parent)
Arguments: Takes in id and node parent.
Returns: RBNode
Description: Rotates the tree to maintain height balancing.

Function: rotateWithLeftChild(RBNode node)
Arguments: node
Returns: RBNode
Description: It rotates with respect to left child.

Function: rotateWithRightChild(RBNode node)
Arguments: node
Returns: RBNode
Description: It rotates with respect to right child.

Function: increase(int id,int count)
Arguments: Takes in id and count.
Returns: int
Description: Increases the count of id by count if present else inserts id,count.

Function: int count(int id)
Arguments: id
Returns: int
Description: Prints the count of id.

Function: inrange(int id1, int id2)
Arguments: Takes in id1 and id2.
Returns: int
Description: Call to inrange(root,id1,id2)

Function: inrange(RBNode node, int id1, int id2)
Arguments: Takes in node, id1, id2.
Returns: int
Description: Prints the sum of counts of id’s between id1 & id2.

Function: reduce(int id,int count)
Arguments: Takes in id1, id2.
Returns: int
Description: Reduces the count of id by count if present else prints 0, if count reduces to less than deletes the node.

Function: findClosest(int id)
Arguments: Takes in node, id1.
Returns: int
Description: Returns the closest id.

Function: findMinimum(RBNode node)
Arguments: Takes in node.
Returns: int
Description: Finds minimum in the left subtree.

Function: next(int id)
Arguments: Takes in id.
Returns: int
Description: Finds the next of id.

Function: findMaximum(RBNode node)
Arguments: Takes in node.
Returns: int
Description: Finds the maximum in right subtree.

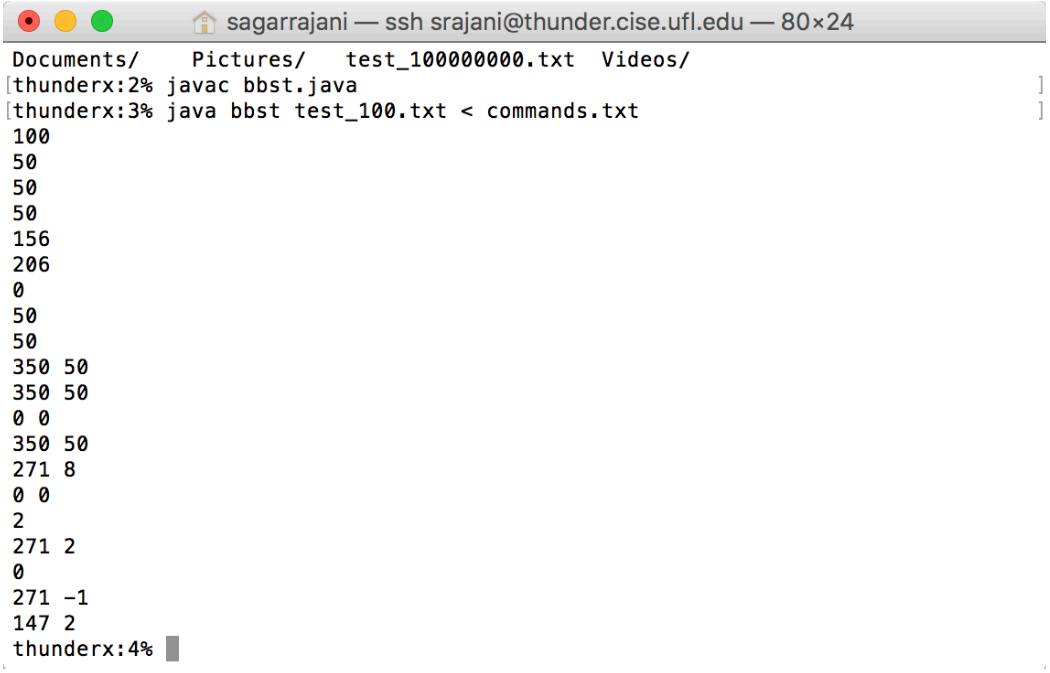
Function: previous(int id)
Arguments: Takes in id.
Returns: int
Description: Finds the previous of id.

Function: search(int id)
Arguments: Takes in id.
Returns: boolean
Description: Searches for the id if present.

