Stock Maximize

by HackerRank

Your algorithms have become so good at predicting the market that you now know what the share price of Wooden Orange Toothpicks Inc. (WOT) will be for the next N days.

Each day, you can either buy one share of WOT, sell any number of shares of WOT that you own, or not make any transaction at all. What is the maximum profit you can obtain with an optimum trading strategy?

Input Format

The first line contains the number of test cases T. T test cases follow:

The first line of each test case contains a number N. The next line contains N integers, denoting the predicted price of WOT shares for the next N days.

Constraints

- $1 \leq T \leq 10$
- $1 \leq N \leq 50000$
- All share prices are between 1 and 100000

Output Format

Output T lines, containing the maximum profit which can be obtained for the corresponding test case.

Sample Input

```
3
3
5 3 2
3
1 2 100
4
1 3 1 2
```

Sample Output

```
0
197
3
```

Explanation

For the first case, you cannot obtain any profit because the share price never rises.
For the second case, you can buy one share on the first two days, and sell both of them on the third day.
For the third case, you can buy one share on day 1, sell one on day 2, buy one share on day 3, and sell one share on day 4.

Submissions: 20945

Max Score: 50
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

Need Help?

Dynamic Programming Basics

More

Current Buffer (saved locally, editable)

Java 8

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int t = in.nextInt();
        for(int a0 = 0; a0 < t; a0++){
            int N = in.nextInt();
            long[] prices = new long[N];
            for(int prices_i=0; prices_i < N; prices_i++){
                prices[prices_i] = in.nextLong();
            }
                    System.out.println(dp(prices, new HashMap<Integer, Long>()));
        }
    }

    public static long dp(long[] stocks, Map<Integer, Long> map){


        long previousStockVal = stocks[stocks.length - 1];
        long output = 0;

        for(int i = stocks.length - 2 ; i >= 0 ; i--){

            long currentStockVal = stocks[i] ;

            if(currentStockVal < previousStockVal){
                output += previousStockVal - currentStockVal;
            }
            else{
               previousStockVal =  currentStockVal;
                output += previousStockVal - currentStockVal;
            }

        }
        return output;
    }

}

```

Line: 1 Col: 1

Upload Code as File   □ Test against custom input   Run Code   Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature