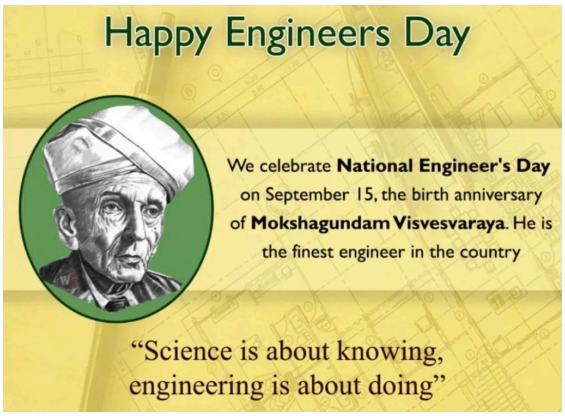


Today's Content

- Max consecutive 1's by
  - a) atmost 1 replace
  - b) atmost 1 swap
- Count triplets → { Goldman Sachs hiring challenge }
- Josephus Problem

{ ans → private chat
ques → public chat. }

Q) Given a binary (0's and 1's) 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
4 6 5 3

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

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
③ + 1 + ② = 6
7
8 3 5.

idea:

For every '0'

→ no. of consecutive 1's on l.h.s = (l)

→ no. of consecutive 1's on r.h.s = (r)

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

Edge case: if all 1's are there ⇒ N.

pseudo-code :

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

T.C → O(n)
S.C → O(1)

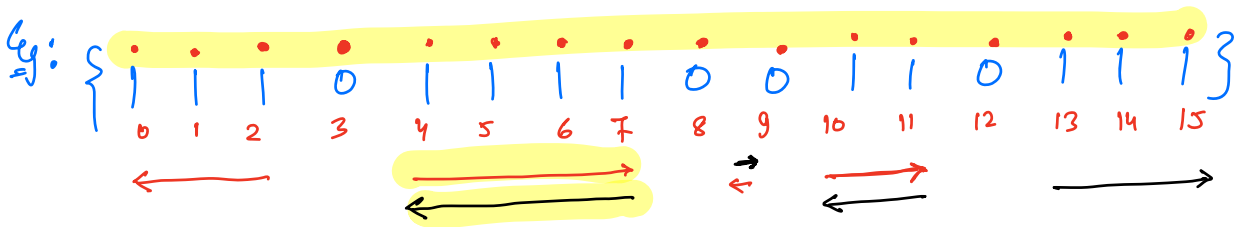
TC : Discussion

```
for ( i = 1 ; i <= 3 ; i++) {
    for ( j = 1 ; j <= n ; j++) {
        ======
    }
}
```

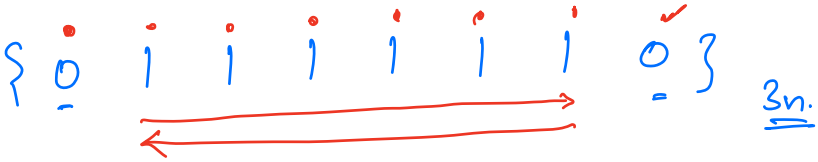
i	j	iterations
1	[1,n]	n
2	[1,n]	n
3	[1,n]	n
		3n.

T.C → O(N)

Eg:



At max, every element of the array is touched 3 times.



Note. → If break is present in nested loops, be careful while calculating 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

[ans = 6]

[l: 2, r: 3] → c = l+r+1 = 6.

[l: 3, r: 1] → c = l+r+1 = 5

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

[ans = 5]

l: 0, r: 3 → c = l+r+1 = 4.

l: 3, r: 2 → c = l+r = 5

l: 2, r: 0 → c = l+r+1 = 3

l: 0, r: 0 → c = l+r+1 = 1

count = 12

Eg: { 1 1 1 0 1 1 0 0 1 1 1 1 0 1 1 1 }
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

l: 3
r: 2

l+r+1

⇒ 8.

