Buy and sell stocks

Max Profit with atmost 1 transaction

Question:

→ Max Profit I can make by atmost 1 transaction

A =	4	2	9	3	8	5	9	7
	0	1	2	3	4	5	6	7

Ans = 9 - 2 = 7

A =	5	8	7	6	5	4
	0	1	2	3	4	5

profit = 8 - 5 = 3

A =	1	2	3	4	5
	0	1	2	3	4

Profit = 5 - 1 = 4

A = 4 3 2 1

Profit = 0

Brute force

Consider all pairs (i, j) such that $i < j$ (Buy ↗ i, sell ↙ j)

T.C: $O(n^2)$

S.C: $O(1)$

Approach2:

[0..... i-1]

A = 4 2 3 8 5 9 7
0 1 2 3 4 5 6

profit = 8-2 = 6
= 5-2 = 3

T.C to find minimum in left Subarray: O(n)

If I want to sell stock on Day i
when would I buy : min (0.... i-1)

min_so_far = ~~4~~ 2
profit = ~~X~~ ~~6~~ 7

A = 4 2 3 8 5 9 7
0 1 2 3 4 5 6

7-2 = 5
5-2 = 1
2-4 = -2
8-2 = 6
9-2 = 7

T.C: O(n)
S.C: O(1)

i (i+1..... n-1)

max_so_far = ~~7~~ 9
profit = ~~0~~ 4

Approach3 : → 4 2 3 8 5 9 7
0 1 2 3 4 5 6

max_so_far = 7

max_so_far = A[n-1];
for (i = n-2 ; i >= 0; i--) {

{
...
}

T.C: O(n)
S.C: O(1)

sell = 7
buying = 9

7-9 = -2
7-9 = -2

Question: You can make as many transaction as possible

→ You cannot cross in multiple transaction at any point

	0	1	2	3	4	5	6
A =	10	14	23	34	7	51	67

Profit: (10, 34) (7, 67)

24 + 60 = 84

	0	1	2	3	4	5	6	7	8	9
A = 2	14	12	7	9	10	12	5	14	11	15

Buy

sell

Profit = (7, 12) (5, 14) (11, 15)

5 + 9 + 4 = $\boxed{18}$

profit = 15-2 = 13

A = 1 2 3 4 5

(1, 5) ⇒ 4

A = [0]

A[0...0]

maxProfit(0):

Approach 1:

Buy j

12 − price[j] + maxProfit(j−1)

A[0.... 4]

	0	1	2	3	4	5
A:	14	12	7	9	10	12

maxProfit(4)

if (i == 0)

i = −1

Recursion

Choices

1) Don't sell the stock: maxProfit(i − 1)

2) Sell the stock: 12−10 + maxProfit(3)

12 − 9 + maxProfit(2)

5 ← $\boxed{12-7}$ + maxProfit(1)

0 + maxProfit(0)

12 − 14 + maxProfit(−1)

max

$\max(\text{price}[i] - \text{price}[j] + \text{maxProfit}(j-1))$

$\uparrow$ sell $\qquad$ $\uparrow$ buying

$j \in [0, \dots i-1]$

index

```
int maxProfit(i) {
    if(i <= 0) return 0;

    if(dp[i] != -1) return dp[i]

    profit = maxProfit(i-1);
    for(j = i-1; j >= 0; j--) {
        profit = max(profit,
                     price[i] - price[j] + maxProfit(j-1));
    } dp[i] = profit
    return profit
}
```

O(n)

A =	14	12	7	9	6	13
					4	5
	0	1	2	3		

maxProfit(3), (2), (1), (0)

maxProfit(2), (1) (0)..

