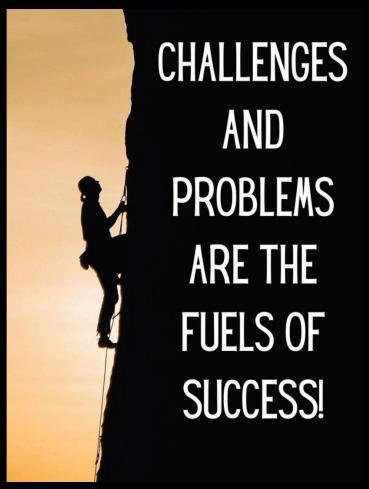


Today's Content:

→ Max Consecutive 1's by
  a) Atmost 1 replace
  b) Atmost 1 swap

→ Count Triplets {Goldman Sachs Hiring Challenge}

→ Josephus Problem

Q) Given a binary arr[]. We can atmost replace a single 0 with 1. find the maximum consecutive 1's we can get in the given arr[].

Eg → [1 1 1 (0) 1 1 0 1] [ans = 5]
0 1 2 3 4 5 6

Q → [0 1 1 1 0 1 1 0 1 1 0] [ans → 6]
0 1 2 3 4 5 6 7 8 9 10

Q → [1 1 1 1 1 1] [ans → 6]
0 1 2 3 4 5

[ans = 7]

Eg → [1 1 1 (0) 1 1 0 1 1 1 1 0 0 1 1 0 1 1]
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

l = 3, r = 2

l+r+1 ⇒ 3+2+1 = 6.

2+4+1 = 7.

4+0+1 = 5

0+2+1 = 3

2+2+1 = 5.

idea : for every '0'

→ find count of consecutive 1's on l.h.s = l
→ find count of consecutive 1's on r.h.s = r
→ if (l+r+1 > ans) update {ans = l+r+1}

Edge case → If all 1's are present ? → return N.
0's " " → return 1.

pseudo-code

```
int maxConsecutiveOnes(arr, N){
    count = 0, ans = 0
    for(i = 0; i < N; i++){
        if (arr[i] == 1) { count++ }
    }
    if (count == N) { return N }
    if (count == 0) { return 1 }
    for(i = 0; i < N; i++){
        if (arr[i] == 0){
            l = 0, r = 0
            // find consecutive 1's on l.h.s
            for(j = i-1; j >= 0; j--){
                if (arr[j] == 1) { l++ }
                else { break }
            }
            // find consecutive 1's on r.h.s
            for(j = i+1; j < N; j++){
                if (arr[j] == 1) { r++ }
                else { break }
            }
            if (l+r+1 > ans) { ans = l+r+1 }
        }
    }
    return ans.
}
```

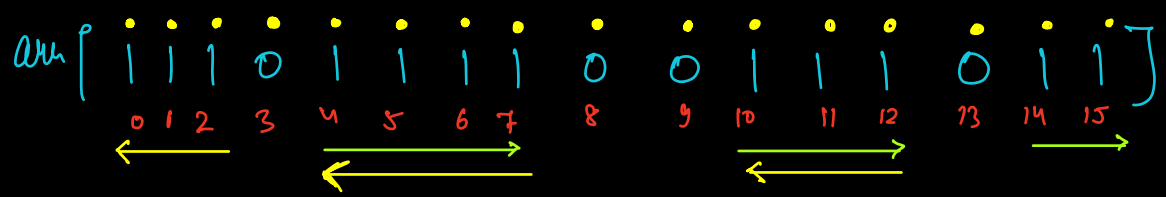
T.C → O(N)
S.C → O(1)

Time Complexity

```
for( i=1 ; i<=3 ; i++) {
    for(j=0 ; j<N ; j++) {
        // print —
    }
}
```

i=1 → N
i=2 → N
i=3 → N

3N iterations



[At max, how many times are we touching an element of the array ⇒ 3 times.]

Eg- [0 1 1 1 1 — — — 1 0]
0 1 2 3 N-2 N-1

⇒ 3N iterations

Eg- [1 1 1 1 1 0 1 1 1 1 1 1]
0 1 2 3 4 5 6 7 8 9 10 11

⇒ 2N iterations.

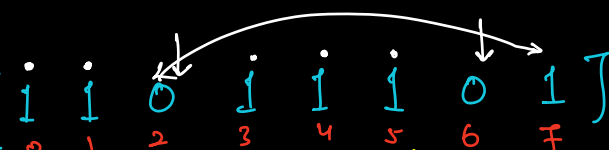
Eg- [0 1 0 1 0 1 0 1]

Be very careful while calculating the time complexity if, break statement is present in loops.

Q) Given a binary arr[]. Find max no. of consecutive 1's we can get by atmost 1 swap.

[We can swap with the values present in arr]

Eg: [1 1 0 1 1 1 0 1] [ans = 6]



l+r+1
⇒ 2+3+1 = 6

l: 3
r: 1
l+r+1
⇒ 3+1+1 = 5

Eg: [0 1 1 1 0 1 1 0 0] [ans = 5]

l: 0
r: 3
l+r+1
⇒ 0+3+1
= 4

l: 3
r: 2
l+r
⇒ 3+2
= 5

l: 2
r: 0
l+r+1
⇒ 2+0+1
= 3

l: 0
r: 0
l+r+1
⇒ 0+0+1
= 1

[fans = 5]

pseudo-code.

```
int maxConsecutiveOnes(arr, N){
    count = 0, ans = 0
    for(i = 0; i < N; i++){
        if (arr[i] == 1) { count++ }
    }
    if (count == N) { return N }
    if (count == 0) { return 0 }
    for(i = 0; i < N; i++){
        if (arr[i] == 0){
            l = 0, r = 0
            //find consecutive 1's on left:
            for(j = i-1; j >= 0; j--){
                if (arr[j] == 1) { l++ }
                else { break }
            }
            // find consecutive 1's on right:
            for(j = i+1; j < N; j++){
                if (arr[j] == 1) { r++ }
                else { break }
            }
            K = l + r
            if { K < count } { K += 1 }
            if ( K > ans } { ans = K }
        }
    }
    return ans
}
```

T.C → O(N)
S.C → O(1)

Q:) **No. of Triplets**

Given arr[N] elements, calculate no. of triplets i, j, K such that $i < j < K$ and $arr[i] < arr[j] < arr[K]$.

Eg: arr → [2 (0) 6 (1) 9 (2) 4 (3) 10 (4)]

[ans = 5]

i	<	j	<	K
0		1		2
0		3		4
0		2		4
0		1		4
1		2		4

arr[i]	<	arr[j]	<	arr[K]
2		6		9
2		4		10
2		9		10
2		6		10
6		9		10

Eg. arr → [4 (0) 1 (1) 2 (2) 6 (3) 9 (4) 7 (5)]

[ans → 9]

i	<	j	<	K
0		3		4
0		3		5
1		2		3
1		2		4
1		2		5
1		3		4
1		3		5
2		3		4
2		3		5

arr[i]	<	arr[j]	<	arr[K]
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

idea-1 Consider all the triplets & increment count when they satisfy the condition.

```
count = 0
for( 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]){
                count++
            }
        }
    }
}
return count
```

T.C → $O(N^3)$
S.C → $O(1)$

Hint for Optimisation :

arr[] → [4 (0) | 1 (1) | 2 (2) | 6 (3) | 9 (4) | 7 (5)]

In how many triplets, idx 3 [6] will be the middle element?

left	middle	right
4	6	9
1		7
2		

triplets = 3*2 = 6.

idea ⇒ Consider every element as middle elemnt, then find no. of smaller elements on l.h.s & greater elemnts on r.h.s.

arr[] → [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

count = 0 + 0 + 3 + 6 + 0 + 0 = 9.

pseudo-code.

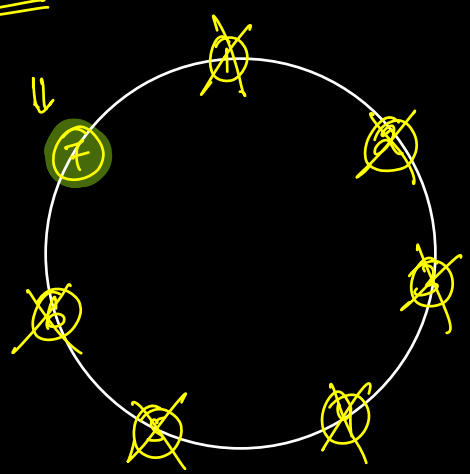
```
ans = 0
for( j = 1 ; j < N-1 ; j++){
    //count of smaller nos on l.h.s
      l = 0
      for( i = j-1 ; i >=0 ; i--){
        if(arr[i] < arr[j]) { l++ }
      }
    //count of greater nos on r.h.s
      r=0
      for( k= j+1 ; k < n ; k++){
        if (arr[k] > arr[j]) { r++ }
      }
    //update ans.
    ans += (l*r)
}
return ans;
```

T.C → O(N²)
S.C → O(1)

