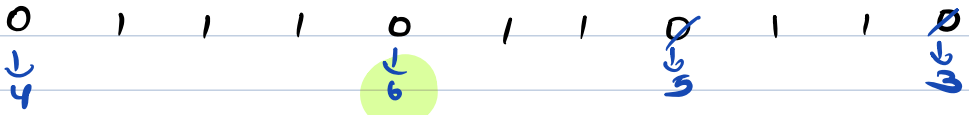
Today's Agenda :-

Max Consecutive 1's.

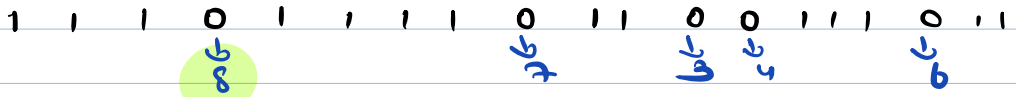
Ques) Max Consecutive 1's.

Given a Binary (0 & 1's) array, we can almost replace a single 0 with 1, find the max consecutive 1's we can get.

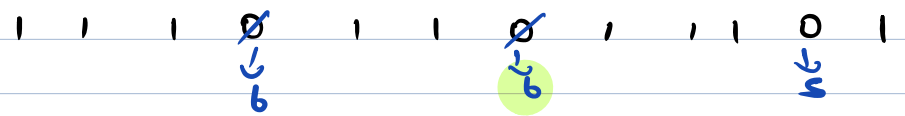
Ex1)



Ex2

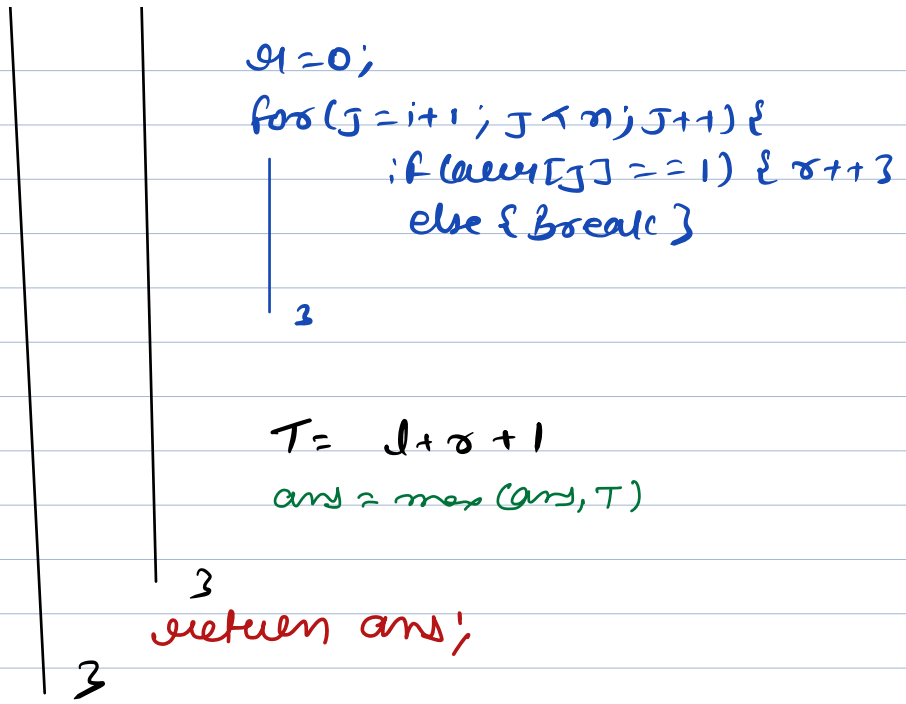


Ex3 :-

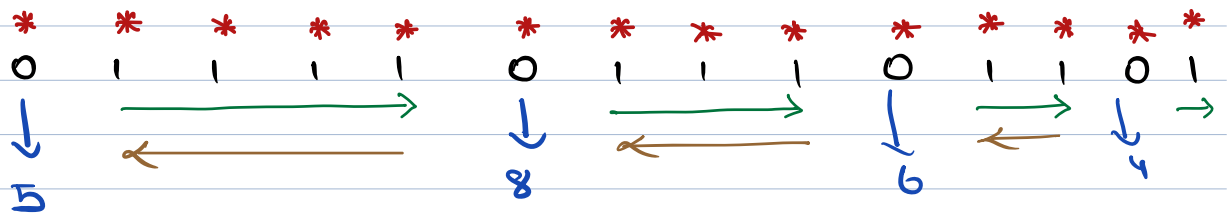


Brute force :-

```
ans = 0;
// Iterate & get count of 1's.
if (cnt == n) { return n }
for (i=0; i<N; i++) {
    if (arr[i] == 0) {
        l = 0;
        for (j = i-1; j >= 0; j--) {
            if (arr[j] == 1) { l+1 }
            else { Break }
        }
```



