PS (Sunday's Session)
↳ till CF (incl. CF)

Today: [Arrays: Subarrays → Interview Probs - 2]

Q1: Arrays - Interviews - 1 (HW Q2)

Max positivity

Given an a[]. Return max size subarr of a[] having non negative elements only. (>= 0) If same 2 subarr have same len, return the one which occurs first

Eg: A = [5, 6, -1, 7, 8]

↓

ans = {5, 6}

A = [1, 2, 3, 4, 5, 6]

↳ ans = {1, 2, 3, 4, 5, 6}

Brute force:

↳ use 2 loops to generate subarrays

↳ 3rd loop to check if subarr contains non-negative elements

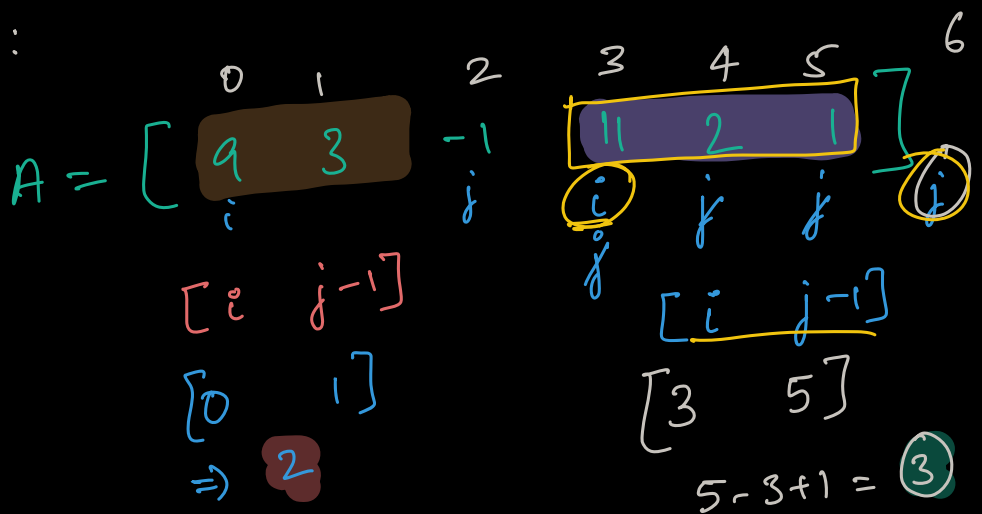
↳ Compare & update ans.

```
for (i=0; i<n; i++) {
    for(j=i; j<n; j++) {
        // subarr [i, j]
        for(k=i; k<=j; k++) {

        }
    }
}
```

→ TC: $O(N^3)$
SC: $O(N)$

idea 2 :



O(n)

```
int[] solve(int a[], int N) {
    i = 0
    max_len = 0
    start_idx = 0, end_idx = 0
    while(i < n) {
        if(A[i] >= 0) {
            int len = 0
            int start = i
            while(start < n && A[start] >= 0) {
                len++
                start++
            }
            if(len > max_len) {
                max_len = len
                start_idx = i
                end_idx = start - 1
            }
            i = start + 1
        }
        else {
            i++
        }
    }
```

```
for(j = start_idx ; j <= end_idx ; j++){
    ans.add(A[j])
}
return ans
}
```

TC: O(N)
SC: O(1)

Q2: Subarrays, HW B2

Alternating Sub-arrays Easy

Given a binary a[] ({0,1}) & int B. Find all the idx of a[] that can act as centre of 2B+1 subarr[], alternating subarr.

