Interview Problems

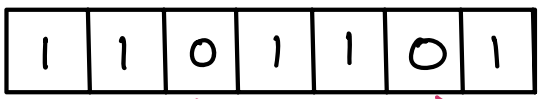
Good Evening ☺

Content ⟶

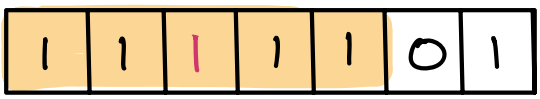
01. Max consecutive 1's by
    (a) Atmost 1 replace
    (b) Atmost 1 swap
02. Count triplets
03. Josephus problem (puzzle)

01. Given a binary ar[ ], we can atmost replace a single 0 with 1. Find the maximum consecutive 1's we can get in an ar[ ].

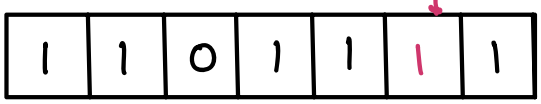
Eg :-



Ans →



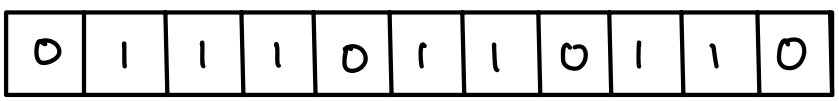
Ans = 5



Ans = 4

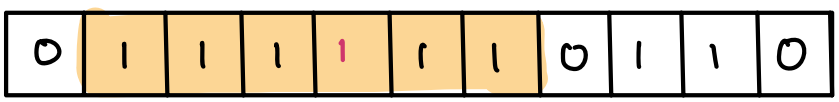
Ans = 5

Eg :-

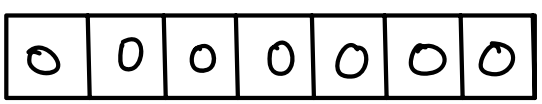


Ans = 6

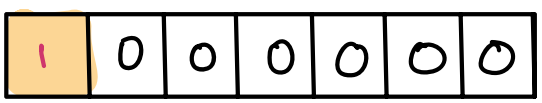
Ans →



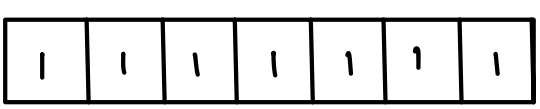
Eg :-



Ans = 1

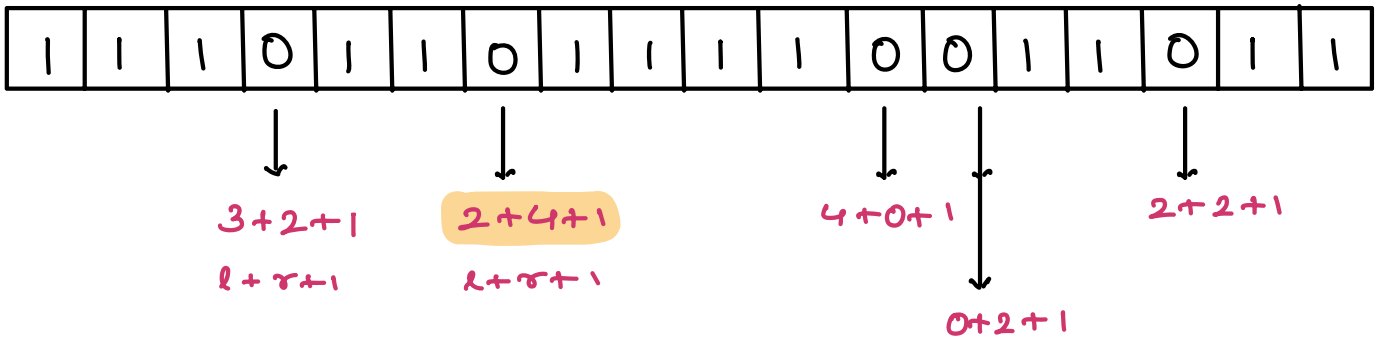


Eg



Ans = 7

Eg:-



Brute force →

For every 0

01. calculate the no. of consecutive 1's in LHS

02 calculate the no. of consecutive 1's in RHS

```
if ( l + r + 1 > ans) ans = l + r + 1
```

