Array : Important Problems

Question 1

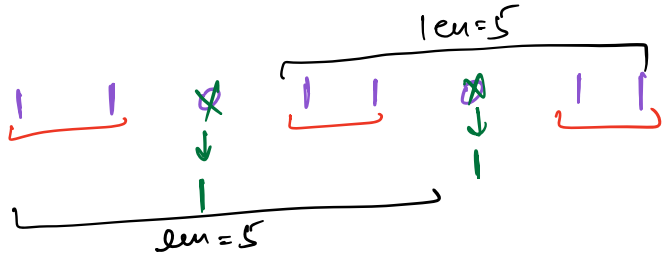
a[i] is either 0 or 1

Given a binary array, we can at most replace a single <=1

0 or 1.

find max consecutive 1's you can get in the array.

eg



curr length = 2

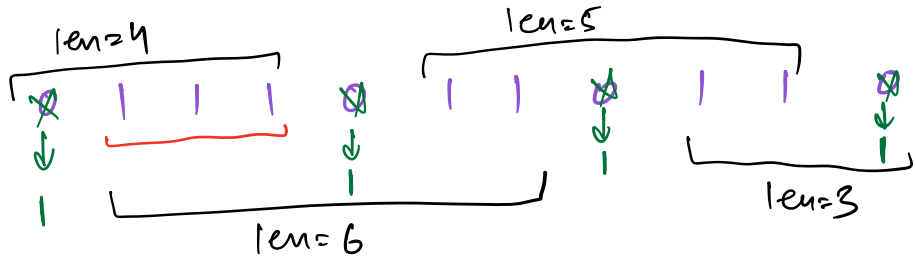
Ans = 5

eg

curr length = 6

Ans = 6

eg



curr length = 3

Ans = 6

Idea :

for each zero :

1. count consecutive 1's in the left = l
2. " " 1's in the right = r
3. len = l + r + 1

Code

```
int Replace(a[]) {
    n = a.length
    c=0
    for(i=0; i<n; ++i)
        c += a[i]
```

if a[i]==1 ⇒ c++
else ⇒ nothing

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

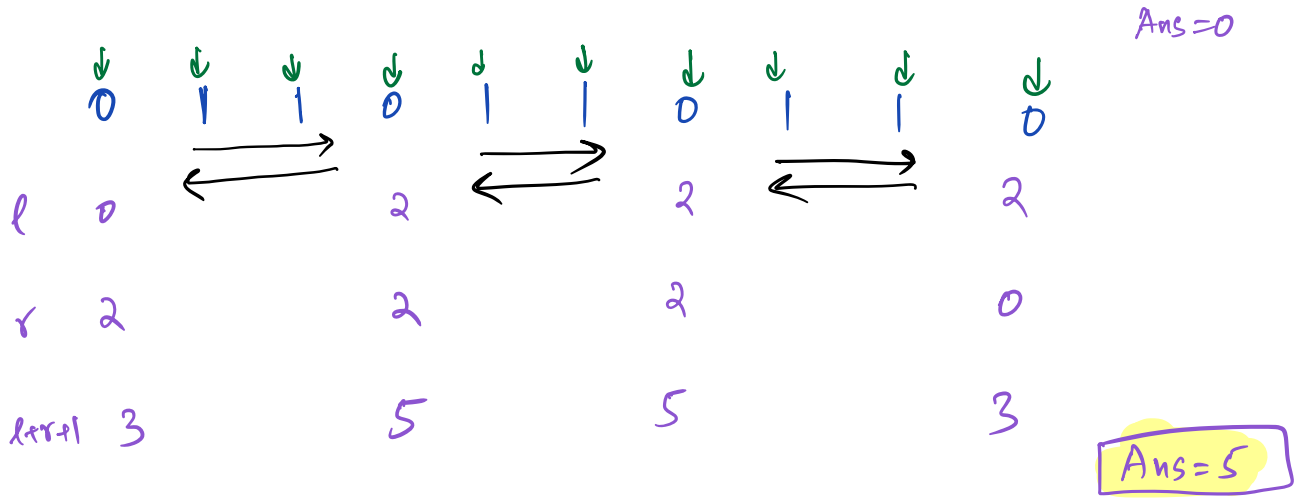
left part of consecutive 1's

right part of consecutive 1's

TC: O(N²) ~~O(N²)~~

SC: O(1)