A = [1 0 1 0 1] (idx 0 1 2 3 4)

B = 1

↳ 2B+1 = 3

↳ ans = {1, 2, 3}

[1, 0, 1, 0, 1] ✓

[0, 1, 0, 1, 0] ✓

[0, 1, 1, 0] ↳ X

[0, 0, 1, 0, 1, 0] ↳ X

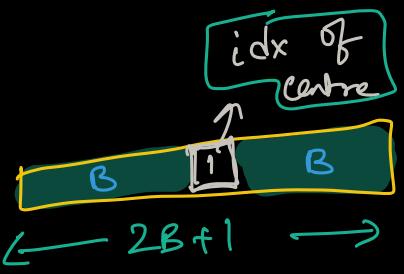
[0 2] = {1 0 1} (idx 0 1 2) ✓

[1 3] = {0 1 0} (idx 1 2 3) ✓

[2 4] = {1 0 1} (idx 2 3 4) ✓

ans = {1, 2, 3}

2B+1 ↳ odd → B + 1 + B



Brute force: Consider every idx i as middle idx / centre

↳ Check B elements on left → left

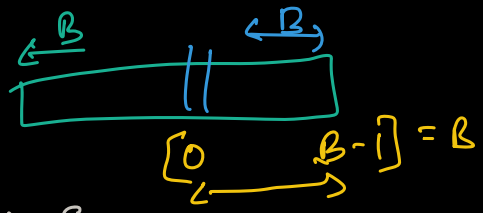
↳ Check B elements on right → right

↳ if both left & right, ↳ add i to ans

```
int[] solve(int a[], int N, int B) {

    int ans[]
    for(i = B ; i < N-B ; i++) {
        // i is middle element

        // Check left of i
        bool left = true
        for(j = i-1 ; j >= i-B ; j--) {
            if(A[j+1] == A[j]) {
                left = false
                break
            }
        }
        // Check right of i
        bool right = true
        for(j = i+1 ; j <= i+B ; j++) {
            if(A[j] == A[j-1]) {
                right = false
                break
            }
        }
        if(left == true && right == true) {
            ans.add(i)
        }
    }
    return ans
}
```



$[x \quad i-1] = B$

$i-1-x+1 = B$

$x = i-B$

$[i+1 \quad y] = B$

$y-i-1+1 = B$

$y = i+B$

TC: $(N-2B)*(B+B+1)$

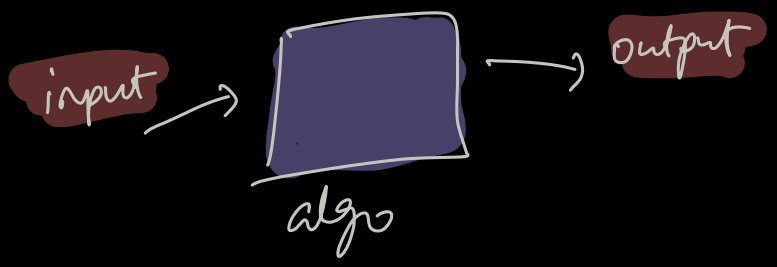
$= O(N^2)$

SC: $O(1)$

$[B \quad N-B-1]$

$\Rightarrow N-B-1-B+1$

$= N-2B$



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

5 | x

B = 1

2B+1 = 3

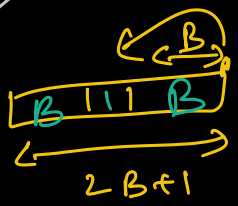
Code:

```
int[] Solve(int a[], int N, int B) {
    len = 2B+1
    curr_window_len = 1
    ans = []
    if(B==0) {ans.add(0)}
    for(i=1; i<n; i++) {
        if(A[i-1] == A[i]) {
            curr_window_len = 1
        }
        else {
            curr_window_len++
        }
        if(curr_window_len >= len) {
            ans.add(i-B)
        }
    }
    return ans
}
```

edge case

TC: O(N)

SC: O(1)



B = 1 2

len = 3

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

i

cwl: 4

ans: (1, 2, 3, 6, 7)

$B = 0 \Rightarrow 2B+1 = 1$

Break: 10 mins : 8:45 am

Magic No.

5, 25, 30, 125, 130, 150, 155

625, 630, 650, 655

625 + 5 | 625 + 25 | 625 + 25 + 5

$5 + 25 + 125$

$5^1 \quad 5^2 \quad 5^3$

Q3: Interview Prob -1, HW Q1

Christmas Trees

Given an A[], having height of trees & B[] having cost of each tree.

Choose 3 trees p, q, r st.

