Best Time to Buy and Sell Stock → DP

Logic:

We want the max profit that we can achieve.

→ use dynamic programming again.

table / approach =

i = current day

j = next day

i = j = length of the prices list

i \ j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	4	2	5	3
3	0	0	0	0	0	1	0
4	0	0	0	0	0	3	1
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

From the table, we will then look for the largest number which we know is 5 which is basically the profit that we were able to earn technically. The table will then produce the profit that helps us to answer the Q.

prices = [7, 1, 5, 3, 6, 4] (1 2 3 4 5 6)

profit = 5

prices = [7, 6, 4, 3, 1]

profit = 0 → seems like when the list is in descending order, the profit will naturally be 0 because we are not gaining any kind of profit. The values after the day 1 will keep on deteriorating from then and then it keeps going downward from then.

if prices[i] > prices[j]

put 0

(because profit is in negative which means no profit technically)

if prices[i] < prices[j]

dp[i][j] = prices[j] - prices[i]

if a neg value is obtained, put 0.

if (i == j)

dp[i][j] = 0 → Because no profit if we are coming the same day basically

To do this properly, we will have our prices list with our two pointers called i and j. These pointers will allow us to fill our table. The recurrence relation is as such =

$$T(n) = \begin{cases} dp[i][j] = 0, & \text{if } i == j \text{ (no profit on the same day)} \\ dp[i][j] = 0, & \text{if } prices[i] > prices[j] \\ dp[i][j] = prices[j] - prices[i], & \text{given that } prices[i] < prices[j] \end{cases}$$

The time complexity for this question will be $O(n^2)$. The potential is the memory overhead. So instead of this, we could have a variable to keep track of our maxValue. This will make it completed by $O(n)$ time.