Best Time to Buy and Sell Stock IV

Question: Say you have an array for which the ith element is the price of a given stock on day i. Design an algorithm to find the maximum profit. You may complete at most k transactions.

Note: You may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

Solutions:

```
class Solution:
    """
    @param k: an integer
    @param prices: a list of integer
    @return: an integer which is maximum profit
    """
    def maxProfit(self, k, prices):
        if prices is None or len(prices) <= 1 or k <= 0:
            return 0

        n = len(prices)
        # k >= prices.length / 2 ==> multiple transactions Stock II
        if k >= n / 2:
            profit_max = 0
            for i in range(1, n):
                diff = prices[i] - prices[i - 1]
                if diff > 0:
```

```
                profit_max += diff
            return profit_max


        f = [[0 for i in range(k + 1)] for j in range(n + 1)]
        for j in range(1, k + 1):
            for i in range(1, n + 1):
                for x in range(0, i + 1):
                    f[i][j] = max(f[i][j], f[x][j - 1] + self.profit(prices, x + 1, i))


        return f[n][k]


    # calculate the profit of prices(l, u)
    def profit(self, prices, l, u):
        if l >= u:
            return 0
        valley = 2**31 - 1
        profit_max = 0
        for price in prices[l - 1:u]:
            profit_max = max(profit_max, price - valley)
            valley = min(valley, price)
        return profit_max


Solution().maxProfit(8,[1, 4, 8, 1, 2, 10, 20, 30, 5, 3])
```