Ex :- 1 1 1 1 1 -> 5

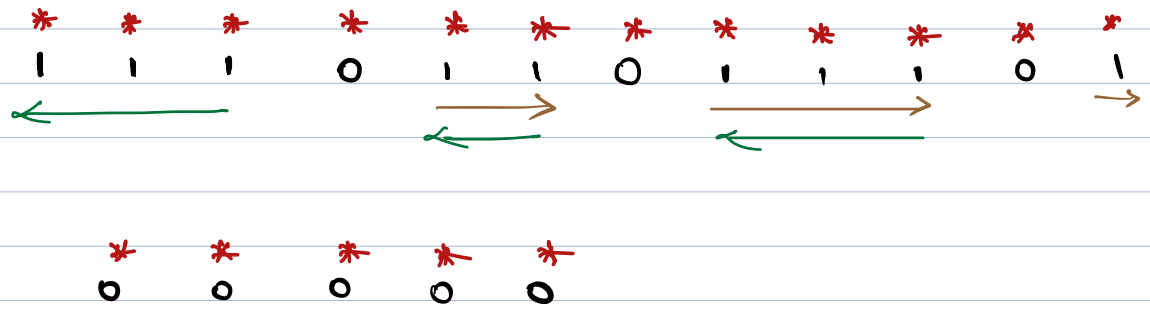


All the zeros are visited how many times → 1

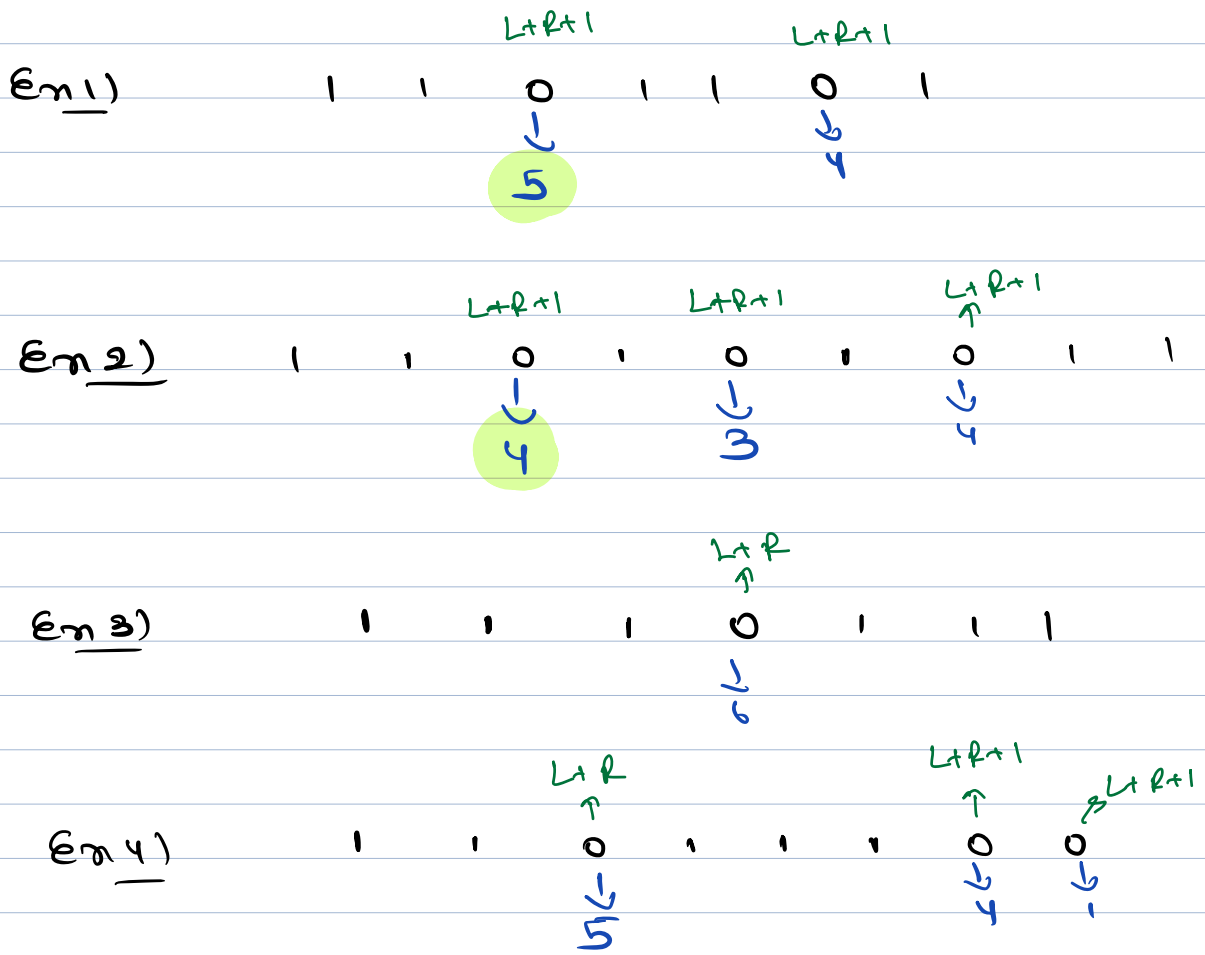
All the 1's are visited how many times → 3

