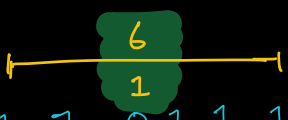
Q) Given an array of 1 & 0s. We can replace one of the 0 with a 1. Return the count of max consective 1's in the array.



Ex1 : 1 1 0 1 1 0 1 1 1 ⇒ ans = 6

Ex2 : 0 1 1 1 0 1 1 0 1 1 0 = ans = 6

OBSERVATION !

For every 0, find

consective 1's on left = l

consective 1's on right = r

consective 1's if this 0 is replaced = l+r+1

1 0 0 1 1 0 0 0

1: l = 1, r = 0, len = 2

0: l = 0, r = 2, len = 3

0: l = 0, r = 0, len = 1

Pseudo Code

```
if (n == 0) return 0;
int ans = 0;
for (int i = 0; i < n; i++) {
    if (arr[i] == 0) {
        l = 0
        for (int j = i-1; j >= 0; j--) {
            if (arr[j] == 1)
                l++
            else
                break;
        }
        r = 0
        for (int j = i+1; j < n; j++) {
            if (arr[j] == 1)
                r++;
            else
                break;
        }
        ans = max(ans, l+r+1);
    }
}
if (ans == 0) return n;
else return ans;
```

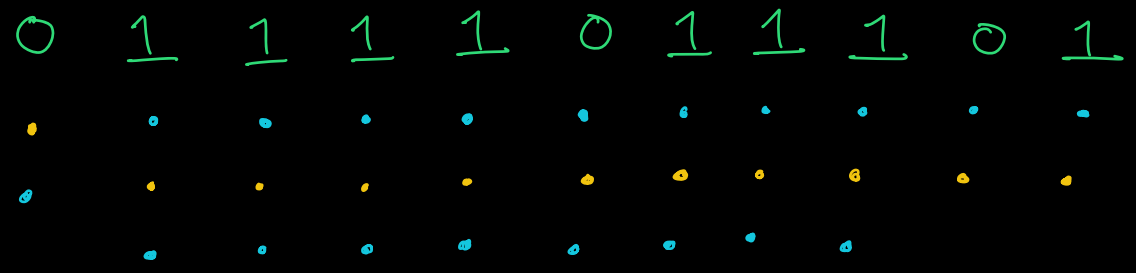
i = 0

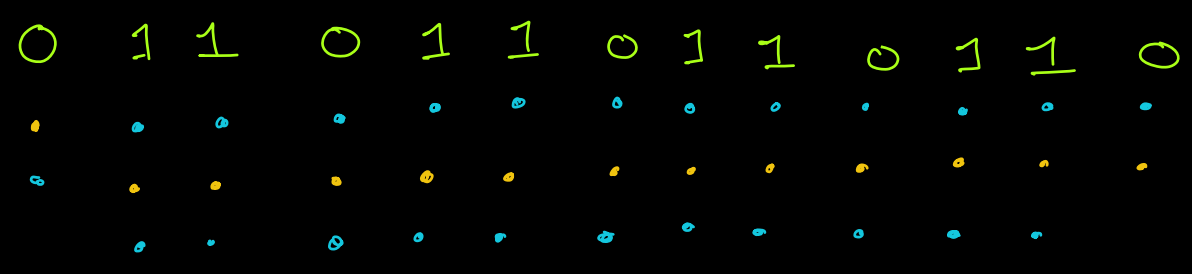
j = i - 1

l[i] ⇒ r[i] =

Edge Case

1 1 1 1 1





↳ main loop visit

↳ count for L

↳ count for R

↳ main loop visit.

↳ count for L

↳ count for R.

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

	0	1	1	1	1	0	1	1	1	0	1
l[i] =	0	0	0	0	0	4	0	0	0	3	0
r[i]	4	0	0	0	0	3	0	0	0	1	0

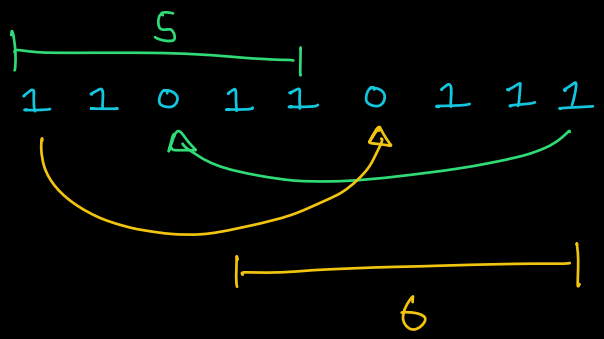
Q2 [MODIFIED]

Given a Binary array.