Pseudo-code:

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

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 (0) 6 (1) 9 (2) 4 (3) 10 (4) } ⟹ ans = 5

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

Eg2: arr[6] : { 4 (0) 1 (1) 2 (2) 6 (3) 9 (4) 7 (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 every triplet & check if it satisfy the condns.

```
int countTriplets( arr , N ) {
    count = 0
    for( i → 0 to N-1 ){
        for( j → i+1 to N-1 ) {
            for( k → j+1 to N-1 ) {
                if (arr[i] < arr[j] and arr[j] < arr[k]){
                    count += 1
                }
            }
        }
    }
    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 middle element.
{idx-3}

left	middle	right
{4, 1, 2}	6	{9, 7}
	[j]	

triplets.
left * right = 3 * 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. total triplets.

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

T.C → O(N²)
S.C → O(1)

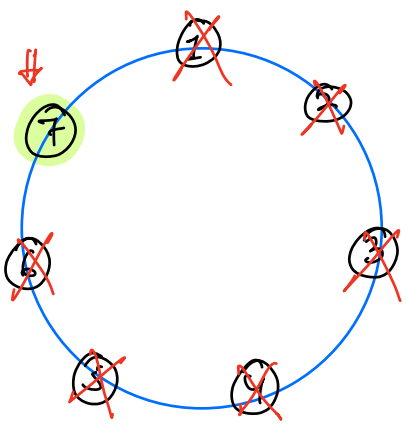
Josephus Problem

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

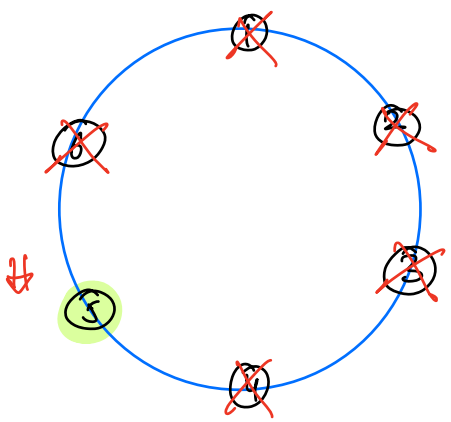
Repeat the process until a single person is alive.

Find the last person standing.

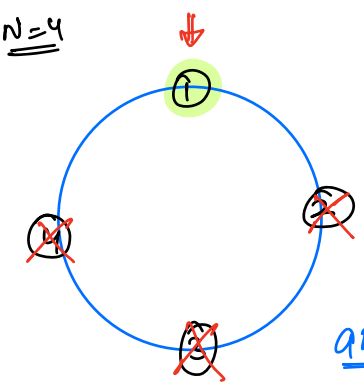
N=7.



N=6



N=4



ans=1

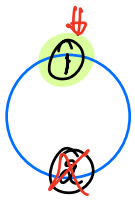
observations

① if n is odd ⇒ N
n is even ⇒ N-1] p

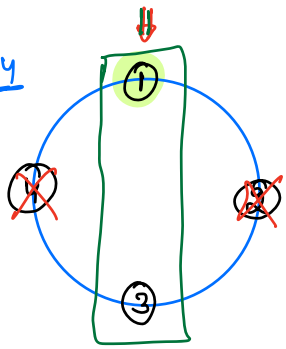
② largest prime <= N] p

N=1

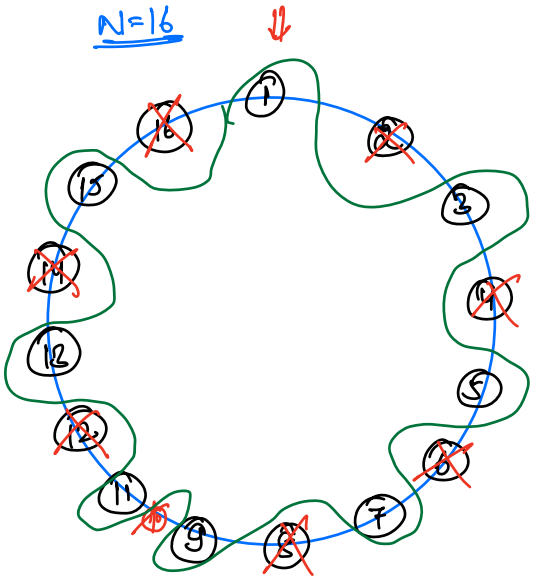
N=2.



N=4



N=16

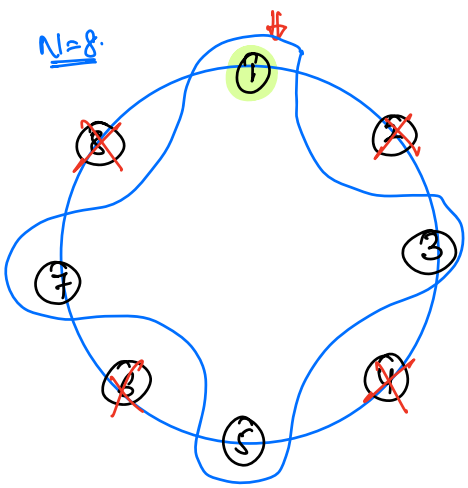


ans = 1.

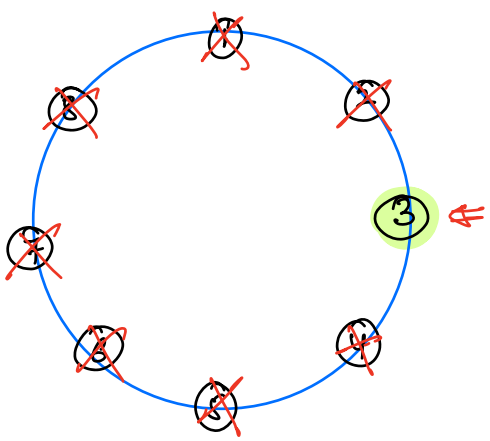
N = 32. 1 has knife ⇒ 1

[observation: If N is a power of 2, 1 starts the game, ans = 1]

N=8.

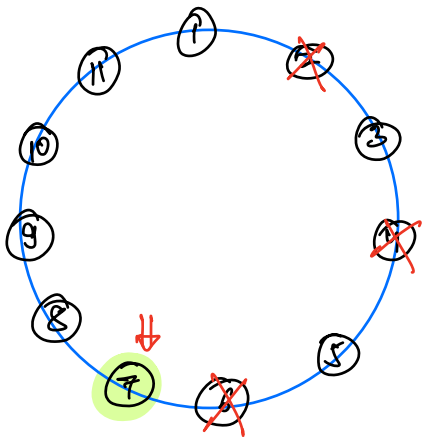


N = 8

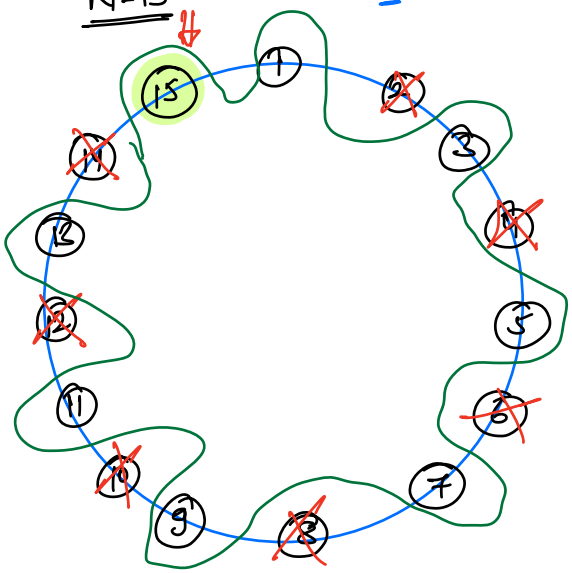


[observation 2.: If N is a power of 2, whoever starts the game will win the game.]