T.C → O(3n) → O(n)

S.C → O(1)



Ques) Given a **Binary Array**, we can almost swap a single 0 with 1, find maxm consecutive 1's we can get.



```
ans = 0;
// Iterate & get count of 1's.
if (cnt == n) { return n }
for(i=0; i<N; i++) {
    if (arr[i] == 0) {
        l = 0;
        for (j = i-1; j >= 0; j--) {
            if (arr[j] == 1) { l++ }
            else { Break }
        }

        r = 0;
        for (j = i+1; j < n; j++) {
            if (arr[j] == 1) { r++ }
            else { Break }
        }
        if (cnt > l+r) {
            ans = max(ans, l+r+1);
        } else { ans = max(ans, l+r); }
    }
return ans;
}
```

T.C -> O(n);
S.C -> O(1);

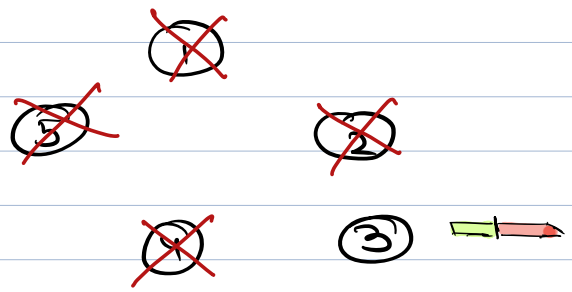
Break 9:53 - 10:10 pm

Majority -> Democracy

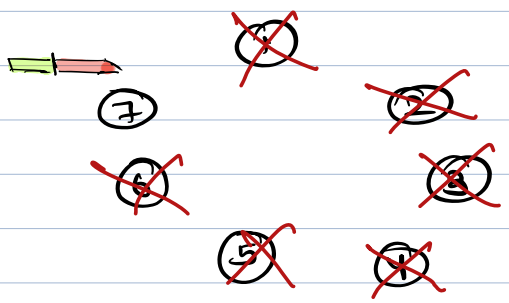
The only way out is through the process.

Ques) Josephus Problem :- → Google / Adobe.

