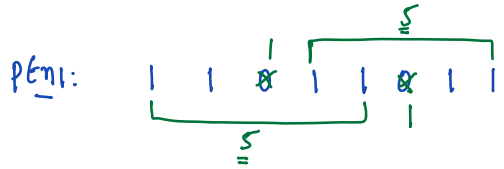
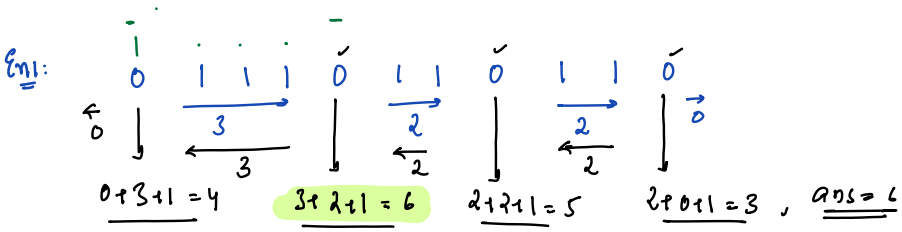
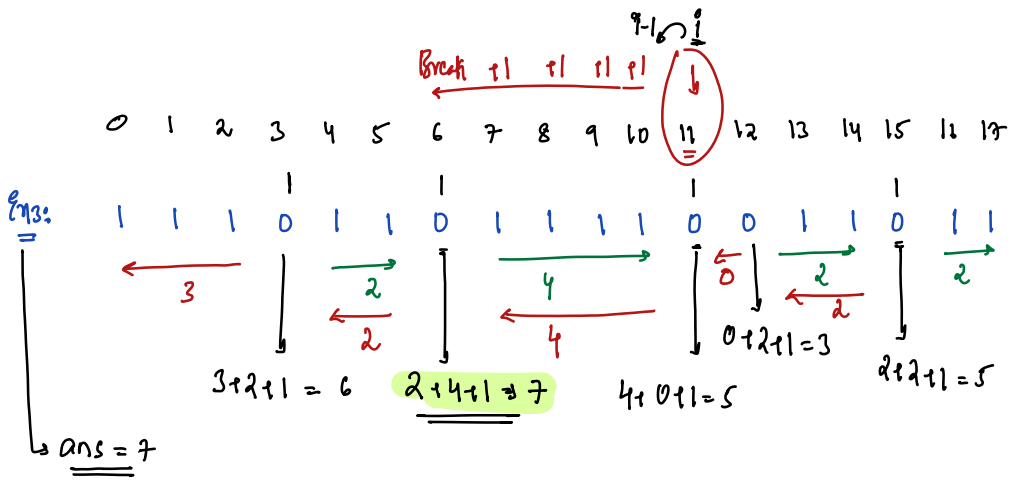
Todays Content :

- Max Consecutive 1s by
  a) Atmost 1 Replace
  b) Atmost 1 Swap
- Count Triplets
- Josephus Problem

Q8 Given a binary (0/1) ar[], we can atmost replace a single 0 with 1 find the max consecutive 1's we can get in your arr[]







Idea: For every '0'

a) Count no: of consecutive 1's on left = l

b) Count no: of consecutive 1's on right = r

c) if (l + r + 1 > ans) { ans = l + r + 1 }

Edge Case: If all are 1's, return N

Idea & Pseudo Code :

```
int Replace(int ar[]){
    int n = ar.length;
    int c = 0;
    for(int i=0; i<n; i++) { c = c + ar[i] }
    if(c==n) { return n }
    ar[i]==0: c = c+0
    c = c + ar[i]  ↳ ar[i]==1 c = c+1
    ------------> if(c==0) { return 1 }
    int ans = 0
    for(int i=0; i<n; i++){ -> 3N iterations
        if(ar[i]==0){
            int l, r = 0;
            for(int j=i-1; j>=0; j--){
                if(ar[j]==1) { l = l+1 }
                else { break }
            }
            for(int j=i+1; j<n; j++){
                if(ar[j]==1) { r = r+1 }
                else { break }
            }
            int cnt = l+r+1
            if(cnt > ans) { ans = cnt }
        }
    }
    return ans;
}
```

