Interview Problems

Jul 26, 2023

AGENDA

- Max no. of consecutive 1's
- No. of triplets
- Josephus problem

Tips on constraints

10^8 Iterations are allowed..

$1 <= n <= 10^6 \longrightarrow O(n),\ O(n \log n)$

$1 <= n <= 20 \longrightarrow O(2^n),\ O(n!)$

$1 <= n <= 10^{10} \longrightarrow O(\log n),\ O(\sqrt{N})$

1. Amazon, Microsoft, Adobe.

Given an array of 0's and 1's, you are allowed to replace only one 0 with 1. Find the max. no. of consecutive 1's that can be obtained after making the replacement.

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

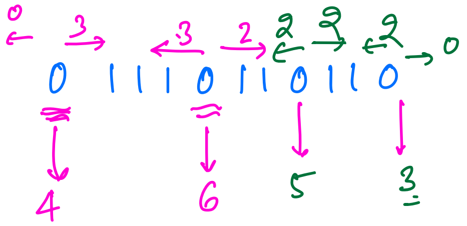
1 (under the struck 0)

Ans=5.

1 1 0 1 1 0 1 1 1

5 (under the first 0) 6 (under the second 0)

Ans= 6



Ans=6

B.F.

1 1 1 0 1 1 1 1 0 11 0 0 111 0 1 1

```
ans=0
for (int i= 0; i< n; i++)
{
        if (arr[i] == 0)
        {
                L= consecutive 1's to the left
                R=            "          "   right
                total = L+R+1
                ans= max(ans, total)
        }
}
return ans
```

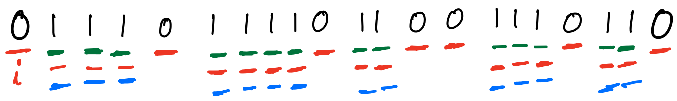
```
ans=0
cnt_ones =   // count no. of 1's   present in the array.
if(cnt_ones ==n)
        ans=n
for (int i= 0; i< n; i++)
{
          if (arr[i] == 0)
          {
                //L= consecutive  1's  to  the  left
                j=i-1 , cnt=0
                while(j>=0  &&  arr[j]== 1)
                {      cnt++;
                       j-- ;
                }
                L=cnt
                 //   R=               "           "       right
                j=i+1 ,cnt=0
                while ( j<n  &&  arr[j]== 1)
                {
                       cnt++
                       j++
                }
                R=cnt
          total = L+R+1
          ans= max  ans, total)

}
return ans
```

1 1 1 0 1 1 1 1 0 11 0 0 111 0 11

i j

Time complexity



* Every element is getting accessed max 3 times.

T.C. → O(3n)

= O(n)

1 1 1 1 1 1 1 1 1 0

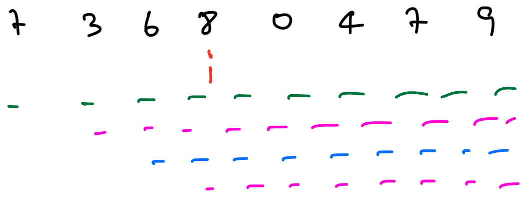
i

0 1 1 1 1 1 1 1 1 1 1 1 = O(n)

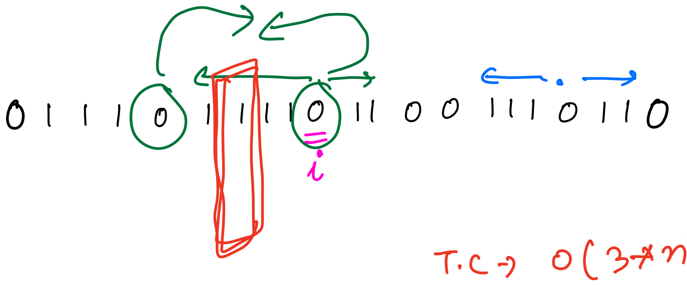
i

0 1 1 1 1 1 1 1 1 0

T.C. → O(n*n)

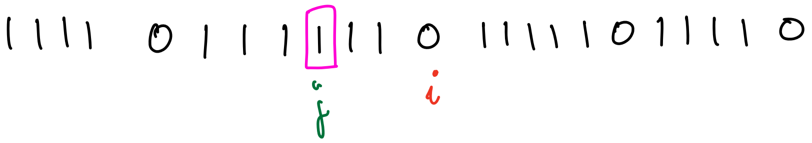


```
(i=0; i<n; i++)
{
    (j=i ; j<n; j++)
    {
```



T.C → O(3*n) = O(n)

i, j



Every elem gets accessed max 3 times.