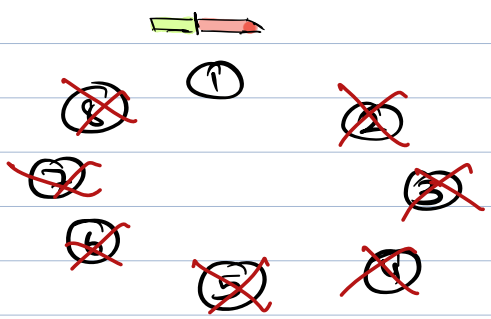
N = 5



N = 7



N = 8



N	ans
1	1
2	1
3	3
4	1
5	3
6	5
7	7
8	1
9	3
10	5

Observations :-

1) only odd can be ans.
2) for N: 2^n ans → 1

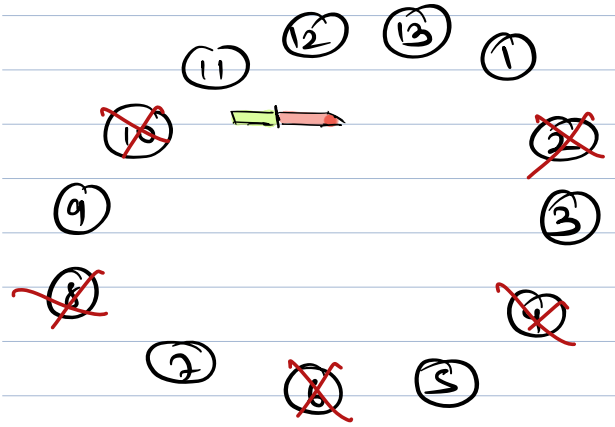
N=11, y=8

x = N-y+1 = 3.

N=11

remaining people	Starting place
11	1
10	3
9	5
8	7

N = 13



Remaining People	Starting Place
13	1
12	3
11	5
10	7
9	9
8	11

N, 1 = 3
N, 2 = 5
N, 3, = 7
N, x = 2x + 1. ← Starting place.

find the nearest power of 2
$x = N \rightarrow y$.

N − x = y. → nearest power of 2.
↓ ↓
Total people — x people got killed

$\Rightarrow \quad x = N - y$.

ans = 2x + 1

N = 13. y = 8. ∴ 13 − 8 = 5 ⇒ x.

ans = 2(5) + 1 ∴ 11.

```
cp = 1                                   N = 19.
while (cp <= N) {
        cp = cp * 2
}
```

y. = cp = cp/2. T.C : log N

Ques) **Doors**

There are n doors.
Initially all are closed.

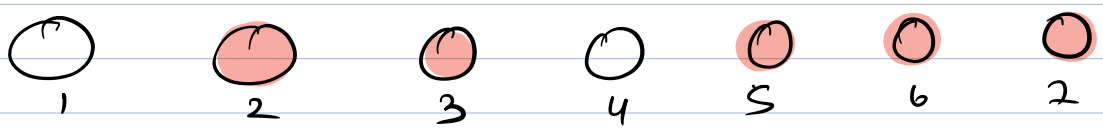
All the doors having 1 as their multiple will get toggled.

All the doors having 2 as their multiple will get toggled.

All the doors having 3 as their multiple will get toggled.

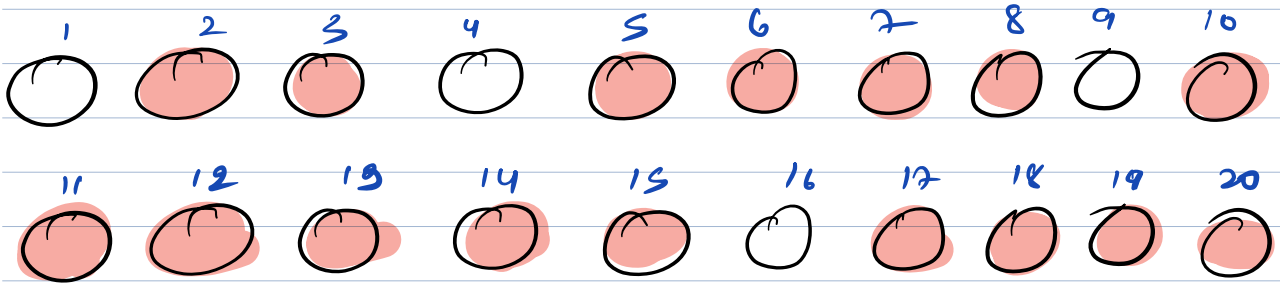
:

N = 7



Ans = 2

N = 20



Ans = 4.

9 → 1, 3, 9
15 → 1, 3, 5, 15
18 → 1, 2, 3, 6, 9, 18

if n has odd factors → open
if n has even factors → close.

Perfect square have odd factors.

100

1	100
2	50
4	25
5	20
10	10

Ans → all the perfect squares till no.

N = 100 → 1 to 100 → how many are perfect squares.

↓

1 to 10

→ return sqrt(N).

T.C → O(√N) → O(log N)

↓

binary search.

S.C → O(1).