Edge Case:

	0	1	2	3
ar[i]:	0	0	0	0
l =	0	0	0	0
r =	0	0	0	0
l+r+1 =	1	1	1	1

$\Rightarrow |ar| = N \leq 10^5$

TC: $O(N^2)$ → $O(N)$ → will work

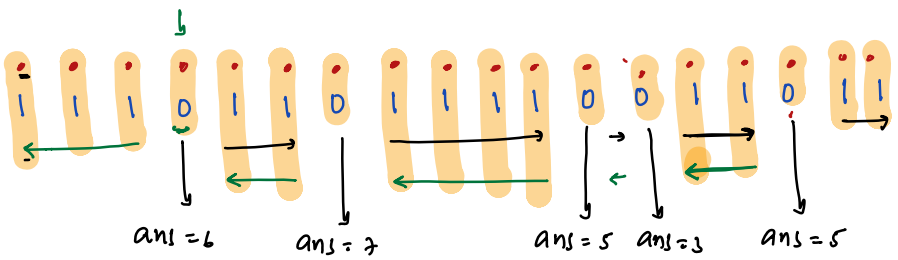
SC: $O(1)$

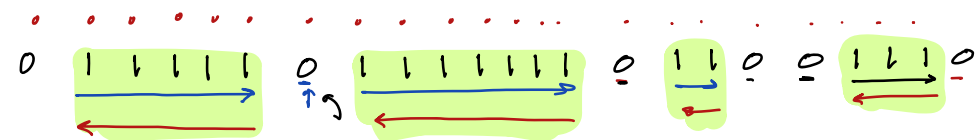
Note: If we have break inside Nested loop, be very careful when we calculate TC.

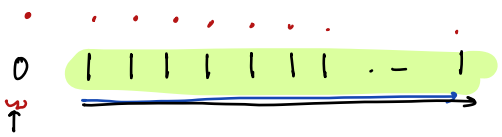
TC : Discussion :

```
for(i=0; i<3; i++){ => // We iterate on array 3 times
    for(j=0; j<n; j++){
        print(car[j])
    }
}
```

Iterations ⇒ 3N ⇒ TC: O(N)

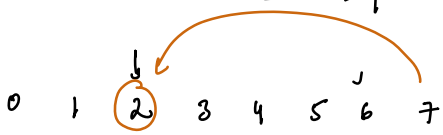






2Qs: Given a binary arr[], find max no of consecutive 1 we can get by atmost 1 swap.

↳ {We can only swap with data in arr[]}

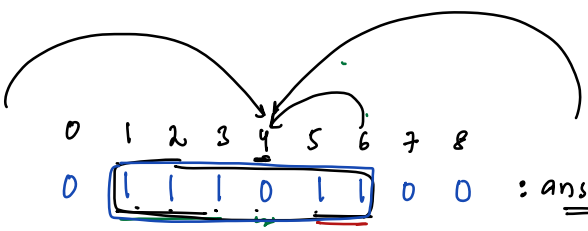


ans = 6

Ex1:

[l: 2, r: 3] I have extra 1 to swap at 2nd index

cnt = l + r + 1



Ex2: : ans = 5

{l: 3, r: 2} Since there is no extra 1

cnt = l + r

Note: If l + r == (Total no. of 1's)

: No Extra 1's

else : We can have Extra 1's

Pseudo Code :

```
int swap(int ar[]){ TC → O(N)  SC :→ O(1)
    int n = ar.length;
    int c = 0;
    for(int i = 0; i < n; i++) { c = c + ar[i] }   ar[i] == 0 : c = c + 0
    if(c == n) {return n}                          c = c + ar[i]  ↳ ar[i] == 1  c = c + 1

    int ans = 0
    for(int i = 0; i < n; i++){ → 3N iteration
        if(ar[i] == 0){
            int l, r = 0;
            for(int j = i-1; j >= 0; j--){
                if(ar[j] == 1) { l = l + 1 }
                else { break }
            }
            for(int j = i+1; j < n; j++) {
                if(ar[j] == 1) { r = r + 1 }
                else { break }
            }
            int cnt = l + r;                       Total is in array → if(l+r < c) {:
            if( l+r < c) { // Extra 1 is present to swap at i'th index     { There is extra 1 present other than l & r }
                cnt = cnt + 1
            }
            if (ans < cnt) { ans = cnt }
        }
    }
    return ans;
}
```