T.C: $O(\#\text{states} \times \text{T.C per state})$

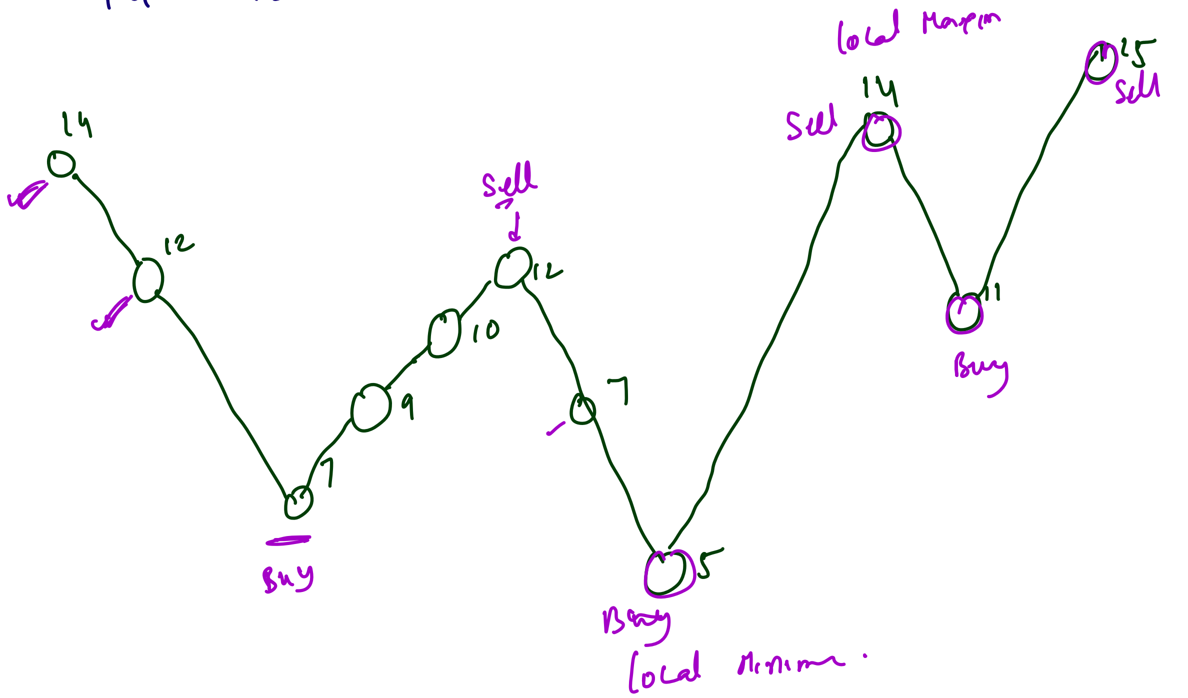
$\#\text{states} \to n$, $\text{T.C per state} \to O(n)$

$= \boxed{O(n^2)}$

S.C: $O(n)$

Efficient Approach

	0	1	2	3	4	5	6	7	8	9
A =	14	12	7	9	10	12	5	14	11	15



Buy at local Minima
sell at local Mnima.

local Minima: if (price[i] < price[i-1] && price[i] < price[i+1]) Local Minima.

T.C: O(n)

S.C: O(1)

Question: At most 2 transactions

A =	7	19	8	12	17	11	23
	0	1	2	3	4	5	6

(7, 19) (8, 23)