Edge cases → if (all 1's present) → return n;

if (all 0's present) → return 1;

```
int maxones ( int [] ar )
{
    int count = 0

    for( i=0 ; i<n ; i++ ){
        if (ar[i] == 1) count += 1 ;
    }
    if (count == n) return n;
    if (count == 0) return 1;
```

} Edge case

```
for (i=0; i<n; i++)
    if (ar[i] == 0){
        // count no. of 1's on LHS
        int l=0
        for (j=i-1; j≥0; j--){
            if (ar[j] == 1){l++;}
            else break;
        }
        // count no. of 1's on RHS
        int r=0
        for (j=i+1; j<n; j++){
            if (ar[j] == 1){r++;}
            else break;
        }
        if (l+r+1 > ans){ans=l+r+1};
    }
return ans;
}
```

TC: O(3n) ≅ O(n)

SC: O(1)

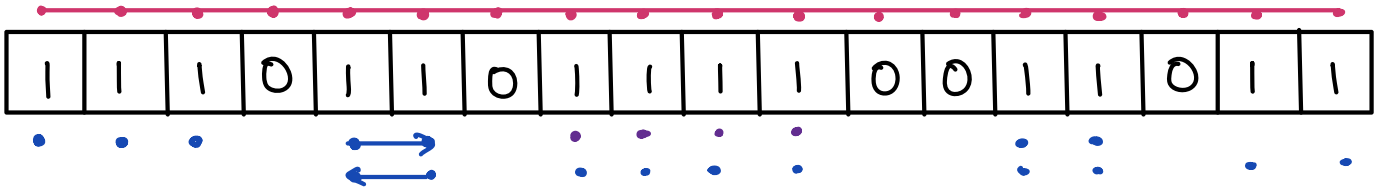
Analysis of TC

```
for( i=1 ; i≤3 ; i++){
    for (j=1 ; j≤n ; j++){
        //Stmts
    }
}
```

3 (outer) × 3 (inner)

$TC = 3n$

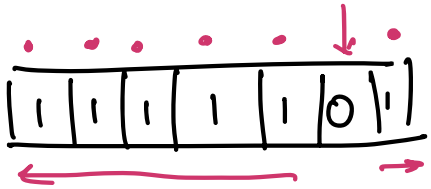
$TC \simeq O(n)$



Claim → At max, we are going to iterate on an ele 3 times

Eg: {1 0 1 0 1 0 1} — Max = 3 times

* If for each & every ele
  ↳ we are performing a task of n iterations = $O(n^2)$



1 time = n−1 ele

01. Given a binary ar[ ], we can atmost swap a single 0 with 1. Find the maximum consecutive 1's we can get in an ar[ ].

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

Ans → { 1 1 1 1 1 1 0 0 } Ans = 6

{ 1 0 0 1 1 1 1 1 } Ans = 5

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

{ 1 1 1 1 1 1 0 } Ans = 6

Eg :- { 0 1 1 1 0 1 1 0 0 } count of 1's = 5

l = 0
r = 3
val = 0+3+1
= 4

l = 3
r = 2
val = l+r

l = 2
r = 0
val = 2+0+1

l = 0
r = 0
val = 0+0+1

{ No extra 1 is present to swap this 0 }

Idea

For every 0'

01. calculate the no. of consecutive 1's in LHS

02 calculate the no. of consecutive 1's in RHS

```
if (l + r == count of 1's)
    val = l + r
else
    val = l + r + 1

if ( val > ans ) { ans = val }
```

