Given a 2D matrix of size N x N

Google Adobe Oracle Apple FB

simulation problems.

N=6

i \ j	0	1	2	3	4	5
0	4	1	3	6	9	7
1	14	21	32	8	5	16
2	19	18	40	25	11	12
3	15	42	33	13	17	24
4	50	16	61	52	62	22
5	64	28	54	27	63	65

[0,0] — [0,4] → N-1

[0,5] — [4,5] → N-1

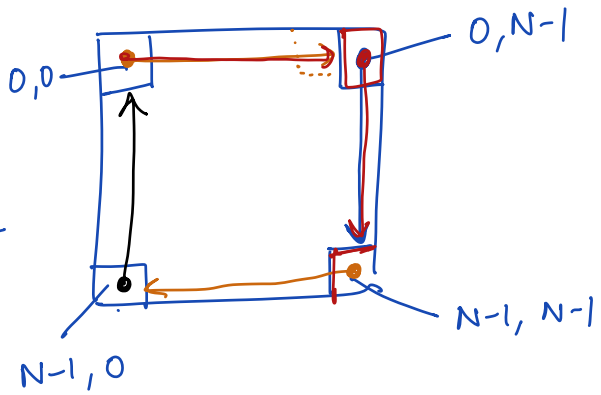
[5,5] — [5,1] → N-1

[5,0] — [1,0] → N-1

N=6 i=0 j=0 N=4 i=1 j=1 N=2 i=2 j=2 N=0

Print

4 1 3 6 9 7 16 12 24 22 65 63 27 54 28 64 50 15 19 14



① Every for loop will run for N-1 times

② As soon as we print 1 set, we reach to the starting of next set.

N=6

i=0 j=0

```
while (N > 1){

    for(K = 1; K<N; K++){
        print(mat[i][j])
        j++
    }

    for(K=1; K<N; K++){
        print(mat[i][j])
```

i=0 j=0

K=1 i=0 j=1

K=2 i=0 j=2

K=3 i=0 j=3

K=4 i=0 j=4

K=5 i=0 j=5

K=6

```
        i++
    }

for(K=1; K<N; K++){
    print(mat[i][j])
    j--
}

for(K=1; K<N; K++){
    print(mat[i][j])
    i--
}

// i=0  j=0
N = N-2
i++  j++
}

if(N==1){
    print(mat[i][j])
}
```

i=0 j=5

K=1 i=1 j=5

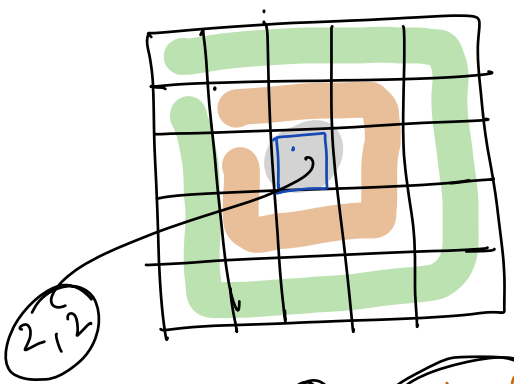
K=2 i=2 j=5

K=3 i=3 j=5

K=4 i=4 j=5

K=5 i=5 j=5

K=6



i=0 N=5
j=0

i=1 N=3
j=1

i=2 N=1
j=2

N=6

N=4

N=2

N=0

TC → O(N*N)

SC → O(1)

Barclays & Citicorp

Amazon

→ none or 1

Ques. Given a Binary String, atmost replace one 0 with 1. Find the max consecutive 1s.

MAX CONSECUTIVE 1s.

