RB Tree: A Programmer's Guide

1. Overview:

This report briefly describes the architecture and layout of the code. It describes the API's exposed by bbst.java. The main () has the implementation of the application which uses these APIs to execute the commands read from standard input.

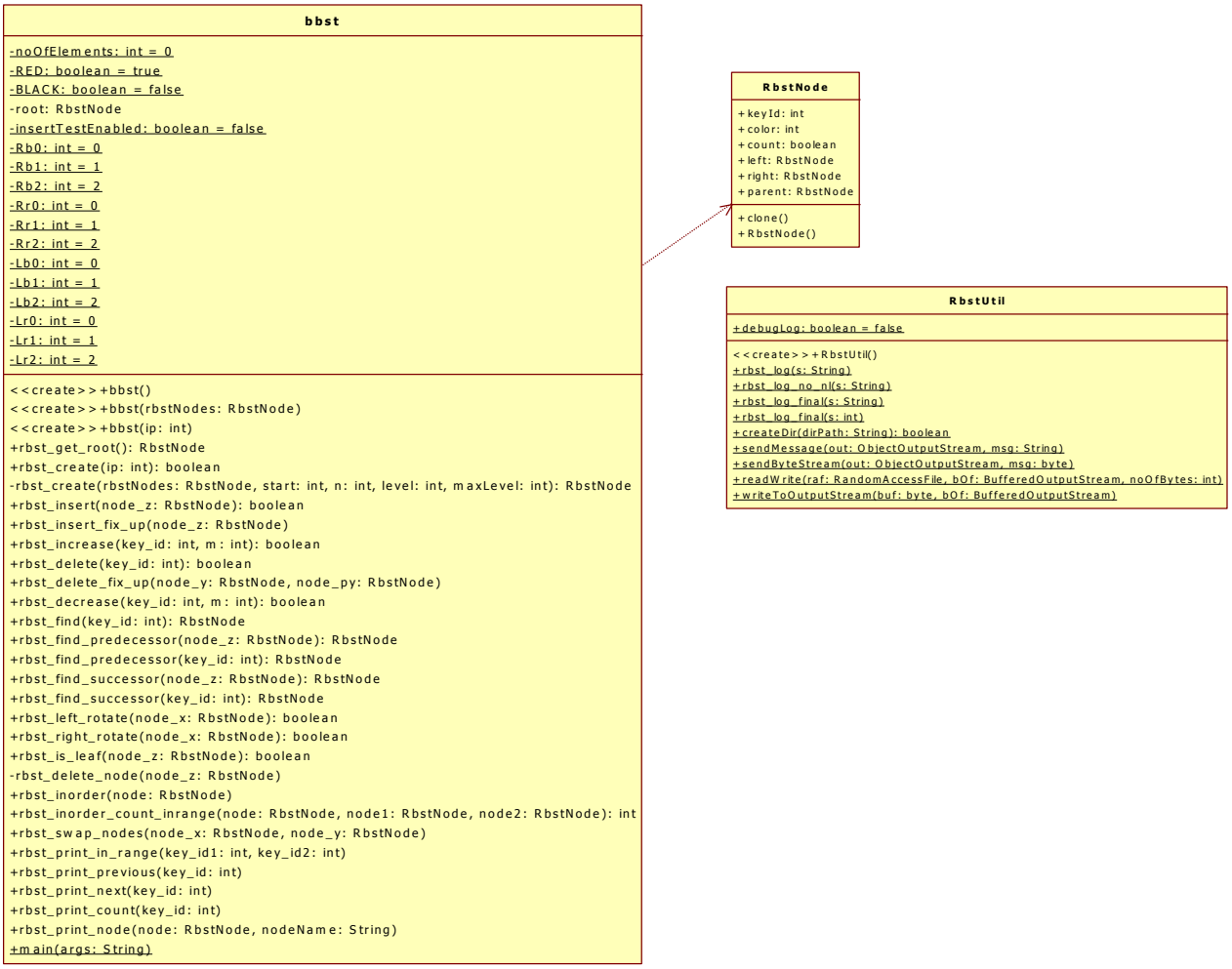

Figure 1: Class diagram of bbst.java and RbstUtil.java

2. Architecture:

The architecture is as depicted in the above class diagram. Bbst.java is the class which exposes all the APIs of an RBST. It has a subclass RbstNode.java. RbstUtil.java is the utility class which exposes utility functions meant to be used only by bbst.java to carry out secondary functionalities like logging etc.