8:30 → 8:38 am

3cg) No of Triplets:

Given ar[N] elements, calculate no: of triplets i, j, k such that

$i < j < k$ && $ar[i] < ar[j] < ar[k]$, i, j, k are indices

Ex1: ar[5] = { 2 6 9 4 10 }

0	1	2	3	4
2	6	9	4	10

i < j < k			ar[i] < ar[j] < ar[k]			→ ans = 5
0	1	2	2	6	9	
0	1	4	2	6	10	
0	3	4	2	4	10	
0	2	4	2	9	10	
1	2	4	6	9	10	

Ex2: ar[6] = { 4 1 2 6 9 7 }

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

i < j < k			ar[i] < ar[j] < ar[k]			→ ans = 9
0	3	4	4	6	9	
0	3	5	4	6	7	
1	2	3	1	2	6	2, at center
1	2	4	1	2	9	3 triplets
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	

Idea: For every triplet i, j, k check, if it satisfies condition

```
int countTriplets(int ar[]) { → TC: O(N3)  SC: O(1)
    int n = ar.length;
    for(i=0; i<n; i++) {
        for(j=i+1; j<n; j++) {
            if(ar[i] >= ar[j]) break
            for(k=j+1; k<n; k++) {
                if(ar[k] > ar[j] && ar[j] > ar[i]) {
                    c=c+1
                }
            }
        }
    }
    return c;
}
```

mid

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

Hint2: fn2: ar[6] = {4, 1, 2, 6, 9, 7}

↳ # In how many triplets is index's middle element =

left | middle | right | # Triplets

left: 4, 1, 2 → 3 ; middle: 6 ; right: 9, 7 → 2 ; 3 * 2 = 6

i < j < k

i is on left of j, j is on left of k

fn2: ar[6] = {4, 1, 2, 6, 9, 7}

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

= ans = 9, total triplets

Idea: for every element ar[i]

→ get no: of elements less than that in left side = l

→ get no: of elements greater in right side = r

→ cnt = l * r

→ ans = ans + cnt

```
int Triplets(int ar[]){
    int n = ar.length
    int ans = 0;
    for(int j=0; j<n; j++){
        int l=0, r=0
        //Elements < ar[j] in leftside
        for(int i=0; i<j; i++){
            if(ar[i] < ar[j]) { l = l+1}
        }
        //Elements > ar[j] in rightside
        for(int k=j+1; k<n; k++){
            if(ar[j] < ar[k]) { r = r+1}
        }
        ans = ans + l*r
    }
    return ans;
}
```

TC: O(N * (N-1)) ≈ O(N^2)

SC: O(1)

O(N3) → O(N2) → O(NlogN) : Balanced Binary Search Trees

→ O(NlogN) : Segment Trees

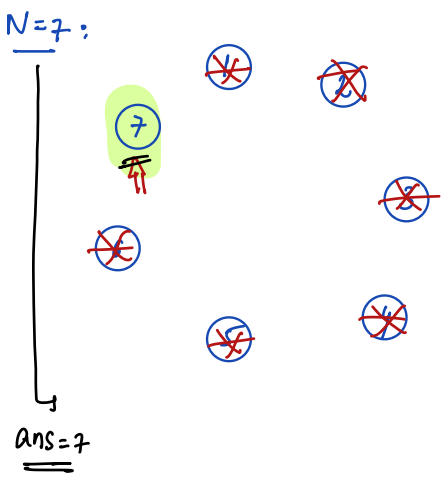
Josephus Problem : → { Squid game in Programming }