We are allowed to swap atmost one 0 with 1.

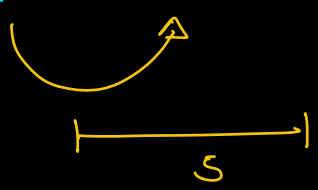
Return the length of max Consecutive 1s.

Ex1 :



ans = 6

Ex2 :



ans = 5

CASE 1: total_ones = 7

1 1 0 1 1 0 1 1 1

if ((l+r) < total_ones)

len = l+r+1

CASE 2

1 1 0 1 1 1

if ((l+r) == total_ones)

len = l+r;

Pseudo Code

```
if (n == 0) return 0;
int ans = 0;
for (int i = 0; i < n; i++) {
    if (arr[i] == 0) {
        l = 0
        for (int j = i-1; j >= 0; j--) {
            if (arr[j] == 1)
                l++
            else
                break;
        }
        r = 0
        for (int j = i+1; j < n; j++) {
            if (arr[j] == 1)
                r++;
            else
                break;
        }
        int len;
        if ((l+r) < total_ones) len = l+r+1
        else len = l+r;
        ans = max(ans, len);
    }
}
if (ans == 0) return n;
else return ans;
```

```
int total_ones = 0
for (int i = 0; i < n; i++) {
    if (arr[i] == 1)
        total_ones++;
}
```

TC: O(n)

SC: O(1)

Q Find no of triplets i, j, k [indices]

such that $i < j < k$ &&

$a[i] < a[j] < a[k]$

Ex :

0	1	2	3	4
3	4	6	9	2

i	j	k	arr[i] < arr[j] < arr[k]		
0	1	2	3	4	6
1	2	3	4	6	9
0	2	3	3	6	9
0	1	3	3	4	9

↳ ans = 4 triplets

Brute force : $i < j < k$

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

3

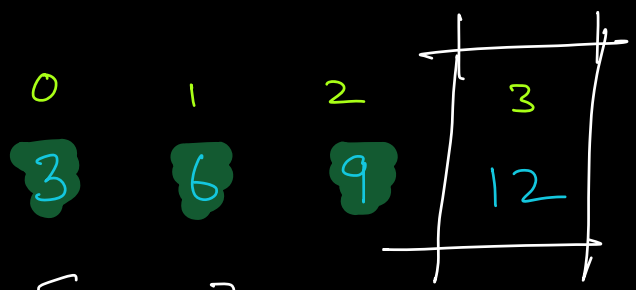
3

3

```
return cnt;
```

Tc: $O(n^3)$

Sc: $O(1)$



$i = [0, j-1]$ j $R = [j+1, n-1]$

1) $i < j$ ✓

3	12	16
6	12	16
9	12	16

3	12	17
6	12	17
9	12	17

	0	1	2	3	4	5	6	7
	3	6	9	12	5	16	8	7
left	-	1	2	3	1	5	3	-
right	-	5	2	1	3	0	0	-

$0 + 5 + 4 + 3 + 3 + 0 + 0 + 0$

$\Rightarrow \boxed{15}$

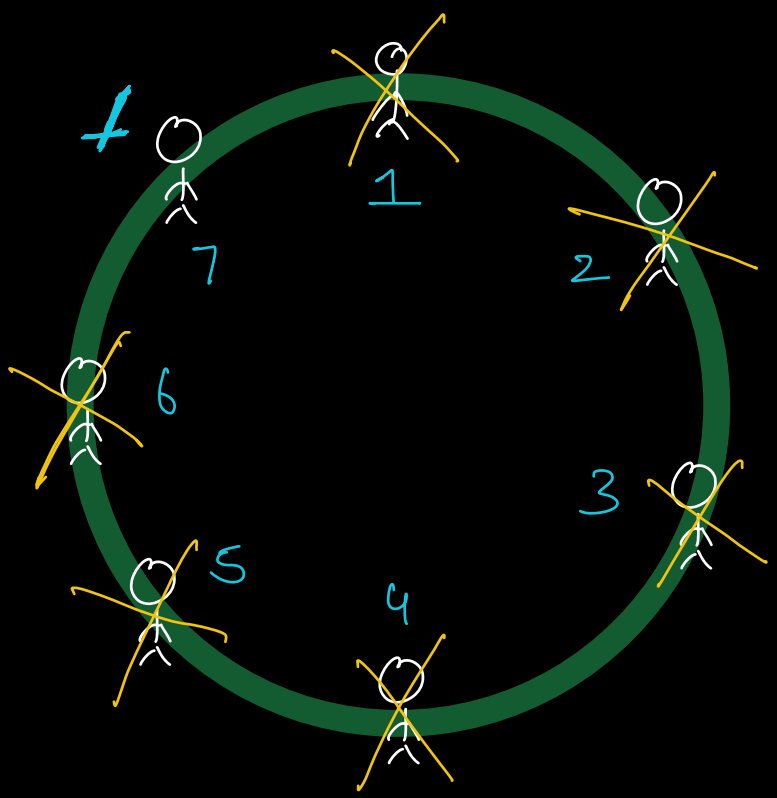
Pseudo Code

```
int cnt = 0;

for (int j = 1 ; j < (n-1) ; j++) {

    left_small  =>  [ iterate from [0, j-1]
                      & find element smaller
                        than arr[j] ]

    right_big   =>  [ iterate from [j+1, n-1]
                      & find element greater
                        than arr[j] ]

    cnt = cnt + ( left_small * right_big);

}

return cnt;
```