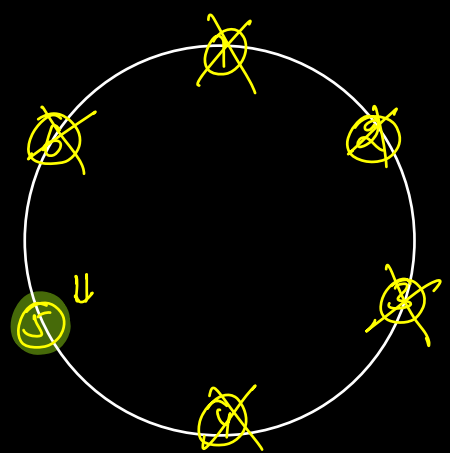
Josephus Problem

N people are standing in a circle. Person 1 has knife, he kills next person in clockwise direction & passes on the knife to next alive person in clockwise direction. Repeat the process until a single person is alive. Find the last person standing.

N=7



N=6

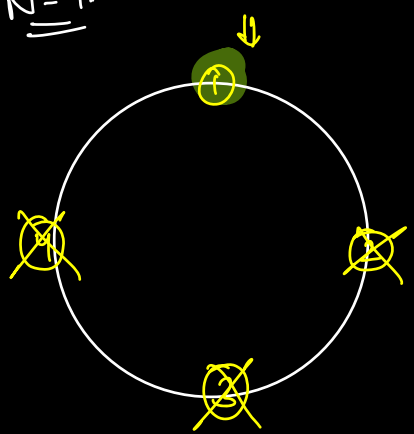


① if N is odd → N
if N is even → N-1 ✓

② largest prime <= N ✓

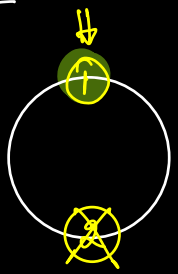
③ largest odd ✗

N=4.