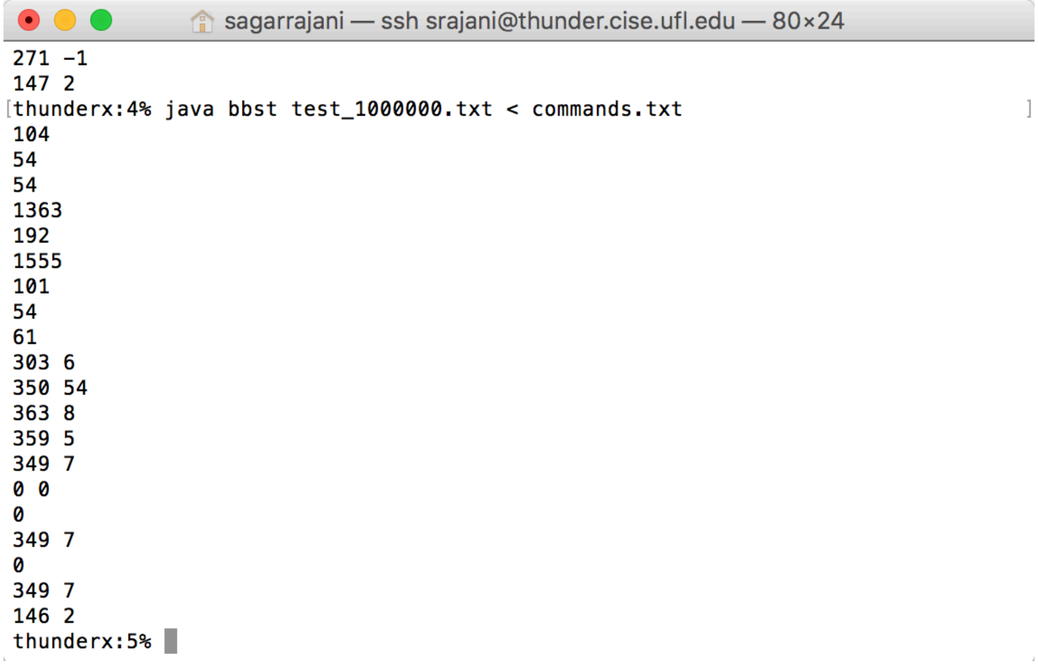
class bbst
It is the class in which all functions are invoked and test and command files are read.

Function: main(String[] args)
Arguments: String args[]
Returns: void
Description: Main function to invoke all functions and read files.

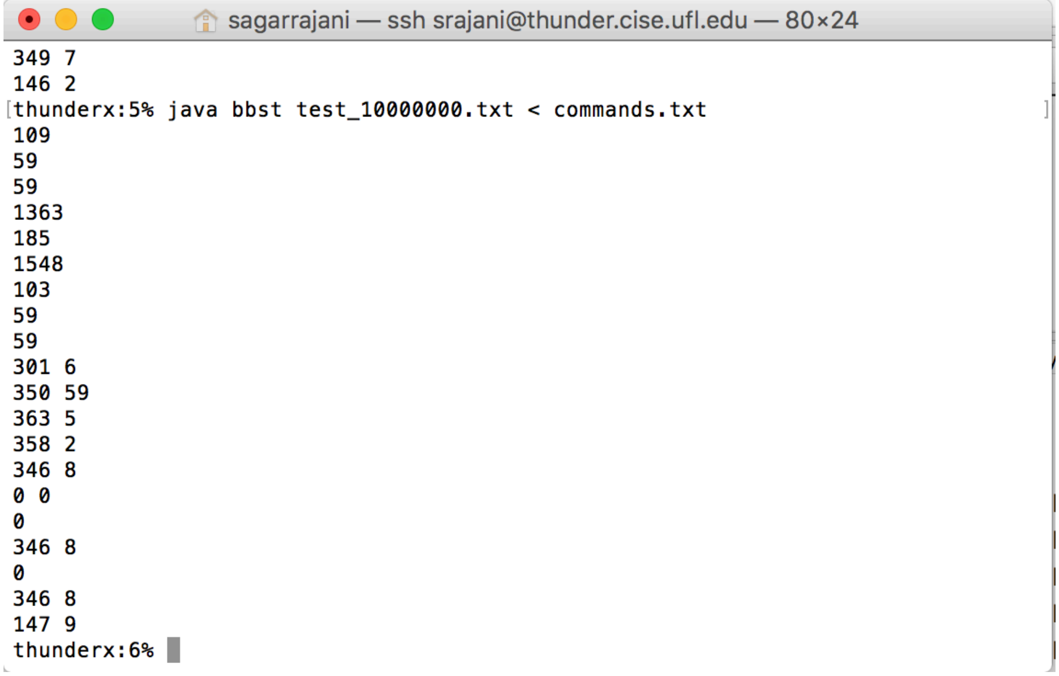
4. Outputs



```
Documents/    Pictures/   test_100000000.txt  Videos/
thunderx:2% javac bbst.java
thunderx:3% java bbst test_100.txt < commands.txt
100
50
50
50
156
206
0
50
50
350 50
350 50
0 0
350 50
271 8
0 0
2
271 2
0
271 -1
147 2
thunderx:4%
```



```
271 -1
147 2
thunderx:4% java bbst test_1000000.txt < commands.txt
104
54
54
1363
192
1555
101
54
61
303 6
350 54
363 8
359 5
349 7
0 0
0
349 7
0
349 7
146 2
thunderx:5%
```



```
349 7
146 2
thunderx:5% java bbst test_10000000.txt < commands.txt
109
59
59
1363
185
1548
103
59
59
301 6
350 59
363 5
358 2
346 8
0 0
0
346 8
0
346 8
147 9
thunderx:6%
```

5. Conclusion

Red Black tree is used to implement event counter. This event counter works perfectly according to the prescribed specifications. The counter supports various specified operations in specified time complexity.

6. References

- **http://www.sanfoundry.com/java-program-implement-red-black-tree**
- **http://users.cis.fiu.edu/~weiss/dsaajava3/code/RedBlackTree.java**
- **http://algs4.cs.princeton.edu/33balanced/RedBlackBST.java.html**
- **https://www.quora.com/How-can-you-find-successors-and-predecessors-in-a-binary-search-tree-in-order**
- **http://algorithms.tutorialhorizon.com/inorder-successor-in-binary-tree/**