COP 5536: Spring 2016
Programming Assignment - I
Red Black Tree Implementation

Author: Ananda Kishore Sirivella | UFID: 9951-5080

Language and Compiler Information:-

- Java – version 1.8.0_65
- Laptop specifications - Core i5 (2.3GHz), 8GB RAM & windows 10
- Tested for tree sized of node 10^2, 10^6. 10^7 & 10^8 (by increasing heap-size to 8096M with –Xms8096 argument).
- Directory structure:
- Step to compile: Javac bbst.java
  Creates the following files:
  - Bbst.class
  - Bbst$1.class
  - Bbst$Opt.class
  - Bbst$Color.class
  - Bbst$Node.class
- Steps to execute:
  - Java bbst <Filename/>
  - Filename: File with initial configuration for building a red-black tree
  - Once the tree is build, program accept arguments to perform increase, decrease, count, inrange, next & previous operations.
  - Exits on typing “quit”

Function Prototypes and Description:-

Node

Class utilized in creation of red-black tree node with its data (key, value & color), parent & children linkages. Hold some of the help function to fetch & update.

main(String args[])

Main function to initialize red-black tree, read arg[0] build a red-black tree with it.

Input:

args[]: command line arguments

loadbstfromfile(String filename)

Function to load data from a file and initialize a red-black tree in O(n) time.

Input:

Filename: Filename with prelim re-black tree data.

Increase(int key, int value)

Function to find a node give its key (if not found create a key & insert it) and increments its count by specified value.

Input:

Key: key of the node to update

Value: value to be incremented.

Output:

Print the final incremented count of the node.

Reduce(int key, int value)

Function to find a node given its key and decrements its count by specified value (if the new value is less than or equal to 0 then delete the node).

Input:

Key: key of the node to update

Value: value to be decremented.

Output:

Print the final decremented count (0 in case the node is marked for deletion) of the node.

Count(int key)

Function to find a node given its key and decrements its count by specified value (if the new value is less than or equal to 0 then delete the node).

Input:

Key: key of the node to update

Value: value to be decremented.

Output:

Print the final decremented count (0 in case the node is marked for deletion) of the node.

InRange(int id1, int id2)

Function to return the sum of counts of different node corresponding to the range id1 to id2.

Input:

Id1: lower bound of the ID range

Id2: upper bound of the ID range

Output:

Print the final sum of the counts in the range of id1 & id2.

Next(int key)

Function to find the lowest key that is greater than the input key and print its key & value.

Input:

Key: key to search with

Output:

Print the key and the count of the event with the lowest key that is greater than the input key. Print “0 0”, if there is no next key.

Previous(int key)

Function to find the greatest key that is less than the input key and print its key & value.

Input:

Key: key to search with

Output:

Print the key and the count of the event with the greatest key that is less than the input key. Print "0 0", if there is no next key.

insertSorted(int[] key, int[] value, int start, int end, int count, int height)

Function to create a red-black tree give list of keys & values. This function creates a simple Red-black tree in O(n) time.

Input:

key: list of keys.

value: list of value.

start & end: the start & end of the sub-tree.

count: present height of the tree, 0 at the root and log(n) at the leafs.

height: the maximum value of the height a tree can grow when constructed evenly.

Output:

Returns the root node on complete creation of the tree.

insert(Node n)

Function to insert a node into the Red-black tree given the node object. Execution time O(logn).

insertFixup(Node n)

Function to fix-up the color structure of the Red-black tree in inserting a node.

transplant(Node u, Node v)

Function to swap node u with node v.

delete(Node n)

Function to delete a node from the given Red-black tree given the node object. Execution time O(logn).

deleteFixup(Node n)

Function to fix-up the color structure of the Red-black tree in deleting a node.

toString(Node node)

Function to print the sub-tree details given a node.

Input:

Node: root of the sub-tree to print

Output:

Print the sub-tree nodes in key, color, left-child & right-child

inorderRange(Node node, int id1, int id2, ArrayList<Integer> values)

Helper function to InRange(), helps in summing of the counts for range of IDs.

leftRotate(Node x)
Helper function to left-rotate the sub-tree along the node n. Runs at O(1).

rightRotate(Node x)
Helper function to right-rotate the sub-tree along the node n. Runs at O(1).

findSuccessor(int key)
Helper function to find the successor of a node given it key.

findPredecessor(int)
Helper function to find the predecessor of a node given it key.

findSuccessor(Node)
Helper function to find the successor of a node given the node.

findPredecessor(Node)
Helper function to find the predecessor of a node given the node.

min(Node n)
Helper function to find the minimum in the sub-tree under the node n.

max(Node n)
Helper function to find the maximum in the sub-tree under the node n

add(int key, int value)
Helper function to create and insert a new node into red-black tree given its key and count.

