CSE 2010, HW4

Due Thu Mar 7 at the start of your lab section; Submit Server: class = cse2010, assignment = hw4S*x*Individual

Due Thu Mar 7 at the end of your lab section; Submit Server: class = cse2010, assignment = hw4S*x*GroupHelp

x is merged section: 14, 23

Bitcoin, or other kinds of cryptocurrency, has been frequently in the news. Besides the technology around Bitcoin, the price of a Bitcoin has been fluctuating in large amounts on different exchanges. How would you design an exchange system to match buyers and sellers of Bitcoins?

The goal of HW4 is to design an exchange system that can efficiently match buyers and sellers of "Fitcoins." The system allows users to enter orders. Each buy/sell order consists of a buyer/seller, a price, a quantity, and a timestamp. A buy order and a sell order are executed if they have the same price. During execution, if the buy (or sell) quantity is larger, the buy (or sell) order is updated with the remaining buy (or sell) quantity; however the timestamp remains the same. If two buy (or sell) orders have the same price, the earlier order has a higher priority. In the unlikely case that the buy order has a higher price than the sell order, the orders are executed at the average of the buy and sell prices. To manage and match buy and sell orders efficiently, use two *heap-based* priority queues: one for the sellers and one for the buyers.

To implement the priority queue, you may borrow/modify Code Fragment 9.8 on pp. 377-378 in Goodrich et al. We will evaluate your submissions on code01.fit.edu so we strongly recommend you to test your programs on code01.fit.edu. To preserve invisible characters, we strongly recommend you to download, NOT copy and paste, input data files.

Input: The command-line argument for HW4.java is the name of the input file, which has:

- EnterBuyOrder *time buyer price quantity*
- EnterSellOrder *time seller price quantity*
- DisplayHighestBuyOrder *time*
- DisplayLowestSellOrder *time*

Time is an integer in HHMMSS format, where HH is 00-23 and MM/SS is 00-59 (leading zeros are optional). Sample input files are Canvas.

Output: Output goes to the standard output (screen), each line corresponds to an action:

- EnterBuyOrder *time buyer price quantity*
- EnterSellOrder *time seller price quantity*
- DisplayHighestBuyOrder *time buyer orderTime price quantity*
- DisplayLowestSellOrder *time seller orderTime price quantity*
- ExecuteBuySellOrders *price quatity*
  Buyer: *buyer remainingBuyQuantity*
  Seller: *seller remainingSellQuantity*

Sample output is on the course website.

Extra Credit (10 pts): Separate submission via HW4Extra.java. Consider the condition of the buyer and sellers can change their orders. When an order is changed, the timestamp for the order is updated. For simplicity, each user can have only one order and the timestamps are unique. Additional possible input action is:

- ChangeBuyOrder *time buyer price quantity*
- ChangeSellOrder *time seller price quantity*
- CancelBuyOrder *time buyer*
- CancelSellOrder *time seller*

and output result is:

- ChangeBuyOrder *time buyer price quantity* [noBuyerError]
- ChangeSellOrder *time seller price quantity* [noSellerError]
- CancelBuyOrder *time buyer* [noBuyerError]
- CancelSellOrder *time seller* [noSellerError]

Although the priority queue is designed to find the lowest/highest price quickly, it is not designed to find a buyer/seller quickly—faster than $O(N)$, where N is the number of buyers/sellers.

1. Design and implement an additional data structure that can help find a buyer/seller and update the priority queue faster than $O(N)$.
2. Note that changeBuy/SellOrder might not udpate the entry with the lowest/highest price in the priority queue.
3. In the comments at the top of your program (or in a separate PDF file):
   - explain why your additional data structure can help changeBuy/SellOrder become faster than $O(N)$ with an analysis of the time complexity.
   - when changeBuy/SellOrder does not remove the entry of lowest/highest price, discuss how the heap (priority queue) can be adjusted in $O(\log N)$ time.

[It's OK if EnterBuy/SellOrder may become slower than $O(\log N)$ due to the additional data structure, which can be remedied with data structures to be discussed later in the course.]

Submission: Submit HW4.java that has the main method and other program files. Submissions for Individual and GroupHelp have the same guidelines as HW1. Note the late penalty on the syllabus if you submit after the due date and time as specified at the top of the assignment.

GroupHelp and late submissions are not applicable for Extra Credit.