Prompt   Scratchpad   Our Solution(s)   Video Explanation   Run Code

Solution 1   Solution 2

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
#include <climits>
using namespace std;

// O(nk) time | O(nk) space
int maxProfitWithKTransactions(vector<int> prices, int k) {
  if (prices.size() == 0) {
    return 0;
  }
  vector<vector<int>> profits(k + 1, vector<int>(prices.size(), 0));
  for (int t = 1; t < k + 1; t++) {
    int maxThusFar = INT_MIN;
    for (int d = 1; d < prices.size(); d++) {
      maxThusFar = max(maxThusFar, profits[t - 1][d - 1] - prices[d - 1]);
      profits[t][d] = max(profits[t][d - 1], maxThusFar + prices[d]);
    }
  }
  return profits[k][prices.size() - 1];
}
```

Prompt   Scratchpad   Our Solution(s)   Video Explanation   Run Code

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
#include <climits>
using namespace std;

// O(nk) time | O(nk) space
int maxProfitWithKTransactions(vector<int> prices, int k) {
  if (prices.size() == 0) {
    return 0;
  }
  vector<vector<int>> profits(k + 1, vector<int>(prices.size(), 0));
  for (int t = 1; t < k + 1; t++) {
    int maxThusFar = INT_MIN;
    for (int d = 1; d < prices.size(); d++) {
      maxThusFar = max(maxThusFar, profits[t - 1][d - 1] - prices[d - 1]);
```