$A_p < A_q < A_r$ where $p < q < r$

Minimise the cost $B_p + B_q + B_r$

0	1	2	3	4	5
A = [1,	6,	4,	2,	6,	9]
B = [2,	5,	7,	3,	2,	7]

$p < q < r$

0	3	4
0	3	5
0	1	5
2	4	5
3	4	5

$A_p < A_q < A_r$

1	2	6
1	2	9
1	6	9
4	6	9
2	6	9

cost

$B_p + B_q + B_r$

$2 + 3 + 2 = 7$

$2 + 3 + 7 = 12$

$2 + 5 + 7 = 14$

$7 + 2 + 7 = 16$

$3 + 2 + 7 = 12$

0	2	4	2+7+2 = 11
0	2	5	2+7+7 = 16
0	4	5	2+2+7 = 11
1	4	6	
1	4	9	
1	6	9	

Brute force:

↳ 3 nested loops

⇒ check for $A_p < A_q < A_r$

↳ minimise $(B_p + B_q + B_r)$

Keep in middle

$p < q = i < r$

→ For every element i, keep i in middle

Check left

find elements lesser than A[i]

&&

min (Bp)

Check right

find elements greater than A[i]

&&

min (Br)

if found Bp && Br

ans = min (ans, Bp + Br + Bi)

Code's To-do

↳ ask TA

↳ ask Batchmates

↳ ask Me

Q:4 N/3 majority element

Interview Prob-2, HW Q1

Given a[], find if any element occurs more than N/3 times.

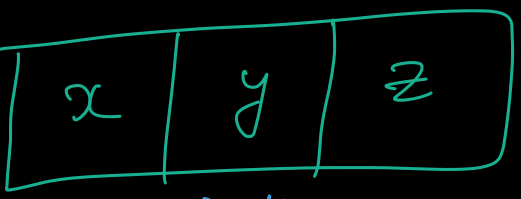
A = [1 2 3 1 1] N = 5

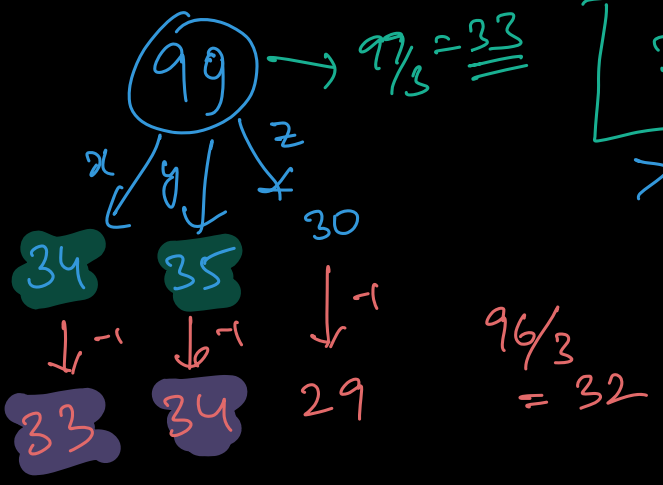
freq(1) = 3 5/3 = 2

ans = 1

maj elements = 2

99 → 99/3 = 33





Eg: A[] = 2 2 2 1 4 2 3 2 1 4 4 4 3 4

Me1 = 2

freq1 = —

Me2 = 4

freq2 = —

// Cross-check

N = 14

14/3 = 4

freq(2) = 5

freq(4) = 5

```
int repeatedNum(int a[], int N) {
    freq1 = 0      freq2 = 0
    ele1 = 0       ele2 = 0

    for(i=0; i<n; i++) {
        if(freq1 == 0 && A[i] != ele2) {
            ele1 = A[i]
            freq1 = 1
        }
        else if(freq2 == 0 && A[i] != ele1) {
            ele2 = A[i]
            freq2 = 1
        }
        else if(A[i] == ele1) {
            freq1++
        }
        else if(A[i] == ele2) {
            freq2++
        }
        else {
            freq1--
            freq2--
        }
    }

    // Cross-check
    c1 = 0, c2 = 0
    --) To-do
}
```

2 distinct elements

TC: O(n)
SC: O(1)

