> Anyone who keeps learning stays young.
>
> — Henry Ford —

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 (0, 1) arr[], we can atmost replace a single 0 with 1. Find the maximum consecutive 1's we can get in given arr[].

Eg1 { 1 1 0 1 1 0 1 } {ans = 5}
0 1 2 3 4 5 6

Eg2 { 0 1 1 1 0 1 1 0 1 1 0 }
0 1 2 3 4 5 6 7 8 9 10 {ans = 6}

Eg3 { 1 1 1 1 1 1 }
0 1 2 3 4 5 {ans = 6}

Eg4 { 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 {ans = 7}

3 2
3+2+1. 2+4+1.
4+0+1 0+2+1 2+2+1

idea:
For every '0'
→ count no. of consecutive 1's on l.h.s = (l)
→ count no of consecutive 1's on r.h.s = (r)
→ if (l+r+1 > ans) update ans = l+r+1

Edge case: What if all 1's are present? → return N.

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 (int i = 0 ; i < N ; i++) {
        if (arr[i] == 0) {
            // no. of consecutive 1's on l.h.s.
            l = 0
            for( j = i-1 ; j >= 0 ; j--){
                if (arr[j] == 1){ l ++ }
                else { break }
            }
            //no of consecutive 1's on r.h.s.
            r = 0
            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)

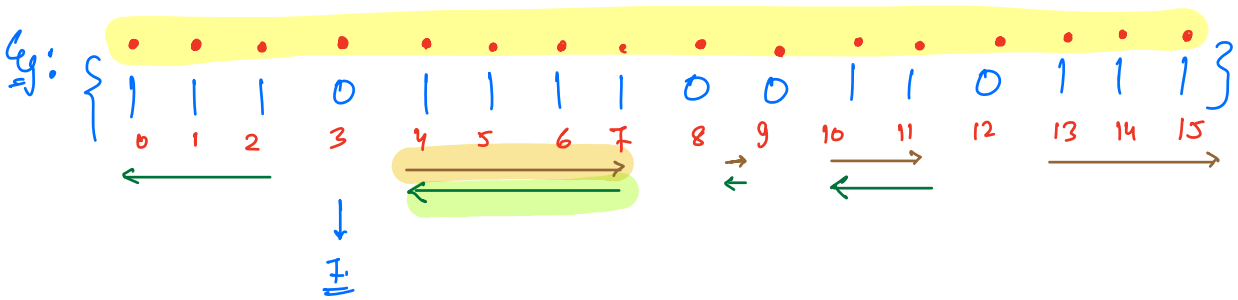
TC :

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

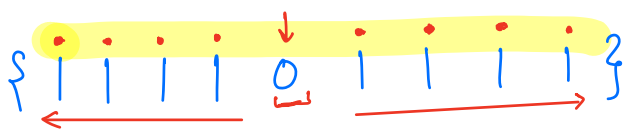
i = 1 → N
i = 2 → N
i = 3 → N

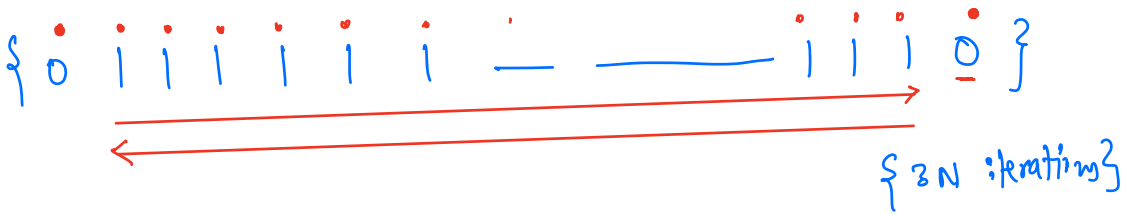
3N

[T.C → O(N)]



[At max, we are touching all elements → 3 times.]