2.1. How does it create an RB tree in O(n) time from a sorted array of ips:

It recursively creates the RB tree by taking the middle element as the root of the current subtree and the middle element of the subarray to its left as its left child and the middle element of the subarray towards its right as the right child. The middle element of the original array is the root. A node is colored as RED if it is found to be at the deepest level $\log_2 n + 1$, while doing the recursion. The resulting tree is a complete binary tree with all BLACK nodes except the nodes at the deepest level (level $\log_2 n + 1$).

3. Compiler Version

Following are the compiler versions in which the program has been tested:

java version "1.7.0_95"
OpenJDK Runtime Environment (IcedTea 2.6.4) (7u95-2.6.4-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.95-b01, mixed mode)

java version "1.8.0_74"
Java(TM) SE Runtime Environment (build 1.8.0_74-b02)
Java HotSpot(TM) 64-Bit Server VM (build 25.74-b02, mixed mode)

4. Bbst.java and its APIs:

This section describes the layout of the class bbst.java. It throws some light on its various methods followed by the description of the API's it exposes.

4.1. Constructors:

bbst() : This is the default constructor. It does not take any argument. If an object of this class is instantiated using this constructor then the user will have to call rbst_create() separately.

bbst(RbstNode []rbstNodes) : This constructor takes an array of type RbstNode (see class diagram). It creates an RB tree from the nodes contained in the array.

bbst(int []ip) : This constructor takes an array of integers. This integer array contains (ID, count) pair in locations (i, i+1), 0 <= i <= ip.length() -1. It creates an RB tree from this array.

4.2. Methods:

public boolean rbst_create(int []ip) :

access category	Public	Publicly accessible
arguments	int[] ip	contains (ID, count) pair in locations (i, i+1), 0 <= i <= ip.length() -2
output	Boolean	Returns true or false
Description: Creates an RB tree from ip[]. It uses rbst_insert() internally to insert a new node into the tree. It takes O(nlogn) time.		

private RbstNode rbst_create(RbstNode []rbstNodes, int start, int end, int level, int maxLevel)

access category	private	Not accessible to the user
arguments	RbstNode []rbstNodes	an array of type RbstNode (see class diagram)
	int start	Start index of rbstNodes[]
	int end	End index of rbstNodes[]
	int level	Current level in the tree
	int maxLevel	Maximum level of the tree (log2n + 1)
output	RbstNode	Returns a reference to the node which is supposed to be the root
Description: Recursively creates a RB tree from rbstNodes[]. See *2.1* above for details. It takes O(n) time.		

public boolean rbst_insert(RbstNode node)

access category	public	Publicly accessible
arguments	RbstNode node	Node to be inserted
output	boolean	Returns true or false
Description: Inserts node to the RB tree in O(logn) time.		

private void rbst_insert_fix_up(RbstNode node)

access category	private	Not accessible to the user
arguments	RbstNode node	Root of the subtree where the insertion has been done
output	void	Returns void
Description: Does readjustment after insertion of tree rooted at 'node'. readjustment is done from 'node' upwards till root node.		

public boolean rbst_delete(int key_id)

access category	public	Publicly accessible
arguments	int key_id	ID of the node to be deleted
output	boolean	Returns true or false
Description: Deletes node with ID ‘key_id’ in O(logn) time.		

private void rbst_delete_fix_up(RbstNode node_y, RbstNode node_py)

access category	private	Not accessible to the user
arguments	RbstNode node_y	Root of the deficient subtree
	RbstNode node_py	node_y’s parent
output	void	Returns void
Description: Removes deficiency of tree rooted at 'node_y'. Deficiency is removed through repeated readjustment done from 'node_y' towards root		

private RbstNode rbst_find(int key_id)

access category	private	Not accessible to the user
arguments	int key_id	ID of the node to search
output	RbstNode	Returns reference to the node if found otherwise reference to the leaf node which

		should be the parent of the node if inserted. Returns null if root is null.
Description: Performs binary search to find node with ID ‘key_id’. If the node is not found then it returns the leaf of node which should be the parent of the node containing the input 'key_id'. If root is null then it will return null.		

