prices array ⇒ {7, 6, 4, 11, 9} → n days

you are allowed to have k transactions

whenever we buy, no. of transactions ~~increase~~

$dp(i)(k)(x)$ → numbers of stocks you are holding

i → days; $(0, n-1)$, $(0, k)$, $(0, 1)$

maximum profit i can make on 3rd day with k transactions and with x stocks in my hand.

$dp(2)(0)(1) = -3$

$dp(2)($

$\boxed{ans = dp(n-1)(k)(0)}$

base cases:

$dp[-1][k][0] = 0$ ← day, tra

$dp[-1][k][1] = -\infty$ → not possible

$dp(i)(0)(0) = 0$

$dp(i)(0)(1) = -\infty$ → not possible

dp states:

$dp(i)(k)(0) = \max\{rest, sell\}$ — rest: do not buy, $dp(i-1)(k)(0)$ → rest; $(i^{th})(k)(0)$

$dp(i)(k)(1) \Rightarrow \max\{rest, buy\}$ — rest → hold, $dp(i-1)(k)(1)$; buy → no. of transaction increase

$$\boxed{\begin{aligned} dp(i)(k)(0) &\Rightarrow \max\{dp(i-1)(k)(0),\ dp(i-1)(k)(1) + prices(i)\} \\ dp(i)(k)(1) &\Rightarrow \max\{dp(i-1)(k)(1),\ dp(i-1)(k-1)(0) - prices(i)\} \end{aligned}}$$

⇒ 1) $dp\,10 \Rightarrow \max(dp\,10,\ dp\,11 + p)$

2) $dp\,20 \Rightarrow \max(dp\,20,\ dp\,21 + p)$

3) $dp\,11 \Rightarrow \max(dp\,11,\ 0 - p)$

4) $dp\,21 \Rightarrow \max(dp\,21,\ dp\,10 - p)$

$dp\,10$

$$\begin{tikzcd} \boxed{1} \arrow[d] \\ \boxed{4} \arrow[d] \\ \boxed{2} \end{tikzcd}$$

$\boxed{2}$ $\boxed{4}$ $\boxed{1}$ $\boxed{3}$

	0	1
-1	0	0
0	0	0
1	0	1
2	0	2
3	0	2
4	0	3

0

	0	1
-1	$-\infty$	$-\infty$
0	$-\infty$	-1
1	$-\infty$	-1
2	$-\infty$	-1
3	$-\infty$	1
4	$-\infty$	1

$$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \{1, & 2, & 3, & 0, & 2\} \end{array}$$

$-1+2$

$0-2$

$1,\ \underline{-1+3},\ 0-3$

$2,\ -1+0$

$1-0$

$1+2$

k=

	0	⇒ dp10 1	⇒ dp20 2
-1	0	0	0
0	0	0	0
1	0	0	0 0
2	0	2 0	2 0
3	0	2	2
4	2 0	2	2
5	0	3	5
6	0	3	5
7	0	4	6

	0	⇒ dp11 1	⇒ dp21 2
-1	-∞	-∞	-∞
0	-∞	-3	-3
1	-∞	-3, 0	-3
2	-∞	0	0
3	-∞	0,	2
4	-∞	0	2
5	-∞	0 0	2 0
6	-∞	0	2
7	-∞	0	2