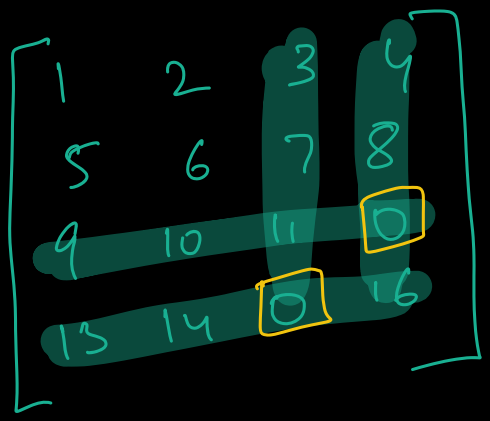
Qs: 2D-Matrix, HW Qs

Row to Col to Zero

Given 2D matrix, make row (=i) & col (=j) as 0 if A[i][j] = 0

Constraint:- $1 \le A_{ij} \le 10^3$

Eg: A =



N×M

$$= \begin{bmatrix} 1 & 2 & 0 & 0 \\ 5 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

ideal (with extra space)

Row[N] = {}
Col[M] = {}

① Traverse the matrix (i, j), if A[i][j] = 0 {
- row[i] = 1
- col[j] = 1

}

TC: O(N×M)
SC: O(N+M)

② Traverse the matrix (i, j):
if(row[i] == 1 || col[j] == 1)
A[i][j] = 0

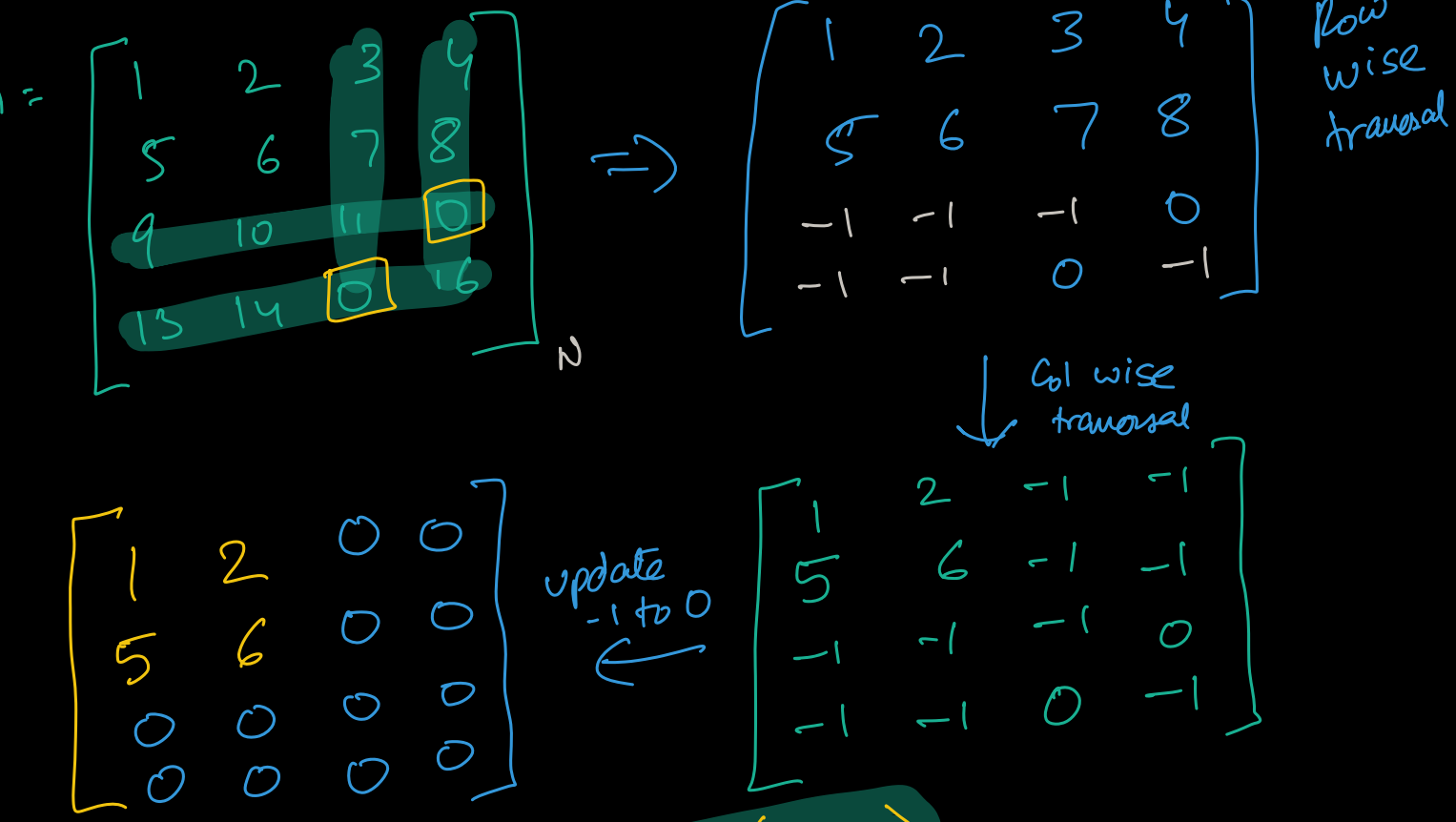
idea 2 (w/o extra space)