private RbstNode rbst_find_predecessor(int key_id)

access category	private	Not accessible to the user
arguments	int key_id	ID of the node whose predecessor is to be searched
output	RbstNode	Returns reference to the predecessor if found. Otherwise return null
Description: Finds the node with the greatest key less than the 'key_id'. Returns null if such a node is not found.		

private RbstNode rbst_find_successor(int key_id)

access category	private	Not accessible to the user
arguments	int key_id	ID of the node whose successor is to be searched
output	RbstNode	Returns reference to the successor if found. Otherwise return null
Description: Finds the node with the lowest key greater than the 'key_id'. Returns null if such a node is not found.		

private boolean rbst_left_rotate(RbstNode node_x)

access category	private	Not accessible to the user
arguments	RbstNode node_x	Root of the subtree based on which the rotation should happen.
output	boolean	Returns true or false
Description: Performs left-rotation of subtree rooted at 'node_x'.		

private boolean rbst_right_rotate(RbstNode node_x)

access category	private	Not accessible to the user
arguments	RbstNode node_x	Root of the subtree based on which the rotation should happen.
output	boolean	Returns true or false
Description: Performs right-rotation of subtree rooted at 'node_x'.		

4.3. API Description

public boolean rbst_increase(int key_id, int m)

access category	public	Publicly accessible
arguments	int key_id	ID of the node
	int m	The value by which the count has to be increased
output	boolean	Returns true or false
Description: Increases the count value of node with 'key_id' if found. Increases the value by 'm'. It inserts a new node if the node is not found.		

public boolean rbst_decrease(int key_id, int m)

access category	public	Publicly accessible
arguments	int key_id	ID of the node
	int m	The value by which the count has to be decreased
output	boolean	Returns true or false
Description: Decreases the count value if node with 'key_id' found. Decreases the value by 'm'. It deletes the node if count is <=0 after decrement. It prints 0 if node is not found.		

public void rbst_print_in_range(int key_id1, int key_id2)

access category	public	Publicly accessible
arguments	int key_id1	ID of the node where the range starts
	int key_id2	ID of the node where the range ends
output	void	Returns void
Description: Prints the count for IDs between 'key_id1' and 'key_id2'. Note that 'key_id1 < key_id2'. In the worst case it takes '4logn + s' or O(logn + s) time where n is the no of elements in the tree and s is the no of elements in the range.		

public void rbst_print_count(int key_id)

access category	public	Publicly accessible
arguments	int key_id	ID of the node whose count is required
output	void	Returns void
Description: Prints the 'count' of the node with id 'key_id'. Print 0 if not found.		

public void rbst_print_previous(int key_id)

access category	public	Publicly accessible
arguments	int key_id	ID of the node whose previous ID is required
output	void	Returns void
Description: Print the 'key_id' and the 'count' of the node with the greatest key less than the 'key_id'. Prints "0 0" if there is no such id.		

public void rbst_print_next(int key_id)

access category	public	Publicly accessible
arguments	int key_id	ID of the node whose next ID is required
output	void	Returns void
Description: Print the 'key_id' and the 'count' of the node with the lowest key greater than the 'key_id'. Prints "0 0" if there is no such id.		

5. RbstUtil.java

5.1. Constructor

RbstUtil() : Deafult constructor.

5.2. Methods

public static void rbst_log(String s)

access category	public	Publicly accessible
arguments	String s	String to be logged
output	void	Returns void
Description: Print String s to the standard output. It adds a end of line character after the string. Build time flag ‘debugLog’ decides whether this log will be enabled or not at run time.		

public static void rbst_log_no_nl(String s)

access category	public	Publicly accessible
arguments	String s	String to be logged
output	void	Returns void
Description: Print String s to the standard output. It does not add a end of line character after the string. Build time flag ‘debugLog’ decides whether this log will be enabled or not at run time.		

public static void rbst_log_final(String s)

access category	public	Publicly accessible
arguments	String s	String to be logged
output	void	Returns void
Description: Print String s to the standard output. It does not add a end of line character after the string. It prints irrespective of the stats of flag ‘debugLog’.		