$x=0$ ⇒

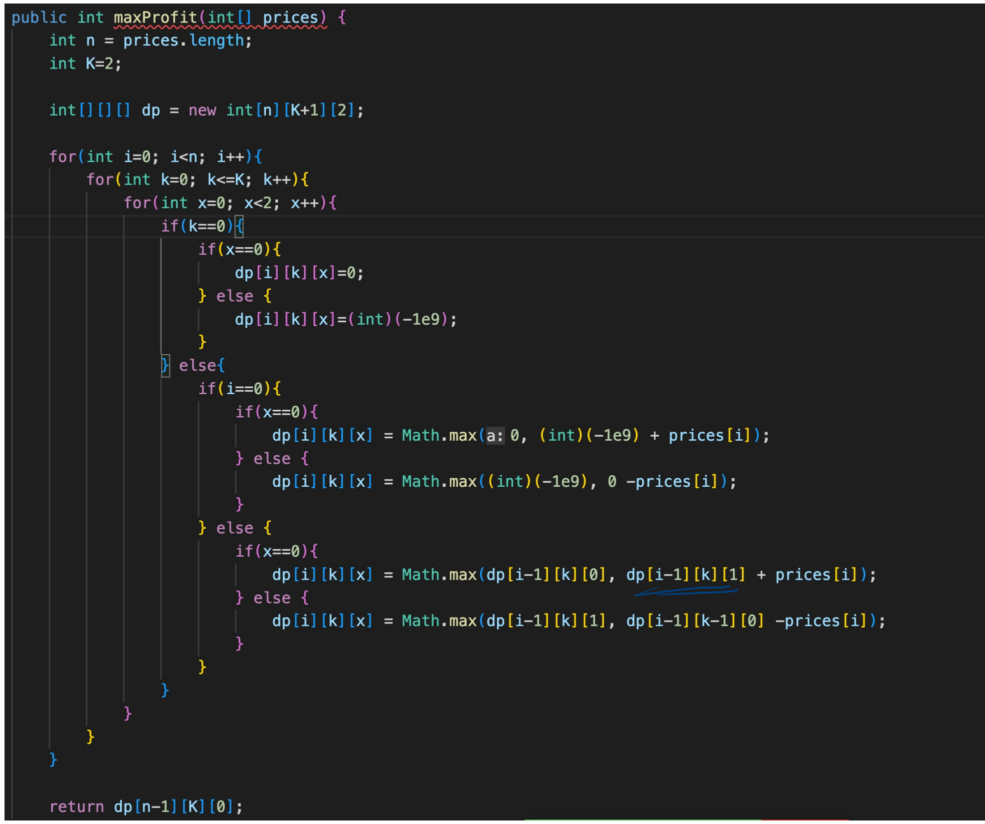
$dp(n-1)(2)(0)$

$x=1$ ⇒

0	1	2	3	4	5	6	7
3	3	5	0	0	3	1	4

② + 3

6 5 6

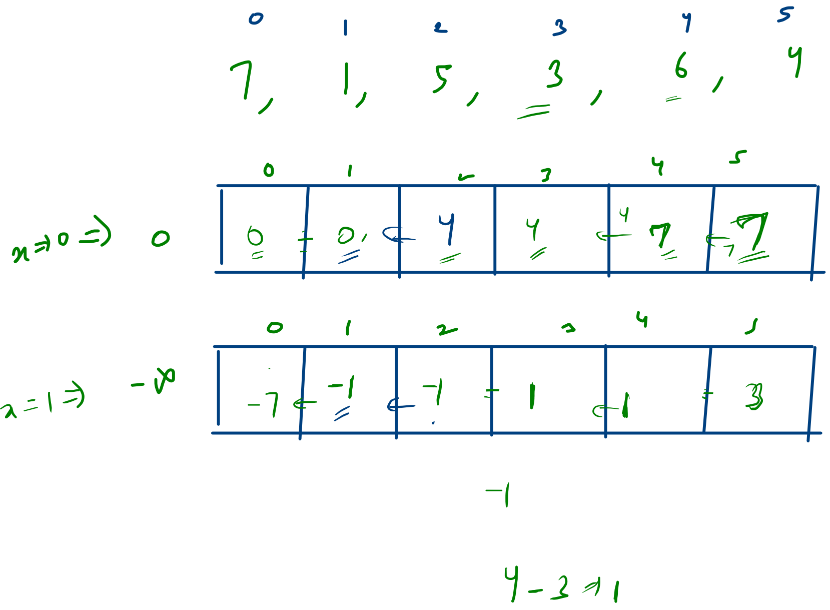


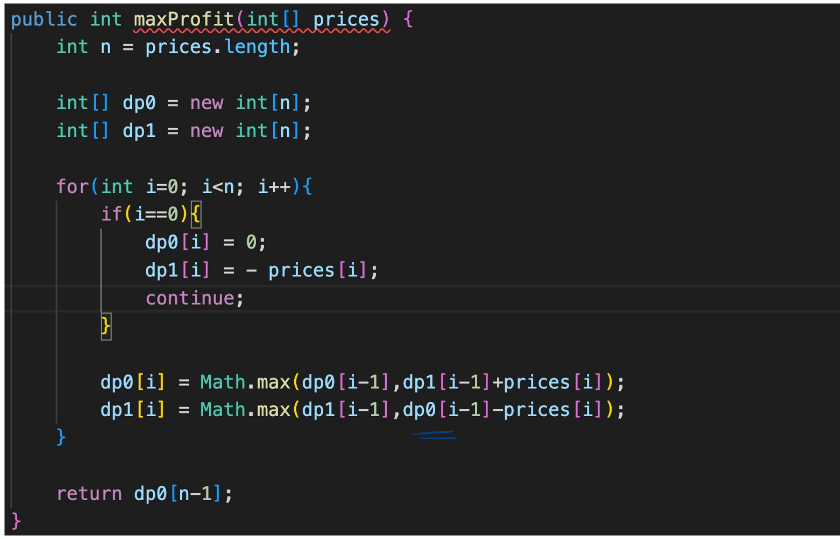
```java
public int maxProfit(int[] prices) {
    int n = prices.length;
    int K=2;

    int[][][] dp = new int[n][K+1][2];

    for(int i=0; i<n; i++){
        for(int k=0; k<=K; k++){
            for(int x=0; x<2; x++){
                if(k==0){
                    if(x==0){
                        dp[i][k][x]=0;
                    } else {
                        dp[i][k][x]=(int)(-1e9);
                    }
                } else{
                    if(i==0){
                        if(x==0){
                            dp[i][k][x] = Math.max(a: 0, (int)(-1e9) + prices[i]);
                        } else {
                            dp[i][k][x] = Math.max((int)(-1e9), 0 -prices[i]);
                        }
                    } else {
                        if(x==0){
                            dp[i][k][x] = Math.max(dp[i-1][k][0], dp[i-1][k][1] + prices[i]);
                        } else {
                            dp[i][k][x] = Math.max(dp[i-1][k][1], dp[i-1][k-1][0] -prices[i]);
                        }
                    }
                }
            }
        }
    }

    return dp[n-1][K][0];
```




```java
public int maxProfit(int[] prices) {
    int n = prices.length;

    int[] dp0 = new int[n];
    int[] dp1 = new int[n];

    for(int i=0; i<n; i++){
        if(i==0){
            dp0[i] = 0;
            dp1[i] = - prices[i];
            continue;
        }

        dp0[i] = Math.max(dp0[i-1],dp1[i-1]+prices[i]);
        dp1[i] = Math.max(dp1[i-1],dp0[i-1]-prices[i]);
    }

    return dp0[n-1];
}
```