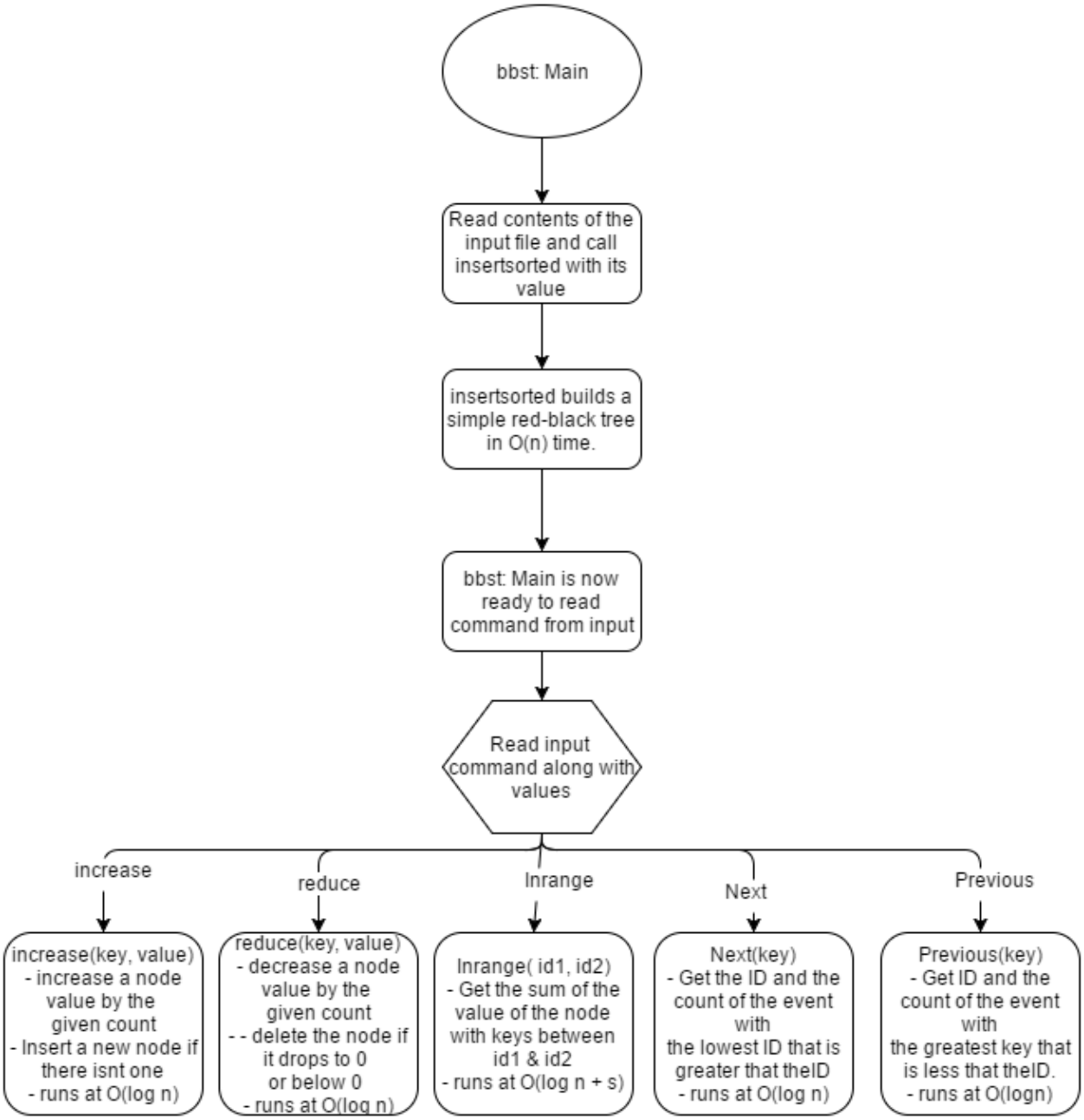
remove(int key)
Helper function to find and delete a node from red-black tree given its key.

search(int key)
Helper function to search for a node in red-black tree given its key.

