Agenda

- Max consecutive 1's caused by
  - Atmost 1 swap
  - Atmost 1 replace
- Count triplets
- Josephus problem
- Comparator

Qu! Given binary arr[n], we can atmost replace a single 0 with 1. find max consecutive 1's in array.

(binary arr[n] → contains 0 and 1)

[Amazon, Adobe, Walmart]

Ex: arr[] = [1 1 0 1 1 0 1 1]

case 1: [1 1 0 1 1 0 1 1]

case 2: [1 1 0 1 1 0 1 1]

ans = 5

Ex: arr[] = [0 1 1 1 0 1 1 0 1 1 0]

case 1: [0 1 1 1 0 1 1 0 1 1 0] 4

case 2: [0 1 1 1 0 1 1 0 1 1 0] 6

Ex: arr[] = [1 1 1 1 1] ans = 5

Ex: arr[] = [1 1 1 0 1 1 0 1 1 1 1 0 0 1 1 0]

consecutive 1's on left [3]
+
consecutive 1's on right [2]
+
1 [1]
[6]

Observation

consecutive 1s on left — prefix[]

consecutive 1s on right — suffix[]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
arr[] =	1	1	1	0	1	1	0	1	1	1	1	0	0	1	1	0
pfx[] =	1	2	3	0	1	2	0	1	2	3	4	0	0	1	2	0
sfx[] =	3	2	1	0	2	1	0	4	3	2	1	0	0	2	1	0

pfx[0] = consecutive 1's from 0th idx to 0th idx.

pfx[1] = consecutive 1's [0, 1] idx

= pfx[0] + if arr[1] == 1 [1]

pfx[3] = consecutive 1's from [0, 3] idx

sfx[i] = consecutive 1's from ith idx to n-1 idx } This statement ?

pfx[i] = " " " 0th " " ith idx

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
arr[] =	1	1	1	0	1	1	0	1	1	1	1	0	0	1	1	0
pfx[] =	1	2	3	0	1	2	0	1	2	3	4	0	0	1	2	0
sfx[] =	3	2	1	0	2	1	0	4	3	2	1	0	0	2	1	0

pf[2] + 1 + sfx[4]

pfx[i-1] + 1 + sfx[i+1]

Edge case:

1) 0th idx = pf[0-1] + 1 + sfx[0+1]

0 + 1 + sfx[1]

2) n-1 idx = pf[n-1-1] + 1 + sfx[n-1+1]

pf[n-2] + 1 + 0

```
int maxConsecutive1s(int[] arr) {
    int[] pfx = consecutivePrefixOnes(arr);
    int[] sfx = consecutiveSuffixOnes(arr);
    int ans = -1;
    // Edge cases.
    1. 0th idx
    if (arr[0] == 0) {
        ones = 1 + sfx[1];
        ans = Math.max(ans, ones);
    }
```

h/w.

2. n-1 idx.

```
if ( arr[n-1] == 0) {
    ones = pf[n-2] + 1;
    ans = Math.max(ans, ones);
}

for(i=1; i<n-1; i++) {
    if ( arr[i] == 0) {
        ones = pfx[i-1] + 1 + sfx[i+1];
        ans = Math.max(ans, ones);
    }
}
if (ans == -1) {
    return arr.length;
}
return ans;
}
```

I have an array with all 1's.

TC : O(n)