Eg 0 1 1 1 0 1 1 0 1 1 0

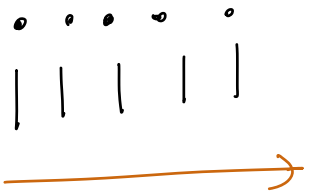
4 6 5 3

Eg 1 1 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1

i-1 i i+1

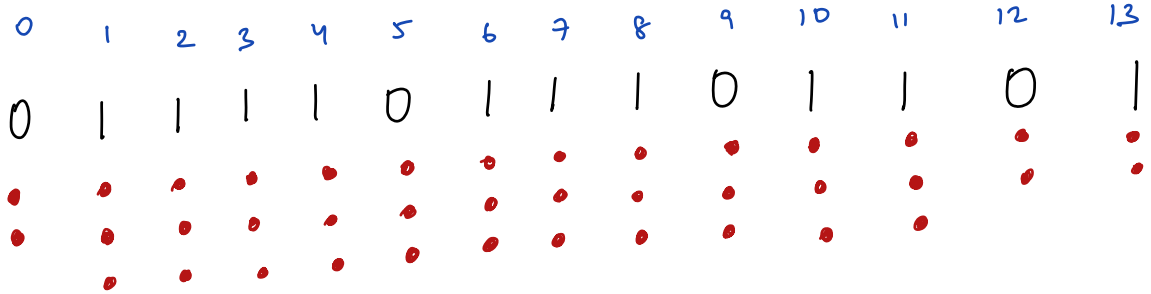
```
ans = 0
flag = false
for(i=0 ; i<N ; i++){
    if( A[i] == '0'){
        flag = true
        // l = count of consecutive 1s on left
        l=0
        for(j=i-1 ; j>=0 ; j--){
            if(A[j] == '1') l++
            else break
        }
        // r = count of consecutive 1s on right
        r=0
        for(j=i+1; j<N ; j++){
```

```
for(j  if (A[j] == '1') k++
       else break
}
ans = max(ans, l+k+1)
}
}
if(flag){
    return ans
}
else return N
```



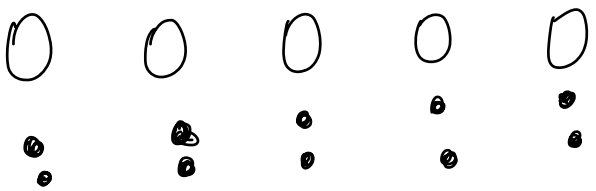
ans = 0

ans = N



At max, we're visiting any ele → 3 times

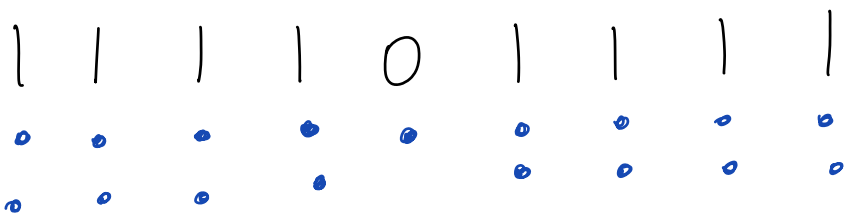
N → 3N → O(N)



$l = 0 \quad r = 0$

ans = 1

$l = 0 \quad i = 0$

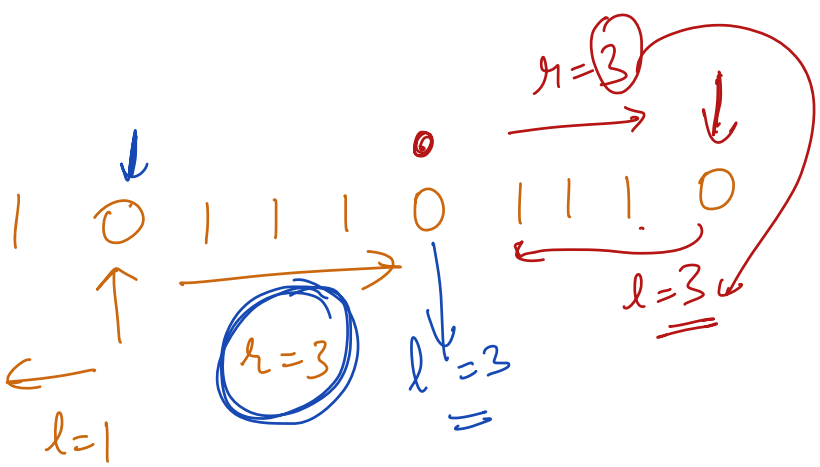


Take more examples & try.