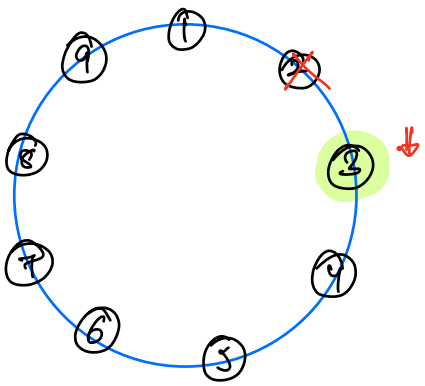
N = 11



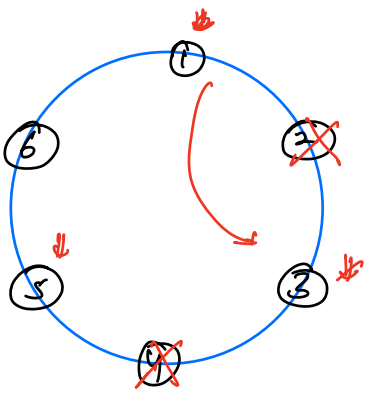
N = 15 → 7 Kills → 8 : 1 + (7)(2) = 15.



N = 9. → 1 Kill → 8 : 1 + (1*2) = 3.



N = 6. → 2 Kills → 4 : 1 + (2*2) = 5



N = 100

Nearest power of 2 which is less than 100 → 64.

no. of kills we need to make in order to attain 64 → 36.

1 + (36 * 2) = 73.

pseudo code. Todo.

step 1. → Find nearcast power of 2 (2^{np}) which is less than or equal to N

step 2. → kills = N - np

step 3 → ans = 1 + (2 * kills)

Doubts :

→ Generalized. → s.p can be any index.

↳ skip k person. {(k+1)th person}

maxProfit = 11

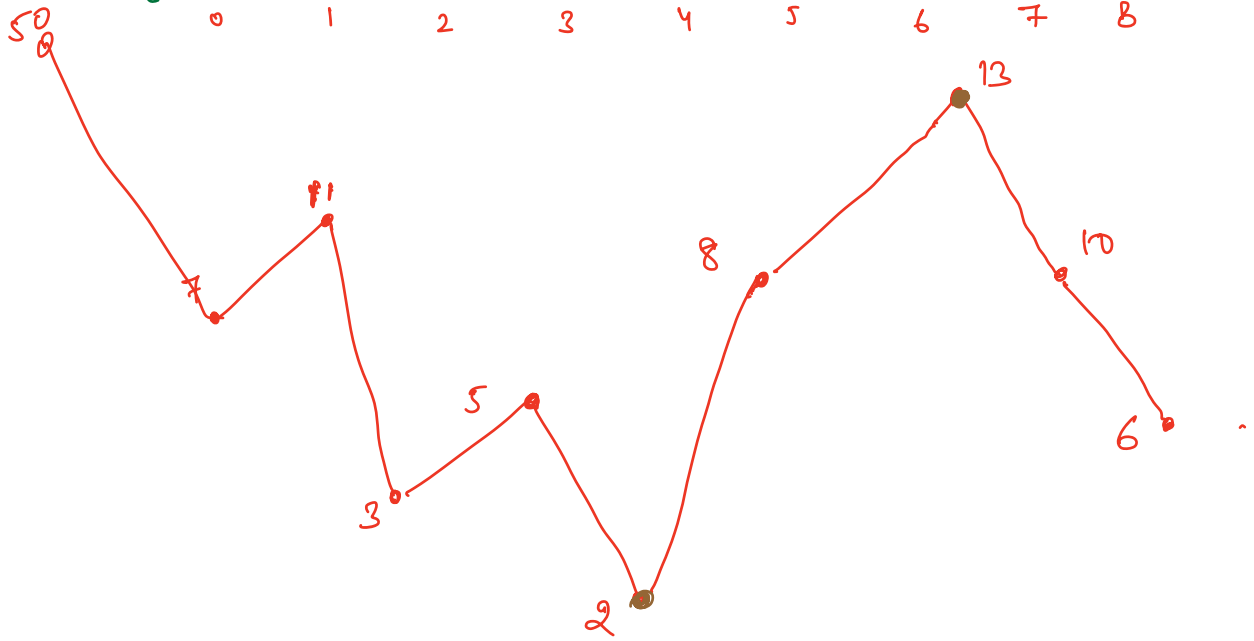
profit = max - arr[i]

Stocks :

max = ~~8~~ ~~10~~ 13

arr → [7 11 3 5 2 8 13 10 6]

0 1 2 3 4 5 6 7 8



Buy → sell.

Q: mat[N][N] => SXS.

$i + j = m + 1$

	0	1	2	3	4
0	1	2	4	7	8
1	3	5	6	9	14
2	10	11	12	13	15
3	16	17	19	22	24
4	18	20	21	23	25

m = 3.