

CSCI 2100: Project (2025)

Prepared by Yufei Tao

The goal of this project is to implement a Real-Time Stock Market Data Tracker.

For each stock, you should store three fields:

- **ID**: an integer uniquely assigned to the stock;
- **price**: a positive float number representing the current price of the stock;
- **vol**: a positive integer representing the trading volume of the stock.

Your system should support the following operations:

- **insert-new-stock**(x, p): insert a new stock with an ID x whose current price is p . You should first check whether a stock with the same id already exists; if so, ignore the operation. Otherwise, create the stock and set its volume to 0.
- **update-price**(x, p): update the price of the stock with ID x to p . You should first check whether a stock with ID x indeed exists; if not, ignore the operation.
- **increase-volume**(x, v_{inc}): increase the volume of the stock with ID x by v_{inc} . You should first check whether a stock with ID x indeed exists; if not, ignore the operation.
- **lookup-by-id**(x): find the price and volume of the stock with ID x if such a stock exists.
- **price-range**(p_1, p_2): return the IDs of all the stocks whose prices are in the interval $[p_1, p_2]$.
- **max-vol**: return the highest volume among all the stocks and one stock having this volume.

If n is the number of stocks in the system currently, then your implementation should have the following guarantees:

- **insert-new-stock**: $O(\log n)$ time.
- **update-price**: $O(\log n)$ time.
- **increase-volume**: $O(\log n)$ time.
- **lookup-by-id**: $O(1)$ expected time.
- **price-range**: $O((1 + k) \cdot \log n)$ where k is the number of IDs reported.
- **max-vol**: $O(\log n)$ time.

Programming Language. You can use C++ (including variants like C, C#, ...), Java, or Python. There are no constraints in the libraries you can use. You are also allowed to use any AI tools (e.g., ChatGPT and Copilot).

Deliverables.

1. Source code.

2. A report, which is a pdf document that explains
 - how you achieve the required time guarantees, detailing the data structures and algorithms deployed;
 - how the above data structures and algorithms are implemented in your source code.
3. A file containing a list of operations that can be used to test your source code. The list should satisfy all the following requirements:
 - It should start with 10000 **insert-new-stock** operations to insert 10000 different stocks into the system. The ID of each stock should be randomly generated in the domain from 1 to 1000000.
 - Each of the 10000 stocks must have its price updated at least once. The new price of a stock should be generated randomly in the range from 1 to 100.
 - Each of the 10000 stocks must have its volume updated at least once. The parameter v_{inc} of each operation should be generated randomly from 1 to 100.
 - After every 1000 **update-price** operations, you should perform a **price-range**($p, p + 2$) operation where p is the price specified in the last **update-price** (among the preceding 1000 **update-price** operations).
 - After every 100 **increase-volume** operations, you should perform a **max-vol** operation.
 - The operation list should end with 10000 **lookup-by-id** operations where every stock has its ID looked up once.

The total number of operations in your list should be at least 40110.

4. A “readme” file explaining your program can be compiled and tested.