Arrays : Interview Problems

Apr 8, 2022

AGENDA:

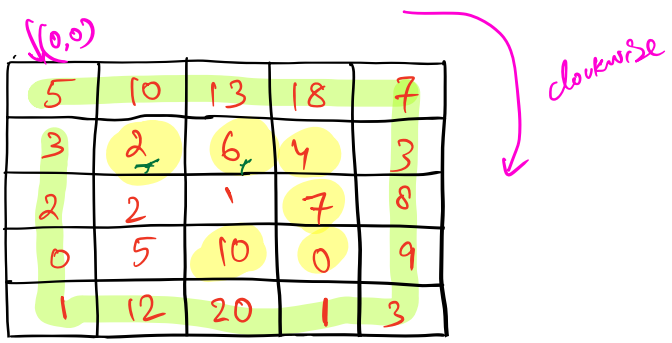
- 2-3 very interesting problems :D

Q.

Given a matrix of size NXN, print the boundary of the matrix in clockwise direction

5	10	13	18	7
3	2	6	4	3
2	2	1	7	8
0	5	10	0	9
1	12	20	1	3

Inp =



Out = 5 10 13 18 7 3 8 9 3 1 20 12

1 0 2 3

- 0th row → O(N)
- N-1th column → O(N)
- N-1th row (reverse) → O(N)
- 0th column (reverse) → O(N)

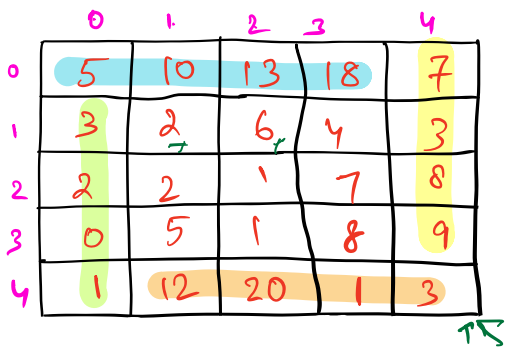
O(N).

4N-4 nos. are printed

O(N).

Approach / Observations.

1.



NxN

{ i=0, j=0 } (j++)

{ i=0, j=4 } (i++)

i=4, j=4

i=4, j=4 j--

i=4, j=0

{ i=0, j=0 } i--

* 4 legs in the boundary.
* Print each leg.

* Quiz No. of elements in each leg of the boundary = N-1

* No. of elements in each boundary = N-1

* Once we are done with printing 1st leg, the ending value of previous leg will be start value of next leg.

```
N=10
i=1          9<10
while(i<N):
    i+=1     9
print(i)
```

i=10

9 and 10.

```
int i;
for(i=1; i<=10; i++)
    // —
print(i)
```

11

N=1

Pseudo-code

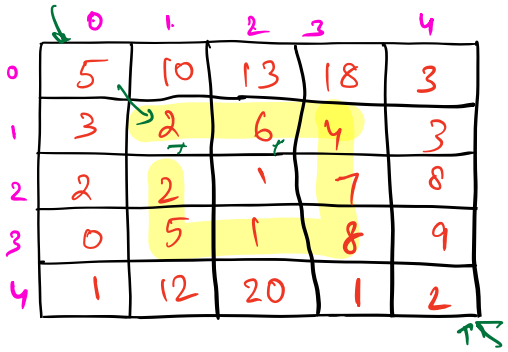
```
//Print 1st leg
i=0, j=0
for _ in range(N-1):      // Run this N-1 times.
    print(mat[i][j])
    j++

// i=0, j=N-1

// Print 2nd leg.

for _ in range(N-1):
    print(mat[i][j])
    i++

// i=N-1, j=N-1

for _ in range(N-1):
    print(mat[i][j])
    j--

// i=N-1, j=0
```

```
for _ in range(N-1):
    print(mat[i][j])
    i--
```

// i = 0 , j = 0

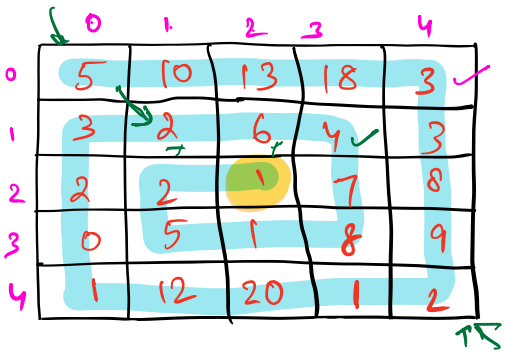
```
i+=1
j+=1
N = N-2
```

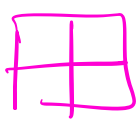
→ Takes us into 1 step inner.



N x N → N-2 x N-2

Q. Given a matrix. print elements in spiral order.

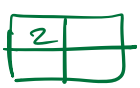




2x2

```
while( N > 1 ) :        <- N/2
    <prev-code>         // Prints outer boundary.
    i+=1
    j+=1                -> O(N)
    N=N-2
```

N → N-2 → N-4



```
if N%2==1:
    print ( arr[N//2] [N//2] )
```

N/2.
5/2 = 2

↳ center element.