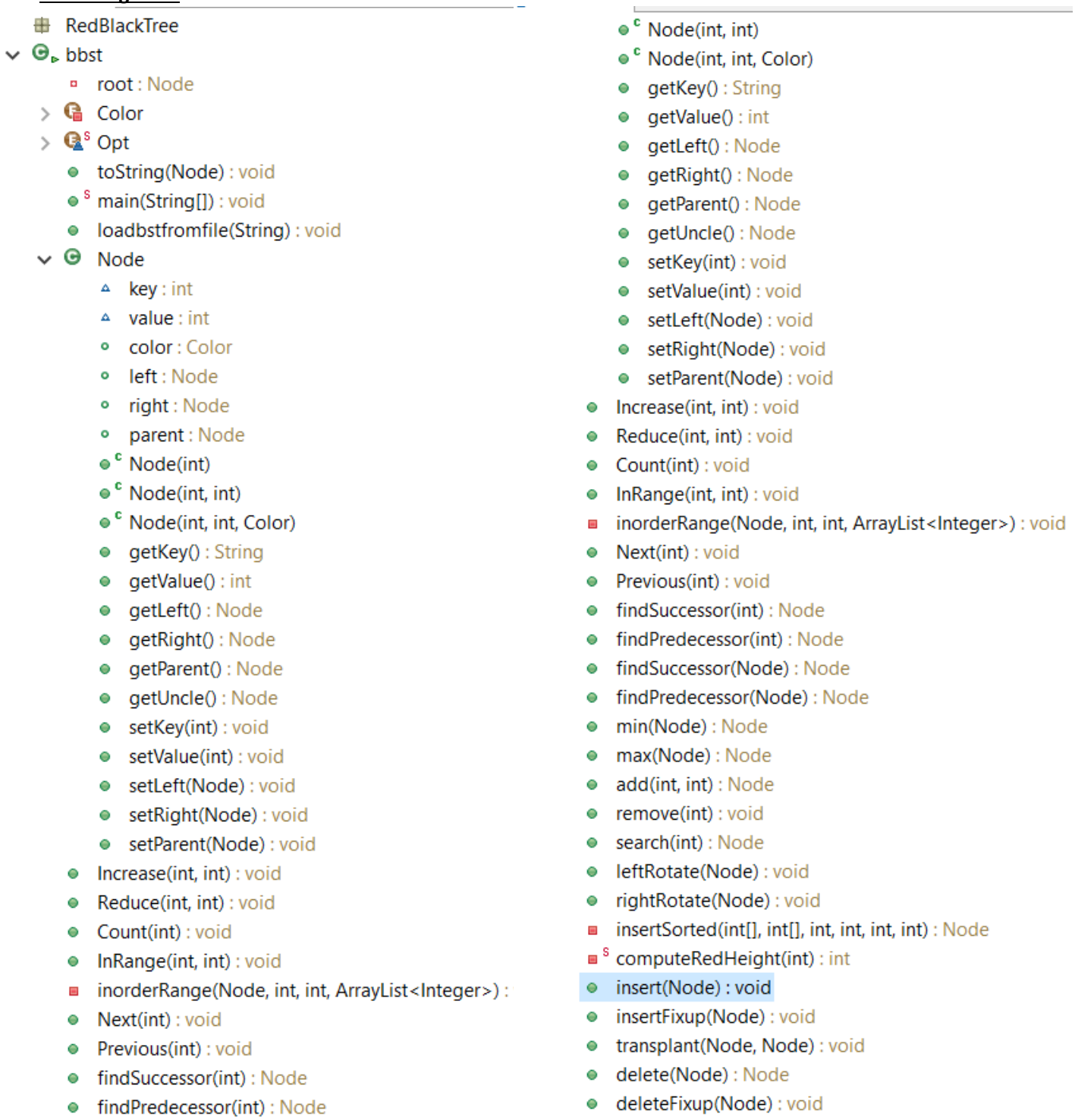
computeRedHeight(int n)
Helper function to compute the min-tree-height given the number of nodes.

All the function are declared public and most function have return type as void until unless it is very much required. InsertSorted is implemented to have a runtime of O(n). While insert, delete, left-rotate and right-rotate run at O(logn), O(logn), O(1) & O(1) respectively.

Code flow:-


bbst: Main
Read contents of the input file and call insertsorted with its value
insertsorted builds a simple red-black tree in O(n) time.
bbst: Main is now ready to read command from input
Read input command along with values
increase
reduce
Inrange
Next
Previous
increase(key, value)
- increase a node value by the given count
- Insert a new node if there isnt one
- runs at O(log n)
reduce(key, value)
- decrease a node value by the given count
- - delete the node if it drops to 0 or below 0
- runs at O(log n)
Inrange(id1, id2)
- Get the sum of the value of the node with keys between id1 & id2
- runs at O(log n + s)
Next(key)
- Get the ID and the count of the event with the lowest ID that is greater that theID
- runs at O(log n)
Previous(key)
- Get ID and the count of the event with the greatest key that is less that theID.
- runs at O(logn)

Class diagram:-



References:-

- Introduction to Algorithms, 3rd Edition (MIT Press) 3rd Edition by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
- Google.com
- Wikipedia.org
- http://www.cise.ufl.edu/~sahni/cop5536/