N people standing in Circle, Clockwise Person 1 has knife, he kills nent perm in clockwise & passes on the knife nent person in clockwise

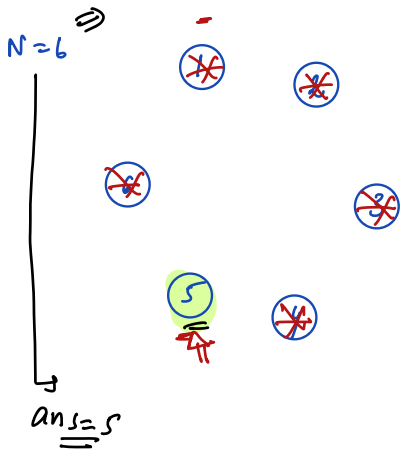
Repeat the process until a single person is alive

find the last person standing?

N=7 :



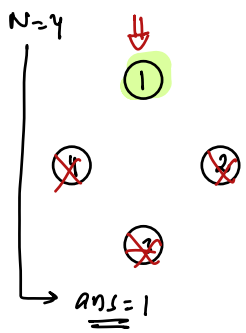
N=6



obs :

1) If N is even → N-1
   If N is odd → N

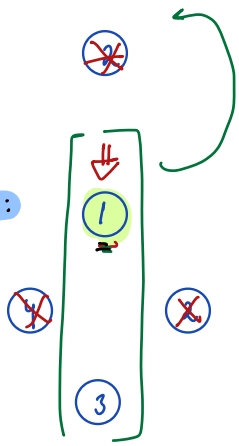
2) largest prime <= N



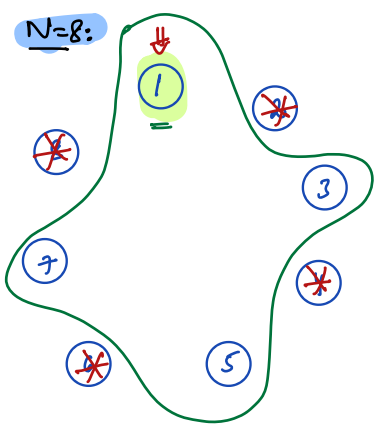
N=1:

N=2:

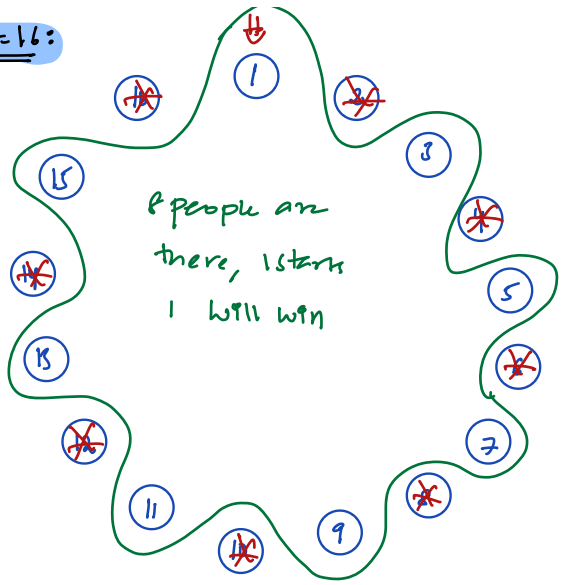
N=4:



N=8:



N=16:

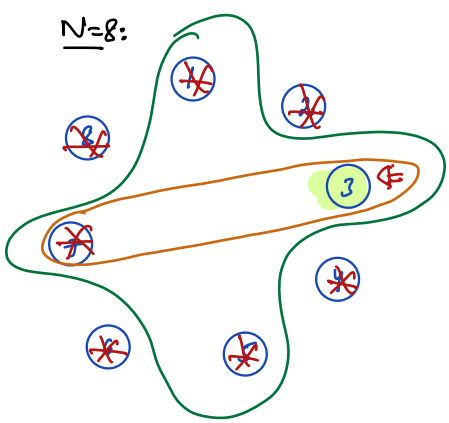


N=32, 1 has knife → 1 will win

obs1

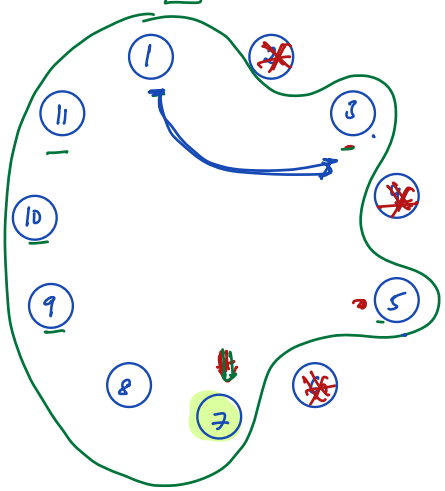
If N is power of 2, and if
1 starts the game → 1 game

N=8:



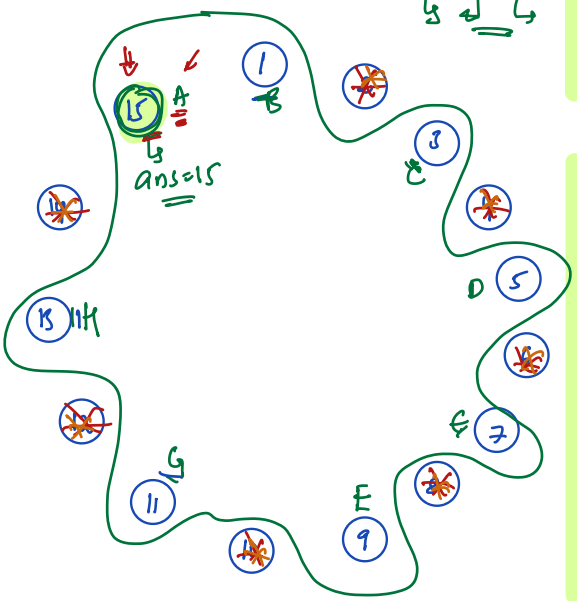
obs2: If N is power of 2, who ever has the knife, wins the game

N=11 → 3 kills → 8 : 1 + (3)(2) ⇒ 7



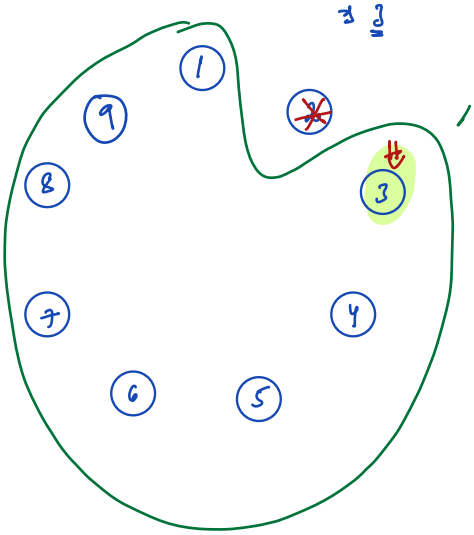
ans = 7

N=15 → 7 kills → 8 : 1 + (7)(2) ⇒ 15



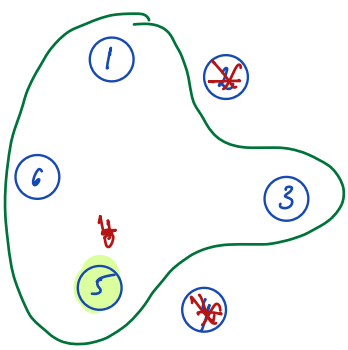
N=9 → 1 kills → 8 : 1 + (2)(1)

⇒ 3



N=6 → 2 kills → 4 : 1 + (2)(2)

⇒ 5



N=100 → 36 kills → 64 : 1 + (2)(36) ⇒ 73 winner

nearer power 2 ⇒ 100

N=4 → 0 kills → 4 : 1 + (2)(0) ⇒ 1 winner

nearer power 2 ⇒ 4

Josephus: → Can start anywhere
↳ Can skip k people at a time