ADS Programming Project Report

Compiler:

java version "1.8.0_71"   javac 1.8.0_71

For run other than 10^8 files: java bbst test_100.txt

For run 10^8 file: java -d64 -Xmx6144m bbst test_100000000.txt

Function Prototypes:

RedBlackTree.java contains the following functions and a Node class.

bbst.java contains the main method using an object of RedBlackTree class to call it's methods.

RedBlackTree.java

- Node initialize(int level, int f, int l, int[] K, int[] V)
- void increase(int theID, int m)
- void insert(int ID, int count)
- void insertFix(Node node)
- void reduce(int theID, int m)
- void remove(Node node)
- void deleteFix(Node node)
- Node binarySearch(Node node, int theID)
- void rotateLeft(Node node)
- void rotateRight(Node node)
- boolean colorOf(Node node)
- Node parentOf(Node node)
- void setColor(Node node, boolean f)
- Node leftOf(Node node)
- Node rightOf(Node node)
- void count(int theID)
- int inrange(int ID1,int ID2)
- void BuildList(Node node, int ID1, int ID2)
- void next(int theID)
- Node succ(int theID)
- void previous(int theID)
- Node prev(int theID)

Program structure: A to B arrow indicates that B function is called in function A.

Build red-black tree from a sorted list of n events – initialize. O(n)

Initialize function recursively builds the left and right subtrees of middle node. To satisfy RBT properties after BST tree is built the last level (not complete) nodes are colored red, rest all are by default colored black.

Increase the count of the event theID by m, if theID is not present, insert it – increase. O(log n)

Increase function calls binary search to find the node with theID. If present its count is increased by m and done. If not present then insert function is called to insert a new node with theID and count m. insert function in turn calls insertFix to maintain RBT properties after insert. Many helper functions are used by insertFix.

$$\begin{tikzcd} \text{increase} \arrow[r] & \text{binarySearch} \end{tikzcd}$$

$$\begin{tikzcd} \text{increase} \arrow[r] & \text{insert} \arrow[r] & \text{insertFix} \arrow[r] & \text{rotateLeft, rotateRight, colorOf,} \\ & & & \text{parentOf, setColor, leftOf, rightOf} \end{tikzcd}$$

Decrease the count of theID by m. If theID's count becomes less than or equal to 0, remove theID from the counter – reduce. O(log n)

reduce function calls binary search to find the node with theID. If not present then done. If present its count is decreased by m. If the newcount is less than or equal to zero then remove function is called to remove the node with theID. remove function in turn calls deleteFix to maintain RBT properties after delete. Many helper functions are used by deleteFix.

$$\begin{tikzcd} \text{reduce} \arrow[r] & \text{binarySearch} \end{tikzcd}$$

$$\begin{tikzcd} \text{reduce} \arrow[r] & \text{remove} \arrow[r] & \text{deleteFix} \arrow[r] & \text{rotateLeft, rotateRight, colorOf,} \\ & & & \text{parentOf, setColor, leftOf, rightOf} \end{tikzcd}$$

Count of theID – count. O(log n)

count function calls binarySearch to check if node is present or not. If present return count.

$$\begin{tikzcd} \text{count} \arrow[r] & \text{binarySearch} \end{tikzcd}$$

Total count for IDs between ID1 and ID2 inclusively – inrange. O(log n + s)

inrange function calls buildList to build a list with counts of nodes with ID in between the given range and finds the sum of these counts

$$\begin{tikzcd}\text{inrange} \arrow[r] & \text{BuildList}\end{tikzcd}$$

ID and the count of the event with the lowest ID that is greater that ID – next. O(log n)

next function calls succ function to find successor node with lowest ID greater than the ID given.

$$\begin{tikzcd}\text{next} \arrow[r] & \text{succ}\end{tikzcd}$$

ID and the count of the event with the greatest key that is less that ID – previous. O(log n)

previous function calls prev function to find predecessor node with greatest ID lower than the ID given.

$$\begin{tikzcd}\text{previous} \arrow[r] & \text{prev}\end{tikzcd}$$