↳ Traverse the matrix, row-wise (i, j)
if A[i][j] = 0
change all non-zero elements in row[i] to -1

→ Traverse the matrix, col-wise
if (A[i][j] == 0)
make all non zero col elements -1

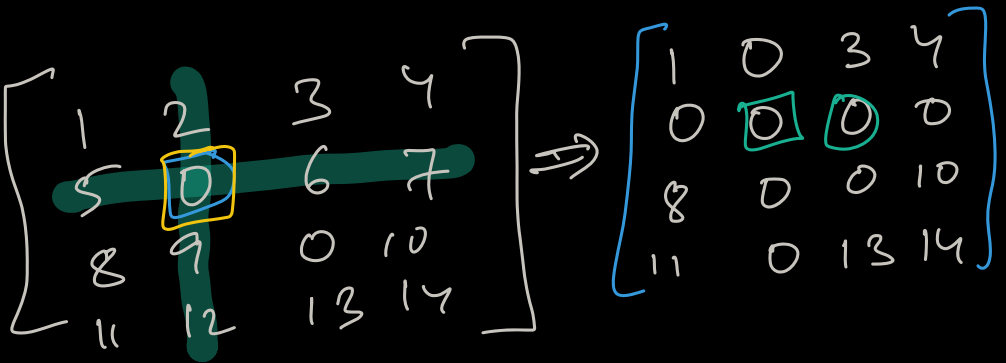
j

→ Traverse the matrix
change -1 to 0.



TC: O(N x M)
SC: O(1)

→ A=



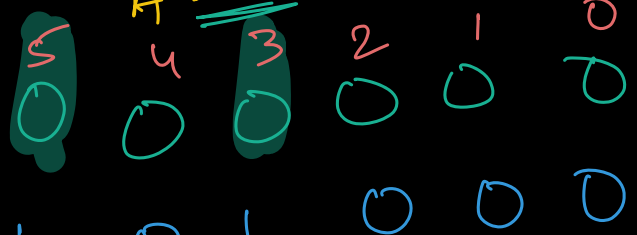
Doubt :

A B

Set A^{th} bit in 0

B^{th} bit in 0

A = 3 , B = 5



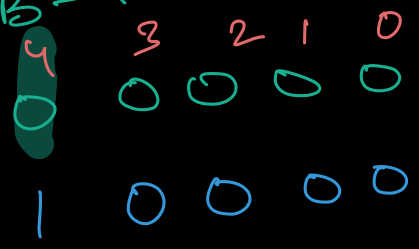
Set ⇒ 1

unset ⇒ 0

$\Rightarrow 2^5 + 2^3 = 32 + 8 = 40$

A = 4

B = 4



$2^4 = 16$

Bit manip 1

How Q3

Subs arr with OR = 1