COL106: Data Structures and Algorithms

Assignment 3

Instructor: Prof. Saroj Kaushik *Deadline: 18/03/2018 11:55pm*

1 Statement

1.1 Preamble

Having worked on a few projects for small tech companies, you plan to try out your luck in the financial markets. Being a smart person that you are, you wish to use machines to make quick trading decisions for you. You believe that data driven computing is the key to succeed in such financial markets. Inspired from the HFT (High Frequency Trading) ideology, you are ambitious about building some computing facility of your own that will help you earn profits in all market conditions. But before you can process or analyze any live data, you need a way to quickly get data from the stock exchange to your compute servers. In such cases, where firms use algorithms to make quick trading decisions, even milliseconds are of much importance and so you need to optimize the time to get data over the network (from the exchange to your servers). Ideally you would want your compute servers to be located right beside the stock exchange so that the network transmission time of the data is minimized. This, however is not possible in a real world scenario. Luckily you have a friend Merlin, who stays very close to the exchange. Merlin has allowed you to run a small setup at his residence but this won't really be sufficient for the computational power you need. So you plan to use this setup to get data to you as fast as possible.

1.2 Statement

In this assignment, we will together build a Market Data Publisher (MDP). This is a component that receives price data for various symbols and broadcasts them (prints on console for our simple system) as per certain rules. A naive publisher would simply publish any data that it gets instantly, however assume that we are operating in a limited bandwidth environment, and do not want to send out price changes unless they exceed some threshold (θ).

Rule: Do not send out any price unless

$$|newPrice - lastPublishedPrice| > threshold$$

(We will skip some other rules that a MDP might need for brevity of the assignment)

You are required to implement an MDP using an AVL Tree. The stocks will be described as a pair (company-id, stock-price). Since the stock exchange exists even before you created your MDP, you first have to construct an AVL tree out of the existing stocks. Your construction of AVL tree must take no more than O(n) time, instead of the usual O(n.log(n)) required when inserting values 1-by-1. This creates the need of pre-sorting the stock values on their company-id. Thus, your assignment gets divided into the following sub-tasks:

- Implement merge sort algorithm for pre-sorting the stocks
- Construct an AVL tree from these sorted key value pairs
- Support the following operations:
  - **register-company** <company-id> <initial-price>: Insert a new node in the AVL Tree with the given company-id and price. Print the company-id and the stock price separated by a space(see output format).

- **deregister-company** <company-id>: Remove all data of the company with the given company-id.
- **update-price** <company-id> <new-price>: You must update the price of the given company-id, if the above given rule is satisfied. Also in case the rule is satisfied, you must broadcast (print) the current price. Ignore operation if given stock does not exist.
- **stock-split** <company-id> <x:y>: Update the price of the company-id given that its stocks are split in the ratio $x - for - y$. (Note: This operation also covers reverse stock split, i.e. it is possible that x < y.) Publish (print) the updated stock price. Ignore operation if given stock does not exist.

2 Input Format

- All IO operations are to be done using standard input/output.
- The first line of the input contains a single integer n - the no. of stocks to be bulk loaded.
- Following n lines contains 2 space separated integers - $comapny - id$ and $initial - price$ - representing the n initial stocks.
- Next line has 2 space separated integers: N - the no. of subsequent operations to be performed on the AVL Tree and T - the threshold parameter of the MDP (see rule above).
- Next N lines consist of 1 operation each in the following format: <operation-code> [params].

Code	Operation
1	register-company
2	deregister-company
3	update-price
4	stock-split

3 Output Format

- Each line in output will have exactly 2 space separated integers: <company-id> <stock-price>
- Operation 1 outputs the initial price of the stock
- There is no output for Operation 2.
- Operation 3 may or may not publish the new price of the stock based on the condition $|newPrice - lastPublishedPrice| > threshold$.
- Operation 4 always publishes the modified price.

4 Limits

- $0 \leq n, T \leq 10^6$
- $1 \leq N \leq 10^6$
- Every parameter of each operations is an integer in the range $[1, 10^9]$
- The value of every stock will always be an integer in the range $[1, 10^9]$.

5 Sample IO

5.1 Input

```
5
11 10
5 4
72 91
35 42
12 12
10 5
1 100 100
3 12 20
3 12 21
3 11 31
3 12 15
3 12 14
4 100 2:1
2 100
3 100 70
3 11 30
```

5.2 Output

```
100 100
12 20
11 31
12 14
100 50
```

6 Submission Instructions

- You are required strictly follow the instructions given below.
- You must submit a single program file which must be named with your *KerberosID.cpp* in small case. For example, if your kerberos id is cs1150999 then the file name should be cs1150999.cpp. WARNING: Do not submit any other file formats just renaming them to cpp.
- **You will be penalized if your submission does not conform to this requirement**.

7 Other Instructions

- The assignment is to be done individually.
- All submissions must use C/C++ for the programming assignment.
- You are **not** allowed to used any standard implementations/libraries (except for IO) in any part of the program.
- You are not allowed to discuss or take help from anybody outside the class. You may only discuss but are not allowed to share code with anyone inside the class.
- We will run plagiarism detection on your submissions. People found guilty will be penalised as per the instructions mentioned in the beginning of the course