COP 5536 Programming Project

An event counter c++ implementation using Red Black Tree

Submitted by : Varun Vyas
UFID : 11746731

Sections:

1. Compiler used

```
Apple LLVM version 7.0.2 (clang-700.1.81) 4.2.1
Target: x86_64-apple-darwin15.3.0
Thread model: posix
Basically g++ 4.2.1
```

2.Function Prototypes explained from RBTree:

a. List of functions:

```
private:
     void leftrotate(NODEPTR treeroot, NODEPTR x)
    void rightrotate(NODEPTR treeroot, NODEPTR y)
    void rbinsertfixup(NODEPTR treeroot, NODEPTR z)
    void rbtransplant(NODEPTR treeroot, NODEPTR u, NODEPTR v)
    void rbdeletefixup(NODEPTR treeroot, NODEPTR x)
public:
    // Constructor
    RBTree() { root = NULL }
    RBTree(NODEPTR _root) { root = _root }
    void inorder(NODEPTR x)
    NODEPTR search(NODEPTR root, int k)
    NODEPTR minimum(NODEPTR root)
    NODEPTR maximum(NODEPTR root)
    NODEPTR successor(NODEPTR root, int x)
    NODEPTR predecessor(NODEPTR root, int x)
    NODEPTR getClosestNode(NODEPTR pRoot, int value)
    NODEPTR sortedArrayToBST(vector<inputPairPTR> &num)
    void makeLeafRed(NODEPTR root)
    void rbinsert(NODEPTR treeroot, int z, int _count)
    void rbdelete(NODEPTR treeroot, int z)
    void Increase(int theID , int m , NODEPTR root )
    void Reduce( NODEPTR root,int theID , int m  )
    void Count(NODEPTR root , int theID )
```

    int InRange(NODEPTR root , int low , int high)

b. **List of typedefs:**
Few typedef used across the project are:-
typedef struct inputPair * inputPairPTR
typedef struct Node *NODEPTR

c. Method functionalities
All the implementations of RedBlack tree are from standard book CLRS.
For the project work I have added requested API functions which will be described below in details.

void Increase(int theID , int m , NODEPTR root)
Increase the count of the event theID by m for which we need to search in the tree. If theID is not present, insert it. Print the count of theID after the addition.
For doing insert we are using rbinsert function.
Time complexity is O(log n)

void Reduce(NODEPTR root,int theID , int m)
Decrease the count of theID by m. If theID's count becomes less than or equal to 0, removes theID from the counter. Print the count of theID after the deletion, or 0 if theID is removed or not present.
Once the count is decreased below 1 element is deleted using rbdelete which internally also uses rbdeletefixup. Implementation is similar to described in CLRS.
Time complexity is O(log n)

void Count(NODEPTR root , int theID)
Recursively searches theID in the redBlack Tree and then print the count of theID. If not present, it prints 0.
Time complexity is O(log n)

int InRange(NODEPTR root , int ID1 , int ID2)
Recursively search for IDs and returns the total count for IDs between ID1 and ID2 inclusively.
Time complexity is O(log n + s) where s is the number of IDs in the range.

NODEPTR successor(NODEPTR root, int x)
Search for the node with event with ID as x. If found then searches in normal successor way. Else if ID is not found it will find the closest ID using getClosestNode function. If the value returned is larger then key we are searching then return the value else search predecessor of value returned. Total time complexity is still O(log n).
Print the ID and the count of the event with the lowest ID that is greater that theID. Print "0 0", if there is no next ID.
Time complexity is O(log n)

NODEPTR predecessor(NODEPTR root, int x)
Search for the node with event with ID as x. If the node is found then find the normal predecessor i.e. maximum in left subtree or if no left subtree then walk up the tree to find the node where search turned to right. But ff ID is not found then find the closest nodes in tree. If the key of closest node found is lower then x then search its successor else return the value which is closest.
Print the ID and the count of the event with the greatest key that is less that theID. Print "0 0", if there is no previous ID.

Time complexity is O(log n)

3. Project file structure

There are four files in the project.

a. bbst.cpp —> Contains the main() function and accepts the commands
b. RedBlackTree.cpp —> Contains the actual implementation of Red Black tree and all the above described APIs used.
c. RedBlackTree.h —> Contains the function definitions and function prototypes with Node and typedef definitions.
d. makefile —> makefile to build the project.
    a. use 'make' to build the project
    b. use 'make cl' to clean the project's executable and object files.

bbst.cpp

Above file is the starting point of the project. Followings functions are performed by it

1. Parse the input file and extract the eventID and counts
2. **Initialise the red black tree in O(n) time.**
3. Instantiate a RBTree and run in infinite loop for accepting the input APIs.

RedBlackTree.cpp

1. Contains the main implementation of Red Black tree and associated APIs asked in the project like increase , next , previous etc .
2. All APIs are explained in above section.

4. Results obtained on Mac book pro 2013 i5 8GB.

```
MacBook-Pro:rbproject varunvyas$ time ./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
267 8
147 2

real  0m0.008s
```

```
user  0m0.002s
sys   0m0.003s



MacBook-Pro:rbproject varunvyas$ time ./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

real  0m1.792s
user  0m1.654s
sys   0m0.077s


MacBook-Pro:rbproject varunvyas$ time ./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

real  0m18.441s
user  0m17.080s
sys   0m0.701s


MacBook-Pro:Archive varunvyas$ time ./bbst ../test_100000000.txt < ../commands.txt
106
56
56
1344
168
1512
93
56
66
303 3
350 56
362 8
358 10
349 10
0 0
0
349 10
0
349 10
149 7
```

```
real	3m2.737s
user	2m36.875s
sys	0m15.093s
```