O(N)

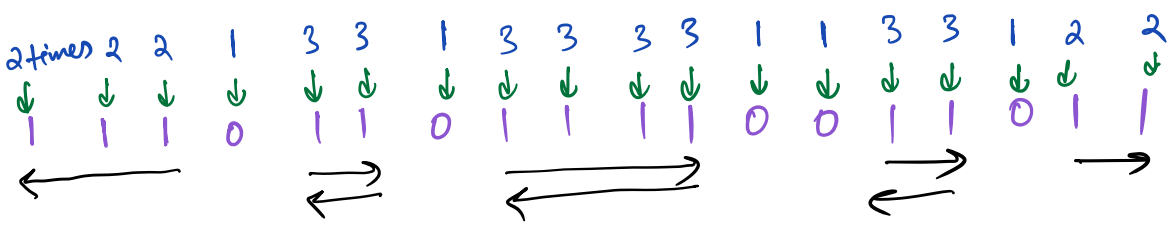


TC discussion

```
for (i=0; i<3; ++i) {
    for (j=0; j<n; ++j) {
        print(a[j])
    }
}
```

→ 3N iterations

TC: O(N)

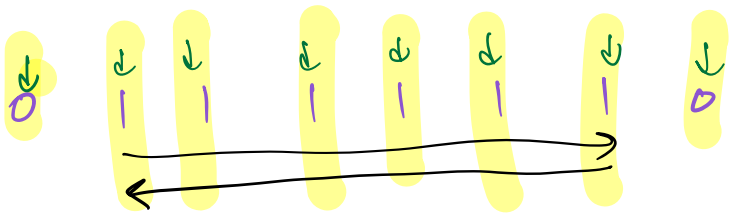


all elements are visited <= 3 times

total iterations <= 3N

TC : O(N)

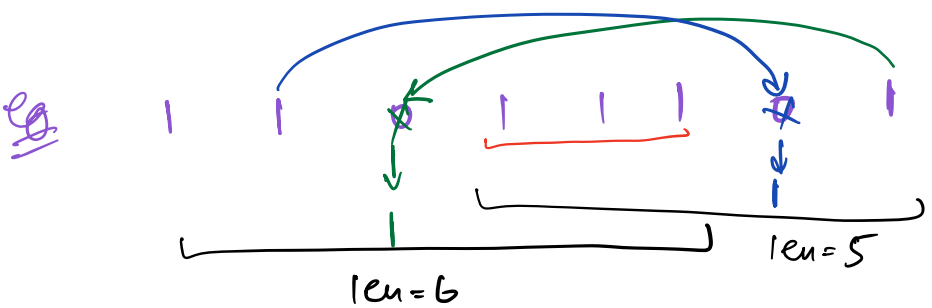
Lesson: If there is a "break" in any loop, then calculate TC carefully.