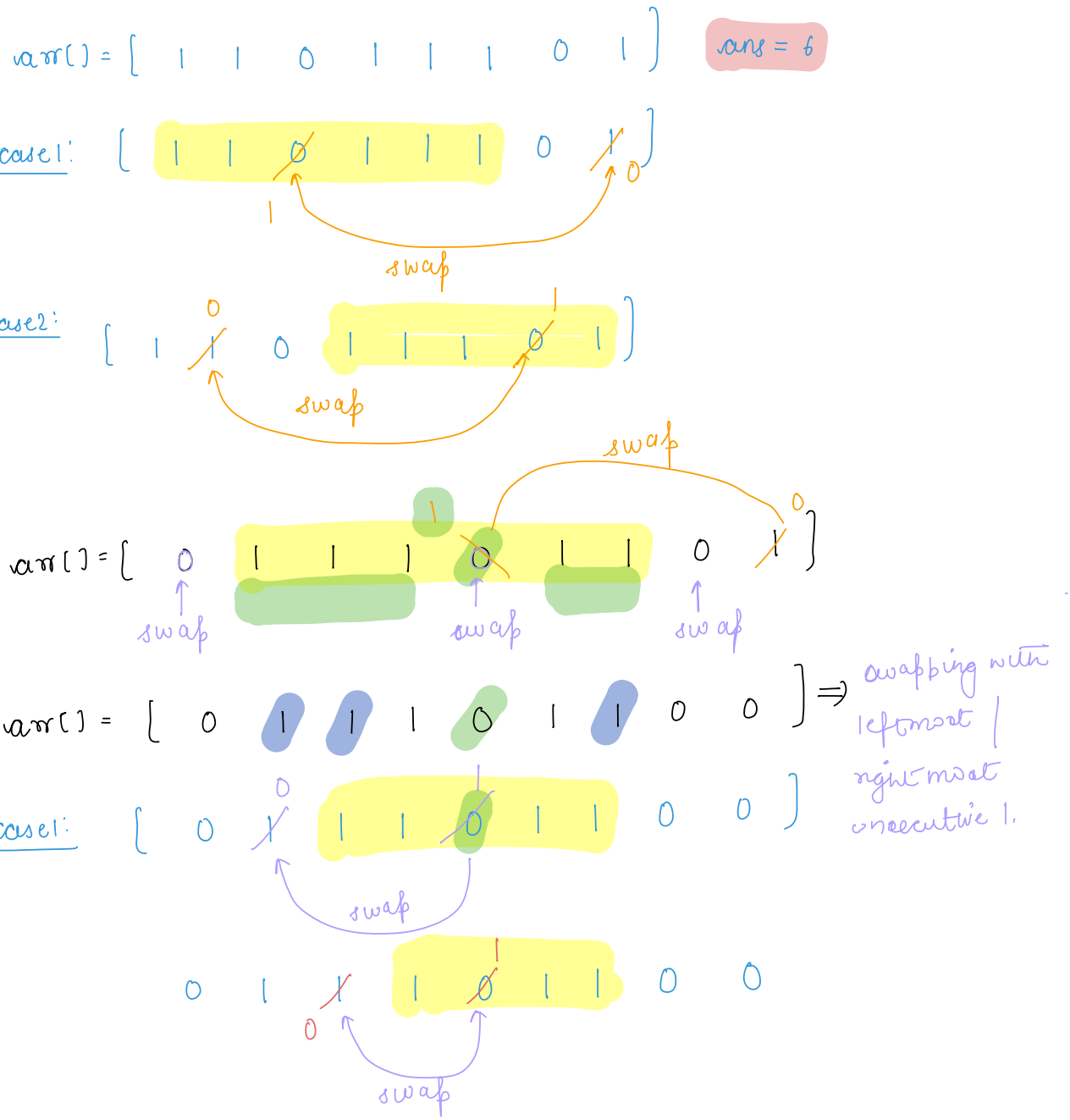
SC : O(n)

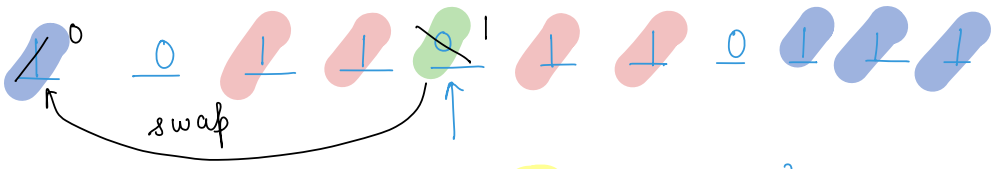
Assignment :- SC: O(1) ?? [Is it possible]

↳ carry forward

Qu2 follow up.

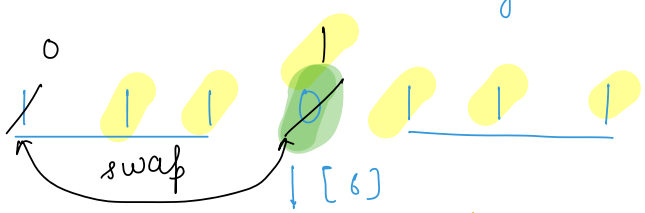
Given binary arr[], find max consecutive 1's. We can do atmost one swap.





pf[i-1] + 1 + sfx[i+1]

if I have extra 1's which are not part of left & right consecutive 1's



pf[i-1] + ~~1~~ + sfx[i+1]

0

if I have no extra 1's. [pf[i-1] + sfx[i+1]]

arr[] = [0 0 1 1 1 0]

pfx[i-1] + sfx[i+1] + 1 (if extra 1's)

0 (if no extra 1's)

swapping ⟶ for any 0 –

pfx[i-1] + sfx[i+1] + if extra 1's (1)

else [0]

```
int maxConsecutive1's (int[] arr) {
    int onesCount = 0;
    for (int el : arr) {
        if (el == 1) {
            onesCount ++;
        }
    }
```

```
int[] pfx = consecutivePrefixOnes(arr);
int[] sfx = consecutiveSuffixOnes(arr);
```

h/w.

```
int ans = -1;
// Edge cases.
1. 0th idx
if (arr[0] == 0) {
    int consecutive1s = sfx[1];
    if (onesCount > consecutive1s) {
        ones = 1 + sfx[1];
        ans = Math.max(ans, ones);
    } else {
        ones = sfx[i];
        ans = Math.max(ans, ones);
    }
}
```

differentiating 1st & 2nd problem

```
:
:
```

Try it out??

