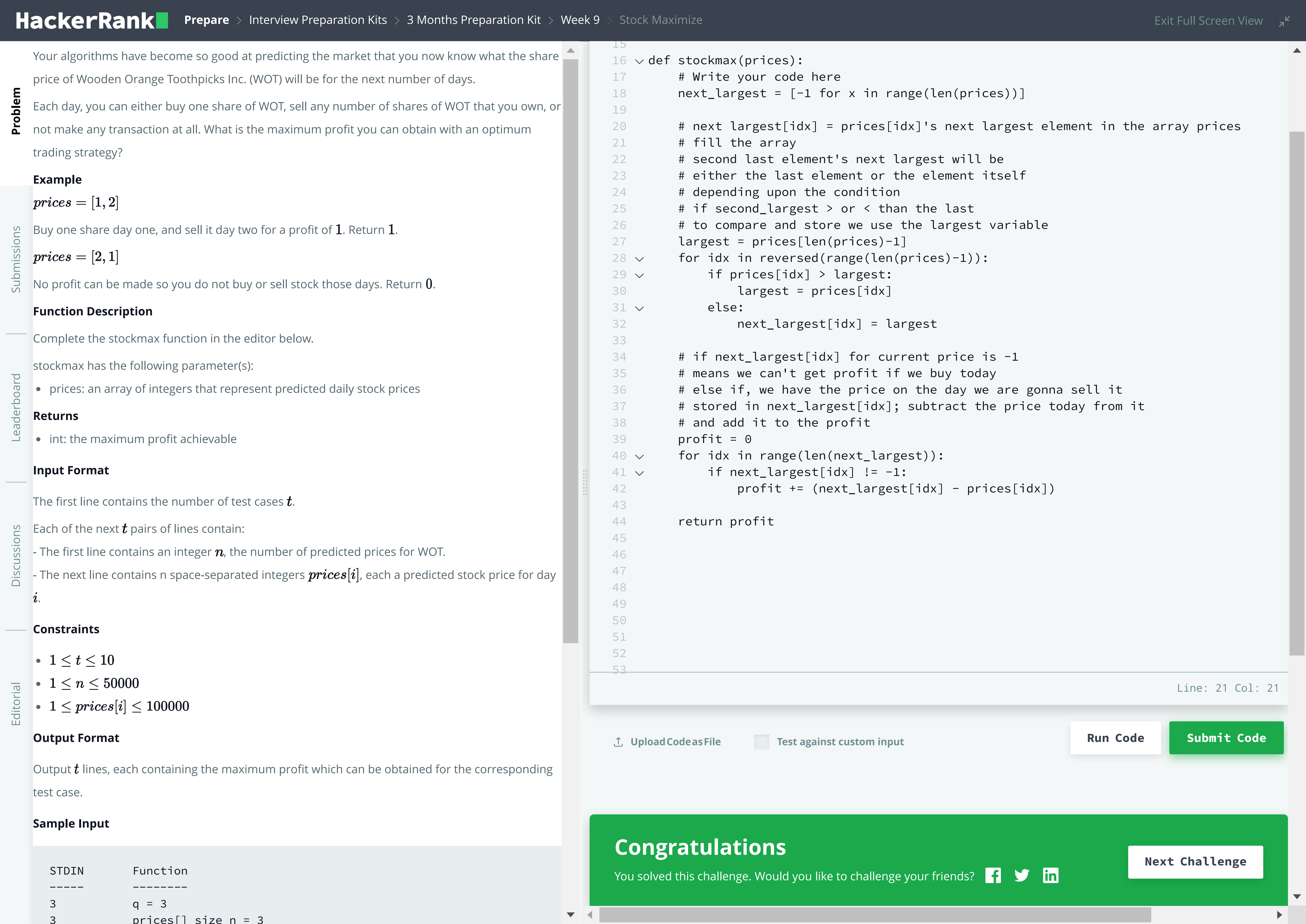
HackerRank
Prepare > Interview Preparation Kits > 3 Months Preparation Kit > Week 9 > Stock Maximize
Exit Full Screen View
Problem
Submissions
Leaderboard
Discussions
Editorial

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 number of 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?

Example

$prices = [1, 2]$

Buy one share day one, and sell it day two for a profit of 1. Return 1.

$prices = [2, 1]$

No profit can be made so you do not buy or sell stock those days. Return 0.

Function Description

Complete the stockmax function in the editor below.

stockmax has the following parameter(s):

- prices: an array of integers that represent predicted daily stock prices

Returns

- int: the maximum profit achievable

Input Format

The first line contains the number of test cases t.

Each of the next t pairs of lines contain:

- The first line contains an integer n, the number of predicted prices for WOT.

- The next line contains n space-separated integers $prices[i]$, each a predicted stock price for day i.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 50000$
- $1 \leq prices[i] \leq 100000$

Output Format

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

Sample Input

```
STDIN       Function
-----       --------
3           q = 3
3           prices[] size n = 3
```

```
def stockmax(prices):
    # Write your code here
    next_largest = [-1 for x in range(len(prices))]

    # next largest[idx] = prices[idx]'s next largest element in the array prices
    # fill the array
    # second last element's next largest will be
    # either the last element or the element itself
    # depending upon the condition
    # if second_largest > or < than the last
    # to compare and store we use the largest variable
    largest = prices[len(prices)-1]
    for idx in reversed(range(len(prices)-1)):
        if prices[idx] > largest:
            largest = prices[idx]
        else:
            next_largest[idx] = largest

    # if next_largest[idx] for current price is -1
    # means we can't get profit if we buy today
    # else if, we have the price on the day we are gonna sell it
    # stored in next_largest[idx]; subtract the price today from it
    # and add it to the profit
    profit = 0
    for idx in range(len(next_largest)):
        if next_largest[idx] != -1:
            profit += (next_largest[idx] - prices[idx])

    return profit
```

Congratulations

You solved this challenge. Would you like to challenge your friends?