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

class Program {
    // O(nk) time | O(n) space
    func maxProfitWithKTransactions(_ prices: [Int], _ k: Int) -> Int {
        if prices.count == 0 {
            return 0
        }

        var evenProfits = Array(repeating: 0, count: prices.count)
        var oddProfits = Array(repeating: 0, count: prices.count)

        for transaction in stride(from: 1, through: k, by: 1) {
            var maxProfitThusFar = Int.min

            if transaction % 2 == 0 {
                secondSolutionHelper(&evenProfits, &oddProfits, &maxProfitThusFar, prices)
            } else {
                secondSolutionHelper(&oddProfits, &evenProfits, &maxProfitThusFar, prices)
            }
        }

        if k % 2 == 0 {
            return evenProfits[prices.count - 1]
        } else {
            return oddProfits[prices.count - 1]
        }
    }

    func secondSolutionHelper(_ currentProfits: inout [Int], _ previousProfits: inout [Int], _ maxProfitThusFar: inout Int, _ prices: [Int]) {
        for day in stride(from: 1, to: prices.count, by: 1) {
            maxProfitThusFar = max(maxProfitThusFar, previousProfits[day - 1] - prices[day - 1])
            currentProfits[day] = max(currentProfits[day - 1], maxProfitThusFar + prices[day])
        }
    }
}
```