Note → If break is present in your loops, be careful while calculating the time complexity

Q) Given a binary arr[], find max no. of consecutive 1's we can get by atmost 1 swap. [We can swap with values present in the array]

Eg: { 1 1 0 1 1 1 0 1 }
0 1 2 3 4 5 6 7

l : 2
r : 3
c = l + r + 1 = 6

l : 3
r : 1
c = l + r + 1 = 5

[ans = 6]

Eg: { 0 1 1 1 0 1 1 0 0 }
0 1 2 3 4 5 6 7 8

l : 0
r : 3
c = l + r + 1
= 4

l : 3
r : 2
c = l + r + 1
{No extra 1's}

c = l + r
= 2 + 3 = 5

l : 2
r : 0
c = l + r + 1
= 3

l : 0
r : 0
c = l + r + 1
= 0 + 0 + 1 = 1

bscudo-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 (int i = 0 ; i < N ; i++) {
        if (arr[i] == 0) {
            // no. of consecutive 1's on l.h.s.
            l = 0
            for( j = i-1 ; j >= 0 ; j--){
                if (arr[j] == 1){ l ++ }
                else { break }
            }
            //no of consecutive 1's on r.h.s.
            r = 0
            for( j = i+1 ; j < N ; j++){
                if (arr[j] == 1) { r++ }
                else { break }
            }
            K = l + r
            if ( l+r < count) { K = K+1 }
            if (K > ans) { ans = K }
        }
    }
    return ans
}
```

T.C → O(N)
S.C → O(1)

[Break 6 minutes]

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[5] : { 2 6 9 4 10 }
0 1 2 3 4

[ans = 5]

i	j	K	arr[i]	arr[j]	arr[K]
0	1	2	2	6	9
0	1	4	2	6	10
0	2	4	2	9	10
0	3	4	2	4	10
1	2	4	6	9	10

Eg2: arr[6] : { 4 1 2 6 9 7 }
0 1 2 3 4 5

[ans = 9]

i	j	K	arr[i]	arr[j]	arr[K]
0	3	4	4	6	9
0	3	5	4	6	7
1	2	3	1	2	6
1	2	4	1	2	9
1	2	5	1	2	7
1	3	4	1	6	9
1	3	5	1	6	7
2	3	4	2	6	9
2	3	5	2	6	7

Idea1 : Consider all the triplets & check if they are satisfying 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[6] = { 4 1 2 6 9 7 }
idx: 0 1 2 3 4 5

In how many triplets, 6 will be the middle element.
(idx → 3)

left → [3] : { 0, 1, 2 }
middle : 3
right → [2] : { 4, 5 }

[triplets = 1 * 2 = 6]

Idea & pseudo-code:

arr[6] : { 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]

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

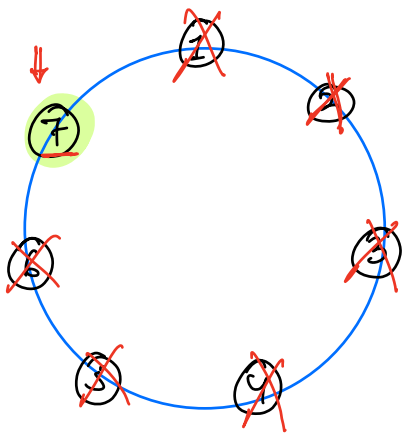
T.C → $O(N^2)$

S.C → $O(1)$

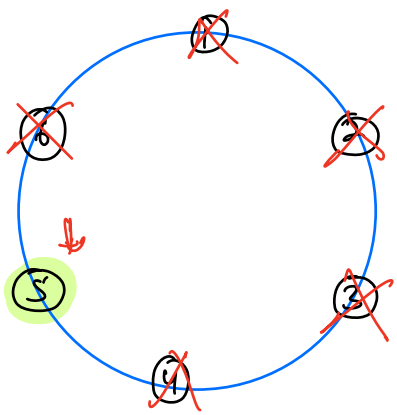
Josephus Problem

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

N=7



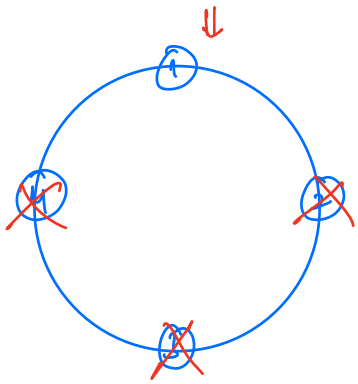
N=6



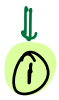
① if n is odd → N
if n is even → N-1] φ

② largest prime ≤ N] L.