Q. Max no. of consecutive 1's.

Given a binary string, at most replace a 0 with a 1. find max no. of consecutive 1's in the resultant string.

Binary string. "0100111011100"

3 4 → (8)

Quiz

0 1 1 1 0 1 1 0 1 1 0

4 6 5 3.

Ans = 6

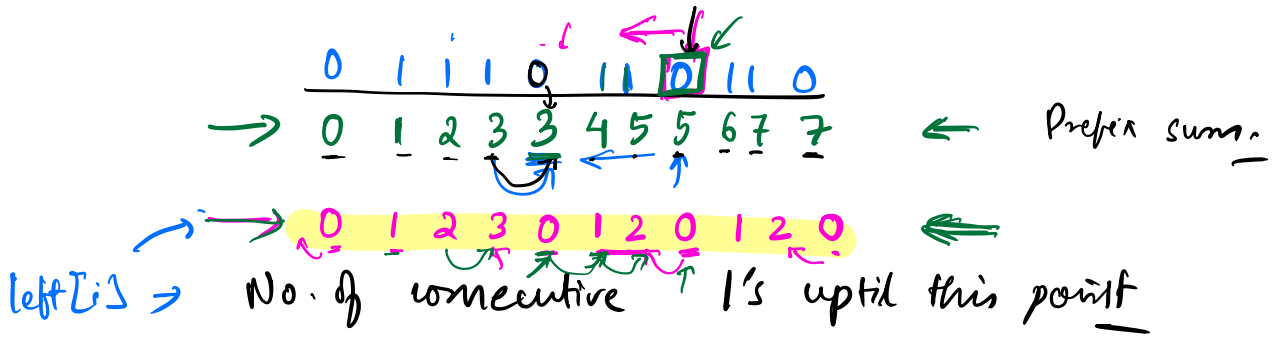
Brute force.

Pseudo-code.

```
ans = 0
for i in range(0, n):
    if ar[i] == '0':
        L = No. of consecutive 1's to left
        R = No. of " " right.
        total = L + R + 1
        ans = max(ans, total)
```

Time consuming → $O(N^2)$

$O(N^2)$ ✗



arr[i] == '0':
↳ L = left[i-1]
prefix

1 0 1 1 1 0
left[i] = A[0]
A[0]
↳ left[0] = ord(arr[0]) - 48
48/49

// Build the left[i] array.
if arr[0] == '0': left[0] = 0 else left[0] = 1

```
for i in range(1, n):
    if s[i] == '0':
        left[i] = 0        // Reset!
    else:
        left[i] = left[i-1] + 1
```

// Build the right[i] array. // Suffix array.

// Handle the case i = n-1:

```
for i in range(n-2, -1, -1):
    if s[i] == '0':
        right[i] = 0
    else:
        right[i] = right[i+1] + 1
```

1 0 1 1 0 1 1
1 0 1 1 0 0 0
1

Final answer.

```
ans = 0
for i in range(n):
    if s[i] == '0':
        L = left[i-1]
        R = right[i+1]
        total = L + R + 1
        ans = max(ans, total)
if ans == 0: return len(s)
return ans.
```

i==0 ? 0 : left[i-1]
i==n-1 ? 0 : right[i+1]

Corner cases.

① 

ans = 1

② 1 1 1 1 1 1 1 1 1 1 1

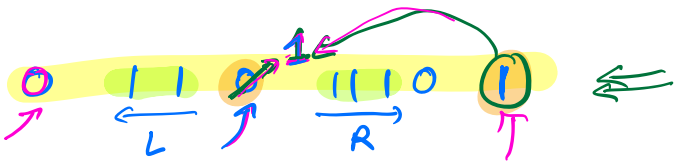
ans = 0. X

ans = len(arr)

Break till 10:40.

Q. Enhancement to the above question:

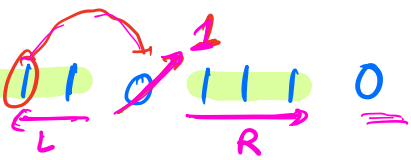
You can replace at most one 0 with a 1, but with a 1 present in array itself. (Extra 1's cannot be used)



Ans = 6

K = Count of 1's.

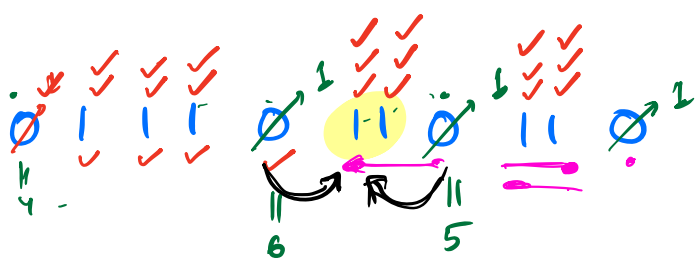
e.g. 0 0 1 1 0 1 1 1 0



Ans = 5

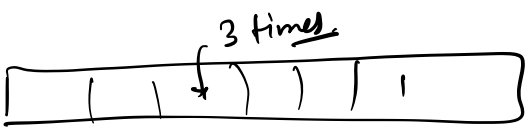
if L+R+1 <= count:
    total = L+R+1
else
    total = L+R.

