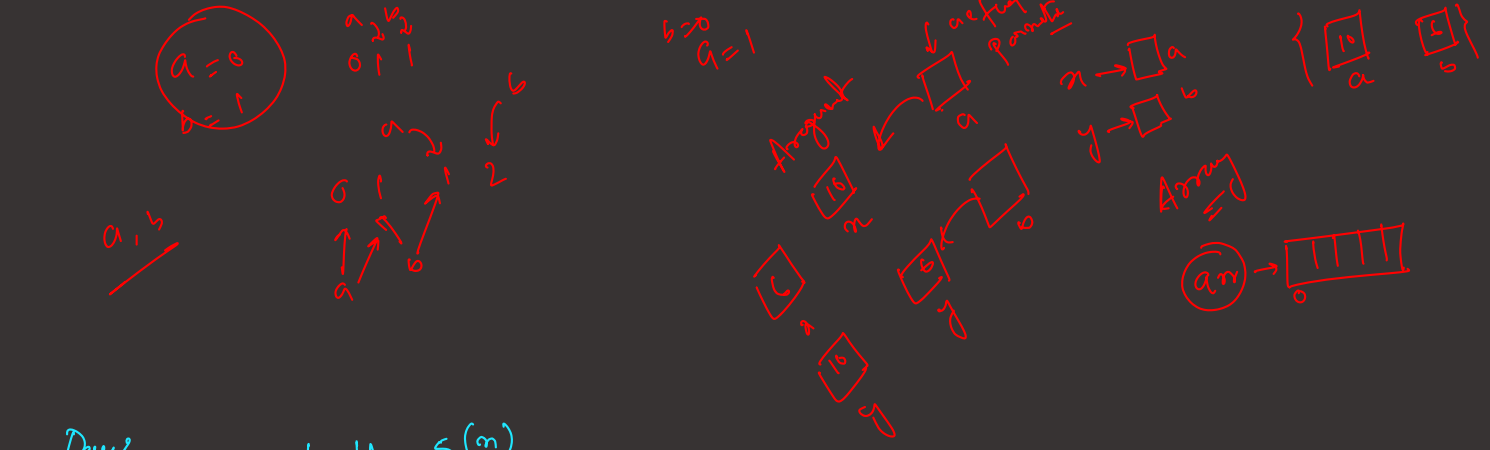


Day 2

arr.length = 5 (n)

arr := | 2 | 4 | 6 | 8 | 10 |

arr[n]

no. of pairs = $1+2+3+\cdots+(n-1)$

$= \frac{n(n-1)}{2}$

pairs: (2,4) (2,6) (2,8) (2,10) → 4

(4,6) (4,8) (4,10) → 3

(6,8) (6,10) → 2

(8,10) → 1

| 2 | 6 | 8 | ✗ not a subarray.

Subarray

↓

Continuous part of array

| 2 | 4 | 6 | 8 | 10 |

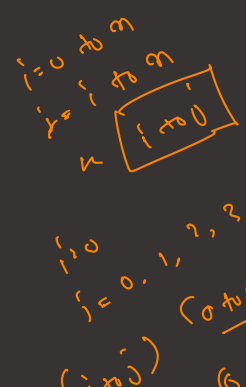
2 | 2 4 | 2 4 6 | 2 4 6 8 | 2 4 6 8 10 → 5

4 | 4 6 | 4 6 8 | 4 6 8 10 → 4

6 | 6 8 | 6 8 10 → 3

8 | 8 10 → 2

10 → 1



i:=0 1 2 3 4

j:= 0 1 2 3 4 (0 to 0) (0 to 1) (0 to 2)

k:= (i to j)

arr[n]

↓

Total subarray = $1+2+\cdots+n$

$= \frac{n(n+1)}{2}$

O(n)

```
for (int i = 0 to n){
    stmt s
}
```

$O(n^2)$

```
for (int i = 0 to n){
    for (int j = 0 to n){
        stmt
    }
}
```

Find Max. or Min Subarray Sum

brute force Approach

T.C = $O(n^3)$

1	-2	6	-1	3

```
Maxsum = -∞
for (int i = 0 to n) {
    for (int j = i to n) {
        sum = 0 ;
        for (k = i to j) {
            sum → (i, j)
        }
        Maxsum = max (Maxsum, sum) ;
    }
}
return Maxsum.
```

```
Maxsum = -∞
for (i = 0 to n)
    for (j = i to n) {
        sum = 0
        for (k = i to j) {
            sum = sum + arr[k]   (sum ↓ (i to j))
        }
        Maxsum = max (Maxsum, sum)
```

$-\infty$ = Integer.MIN_VALUE

Approach 2 : Prefix Sum Method, T.C = $O(n^2)$

arr =

1	-2	6	-1	3
0	1	2	3	4

n

if i = 0, sum(i, j) = sum(0, j) = prefix[j]

sum(i, j) = prefix[j] − prefix[i − 1] → if i ≠ 0

sum(1, 3) = prefix[3] − prefix[0]

= 4 − 1 = 3

[sum(2, 4) = sum(0, 4) − sum(0, 1)]

↓

prefix[4]

prefix =

1	-1	5	4	7
0	1	2	3	4

sum(0, 2) ↓ , sum(0, 4) = prefix[4] ↓

prefix[i] = Sum upto ith element of arr

= sum(0, i) ← Sum of element 0 to i

① Calculate prefix Array.

② for (int i = 0 to n)
    for (j = i to n)
        sum(i, j) = prefix[j] − prefix[i − 1]

H.W find for min sub-array sum

Approach 3 — Kadane's Algo — T.C = O(n)

-2	-3	4	-1	-2	1	5	-3

cs	0	0	4	3	1	2	7	4
ms	0	0	4	4	4	4	7	7

0 + (+ve) → cs = +ve

0 + (-ve) → cs = 0

min + Max

-ve + +ve = +ve.

+ve + -ve = +ve ↓ +ve ↑

```
{ sum < 0
  cs = 0 }
{ ms = max(ms, cs) }
```

cs	-2	-3	4	3	1	2	7	4
ms	-2	-2	4	4	4	4	7	7

-∞ → arr[0]

-2 → cs → arr[0]

-2 → ms → arr[0]

cs = max(arr[i], cs + arr[i])

i = 1: cs = max(-3, -2 + -3) = max(-3, -5) = -3

i = 2: cs = max(4, -3 + 4) = max(4, 1) = 4

i = 3: cs = max(-1, -1 + 4) = max(-1, 3) = 3

Q. Buy and sell Stocks

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

buy (Buy price) → sell

max → profit = 6 - 1 = 5

BP = ∞, P = 0

i = 0, Today's pof = 7 - ∞ = -∞

buy Price = 7

i = 1, 1 - 7 = -6

buy Price = 1 → P = 0

i = 2, 5 - 1 = 4

Today's profit = 4

max profit (0, 4) = 4

i = 3, 3 - 1

today's profit = 2

profit (2, 4)

> BP?

profit = prices[i] - buying price

> 0 ↑

< 0 ↓

buy price = ∞, profit = 0

```
for (int i = 0 to n) {
    if (BP < prices[i]) {
        profit = price[i] - B.P.   // day profit.
        max profit = max(max profit, today's profit);
    }
}
```

today's

else {

buying price = prices[i];

}

}