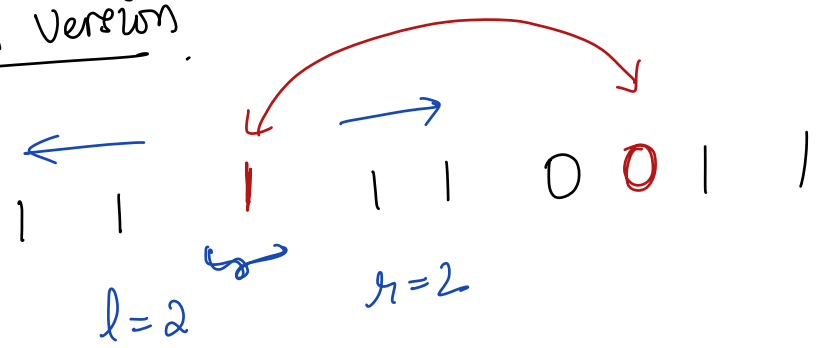
2 pointers

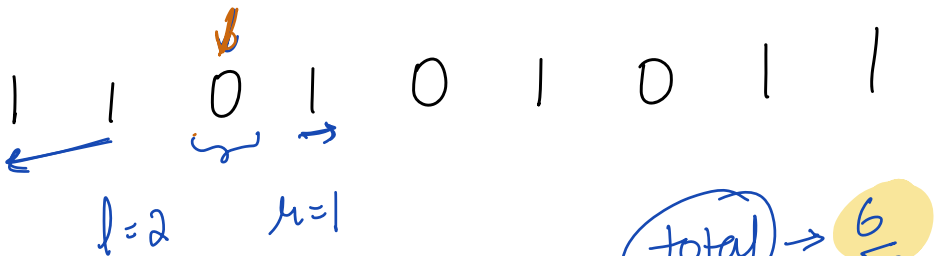
optimization in terms of iteration!

HINT:



Modified Version.

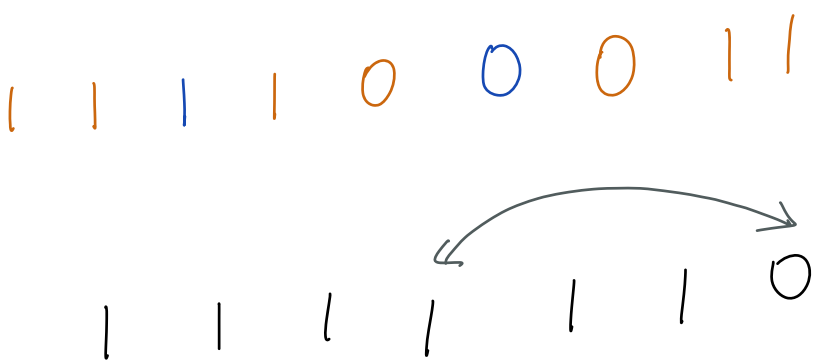




total → 6

l + r = 3

l + r < total



Exactly same just like previous
You still have to find consecutive
1s on left & consecutive 1s on right.

l + 1 + r

1 1 0 1 1 0 1 1 1 0 1

$l = 2 \quad r = 3$

count → 8

$l = 2 \quad r = 2$

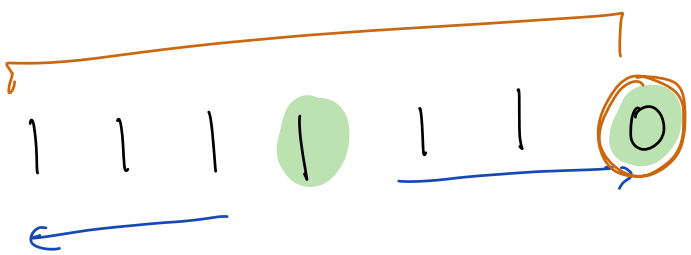
$l + r < count$

$length = l + r + 1$

$l + r = 4$

$length = l + r + 1$

$= 5$.



count = 6

$l = 3 \quad r = 3$

$l + r == count$.

$l + r + 1$?