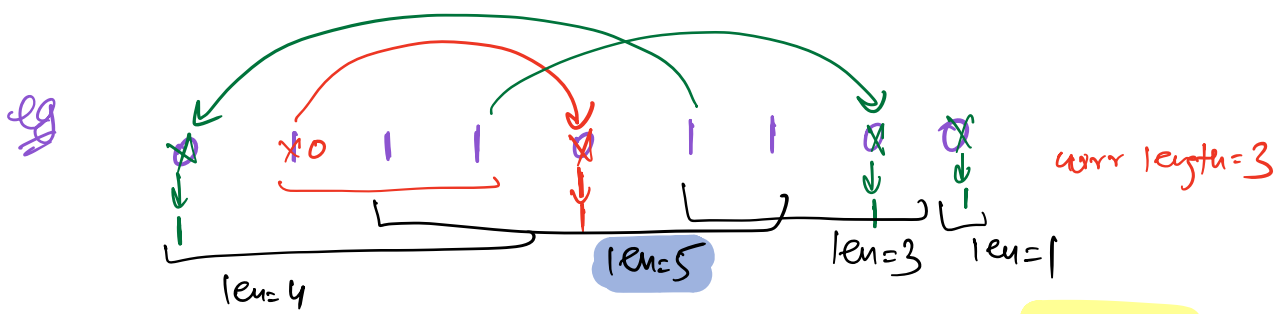


Question 1 – Part 2

Given a binary array, we can at most **swap** single 0 with 1.

find max consecutive 1's.





Part 2 is different from Part 1 in only 1 case:

Your ans can't be greater than total 1's in the array.

Code

```
int Swap (a[]) {
    n = a.length
    c=0
    for(i=0; i<n; ++i)
        c += a[i]                 } if a[i]==1 ⇒ c++
                                    else ⇒ nothing
    if (c==n)
        return n
    ans=0
    for(i=0; i<n; ++i) {
        if(a[i]==0) {
            l=0, r=0
            for(j=i-1; j>=0; --j){      } left part of consecutive 1's
                if(a[j]==1)
                    l++
                else
                    break
            }
            for(j=i+1; j<n; ++j){       } right part of consecutive 1's
                if(a[j]==1)
                    r++
                else
                    break
            }
            ans = max(ans, l+r+1)
        }
    }
```

```
if (ans > c)
    ans = c
```
→ ans can't be greater than total no. of 1's

return ans

3

Question 3

Given a[N] elements, calculate number of triplets i, j, k such that $i < j < k$ and $a[i] < a[j] < a[k]$

(i, j, k: indices of Array)

eg A = 2 6 9 4 10

0 1 2 3 4

i	j	k	a[i]	a[j]	a[k]
(0	< 1	< 2)	2	< 6	< 9
(0	< 1	< 4)	2	< 6	< 10
(0	< 3	< 4)	2	< 4	< 10
(1	< 2	< 4)	6	< 9	< 10
(0	< 2	< 4)	2	< 9	< 10

Ans = 5

Code

```
int triplets ( a[]) {
    n = a.length
    ans=0
    for (i=0; i<n; ++i) {
        for (j=i+1 ; j<n; ++ j ) {
            for (K=j+1; K<n ; ++K ) {
                // triplet (i,j,K)
                if (a[i] <a[j] && a[j] < a[K] )
                    ans ++
            }
        }
    }
    return ans
}
```

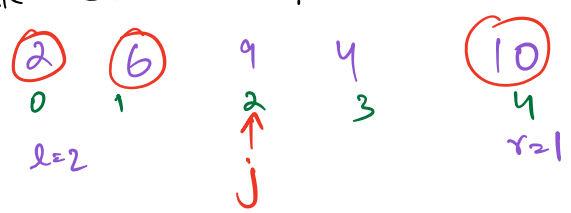
TC: $O(N^3)$

SC: $O(1)$

eg 1 2 3 4 5 6

OPTIMIZE

Hint: In how many triplets, index 3 is the middle element?



Idea:

for every element a[i]:

→ get no. of elements less than a[i] in left = l

→ get no. of elements greater than a[i] in the right = r

→ no. of triplets a[i] will be in the middle = l x r

Code

```
int triplets( a[]) {
    n = a.length
    ans=0
    for ( j=1; j<n; ++j ) { → middle index
        l=0, r=0
        for (i=j-1; i>=0; --i ) { → first index
            if (a[i] < a[j])
                ++l
        }
        for (k=j+1; k<n; ++k ){ → last index
            if (a[j] < a[k])
                ++r
        }
        count = l*r
        ans += count
    }
```

TC: O(N^2)