Break = 10:15 → 10:25 pm

0→ **No. of triplets**

need not be continous

Given ar[N] elements, calculate no. of triplets (i, j, k)

such that $i < j < k$ and $ar[i] < ar[j] < ar[k]$

Eg :- { 2 6 9 4 10 }
0 1 2 3 4

$i < j < k$	$ar[i] < ar[j] < ar[k]$
$0 < 1 < 2$	$2 < 6 < 9$
$0 < 1 < 4$	$2 < 6 < 10$
$0 < 2 < 4$	$2 < 9 < 10$
$0 < 3 < 4$	$2 < 4 < 10$
$1 < 2 < 4$	$6 < 9 < 10$

count = 5

02. Eg :- { 4 1 2 6 9 7 }
0 1 2 3 4 5

$i < j < k$	$ar[i] < ar[j] < ar[k]$
$0 < 3 < 4$	$4 < 6 < 9$
$0 < 3 < 5$	$4 < 6 < 7$
$1 < 2 < 3$	$1 < 2 < 6$
$1 < 2 < 4$	$1 < 2 < 9$
$1 < 2 < 5$	$1 < 2 < 7$
$1 < 3 < 4$	$1 < 6 < 9$
$1 < 3 < 5$	$1 < 6 < 7$
$2 < 3 < 4$	$2 < 6 < 9$
$2 < 3 < 5$	$2 < 6 < 7$

ans = 9

Brute force → Consider all the triplets & check if they satisfy the condn or not.

```
int triplets (int [] ar)
{
    for (i=0; i<n; i++)
    {
        for (j=i+1; j<n; j++)
        {
            for (k=j+1; k<n; k++){
                if (ar[i] < ar[j] && ar[j] < ar[k]){
                    count ++;
                }
            }
        }
    }
}
```

TC: $O(n^3)$

SC: $O(1)$

* Optimised solution

arr = { 4, 1, 2, 6, 9, 7 }

0 1 2 3 4 5

For how many triplets 6 is going to be the middle element

No. of triplets having 6 as the arr[j]

= smaller ele on LHS * Larger ele on RHS

$\Rightarrow 3 * 2 = 6$

* Sorting → Ordering will change * Wrong

$i < j < k$

Idea → Consider each & every ele as your middle ele (arr[j])

- ↳ count of smaller ele on LHS
- ↳ count of larger ele on RHS

$cnt += l * r$

arr = { 4, 1, 2, 6, 9, 7 }
idx: 0 1 2 3 4 5

no. of smaller ele on left = 0 0 1 3 4 4

r = 3 4 3 2 0 0

no. of larger ele on right

0+0+3+6+0+0 ⇒ 9

```
int tripletsoptimised (int []ar)
{
    ans = 0

    for (i=0; i<n; i++)
    {
        int l=0 // smaller ele on left
        for (j=i-1 ; j≥0 ; j--){
            if (ar[j] < ar[i]) l++;
        }

        int r=0 : // greater ele on right
        for (j= i+1 ; j<n ; j++)
        {
            if (ar[j] > ar[i]) r++;
        }
        ans = ans + l * r;
    }
    return ans;
}
```

Note:
Balanced BST
TC = O(nlogn)

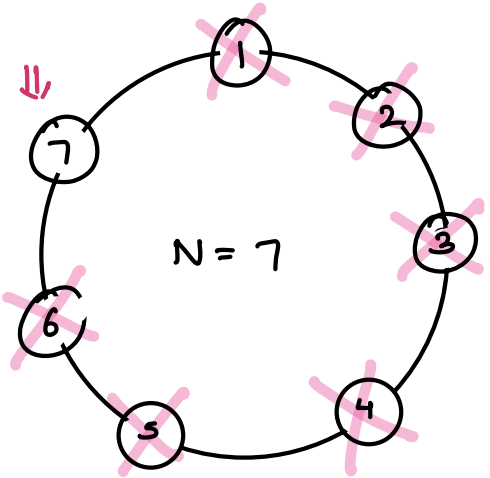
TC: O(n²)
SC: O(1)