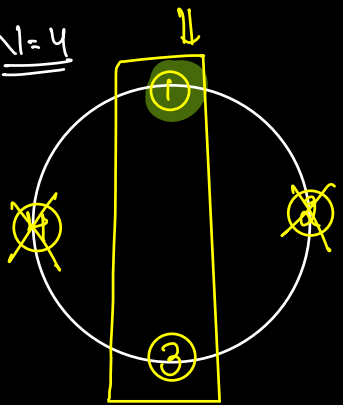


N=1

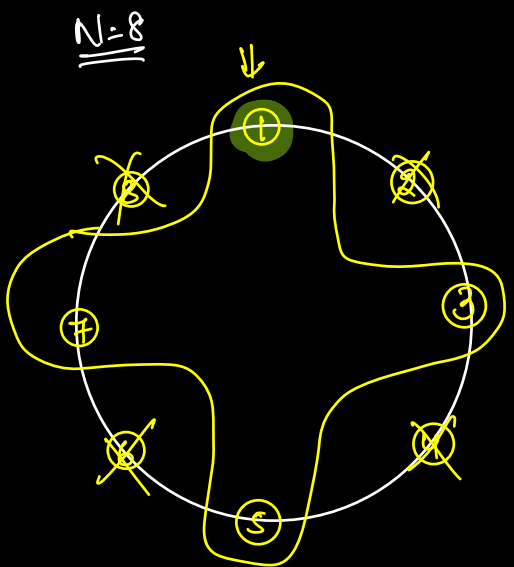
N=2



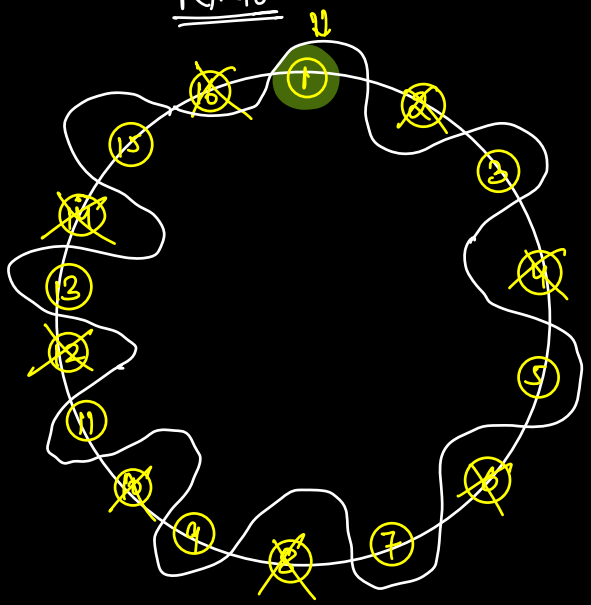
N=4



N=8

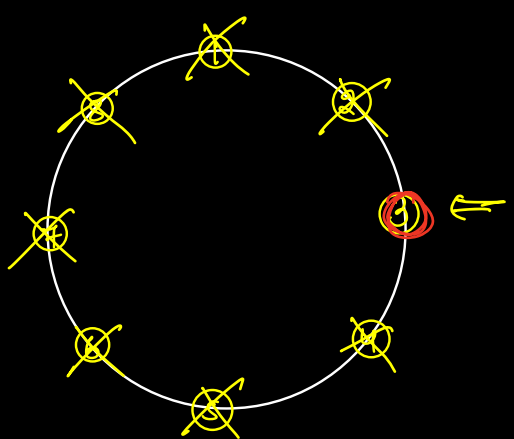


N=16

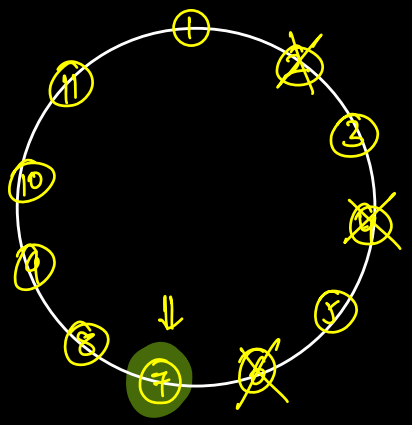


[observation - If N is a power of 2, whoever starts the game, wins the game]

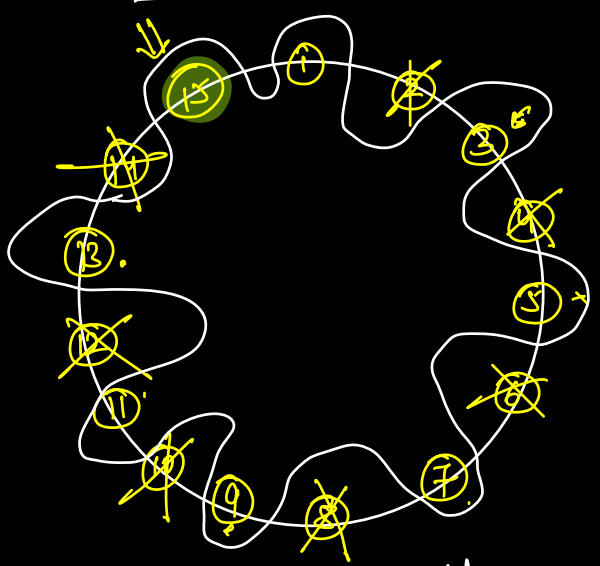
N=8



N=11

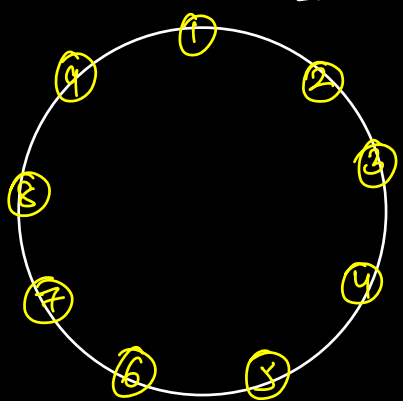


N=15 (7 Kills) → 8



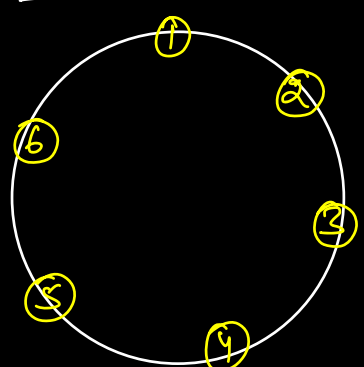
Who will be holding the knife after 7 kills = 1 + (7*2) = 15

N=9. (Kills = 1) → 8



winner = 1 + (1*2)
= 3.

N=6. (Kills = 2) → 4



winner = 1 + (2*2) = 5.

N=100

Closest power of 2 ≤ 100 ⇒ 64

No. of kills to attain 64 ⇒ 100 - 64 = 36

winner = 1 + (36*2) = 73.

pseudo-code.

① Find closest power of 2. [cp] $\leq N$.

② Calculate no. of kills that we have to make in order to attain this value $\Rightarrow$ [kills = N - cp]

③ [ans = 1 + 2 * kills]

→ person is skipping (k-1) persons & killing the k^{th} person.

$\Rightarrow$ [Recursion]

[N = 100.

K = 1 * 2 * 2 * 2 * 2 * 2 * 2

K = 64]

arr = [1 0 1 1 1 0 1 1 1 0 1 0 1 0 1]

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

M = 3.