$3 + 3 + 1 = 7$

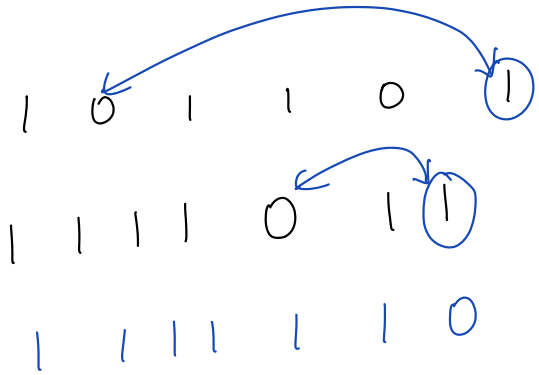
if $l + r < count$ ans → $l + r + 1$

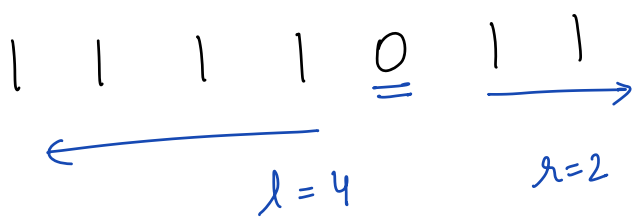
else ans → $l + r$.

```
// count one no. of 1s → C1
ans = 0
flag = false
for(i=0 ; i<N ; i++){
    if( A[i] == '0' ){
        flag = true
        // l = count of consecutive 1s on left
        l = 0
        for( j = i-1 ; j >= 0 ; j-- ){
            if(A[j] == '1') l++
            else break
        }
        // r = count of consecutive 1s on right
        r = 0
        for( j = i+1 ; j < N ; j++){
            if (A[j] == '1') r++
            else break
        }
        if ( l+r < C1) len = l+r+1
        else len = l+r
        ans = max (ans, len)
    }
}
if(flag){
    return ans
}
else return N
```

Break time

1:45





$l + r = 6$

$l = 4 \quad r = 2$

$l + r < c1$

$6 == c1$

$l = 6$

$l = 2 \quad r = 0$

$l + r < c1$

$2 < 6$

$len = 2 + 0 + 1 = 3$

ans = 6

Golmaa

Que. No. of Triplets

$i < j < k$

$A[i] < A[j] < A[k]$

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

3	5	4
6	7	9

X

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

Ans = 9.