T.C. → O(3n) = O(n)

Q. Now, you are allowed to only swap a 1.

1 0 1 1 0 1 → L+R+1

ans = 4

1 1 $\cancel{0}$ 1 1 1

1

ans = 5

1 1 1 1 1 1 1 1 1 $\cancel{0}$ _

i

L + R

1 1 0 1 1 1

L = 2
R = 3

$\boxed{L+R = 5}$

1 1 0 1 1 1 0 1

1

~~L+R~~ L = 2
R = 3

L+R+1

0 1 1 $\cancel{0}$ 1 1 1 0 1

1

L = 2

R = 3

Total 1's = 6

```
ans=0
cnt_ones =    // count no. of 1s   present in the array
if(cnt_ones ==n)
        ans=n
for (int i= 0; i<n; i++)
{
        if (arr[i]== 0)
        {
            //L= consecutive 1's to the left
            j=i-1 , cnt=0
            while(j>=0 && arr[j]== 1)
            {       cnt++;
                    j--;
            }
            L=cnt
              //  R=          "       "    right
            j=i+1,cnt=0
            while ( j<n && arr[j]== 1)
            {
                    cnt++
                    j++
            }
            R=cnt
            if(L+R+1 <= cnt_ones)
                    total=L+R+1
            else
                    total=L+R
            ans= max ans, total)
}
return ans
```

T.C.→ O(n)

S.C.→ O(1)

Break till 8:22 AM

Q. 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

1 2 6

1 2 9

1 2 7

4 6 9

4 6 7

1 6 9

1 6 7

2 6 9

2 6 7

2 6 9 4 10

2 6 9

2 6 10

2 9 10

6 9 10

2 4 10

B.F.

0	1	2	3	4	5
4	1	2	6	9	7
	i	j			k

```
cnt=0
for(int i=0; i<n; i++)
{
    for(j=i+1 ;j<n;j++)
    {
        for(k=j+1 ; k<n; k++)
        {
            if (arr[i] <arr[j] &&
                arr[j] < arr[k]
                cnt++
        }
    }
}
return cnt
```

T.C. → $O(n^3)$

S.C. → $O(1)$

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

_ _ _

middle element

* Try to place each element as the middle element

4 6 7

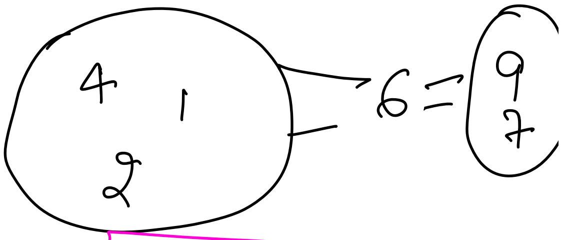
4 6 9

2 6 7

2 6 9

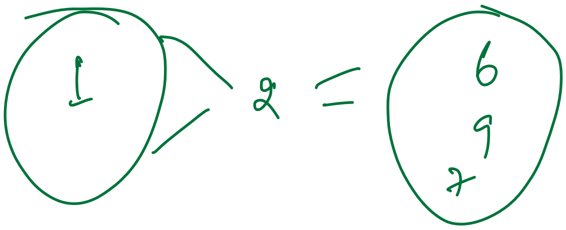
1 6 7

1 6 9



3 * 2 = 6

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



1 * 3 = 3

1, 2, 6

1, 2, 9

1, 2, 7

0	1	2	3	4	
2	6	9	4	10	= 2+2+1 = 5
0	2	2	1	0	

* Pick every element as the middle element of the triplet.