Profit = 12 + 15 = 27

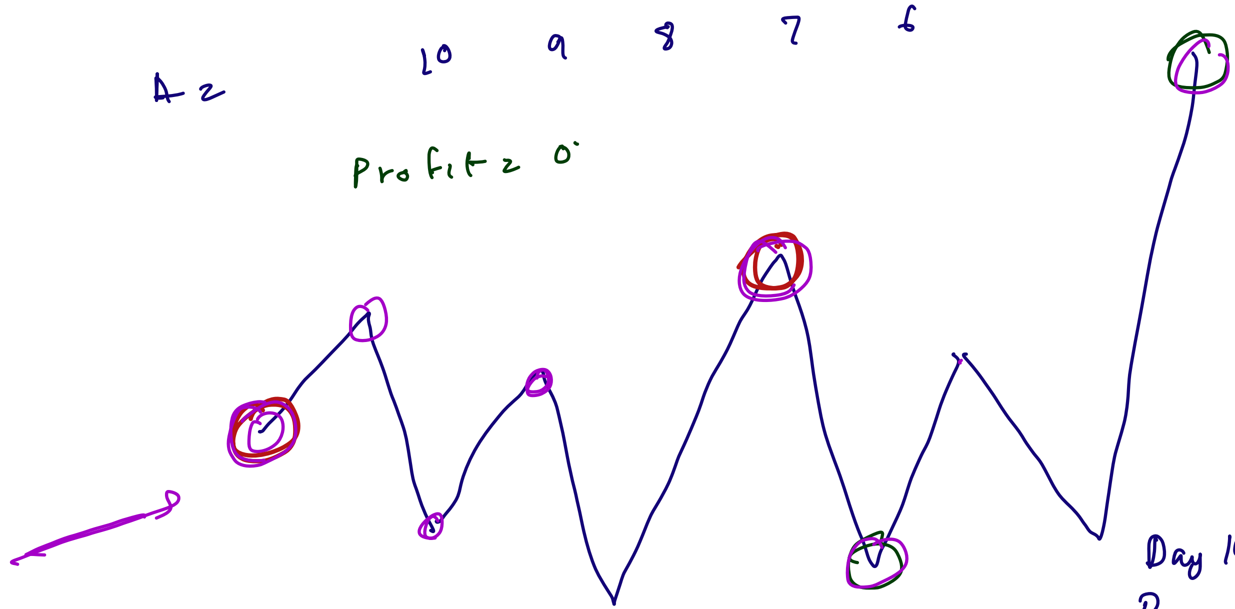
	0	1	2	3	4	5	6
A =	2	30	15	10	8	25	80

(2, 30) (8, 80)

Profit = 28 + 72 = 100

A = 10 9 8 7 6

Profit = 0



Day 10

minimum M1 m2 m3 m4 m5

maximum P1 P2 P3 P4 P5

Day 1

Approach 1:

7 19 8 | 12 | 17 11 23

$O(n)$ $O(n)$

$O(n)$

→ N partitions

1 partition ⇒ $O(n)$

T.C : $O(n^2)$

S.C : $O(1)$

Approach 2:

Let us generate the profit[] array

profit[i] = Max profit I can make until day i with 1 transaction

(23 − 12) + 12 = 23

2nd stage.

max_so_far = 23

```
for(i=n-2; i>=0; i--)
  profit = max(profit,
               max_so_far - price[i] +
               profit[i-1])
```

A =	10	7	19	8	12	17	20	23
	0	1	2	3	4	5	6	7
profit =	0	0	12	12	12	12	13	16

max_profit = ~~15~~ ~~18~~ 23

(12)

profit = 23 − 12 + 12 = 23

23 − 20 + 12 = 3 + 12 = 15

23 − 8 + 12

23 − 17 + 12 = 18

step1: Generate profit array : O(n)

step2: ~ O(n)

T.C: $O(n) + O(n) = O(n)$

S.C: $O(n)$

```
max_profit = 0
max_so_far = price[n-1];
for(i=n-2; i≥0; i--){
    max_profit = max(max_profit, max_so_far - price[i] + profit[i-1]);
    max_so_far = max(max_so_far, price[i]);
```

1 transact

Question: At most K transaction

K = 3

A = 12 14 17 10 14 13 12 15

0 1 2 3 4 5 6 7

(12, 17) (10, 14) (13, 15)