SC: O(1)

return ans

3

eg

	4	1	2	6	9	7	
	0	1	2	3	4	5	
l	0	0	1	3	4	4	
r	3	4	3	2	0	0	
l==r	0	0	3	6	0	0	ans = 9

$O(N^2)$ → $O(N \log N)$

- Balanced binary search tree
- Segment trees

Don't Worry

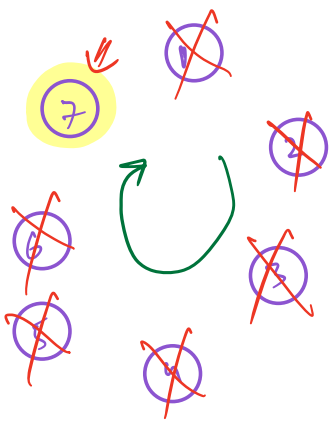
Question - Josephus Problem

N people are standing in circle. Person 1 has a knife. He kills next person in clockwise direction and passes on the knife to the next person in clockwise direction.

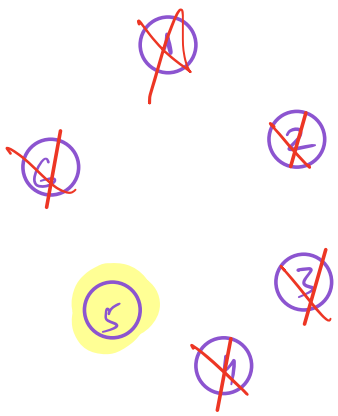
Repeat this until 1 person is alive.

Find the last man standing ?

N = 7



N = 6

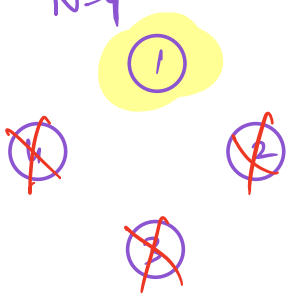


Observations :

1. last odd will be winner
   - if N is odd → N
   - if N is even → N-1
2. largest prime <= N