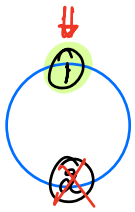
N=4



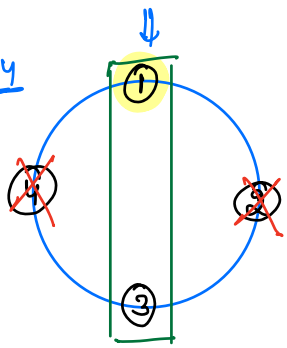
N=1



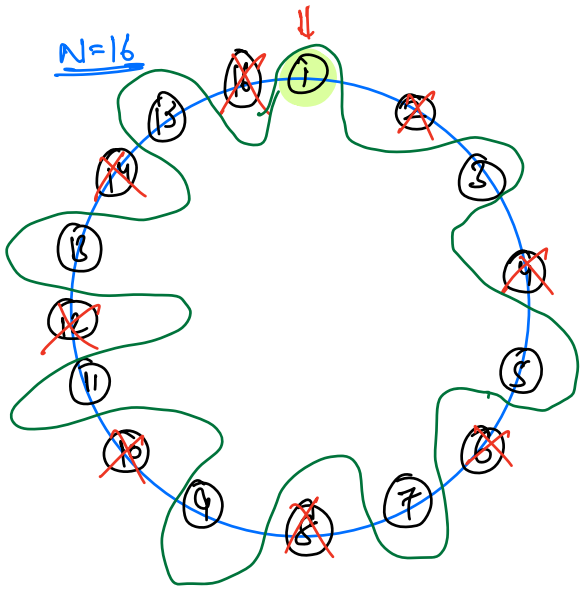
N=2.



N=4



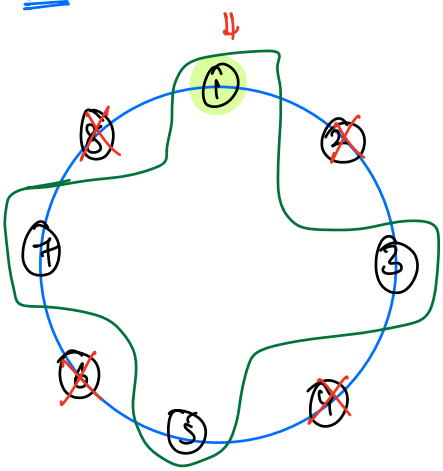
N=16



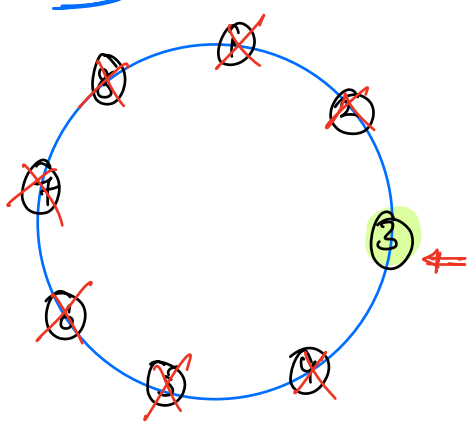
N=32. , 1 is having the knife ⇒ 1

[observation → if N is a power of 2,
1 is starting the game, ans=1]

N=8.

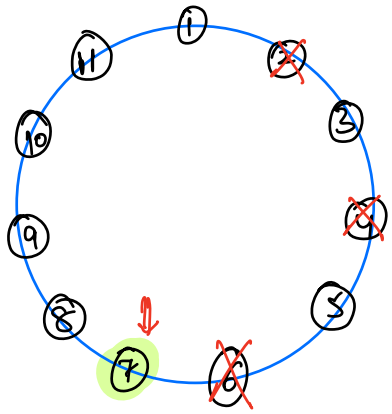


N=8

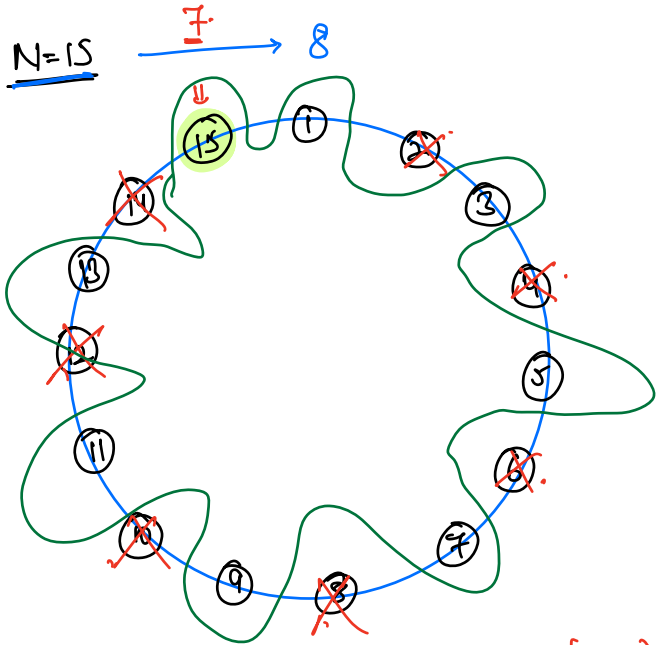


[obs 2. → if N is a power of 2,
whoever is starting the game,
wins the game.]

N=11

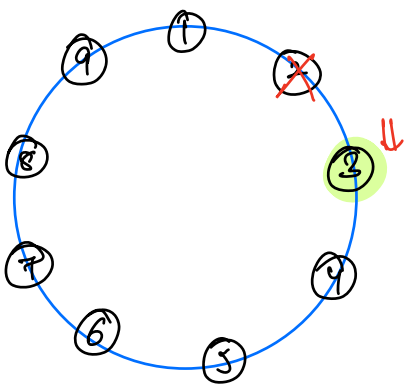


N=15 $\xrightarrow{7}$ 8

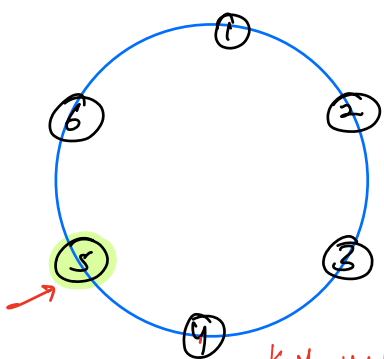


person holding the knife = $1 + (7*2)$
= 15.

N=9 $\xrightarrow{1}$ 8 $\quad 1+(1*2) = 3.$



N=6. $\xrightarrow{2}$ 4



Knife-Holder = $1 + (2*2)$
= 5.

N=100

Closest power of 2 which is less than or equal to 100. → 64

No. of kills, so that persons reduced to 64.

Kills = 100 − 64 = 36

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

pseudo-code. { # to do }.

Step-1. Find closest power (cp) of 2, $\leq N$.

Step-2. Calculate no. of kills that we need to make in order to achieve this. kills = N - cp

Step-3. ans = 1 + (2 * kills)

N = 17

powers: 1, 2, 4, 8, 16

2^0