NAME: Sunil Kumar

UFID: 94018098

Programming Assignment 1

The project is written in Java programming language. It uses the standard Java libraries and compiled using the Java compiler 1.8.0. The Project structure is as follows:

Structure:

There are three .java files in the project as described below:

- Node.java
- RBTree.java
- Bbst.java

Node.java:

This file contains the node class and has the properties:

- int id - it represents the ID of the event represented by the node
- int count - it represents the count of the event represented by the node
- int color - it represents whether the node is red or black. Red and Black nodes are represented by '1' and '0' respectively.
- Node parent - it points to the parent node of the given node.
- Node left - it points to the left child of the node
- Node right – it points to the parent node of the node

The methods and constructors in this class are:

- public Node() - Creates a with Nil node and sets default values for the properties.
- public Node(int[] arr) - Creates a Node from an array containing id and count values.
- public Node(int id, int count) - Creates a Node from the given id and count values.
- There are getter and setter methods defined for each property in the Node class.

RBTree.java:

This file contains the RBTreeclass and has the properties:

- Node root - pointing to the root node of the tree.
- Node nil – The T.Nil node

The methods are:

- public void rbTreeCreate(int[][] initial_array, int size) – Calls the rbInitialize() method to create the initial RB tree from a sorted array.

- public Node rbInitialize(Node root, int[][] initial_array, int low, int high, int height,int level) - Takes a sorted 2D-array of id's count's, precalculated height and start and end indices of the array as input and generates a Red Black Tree.
- public void increase(int id, int count) - Calls the insert() method to insert a new node or increase the count of existing node
- public Node insert(Node root, Node node) – Inserts a new node in the RB Tree or if the event id is already present then increases the count of the event by the given count.
- public void insertFix(Node z) - After a new insert this method re-orients the RB Tree to maintain the RB Tree properties.
- public Node searchNode(int id) – Takes the event id and returns the Node containing the event id.
- public void deleteNode(int id) – Calls the searchNode() method to search for the node containing the given event id and then calls the rbDelete() method to delete the node from the tree.
- public void rbDelete(Node z) – Deletes the given node and calls the deleteFix() method.
- public void deleteFix(Node x) – Fixes the tree after a is deleted to satisfy the RB Tree properties.
- public void rbTransplant(Node u, Node v) - Transplants the given node u with node v.
- public void leftRotate(Node x) - Left rotates the tree at node.
- public void rightRotate(Node x) - Right rotates the tree at node.
- public Node rbMinimum(Node x) – Returns the node with minimum ID in the tree.
- public Node rbMaximum(Node x) - Returns the node with maximum ID in the tree.
- public void inRange(int id1, int id2) - Calls the inRangeCount() to find the sum of count of all events in a given range.
- public int inRangeCount(Node root,int id1,int id2) - Returns the sum ofcount of all events in a given range.
- public Node searchMatch(int id) – Returns the Node having the given event id
- public void nextNode(int id) – Calls the searchMatch() and searchNext() methods to return the ID and the count of the event with the lowest ID that is greater than the given id.
- public Node searchNext(Node root, Node node) – Returns the next ID and the count of the event with the lowest ID that is greater than the given id.
- public void previousNode(int id) - Calls the searchMatch() and searchNext() methods to return the ID and the count of the event with the greatest key that is less than the given id.
- public Node searchPrevious(Node root, Node node) - Returns the ID and the count of the event with the greatest key that is less than the given id.

- public void reduce(int id, int count) – Reduces the count of id of given event and if the counts reduces to zero or less then deletes the node.
- public void countNode(int id) – Returns the count of id of given event.

bbst.java:

This is the main class containing the main function. It initializes a Red-Black Tree and calls the specific methods to perform the operations on the tree. To initialize the tree, the input is taken from a 2D array of events and then waits for the user input from Standard Console and writes to the Standard Console.

Control Flow:

- First the main file bbst.java is compiled and run, then the program reads the input from the input file specified in the command-line arguments and creates a 2D array containing the event count and event ID.
- Next the Red-Black tree is initialized from the sorted array by recursively selecting the middle event of the array as the root of the subtree.

Commands:

Increase - tree.increase(id, count) is called to increase the count of given event or insert the event if not found

Reduce - tree.reduce(id, count) is called to reduce the count of event and delete the event if count goes to zero

Inrange - tree.inRange(id1, id2) is called to return the sum of count of events within the given range.

Count - tree.countNode(id) is called to return the count of given event ID

Next - tree.nextNode(id) is called to return the ID and the count of the event with the lowest ID that is greater than the given id.

Previous - tree.previousNode(id) is called to return the ID and the count of the event with the greatest key that is less than the given id.

Quit - Quit the program execution.