TC : $O(n^2)$

SC : $O(1)$

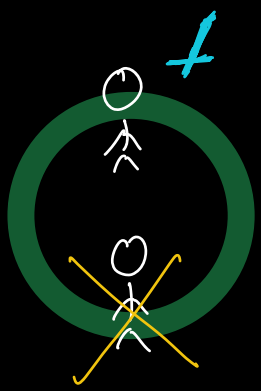
Josephus



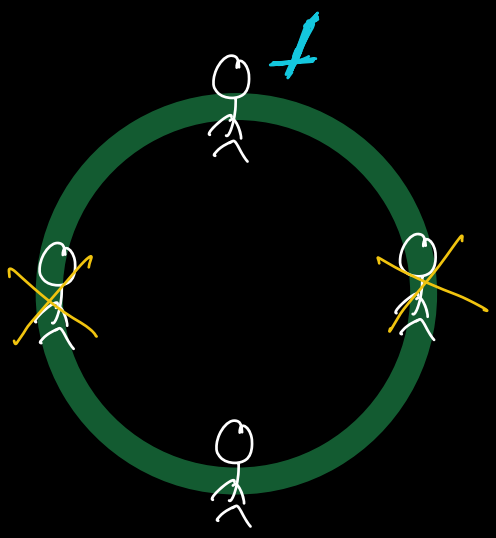
ans = 7

$N = 1$

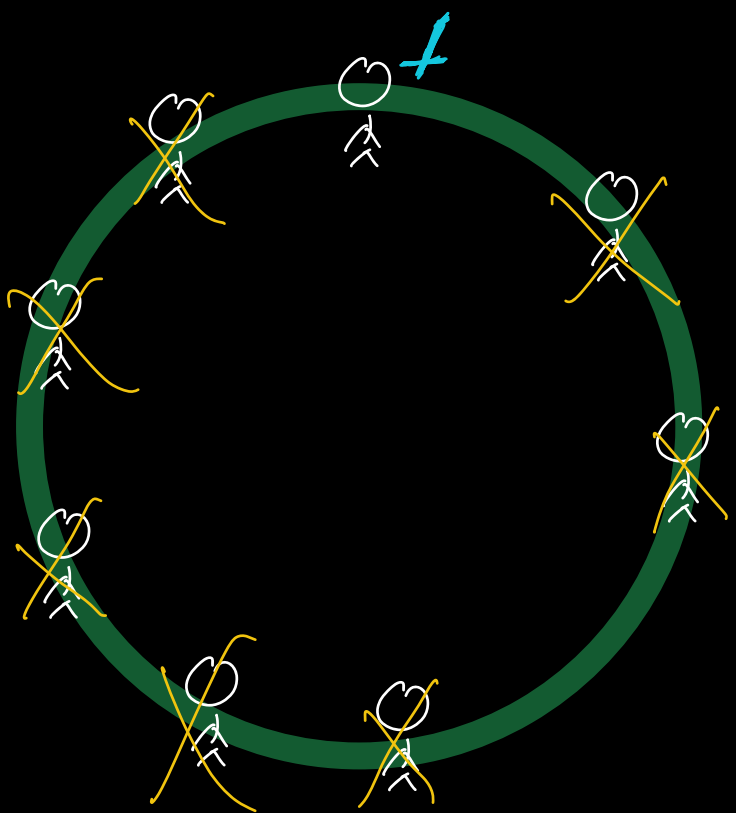
$N = 2$



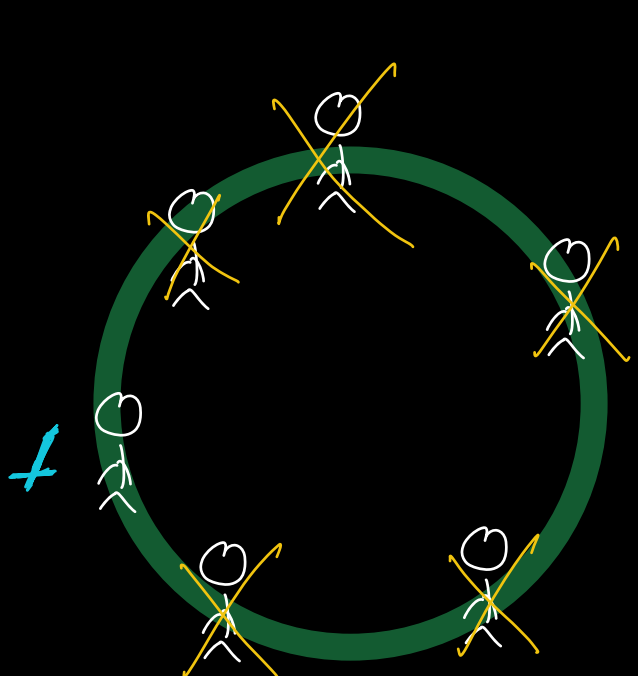
$N = 4$



$N = 8$



$N = 6$



Ans = 5

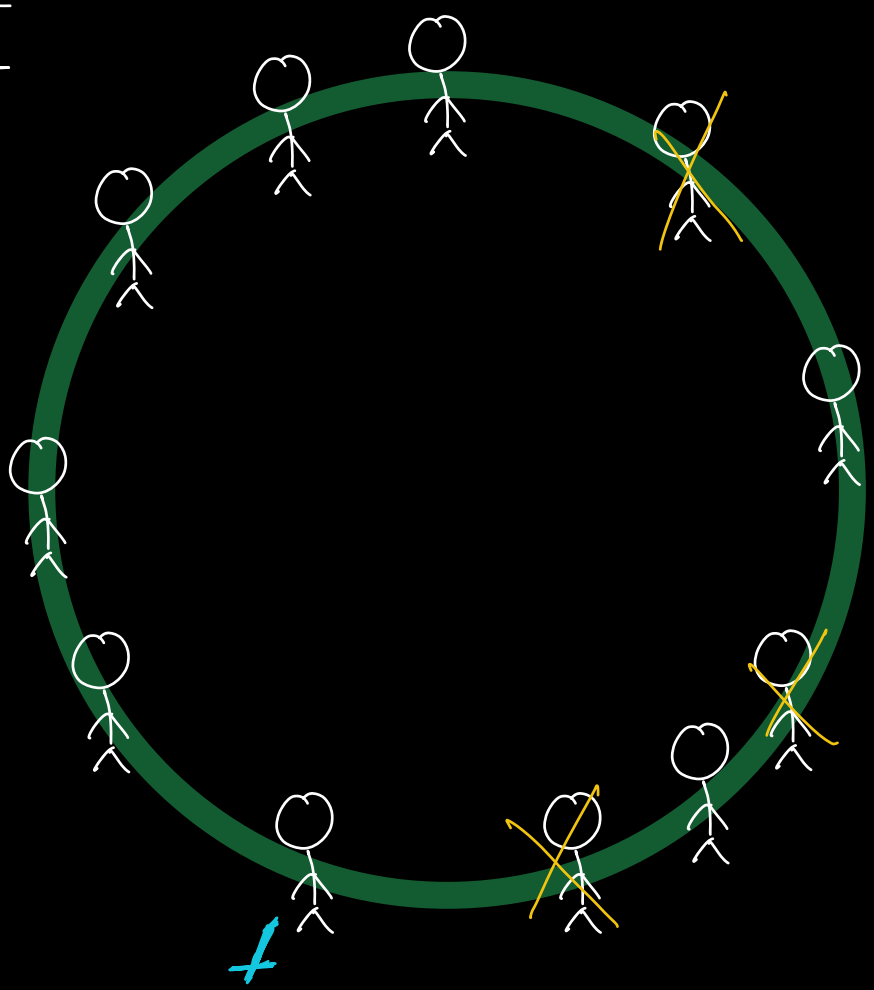
Observation

if $N = 2^x$, the person who starts the Killing is the last man standing.

$N = 11$

$\Downarrow$

8



~~X~~

10

After x people are killed

$\Rightarrow 2x + 1$

$N = 100$

↳ nearest power of 2 $\Rightarrow$ 64

How many need to be killed to reach 64 $\Rightarrow$ 36

Who has the sword after x kills $\Rightarrow 2(x) + 1$

$\Downarrow$

73