

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 package main
4
5 import "math"
6
7 // # O(nk) time | O(n) space
8 func MaxProfitWithKTransactions(prices []int, k int) int {
9     if len(prices) == 0 {
10         return 0
11     }
12     evenProfits := make([]int, len(prices))
13     oddProfits := make([]int, len(prices))
14     var currentProfits, previousProfits []int
15     for t := 1; t < k+1; t++ {
16         maxThusFar := math.MinInt32
17         if t%2 == 1 {
18             currentProfits, previousProfits = oddProfits, evenProfits
19         } else {
20             currentProfits, previousProfits = evenProfits, oddProfits
21         }
22         for d := 1; d < len(prices); d++ {
23             maxThusFar = max(maxThusFar, previousProfits[d-1]-prices[d-1])
24             currentProfits[d] = max(currentProfits[d-1], maxThusFar+prices[d])
25         }
26     }
27     if k%2 == 0 {
28         return evenProfits[len(prices)-1]
29     }
30     return oddProfits[len(prices)-1]
31 }
32
33 func max(arg int, rest ...int) int {
```

Solution 1

Solution 2

Solution 3

```
1 package main
2
3 func MaxProfitWithKTransactions(prices []int, k int) int {
4     // Write your code here.
5     return -1
6 }
7
```

Our Tests

Custom Output

Submit Code

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 package main
4
5 import "math"
6
7 // # O(nk) time | O(n) space
8 func MaxProfitWithKTransactions(prices []int, k int) int {
9     if len(prices) == 0 {
10         return 0
11     }
12     evenProfits := make([]int, len(prices))
13     oddProfits := make([]int, len(prices))
14     var currentProfits, previousProfits []int
15     for t := 1; t < k+1; t++ {
16         maxThusFar := math.MinInt32
17         if t%2 == 1 {
18             currentProfits, previousProfits = oddProfits, evenProfits
19         } else {
20             currentProfits, previousProfits = evenProfits, oddProfits
21         }
22         for d := 1; d < len(prices); d++ {
23             maxThusFar = max(maxThusFar, previousProfits[d-1]-prices[d-1])
24             currentProfits[d] = max(currentProfits[d-1], maxThusFar+prices[d])
25         }
26     }
27     if k%2 == 0 {
28         return evenProfits[len(prices)-1]
29     }
30     return oddProfits[len(prices)-1]
31 }
32
33 func max(arg int, rest ...int) int {
```

```
10 # Returns Report, a BeautifulSoup BeautifulSoup object
11
12 # Run in BeautifulSoup BeautifulSoup BeautifulSoup
13 # Returns Report, a BeautifulSoup BeautifulSoup BeautifulSoup
14
15 #
16
17 # Run in BeautifulSoup BeautifulSoup BeautifulSoup
18 # Returns Report, a BeautifulSoup BeautifulSoup BeautifulSoup
19
20 #
21
22 # Run in BeautifulSoup BeautifulSoup BeautifulSoup
23 # Returns Report, a BeautifulSoup BeautifulSoup BeautifulSoup
24
25 #
26
```

Run or submit code when you're ready.