Profit = 5 + 4 + 3 = 12

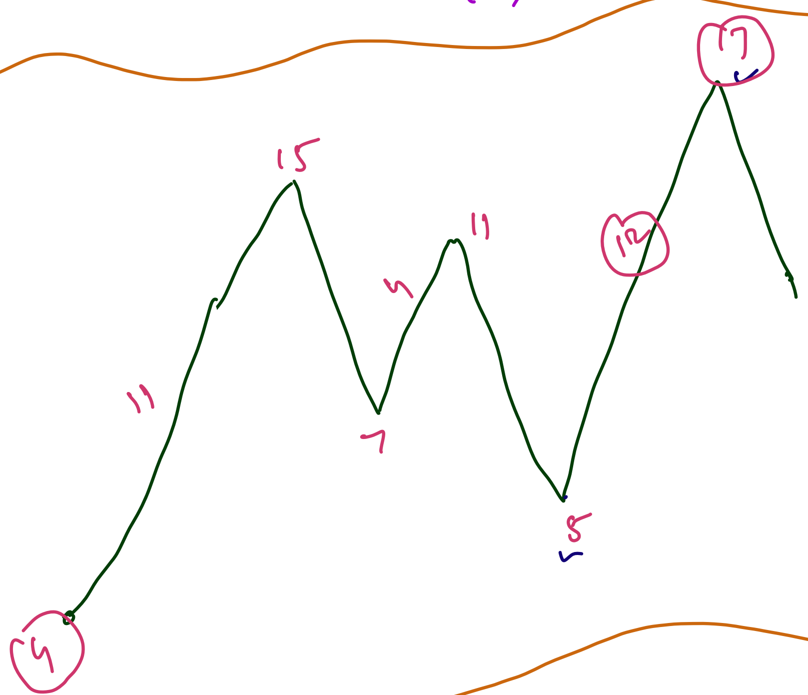
K = 3

A = 100 30 15 10 8 25 80

0 1 2 3 4 5 6

1 transaction

(8, 80) => 72



K = 1

(5, 17)

17 − 4 = 13

	0	1	2	3	4	5	6	7
price =	12	14	17	10	14	13	12	15

Buy

sell

15-14

k = 4

Consider last day,

1) Don't sell : maxProfit(i-1, k)

2) Sell the stock : 15-12 + maxProfit(5, k-1)

15-13 + " (4, k-1)

15-14 + " (3, k-1)

max

index

```
int maxProfit(int i, int k) {
    if(k == 0) return 0;
    if(i <= 0) return 0;
    if(dp[k][i] != -1) return dp[k][i];
    profit = maxProfit(i-1, k)
    for(j = i-1; j >= 0; j--) {
        profit = max(profit,
            (price[i] - price[j] + maxProfit(j-1, k-1)));
    }
    dp[k][i] = profit;
    return profit;
}
```

2D

K x N

O(n)

T.C: O(# state x TC per state)

O(N·K x N) = O(N²·K)

S.C: O(N·K)

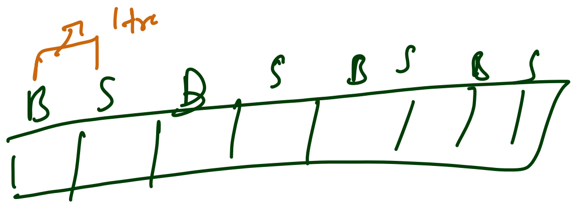
$N \le 1000$

$K \le 10^6$

$(1000)^2 \times 10^6 = \boxed{10^{12}}$

K

Maximum transactions we can make:



if ($K > \frac{N}{2}$)

↓ Find profit with any no. of transactions (Problem)

$O(N)$ ←

else ($K < \frac{N}{2}$)

$K \le \frac{1000}{2} = 500$

$O(n^2 \times K)$

$O(1000 \times 1000 \times 500)$

$= \boxed{5 \times 10^8}$

Optimise it even further?

max(a, b)

max(a, b, c, d, e)

(1) Don't sell stock.

$F(k, i) = \max(F(k, i-1),$

$\max(price[i] - price[j] + F(k-1, j-1) \ \forall j \in [0, i-1]))$

(2) sell

(2) $= \max(price[i] - price[j] + F(k-1, j-1)) \ \forall j \in [0, i-1]$

$\Rightarrow price[i] + \max(F(k-1, j-1) - price[j]) \ \forall j \in [0, i-1]$

(3) $F(k-1, i-2) - price[i-1]$

$i = 2$

(3) $= \max(F(k-1, j-1) - price[j]) \ \forall j \in [0, i-1]$

$j = i-1$

$F(k-1, i-1-1) - price[i-1] = F(k-1, i-2) - price[i-1]$

prevDiff

$i = 2$

Eq1 $j = 0$. $F(k-1, -1) - price[0]$

Eq2 $j = 1$ $F(k-1, 0) - price[1]$

max $\to$ X

$i = 3$ $j = 0, 1, 2$.