2 →

6 → 2 6 9 10

4. 2 4 10

9 → 2 6 9 10

2 6 4

Code

4 1 2 6 9 7

```
ans=0
for(int i= 0;i<n;i++)
{
        // Consider arr[i] as the middle element
           of triplet.

    cnt-left=0
    for(int j=i-1 ; j>=0;j--)
    {
        if(arr[j] < arr[i])
            cnt_left++
    }
```

Count no. of smaller elements on left ←

```
    cnt-right=0
    for(int j= i+1 ;j<n;j++)
    {
        if(arr[j] > arr[i])
            cnt-right++
    }
```

Count no of larger elements on right. ←

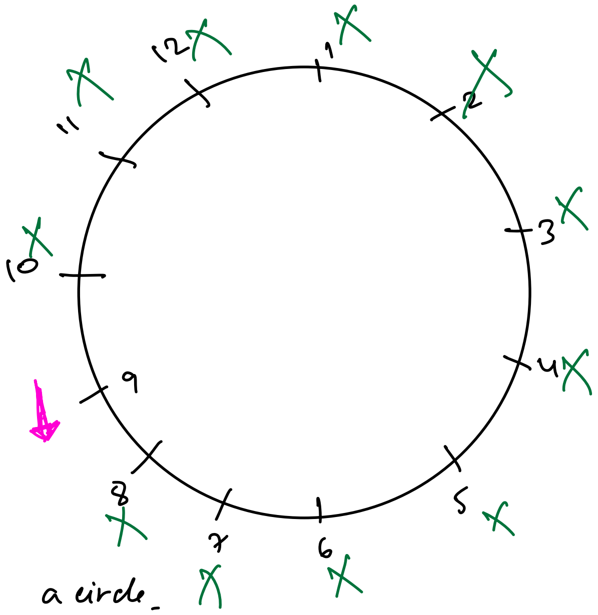
```
    ans += cnt_left*cnt_right

}
return ans.
```

T.C. → $O(N^2)$

S.C. → $O(1)$

Q: Josephus problem

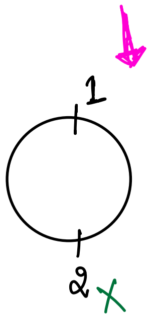


N people in a circle.

Last man standing → 9

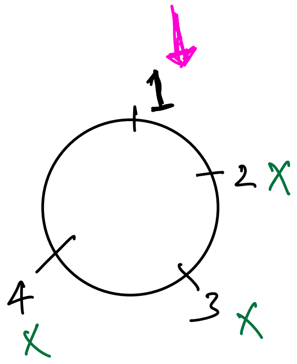
who is the last man standing?
(given N)