Josephus Problem

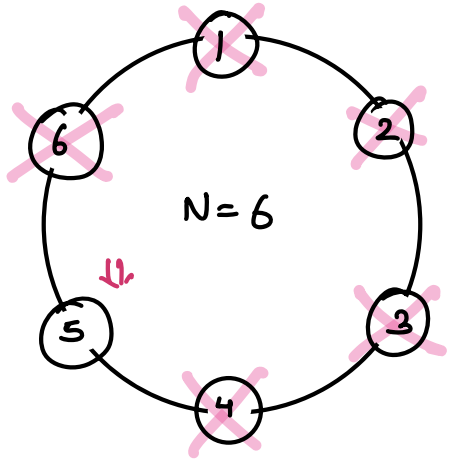
N people are standing in a circle. Person 1 has knife, he kills the next alive person in clockwise direction and passes on the knife to next person in clockwise direction.

Repeat the process until a single person is alive

Find the last person standing.



Ans = 7



Ans = 5

Ans = Last odd no. ✗

for odd → n

for even → n − 1 ✗

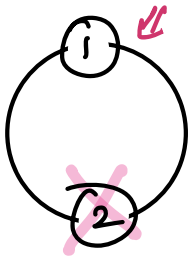
last prime no. ✗

N = 1

ans = 1

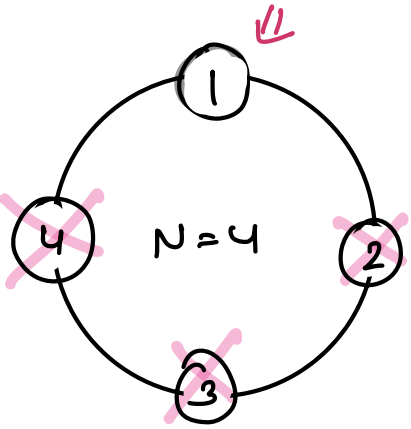
$N = 2^0$

N = 2



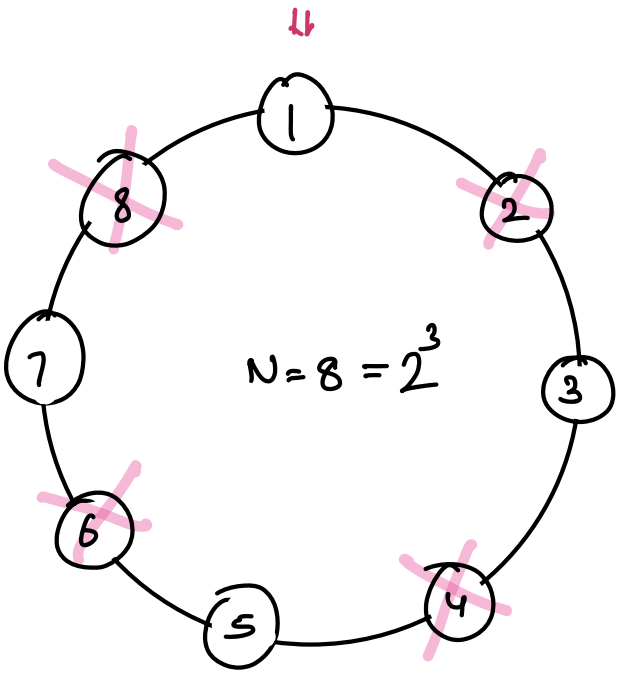
ans = 1

$N = 2^1$

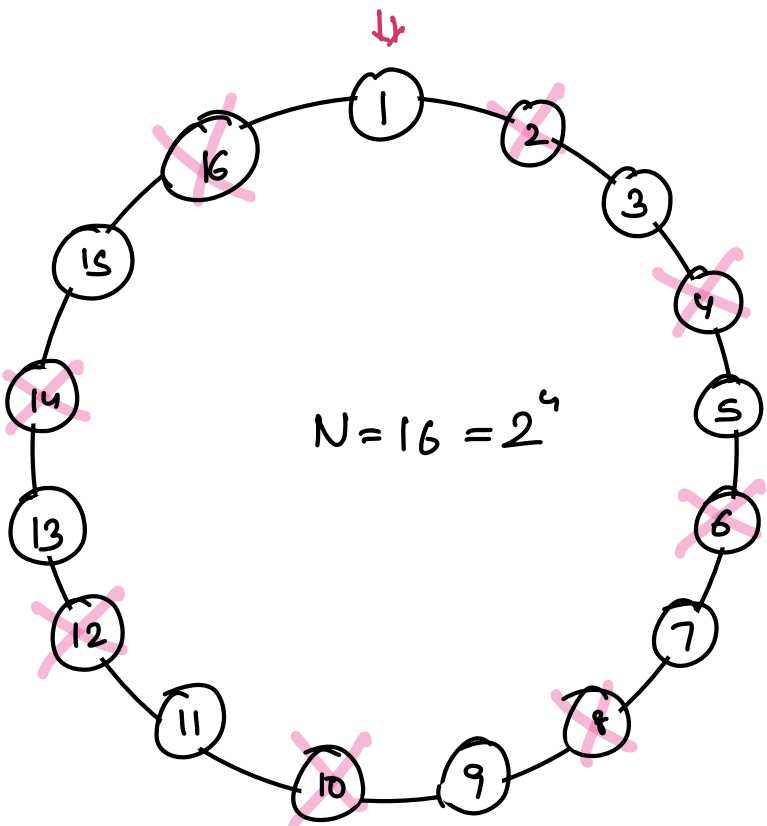


ans = 1

$N = 2^2$

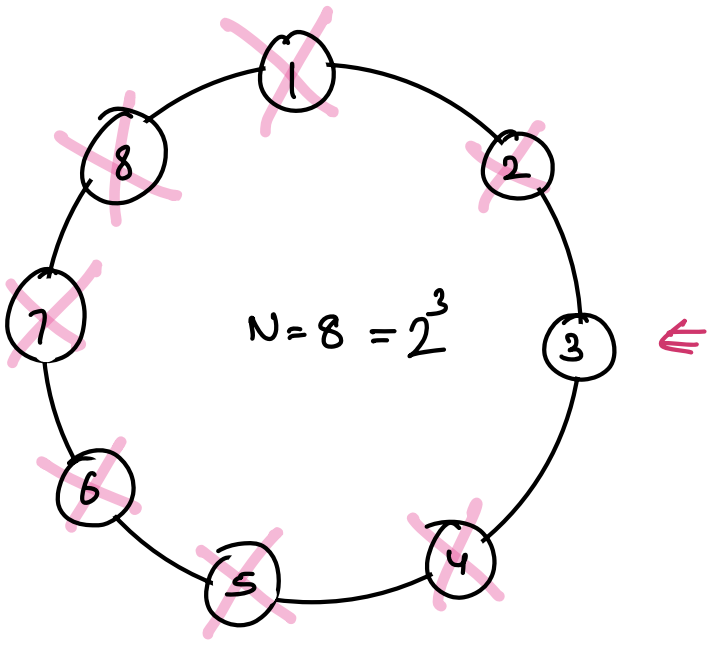


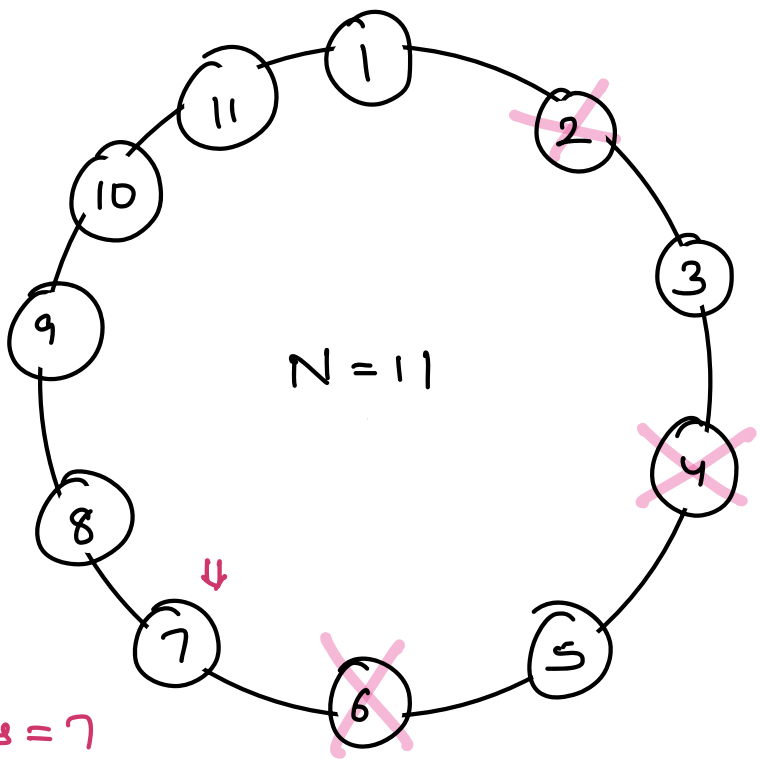
ans = 1



ans = 1

Obs → If n is a power of 2 then person starting the game is going to be our survivor