4 2 6

0 2 3

X

0	1	2	3	4
2	6	9	4	10

HINT

$A[i] < A[j] < A[k]$

2, 6, 9

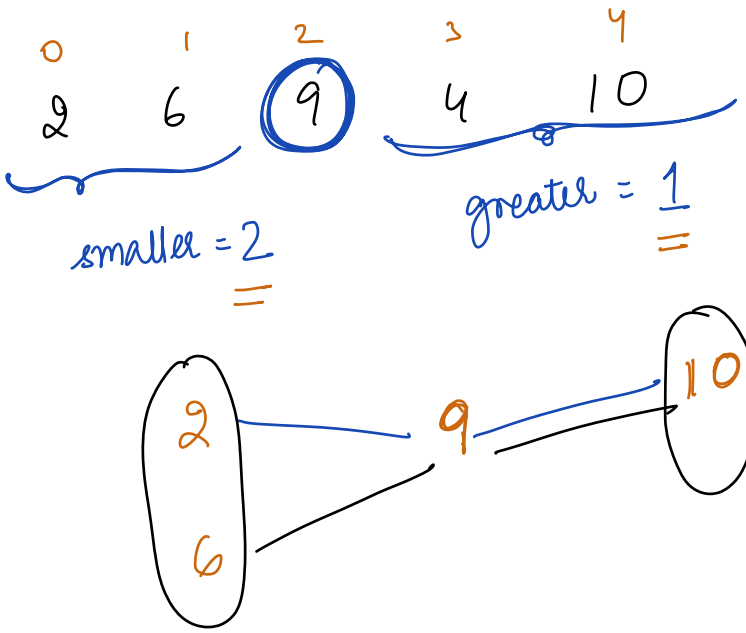
2, 6, 10

2, 9, 10

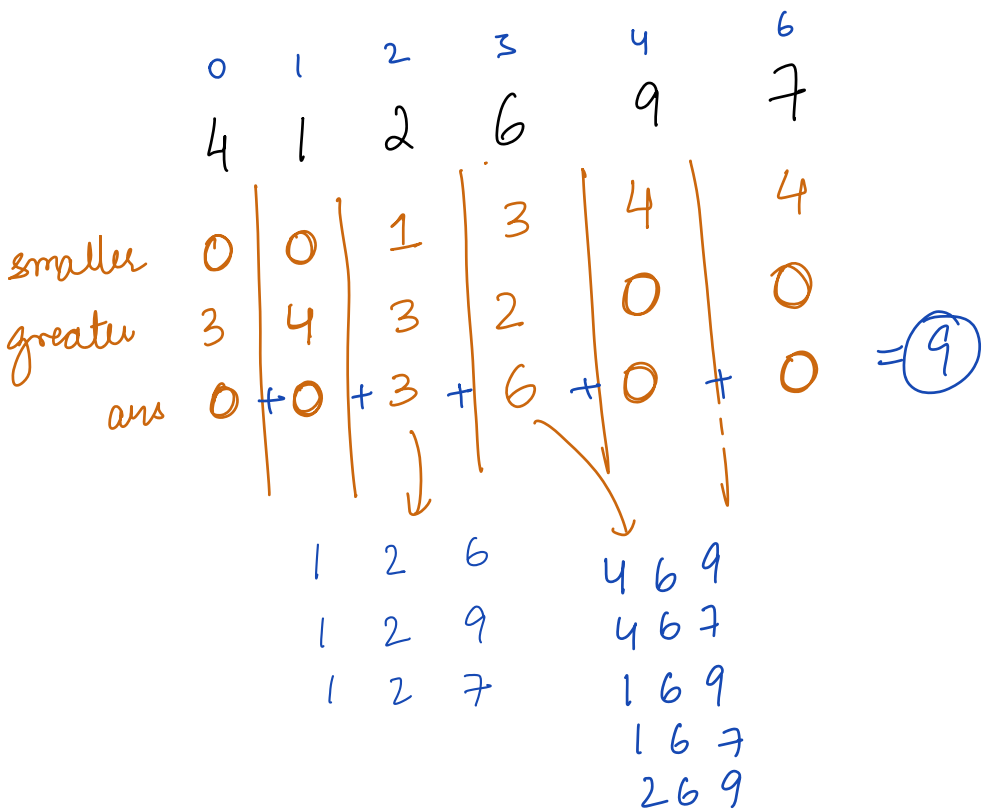
2, 4, 10

6, 9, 10

Ans = 5.



smaller * greater .



```
ans = 0
for ( i = 0 ; i < N ; i++ ) {
    // l = count of smaller to left
    for( j = i-1 ; j >= 0 ; j--){
        if(A[j] < A[i]) l++
    }
    // r = count of greater to right
    for (j = i+1 ; j < N ; j++){
        if (A[j] > A[i]) r++
    }
    ans = ans + (l * r)
}
return ans.
```

TC $O(n^2)$

SC $O(1)$

BST → $O(N \log N)$

TLE

Google → Q1 → observations → coded → TLE

→ Q2

45 min

TLE → optimization 1. → TLE → optimization 2 → TLE

TLE.

1 GHz → 1 B instructions / sec.

10^9

```
for(i=1 ; i<=100 ; i=i+2) {
    print(i)
}
```

(1) (1) (2) (1)

100 × 4

400

4 × 100 → 400 instructions.

10 instructions = 1 iteration

10^9 instruction $= \frac{1}{10} \times 10^9 = 10^8$ iterations.

100 instructions $=$ 1 iteration

10^9 instructions $= \frac{1}{10^2} \times 10^9 = 10^7$

$[10^7 - 10^8]$ iterations

Constraints

$1 \leq N \leq 10^5$

$1 \leq A[i] \leq 10^9$

$O(N^2)$ → 10^{10} ✗

$O(N)$ → 10^5 ✓

Constraint

$1 <= N <= 10^3$

$O(N^2)$ → $(10^3)^2 = 10^6$ ✓

Next Topic → Bit Manipulation

Take

1 0 1 1 0 1 0 1 1

N^2

Iterations