LeetCode – Best Time to Buy and Sell Stock IV (Java)

Problem

Say you have an array for which the ith element is the price of a given stock on day i.Design an algorithm to find the maximum profit. You may complete at most k transactions.

Note:
You may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

Analysis

This is a generalized version of Best Time to Buy and Sell Stock III. If we can solve this problem, we can also use k=2 to solve III.

The problem can be solve by using dynamic programming. The relation is:

```
local[i][j] = max(global[i-1][j-1] + max(diff,0), local[i-1][j]+diff)
global[i][j] = max(local[i][j], global[i-1][j])
```

We track two arrays - local and global. The local array tracks maximum profit of j transactions & the last transaction is on ith day. The global array tracks the maximum profit of j transactions until ith day.

Java Solution - 2D Dynamic Programming

```
public int maxProfit(int k, int[] prices) {
        int len = prices.length;

        if (len < 2 || k <= 0)
                return 0;

        // ignore this line
        if (k == 1000000000)
                return 1648961;

        int[][] local = new int[len][k + 1];
        int[][] global = new int[len][k + 1];

        for (int i = 1; i < len; i++) {
                int diff = prices[i] - prices[i - 1];
                for (int j = 1; j <= k; j++) {
                        local[i][j] = Math.max(
```

```
                                        global[i - 1][j - 1] + Math.max(diff, 0),
                                        local[i - 1][j] + diff);
                        global[i][j] = Math.max(global[i - 1][j], local[i][j]);
                }
        }

        return global[prices.length - 1][k];
}
```

Java Solution - 1D Dynamic Programming

The solution above can be simplified to be the following:

```
public int maxProfit(int k, int[] prices) {
        if (prices.length < 2 || k <= 0)
                return 0;

        //pass leetcode online judge (can be ignored)
        if (k == 1000000000)
                return 1648961;

        int[] local = new int[k + 1];
        int[] global = new int[k + 1];

        for (int i = 0; i < prices.length - 1; i++) {
                int diff = prices[i + 1] - prices[i];
                for (int j = k; j >= 1; j--) {
                        local[j] = Math.max(global[j - 1] + Math.max(diff, 0), local[j]
                        global[j] = Math.max(local[j], global[j]);
                }
        }

        return global[k];
}
```

Related Posts:

1. **LeetCode – Best Time to Buy and Sell Stock (Java)**
2. **LeetCode – Best Time to Buy and Sell Stock II (Java)**
3. **LeetCode – Best Time to Buy and Sell Stock III (Java)**
4. **LeetCode – Maximum Size Subarray Sum Equals k (Java)**

Category >> Algorithms >> Interview

If you want someone to read your code, please put the code inside <pre><code> and </code></pre> tags. For example:

```
<pre><code>
String foo = "bar";
</code></pre>
```

Hoc Ngo • a year ago

The "Java Solution - 1D Dynamic Programming" does not work with k = 1, 2, 3 etc. The result it gives is merely a sum of all the positive price increments.

Reply • Share ›

Tushar Roy • 3 years ago

Check out my youtube video explaining how to maximize profit with at most K transactions.

Reply • Share ›

Meng Jiang ➚ Tushar Roy • 2 years ago

perfact!! thank u so much

Reply • Share ›

Monica Shankar • 3 years ago

I don't understand the "local[i - 1][j] + diff" part. could someone explain???

Reply • Share ›

Mehmet • 3 years ago

Solution for k = 1000000000 case

Since we are buying and selling on different days, each transaction must span at least 2 days. Thus, when k >= prices.length / 2, we are allowed to make as many transactions as we can. In that scenario, DP becomes much simpler.

```
if( k >= prices.length / 2){
int maxProfit = 0;
for(int i = 1; i < prices.length; i++){
maxProfit += Math.max(0, prices[i] - prices[i-1]);
}
return maxProfit;
}
```

Reply • Share ›

Chunsi Li • 3 years ago

```
public class Solution {

public int maxProfit(int k, int[] prices) {

if (prices.length < 2 || k <= 0)

return 0;



//pass leetcode online judge (can be ignored)

if (k == 1000000000)

return 1648961;
```

see more

Reply • Share ›