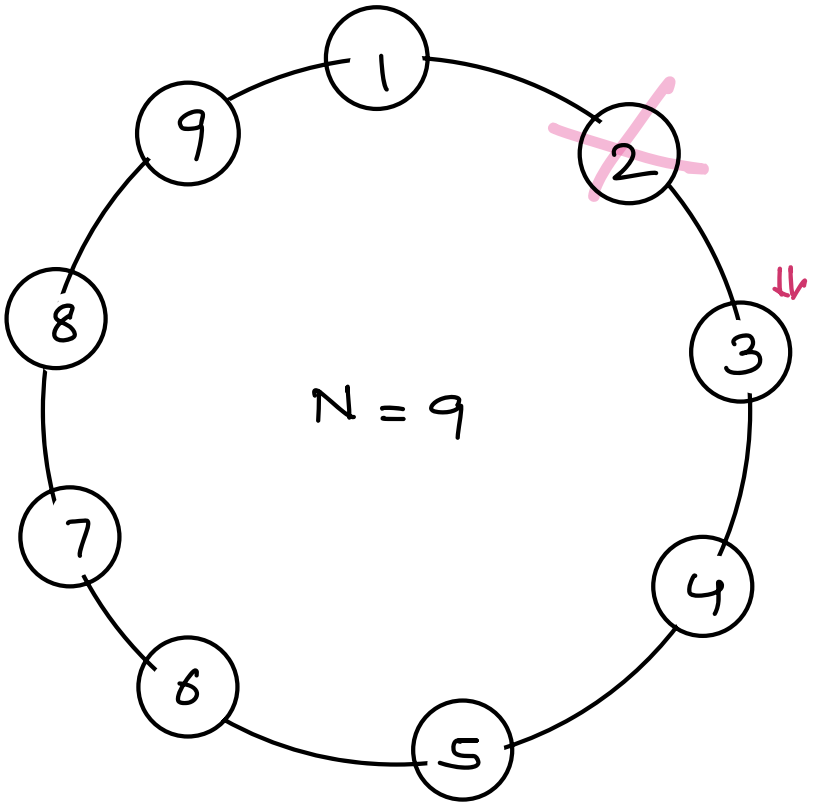




$1 + (3 * 2)$

$= 1 + 6 \Rightarrow 7^{th}$ person

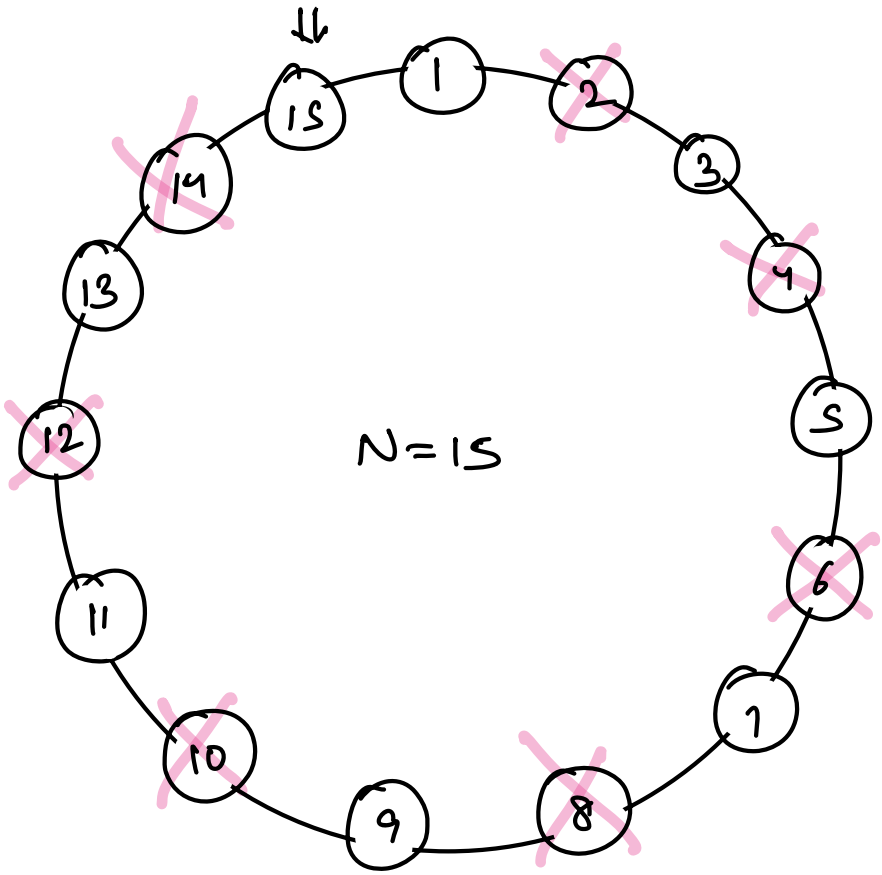
Ans = 7



Ans = 3

1 + (1 kill * 2)

= 1 + 2 = 3rd person



$2^3 = 8$

N = 15

↓ 7 kills

8 people

Person holding the knife after 7 kills $= 1 + (7 * 2)$

$= 1 + 14$

$= 15$

Ans = 15th person is going to survive

N = 100

01. Closest power of $2^x \leq 100 = 64$

02. No. of kills for 64 people = 100 − 64 = 36 standing

03. Who is going to hold the knife after 36 killings $= 1 + (36 * 2)$

$= 73^{rd}$ person

Ans = 73^{rd} person will survive

Steps

01: $2^x \leq n$, find the value of x

02. people to kill = $n - 2^x$

03. Ans = $1 + (2 *$ people to kill$)$

} Write the code

TC :

SC :

Doubts

Π 0, 1, 1, 2, 0, 1, 2, 1

Most basic idea

O(n)

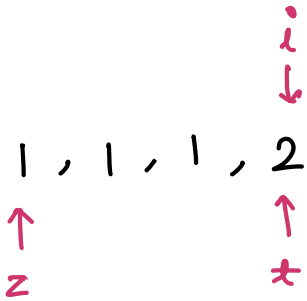
count 0 = 2
count 1 = 4
count 2 = 2

0 0 1 1 1 1 2 2 → count sort

Π" 0, 0, 0, 1, 1, 1, 2, 2

3 pointer approach

O(n)



while (i < t) {

- ar[i] is having 1 → move forward
- ar[i] is having 0 → swap i' idx with z idx
  i++, z++
- ar[i] is having 2 → swap i' idx with t' idx
  t--;

3

Arrays.sort( ) ⟶ $O(n \log n * \sqrt{n})$

Collection.sort( ) ⟶ $O(n \log n * \sqrt{n})$

1K

Google