Quantitative Developer

Technical Assignment

# **Introduction**

In this assignment, the goal is to build a prototype of an exchange, with its basic functionality. In the essence, an exchange has the capabilities of:

* Receiving messages from the actors of the market (market makers and aggressors)
* Keeping an up-to-date LOB (level order book)
* Matching (when possible) buyers & sellers and execute trades

You’re asked to implement a simple framework with 4 components.

# **Components**

## Messages

Exchange should expect 3 messages (Add, Delete, Modify). Your goal is to find a suitable data structure to send necessary order messages to exchange. Messages should contain following fields:

### Add Message:

This is a message from a market actor to place a new order and its required fields are:

* Side (Buy or Sell)
* Price
* Amount

### Delete Message:

This is a message from a market actor to delete an existing order and its required fields are:

* Order id

### Modify Message:

This is a message from a market actor to modify the amount of an existing order and its required fields are:

* Order id
* New amount

## Orders

Exchange should keep all distinct orders at individual Order objects. Receipt of an Add message should generate a new Order object and assign a unique Order id to it. Receipt of a Delete and Modify message should access the respective Order object (identified with its Order id) and either delete or modify it. Please note that a modify message with a higher amount than the original would cause losing the priority of the order.

## Order Book

Exchange should build an LOB, based on price & priority (FIFO). You’re asked to use a suitable data structure that contains Order objects. Order book should allow users to access all orders at a given price level & side; ordered by their priorities. Order with an earlier arrival has a priority over a later order at the same price level.

Example Order Book:

In the below exemplary order book, Amount 0 corresponds to the orders with the earliest arrival and hence the priority in case of an execution.

| Amount n | Amount 2 | Amount 1 | Amount 0 | Bid Price | Ask Price | Amount 0 | Amount 1 | Amount 2 | Amount n |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| … | 30 | 20 | 40 | 9 | 10 | 5 | 100 | 70 | … |
| … | 20 | 20 | 30 | 8 | 11 | 40 | 50 | 30 | … |
| … | 5 | 50 | 50 | 7 | 12 | 20 | 10 | 60 | … |
|  |  |  |  | … | … |  |  |  |  |

## Matching Engine

Matching engine should execute any matching trades and update the LOB. After each message (when processing and LOB update is finalized), the matching engine should observe the LOB and take trade execution actions; if it exists.

Example Execution:

Using the same exemplary order book above, if a new sell order at price of 9 and amount of 55 arrives to exchange, the matching engine should fully fill the bid at 9 with amount of 40 (at Amount 0) and partially fill the bid at Amount 1 (15 is filled and 5 is remaining). The Order object should then be modified to contain the correct amount after the partial fill.

# Tasks

1. Implement a simplistic framework, containing above components with an object-oriented programming language of your choice. Please note that efficiency & performance of your code will be evaluated.
2. Write a script that generates dummy messages and sends these messages to your exchange. There is no need to use real-market data and therefore you can make an imaginary market.
3. Assess the performance of your code theoretically (for each distinct message received and for executions), by means of Big O notation. Comment on the bottlenecks on each cycle.
4. Stress-test your framework with a heavily liquid order book. You may assume 100 price levels at both sides with 2000 orders at each level. Measure and compare processing time of different messages on some random orders.

# Bonus Task

On your liquid order book, implement a function to calculate the equilibrium mid-market price (EP) which is the equilibrium price of cumulative discounted total volume functions of bid and ask side; given the parameter of half-life.

EP is defined as follows:

## Initial conditions:

## Exponential Decay function:

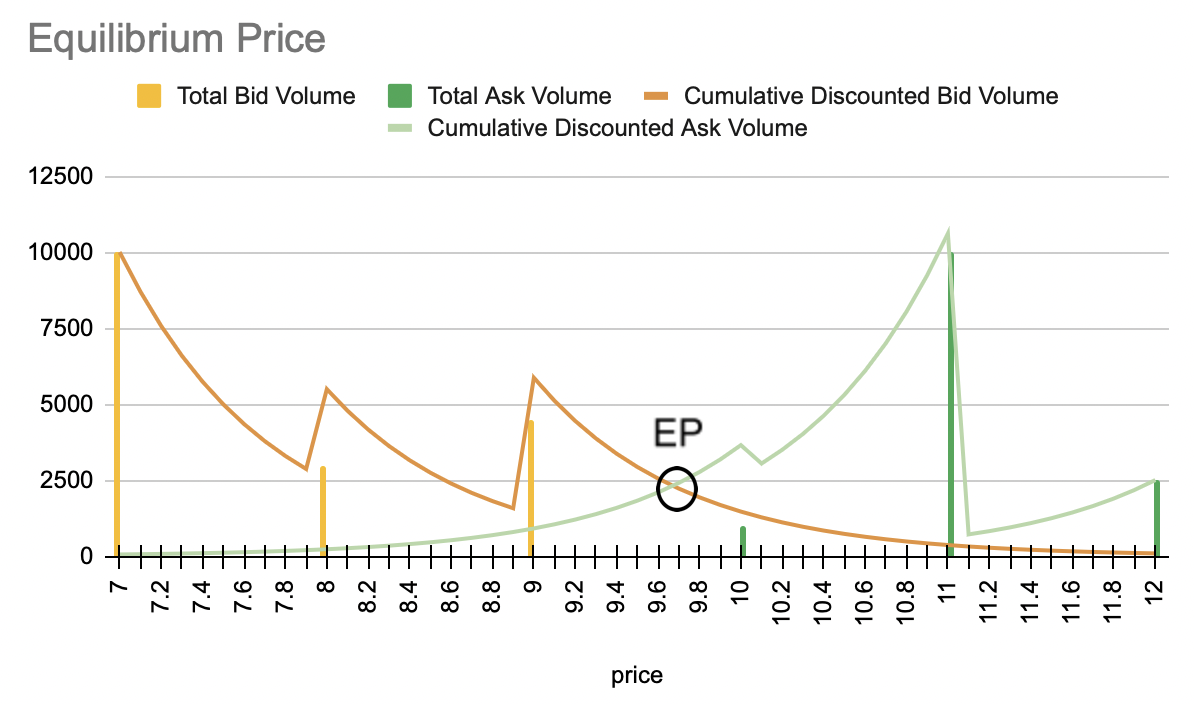
## Example decay function outputs:

## Example EP Calculation:

For an example cumulative book of the following:

| **Total Bid Amount** | **Bid Price** | **Ask Price** | **Total Ask Amount** |
| --- | --- | --- | --- |
| 4500 | 9 | 10 | 1000 |
| 3000 | 8 | 11 | 10000 |
| 10000 | 7 | 12 | 2500 |

EP is found as follows:



Note: Above plots are drawn with half life of 0.5. Selection of is only relevant for the precision of EP and 0.1 is used as an example.

Good luck!