

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 class Program {
4     // O(nk) time | O(n) space
5     public static int maxProfitWithKTransactions(int[] prices, int k) {
6         if (prices.length == 0) {
7             return 0;
8         }
9         int[] evenProfits = new int[prices.length];
10        int[] oddProfits = new int[prices.length];
11        for (int i = 0; i < prices.length; i++) {
12            evenProfits[i] = 0;
13            oddProfits[i] = 0;
14        }
15        for (int t = 1; t < k + 1; t++) {
16            int maxThusFar = Integer.MIN_VALUE;
17            int[] currentProfits = new int[prices.length];
18            int[] previousProfits = new int[prices.length];
19            if (t % 2 == 1) {
20                currentProfits = oddProfits;
21                previousProfits = evenProfits;
22            } else {
23                currentProfits = evenProfits;
24                previousProfits = oddProfits;
25            }
26            for (int d = 1; d < prices.length; d++) {
27                maxThusFar = Math.max(maxThusFar, previousProfits[d - 1] - prices[d - 1]);
28                currentProfits[d] = Math.max(currentProfits[d - 1], maxThusFar + prices[d] - prices[d - 1]);
29            }
30        }
31        return k % 2 == 0 ? evenProfits[prices.length - 1] : oddProfits[prices.length - 1];
32    }
33 }
```

Solution 1

Solution 2

Solution 3

```
1 class Program {
2     public static int maxProfitWithKTransactions(int[] prices, int k) {
3         // Write your code here.
4         return -1;
5     }
6 }
7
```

Our Tests

Custom Output

Submit Code

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 class Program {
4     // O(nk) time | O(n) space
5     public static int maxProfitWithKTransactions(int[] prices, int k) {
6         if (prices.length == 0) {
7             return 0;
8         }
9         int[] evenProfits = new int[prices.length];
10        int[] oddProfits = new int[prices.length];
11        for (int i = 0; i < prices.length; i++) {
12            evenProfits[i] = 0;
13            oddProfits[i] = 0;
14        }
15        for (int t = 1; t < k + 1; t++) {
16            int maxThusFar = Integer.MIN_VALUE;
17            int[] currentProfits = new int[prices.length];
18            int[] previousProfits = new int[prices.length];
19            if (t % 2 == 1) {
20                currentProfits = oddProfits;
21                previousProfits = evenProfits;
22            } else {
23                currentProfits = evenProfits;
24                previousProfits = oddProfits;
25            }
26            for (int d = 1; d < prices.length; d++) {
27                maxThusFar = Math.max(maxThusFar, previousProfits[d - 1] - prices[d - 1]);
28                currentProfits[d] = Math.max(currentProfits[d - 1], maxThusFar + prices[d] - prices[d - 1]);
29            }
30        }
31        return k % 2 == 0 ? evenProfits[prices.length - 1] : oddProfits[prices.length - 1];
32    }
33 }
```

```
10 #Print
11 #Print TestPasses: 0
12 #Print Test = 100
13 #Print assert(assertTrue(assertTestPasses, 10) == 1)
14 }
15
16 #Print
17 #Print TestPasses: 0
18 #Print Test = 10, 100
19 #Print assert(assertTrue(assertTestPasses, 10) == 1)
20 }
21
22 #Print
23 #Print TestPasses: 0
24 #Print Test = 10, 100
```

Run or submit code when you're ready.