Problem Solution - ALG2

Problem Definition:

We are given a array of price predictions for m stocks for n consecutive days. The price of stock i for day j is A[i][j] for i = 1,...,m and j = 1,...,n. You are tasked with finding the maximum possible profit by buying and selling stocks. The predicted price at any day will always be a non-negative integer. You can hold only one share of one stock at a time. You are allowed to buy a stock on the same day you sell another stock. More formally,

Problem 1: Given a matrix A of m × n integers (non-negative) representing the predicted prices of m stocks for n days, find a single transaction (buy and sell) that gives maximum profit.

Problem 2: Given a matrix A of m × n integers (non-negative) representing the predicted prices of m stocks for n days and an integer k (positive), find a sequence of at most k transactions that gives maximum profit. [Hint :- Try to solve for k = 2 first and then expand that solution.]

Task:

Algorithm2: Design a $\Theta(m * n)$ time greedy algorithm for solving Problem 1.

Solution:

For the greedy algorithm, the approach is to iterate over all the stocks and for each stock, finding the minimum price of the stock and subtracting it to all the prices of the next days. Thus we find the transaction with maximum profit for each stocks and find out the maximum profit transaction in the process.

So for m stocks for n days, for each stock 1 to m, the algorithm goes from j = 1 to n, keeps a record of the minimum price from 1 to j and also keeps record of the maximum profit from days 1 to j. Since, a stock can be sold only after the day its bought, the minimum price must be subtracted from the prices after that day only. So, the algorithm greedily keeps a record of the maximum possible profit while going over each day for every stock.

Proof of correctness:

-

Time complexity analysis:

Time complexity for the given task can be calculated as follows: First iterating over all the stocks i.e. m. Then the algorithm goes from j = 1 to n and finds the minimum price and in turn

finds the maximum possible profit from that stock. So, it finds the maximum profit from all the stocks and finds the maximum out of those transaction giving out the maximum profit and the transaction.

So the time complexity for this algorithm will be: **O (m * n)**

Space Complexity analysis:

The algorithm stores the stocks and their prices in a 2d array of size m*n. It also stores the least price, profit, buy and sell as a integer.

So the space complexity for this algorithm will be: O (m * n + c) = **O (m * n)**

[Here c is a constant]

Pseudo Code:

Step 1 : Input A[m][n]

Step 2 : profit = 0, stock = 0, buy = -1, sell = -1, leastPriceSoFar = INT_MAX, minIndex = -1

Step 3 : For I = 1 to m :

```
    For j = 1 to n:
        if leastPriceSoFar > A[I][j]
            leastPriceSoFar = A[i][j]
            minIndex = i
        if profit < (A[i][j] - leastPriceSoFar)
            profit = A[i][j] - leastPriceSoFar
            buy = minIndex
            sell = i
```

Step 4: Return {Profit, stock, buy, sell}