Algorithm Homework 3

Out 11/06/18

Due 11/20/18

This is a programming assignment. You may use Java, Python, C, C++, etc. While you may discuss ideas and problems with others, the final program must be done independently.

Overview

Let S be a dynamic set of integers. We wish to support the following operations on S:

- INSERT(S, x): Insert integer x into S.
- DELETE(S, x): Delete integer x from S.
- REPORT(S, *a, b):* Given integers *a, b* $\in$ S, where a < *b*, report all integers *c* $\in S$ such that a $\leq c \leq b$. For example, if S = {3, 7, 2, 9, 1, 6} and a = 2, *b = 7,* then report 2, 3, 6, 7.
- COUNT(S, *a, b):* Given integers *a, b* as in REPORT, return a count of the integers *c* $\in$ *S* such that $a \leq c \leq b$. In the above example, return the count 4.

The goal is to perform INSERT, DELETE and COUNT in O(log *n)* worst-case time and REPORT in *O(k* + log *n)* worst-case time, where *n* is the current size of S and *k* is the number of integers reported. The data structure should use *O(n)* space. In this assignment, you will solve the above problem by designing and implementing an augmented red-black tree for S. The assignment consists of two parts:

Part 1

In this part you must give a written description of (i) how the data structure is organized, (ii) how the operations are performed, and (iii) establish the correctness of the operations, and analyze the running time and space.
NOTE: An augmented red-black tree is really needed only for COUNT not for REPORT. However, since both queries should be handled by the same data structure you will need to use an augmented red-black tree. (Consider using just a "size" field.)

Part 2

In this part you will implement your solution from Part 1. Starting with an empty tree, you will read in a sequence of operations from the input file (see the next page). The input file consists of a sequence of **`Insert, Delete, Report, Count, Print`** and **`EndOfFile`** commands, one per line. These are abbreviated by single letters as **`I, D, R, C, P, E`**, respectively. In more detail:

- **`I k`** inserts key **`k`** into the tree. You may assume that **`k`** is not already in the tree.
- **`D k`** deletes key **`k`** from the tree. You may assume that **`k`** is already in the tree.
- **`R a b`** and **`C a b`** are the REPORT and COUNT queries discussed before.
- **`P`** asks that you print out the tree in *preorder.* The fields that need to be printed for each node are: **`key, color`** and any augmenting field(s) used. This lets us check your tree periodically.
- **`E`** indicates the end of the input.

What to turn in?

1. Turn in the written description of your solution for Part 1
2. Turn in the hard copy of your program including its output for Part 2.
   (Be sure to retain the electronic copy of your program till the end of the semester; it must be made available if requested.)

Notes

1. For convenience, you may assume that keys are always distinct.
2. Assume "perfect input," i.e., you do not have to check for errors in the input format.
3. The assignment will be graded not only for correctness and completeness, but also for style-your program must be well-structured and well-documented. Format the output so that it is readable. (However, you do not have to go to the extent of printing the tree out in tree format.)
4. This write-up has purposely been left open-ended. You are allowed a fair amount of flexibility in how you set up your program, so long as it conforms to the general guidelines stated here. However, for consistency (and to simplify grading!), you should follow the text's conventions wherever possible. For example, in TREE-DELETE, use TREE-SUCCESSOR (as the book does) rather than TREE-PREDECESSOR (which would also work but would produce a different tree).
5. Be sure to think through both parts (especially Part 1) carefully before you begin coding. And ... start early!

The input

```
I 20
I 29
I 38
I 27
I 12
I 23
I 25
I 21
I 17
I 15
P
R 17 27
C 17 27
R 21 29
C 21 29
D 25
I 25
I 22
P
R 21 29
C 21 29
R 17 21
C 17 21
D 20
D 25
D 27
D 38
D 29
P
R 12 22
C 12 22
R 17 21
C 17 21
E
```