BLG335E – Algorithm Analysis I 17.12.2019
2019–2020 Fall, Homework 3
Submission Deadline: 30.12.2019 23:59

Amazing Inc.

Overview

Elementary data structures such as hash tables or red-black trees are often not enough for solving problems. However, annotating them with additional information may permit new operations to be performed quickly. In such a case, the modified structure is called an augmented data structure.

In this homework you are going to augment red-black tree to solve an order statistic problem.

Problem

You are working for Amazing Inc., world leading e-commerce platform. In its warehouses, a robot called *shipment master* arranges shipments. You are programming the shipment master.

Definition

1 Warehouses / packages

a) A warehouse is where packages are stored. Amazing Inc. keeps ware in packages, and puts a label on each.

b) A label is a two-integer-paper, consisting of package size and an ordinal number. Labels are read by shipment masters.

c) Amazing Inc. has many warehouses, but two of which are really significant: West and East warehouses. Shipments are forwarded to smaller warehouses around the world from these two central warehouses.

d) Both warehouses have a shipment master.

e) A balloon courier called the lead courier goes between West and East warehouses.

2 The Lead Courier / Shipment Master

f) The lead courier works closely with shipment masters. It carries a shipment assigned by a shipment master to the other shipment master (e.g., packages given by the shipment master of the west warehouse is carried by the lead courier to the east warehouse, and vice versa).

g) A shipment is a single package **or** a collection of packages which are about to move.

h) The lead courier begins carrying from **the east** warehouse. The shipment master prepares the first shipment with **the smallest size package** and gives it to the lead courier.

i) Such a shipment is called **smallest size shipment** because it contains the smallest package or packages, if there are more than one package with the smallest size.

j) When the lead courier arrives at a warehouse, it hands in the shipment it was carrying. Shipment master reads the ordinal integer of the package, or **the minimum ordinal integer** of the packages if there are more than one package in the shipment.

k) Assume the ordinal integer is *n*, then shipment master prepares the lead courier a new shipment of the **nth smallest package/s** in the warehouse. Then, shipment master gives it to the lead courier, and unpacks the shipment which was brought by the lead courier and adds the brought packages to warehouse inventory.

l) However, sometimes a warehose **may not have nth smallest package**. (One case is that the count of packages in a warehouse is less than *n.)*

m) When a warehouse does not have nth smallest package, shipment master does not accept this shipment to the warehouse, instead the shipment is forwarded to smaller warehouses around the globe. And it prepares **the smallest size shipment** and gives it to the lead courier, such that courier keeps operating.

n) A warehouse gets completely empty by the time its shipment master gives the last possible shipment to the lead courier. The lead courier carries this last shipment to the other warehouse, and shipment master of the other warehouse directly accepts the shipment without processing it. Packages inside this shipment is added directly to the warehouse inventory and **the whole shipment process ends.**

3 *Your Goal*

o) Since Amazing Inc. serves to millions, shipment masters should be very efficient at finding packages, **especially nth smallest packages,** as well as updating warehouse inventories by insertions and deletions. Therefore as the engineer behind the shipment masters, you decided to program memory of a shipment master as a order statistic tree, an augmented red-black tree whose nodes are associated with packages.

p) Alternatively you would program shipment masters by keeping a few sorted arrays for anything, however this would turn out to be inefficient. This is the motivation behind your decision about using red-black trees.

Implementation

```
HW3_Result hw3(  int eastWarehousePackageCount,
                 int eastWarehousePackageSizes [],
                 int eastWarehousePackageOrdinals [],
                 int westWarehousePackageCount,
                 int westWarehousePackageSizes [],
                 int westWarehousePackageOrdinals [] );
```

You will be knowing:

- `int eastWarehousePackageCount` is the number of packages in the east warehouse
- `int eastWarehousePackageSizes []` is the sizes of packages in the east warehouse
- `int eastWarehousePackageOrdinals []` is the ordinal numbers of packages in the east warehouse
- `int westWarehousePackageCount` is the number of packages in the west warehouse
- `int westWarehouseSizes []` is the sizes of packages in the west warehouse
- `int westWarehouseOrdinals []` is the ordinal numbers of packages in the west warehouse
- `HW3_Result` is a three integer tuple for package count, red node count and black node count after **the whole shipment process ends.**

```
struct HW3_Result {
        int packageCount;       //40 pts
        int redNodeCount;       //30 pts
        int blackNodeCount;     //30 pts
}
```

- `int packageCount` is the total number of packages in whatever warehouse remains
- `int redNodeCount` is the total number of red nodes
- `int blackNodeCount` is the total number of black nodes

  You **must not** declare a main function anywhere in your implementation.

Evaluation

You will be given a header: *hw3.h*

You will be submitting a source: *hw3.cpp*

Your homework will be evaluated in terms of your implementation. Your program should be compilable on ITU's Linux Server by the following command. Yes, you are allowed to use C++11 features in this homework if you wish!

```
>g++ -std=c++0x -Wall -Wextra -Werror hw3.cpp ${OUR_SOURCES} -o hw3
```

Table 1: An example how we run your program

```
>ls
hw3.cpp  hw3.h  ourSources/
>g++ -std=c++0x -Wall -Wextra -Werror hw3.cpp ${OUR_SOURCES} -o hw3
>./hw3
```

q) We will compile your implementation in *hw3.cpp* beside our sources. It is us who will provide **the main function**.

r) Before submission, you can work with a main function while implementing hw3.

s) You are not allowed to include **GNU extension for policy based trees**, which includes red-black tree implementation.

t) Neither do you allow exploit **std::map**, which is based on red-black tree.

u) You are not allowed to include **<algorithm>** any way either.

v) If you do not follow these in your implementation, your code will not be graded even if **it compiles** or **produces correct output**. You need to implement your own augmented red-black tree and use that tree in *hw3* function. You are free to consult to course slides or the reference book for help about augmented data structures.

w) Your homework will be evaluated by using **black-box techniques**.

Policy

- You may discuss the problem addressed by the homework at an abstract level with your classmates, but you should not share or copy code from your classmates or from the Internet. You should submit your own, individual homework.
- Academic dishonesty including but not limited to cheating, plagiarism, collaboration is

unacceptable and subject to disciplinary actions. You are expected to act according to Student Code of Conduct, which forbids all ways of cheating and plagiarism. **Your codes will be checked using plagiarism detection software. In case of cheating occurs, you will be subject to disciplinary actions.**

Submission

- Please submit your files through Ninova e-Learning System.
- You must **only** submit hw3.*cpp*. Do **not** send us back *hw3.h* or anything else.
- All your implementation must be in C++, and we must be able to compile and run it on ITU's Linux Server (you can access it through SSH).
- For Windows users: If you wish, you can use WinSCP to upload and edit your source code into ITU SSH Server, and use PuTTY to compile and run your algorithm. If does not, please make sure that your code is able to **be compiled** and runned on ITU's Linux Server.
- Your code **has to be compiled successfully**. Otherwise you will not be graded. Which will result a grade of **zero** for the homework.
- You should be aware that the Ninova e-Learning System clock may not be synchronized with your computer, watch, or cell phone. Do not e-mail the teaching assistant or the instructors about your submission after the Ninova site is closed for submission. If you have submitted to Ninova once and still want changes in your report, you should do this before the Ninova submission system closes. Your changes will not be accepted by e-mail. Connectivity problems in the last minute about the internet or about Ninova are not valid excuses for being unable to submit. You should not risk your project by leaving its submission to the last minute. After uploading to Ninova, make sure that your homework appears there.

START EARLY.

If a point is not clear, discuss it or e-mail kcengiz@itu.edu.tr