$j = 0$ $F(k-1, -1) - price[0]$

$j = 1$ $F(k-1, 0) - price[1]$

$j = 2$ $F(k-1, 1) - price[2]$

prevDiff

$j = i-1$ max(X,);

for(j = 0; j ≤ i-1; j++)

prevDiff = max(prevDiff,

$j = i$

for(j = 0; j ≤ i-1)

$i = 4$

prevDiff

$j = 0$ $F(k-1, -1) - price[0]$

$j = 1$ $F(k-1, 0) - price[1]$

$j = 2$ $F(k-1, 1) - price[2]$

$j = 3$ $F(k-1, 2) - price[3]$

prevDiff $\forall j \in [0 \ldots i-2]$

$[0 \ldots 2)$

$j = i-1$

$j = i-1$

$i = 5$

~~$i = j$~~

$j = i-1$

$j = 4\ (i-1)$

Buy on Day $(i-1)$

$F(k-1, \sim) - price[\ \sim\ ]$

if prevDiff = max(F(k-1, j-1) - price[j]) $\forall j \in [0, i-2]$

$j = i-1$

$\boxed{prevDiff} =$ max(prevDiff, $\boxed{F(k-1, i-2) - price[i-1]}$)

Pseudocode (Bottom up)

dp[0][i] = 0 , dp[k][0] = 0

$\boxed{\text{if}(k > \frac{N}{2}) \quad \text{Problem 2}}$

(k, i)

for(k...)
 for(r...){
 }
}

for(k=1 ; k ≤ K ; k++) {
 prevDiff = INT_MIN
 for(i = 1 ; i < N ; i++) {
 prevDiff = max(prevDiff, dp[k-1][i-2] - price[i-1]);
 dp[k][i] = max(price[i] + prevDiff, dp[k][i-1]) O(1)
 }
}

O(1)

O(1)

T.C: O(# state × T.C per state)

n × K × O(1)

$\boxed{T.C: \ O(n \cdot K)}$

$K \le \frac{n}{2} \le 500$.

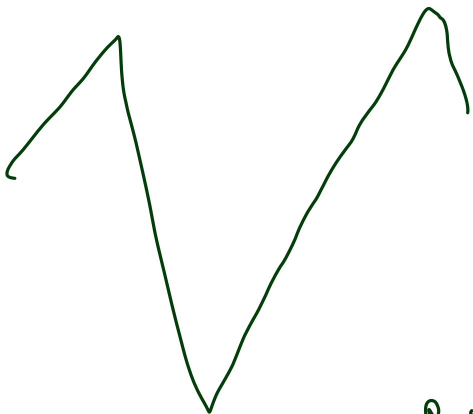
→ Greedy Approach

$\boxed{K \geq \frac{N}{2}}$

1.

$\boxed{\frac{n}{2}}$ → Problem

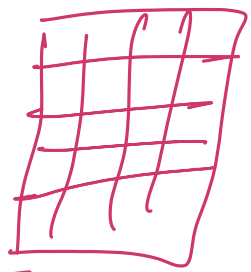
$\boxed{K > \frac{n}{2}}$



Problem

$N \leq 10^3$

$K \leq 10^6$



$N \times K$

T.C: $O(N \times K)$

$N = 10^3, \quad K = 10^6$

$\boxed{10^3 \times 10^6 = 10^9}$

$\boxed{O(n^2 \times K)}$

$K \geq \frac{N}{2}$

if($K \geq \frac{N}{2}$) Problem2

$\boxed{K > \frac{n}{2}}$