Qu2 Goldman sachs

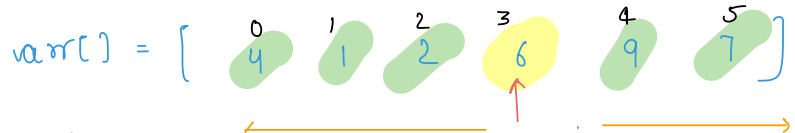
Given arr[n], calculate no of triplets i, j & k such that

$i < j < k$ and $arr[i] < arr[j] < arr[k]$

arr[5] = | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 2 | 6 | 9 | 4 | 10 |

i	j	k	
0	1	2	2 < 6 < 9
0	1	3	
0	1	4	2 < 6 < 10
0	2	3	
0	2	4	
0	3	4	2 < 4 < 10
1	2	3	
1	2	4	
2	3	4	
1	3	4	

Expected TC: $O(n^2)$



arr(i) < arr[j] < arr[k]

4 < 6 < 9

i	j	k
0	3	4
0	3	5
1	3	4
1	3	5
2	3	4
2	3	5

jth idx = 3

└ No of el less than arr[3] on left *

No of el greater than arr(3) on right

3 * 2 = 6

Algo: fix jth idx [1 , n-2 idx]

middle idx

left — find no of el less than arr[j] in left [0, j-1]

right — find " " " greater " " " right [j+1, n-1]

— cnt = left * right

— ans = ans + cnt;

```
int countTriplets(int[] arr) {
    int ans = 0;

    // fix jth idx
    for(j = 1; j <= n-2; j++) {

        // el lesser on left side
        int left = 0;
        for(i = 0; i < j; i++) {
            if(arr[i] < arr[j]) {
                left++;
            }
        }

        // el greater on right side
        int right = 0;
        for(int k = j+1; k < n; k++) {
            if(arr[k] > arr[j]) {
                right++;
            }
        }

        ans = ans + (left * right);
    }

    return ans;
}
```

O(n) — for(j = 1; j <= n-2; j++) {

O(n) — for(i = 0; i < j; i++)

O(n) — for(int k = j+1; k < n; k++)

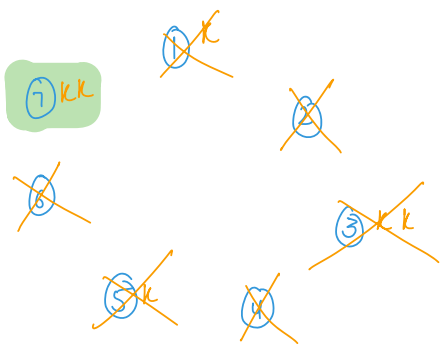
Can prefix solve this??

TC: $O(n^2)$

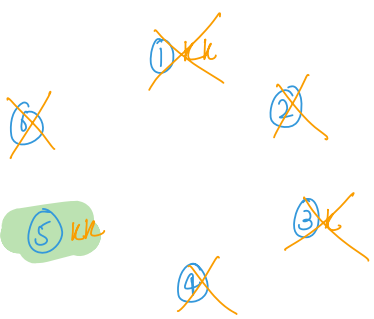
Qu 4 : Squid Game [Josephus problem]

- N people standing in a circle, clockwise
- Person 1 has the knife
- He kills the next person in clockwise manner and passes the knife to next alive person
- Repeat until one person is alive
- find the last man standing