H.W.



R=3

arr[i] → Max. 3 times



O(N)
3N
O(N)

N*N.
N*3

Q.3-

Given an array, count no. of triplets s.t. $i<j<k$ and $arr[i] < arr[j] < arr[k]$.

0	1	2	3	4	5
4	1	2	6	9	7

4 6 9
4 6 7

1 2 6
1 2 9
1 2 7
1 6 9
1 6 7

2 6 9
2 6 7

9

Brute force.

3 loops. count = 0

```
i — [0, n-1]        ⇐ // Not python
    j — [i+1, n-1]
        k — [j+1, n-1]
            if arr[i] < arr[j] < arr[k]:
                count += 1
```

return count.

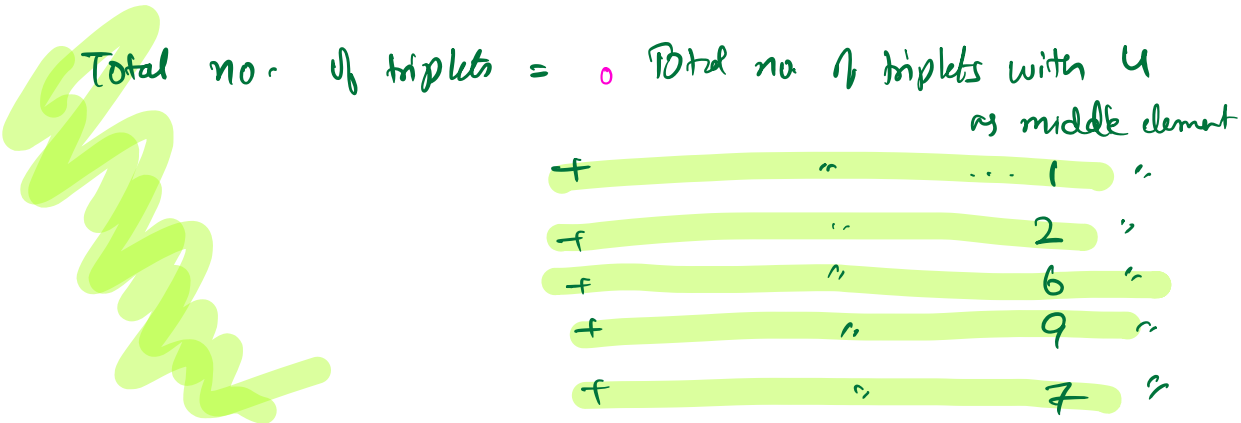
TC: $O(N^3)$. ✓

Expecting a $O(N^2)$ solution.
$O(1)$: SC.

Hint: Take every element as middle element of the triplet.

4 1 2 6 9 7

Total no. of triplets = 0 Total no of triplets with 4 as middle element

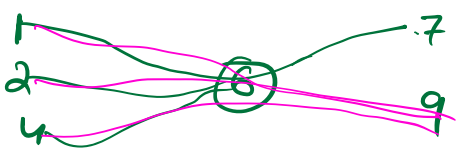


No. of triplets with 6 as the middle element = ?

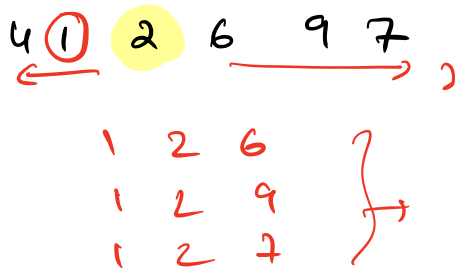
4 1 2 6 9 7.

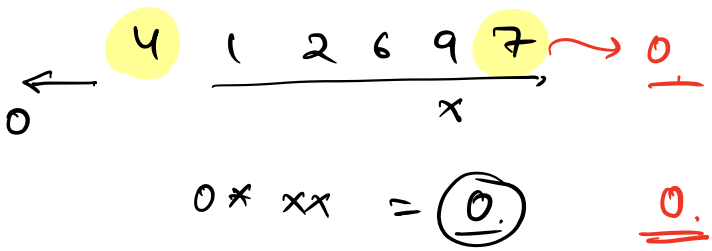
<6. >6.

1,2,4. 9,7



3 * 2 = 6





Pseudo-code

```
sum=0
for i in range(n):
    middle = arr[i]

    count_l = 0
    for j in range(0,i):
        if arr[j] < middle:
            count_l += 1

    count_r = 0
    for j in range(i+1,n):
        if arr[j] > middle:
            count_r += 1

    ans = count_l * count_r
    sum += ans
```

N times.

O(N)

O(N)

return sum.

$O(N^2)$

** Ideal solution → $O(N \log N)$

{Segment Trees.}