N=2



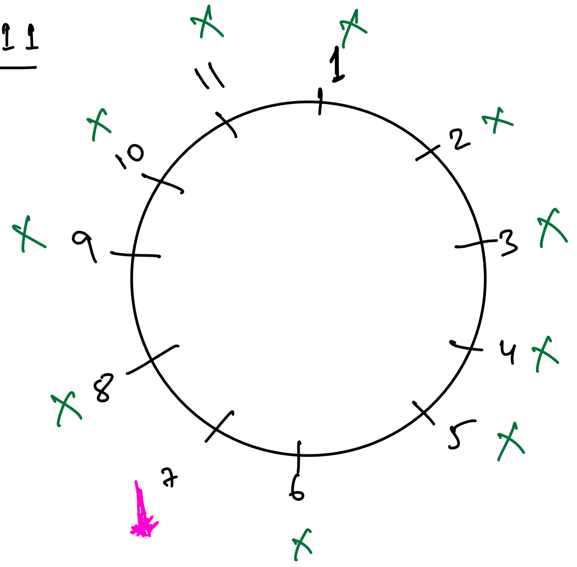
ans=1

N=4

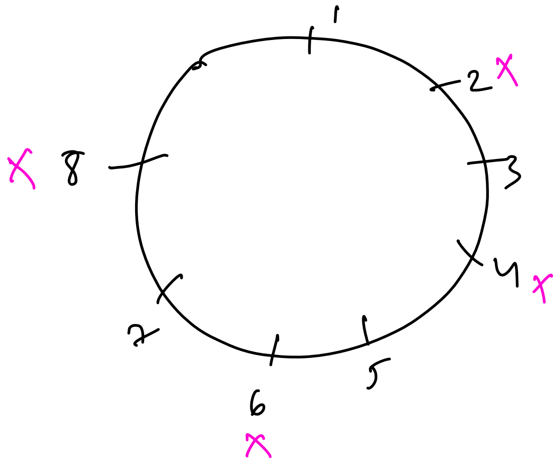


ans=1

N= 11



N		Winner
N=2	→	1
N=4	→	1
N=8	→	
N=6	→	?XX
N=16	→	1

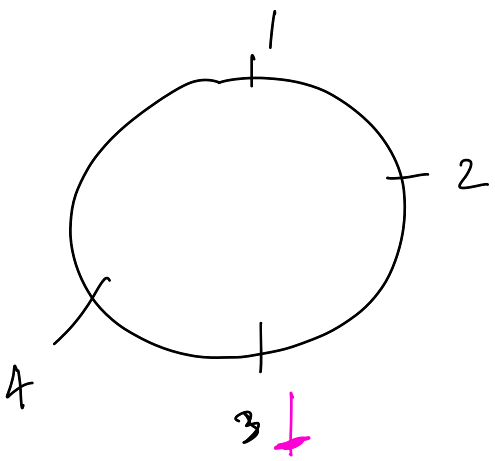


winner is 1

Powers of 2

2, 4, 8, 16, 32, 64, ...

N=4



* if there are 2^k people in the circle, one who has the sword at start wins.

N=11

	no. of people remaining in circle.
1	11
kills 2	
3	10
kills 4.	
5	9
kills 6	
7	8

$\boxed{N=7}$ winner ?

if N = 2, 4, 8, 16, 32, 64.....

Person with sword	no. of people remaining
1	6
3	5
$\boxed{5}$	$\boxed{4}$

$\boxed{Ans = 5}$

$\boxed{N=70}$ → nearest power of 2 <= 70 = $\boxed{64}$

person with sword.	1	3	5	7	9	11	13
no. of people remaining.	70	69	68	67	66	65	64

↓

$\boxed{\text{winner} = 13}$

$int(\log_2 70) = \underline{6}$

$\boxed{2^6 = 64}$

Given N,

find x = Closest power of 2 <= N.

$x = pow(2, int(\log_2 N))$

$37/2 = 18*2$

↓

$\underline{36}$

37 32

↓

$\log_2 37 = 5.....$

↓

2 * 2 * 2 * 2 * 5

32

Reduce N to X people.

Kill N-X persons.

$\boxed{92}$

$\log_2 92 = 6. \ldots$

$2^6 = \boxed{64}$

	Sword is with
After 0 people get killed,	1
1 " ,	3
2 " ,	5
3 " ,	7
4 " ,	9
k " ,	$2*k+1$
$\boxed{N-X}$ " ,	$2*(N-X)+1$

winner = $2*(N-X)+1$

when $X = pow(2, int(\log_2 N))$

N=11

$X = pow(2, int(\log_2 11))$

$= pow(2, 3)$

$= 8$

winner = $2*(11-8)+1$

$= 2*3+1$

$= \boxed{7}$

$N=11$

8

$11 \longrightarrow 8$

$N=70$

$$X = pow(2, \ int(\log_2 N))$$
$$= pow(2, 6)$$
$$= 64$$

← I know the solution when 64 people are remaining

$$N-X = 70-64$$
$$= \boxed{6}$$

← 6 people needs to be killed,

$2*6+1 = 13$ holds the sword when 6 people gets killed.

13 holds the sword when 64 people are remaining.

Contest

↓

Bit manipulation,
Sorting
and modular arithmetic