if N = 4 all observations are wrong

N = 4

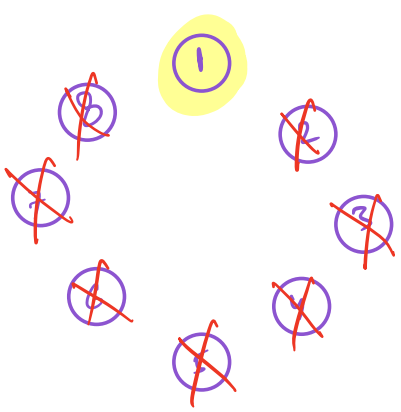


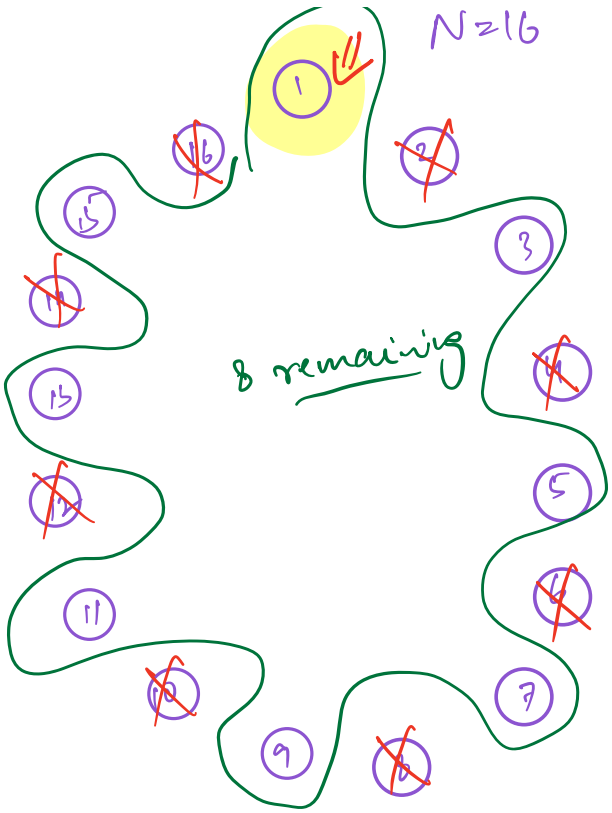
N = 1

N = 4 ⇒

N = 2

N = 8

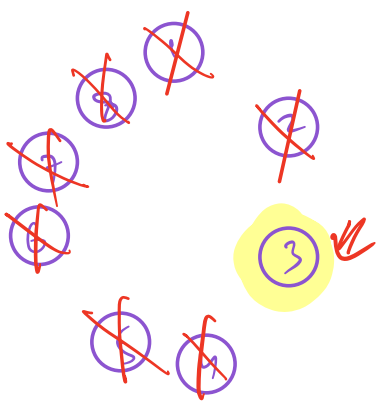




Observation

if N is power of 2,

if 1 starts → 1 wins

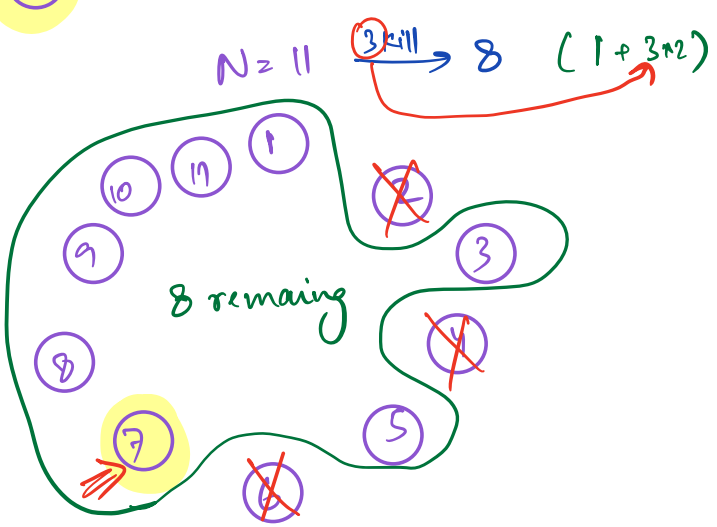


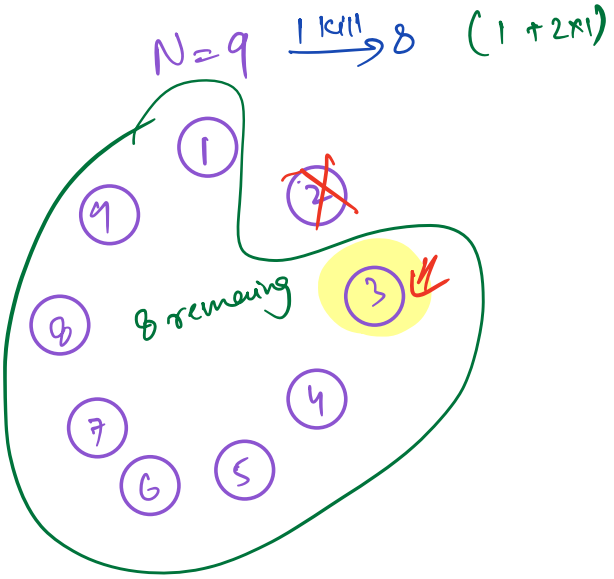
Observation!

if N is power of 2,

whoever starts,

wins the game.





for N = 100 ?

N - Kills = power of 2

$2^6 = 64$, $2^7 = 128$

N - Kills = 64

Kills = 100 - 64 = 36

ans = 1 + 36*2

= 73

Code

N = 100

$\log_2 N = \log_2 100$

$= 6.xxx$

Kills = $N - 2^{\lfloor \log_2 N \rfloor}$

$= 100 - 2^6$

$= 100 - 64 = 36$

ans = 1 + 2 * kills

TC : O(1)

SC : O(1)