AlgoExpert Quad Layout Swift 12px Sublime Monokai 00:00:00

 Prompt
 Scratchpad
 Our Solution(s)
 Video Explanation

Run Code

```
1\, // Copyright @ 2020 AlgoExpert, LLC. All rights reserved.
    class Program {
         // O(nk) time | O(n) space
         \label{lem:func_maxProfitWithKTransactions} \begin{subarray}{ll} func &maxProfitWithKTransactions($\_$ prices: [Int], $\_$ k: Int) $$\rightarrow$ Int {} \end{subarray}
             if prices.count == 0 {
                 return 0
9
10
             var evenProfits = Array(repeating: 0, count: prices.count)
             var oddProfits = Array(repeating: 0, count: prices.count)
11
12
             for transaction in stride(from: 1, through: k, by: 1) {
13
                 var maxProfitThusFar = Int.min
14
15
16
                 if transaction % 2 == 0 {
17
                      {\tt secondSolutionHelper(\&evenProfits, \&oddProfits, \&maxProfitThusFar, prices)}
18
19
                      {\tt secondSolutionHelper(\&oddProfits, \&evenProfits, \&maxProfitThusFar, prices)}
20
21
             }
22
23
             if k % 2 == 0 {
24
                  return evenProfits[prices.count - 1]
25
             } else {
26
                  return oddProfits[prices.count - 1]
27
28
29
30
         func secondSolutionHelper(_ currentProfits: inout [Int], _ previousProfits: inout [Int], _ maxProfitThusFar: inout Int, _ prices: [Int]) {
31
             for day in stride(from: 1, to: prices.count, by: 1) {
32
                 maxProfitThusFar = max(maxProfitThusFar, previousProfits[day - 1] - prices[day - 1])
33
                  currentProfits[day] = max(currentProfits[day - 1], maxProfitThusFar + prices[day])
34
35
36 }
```

Solution 1 Solution 2