n = 7



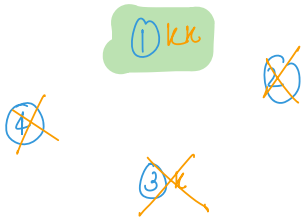
n = 6



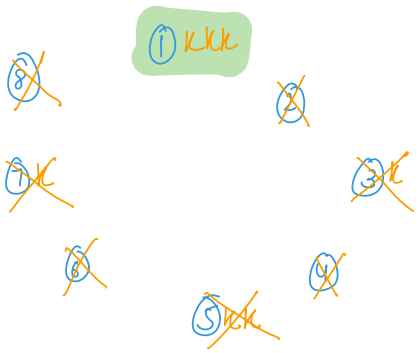
n = 1

n = 2

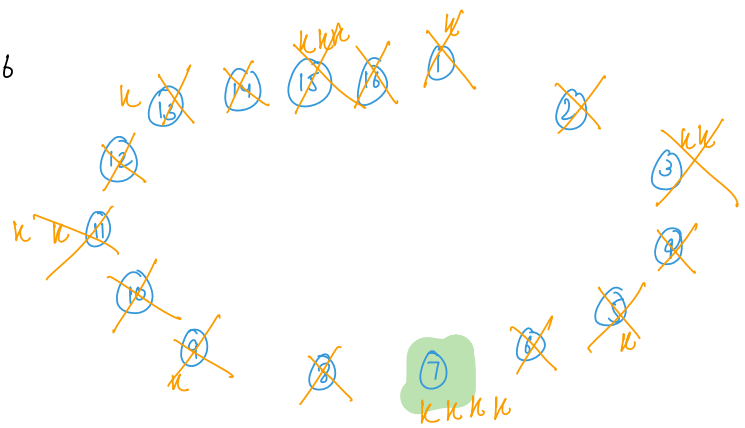
n = 4



n = 8



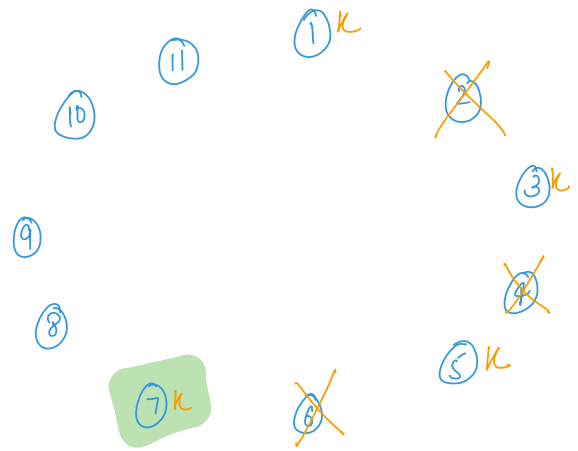
n = 16



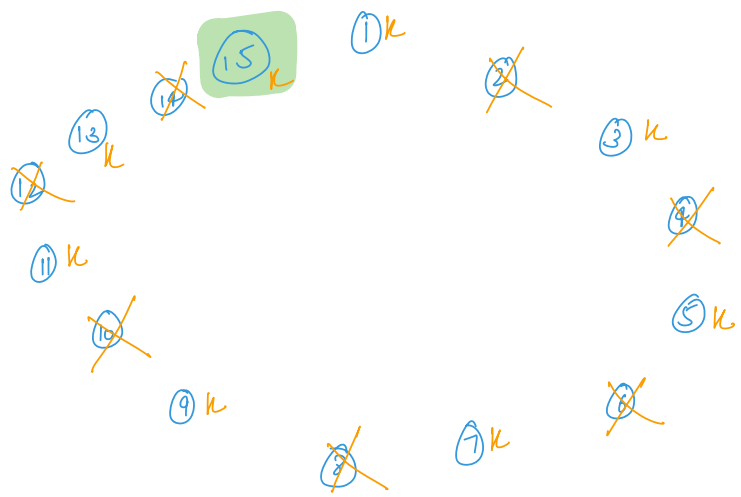
If n is power of 2, whoever has the knife is winner.

[We are killing next person] — α^n.

n=11 → 3 kill → 8

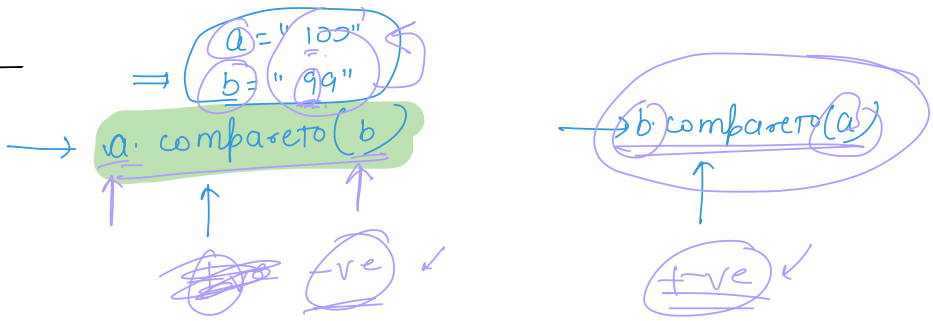


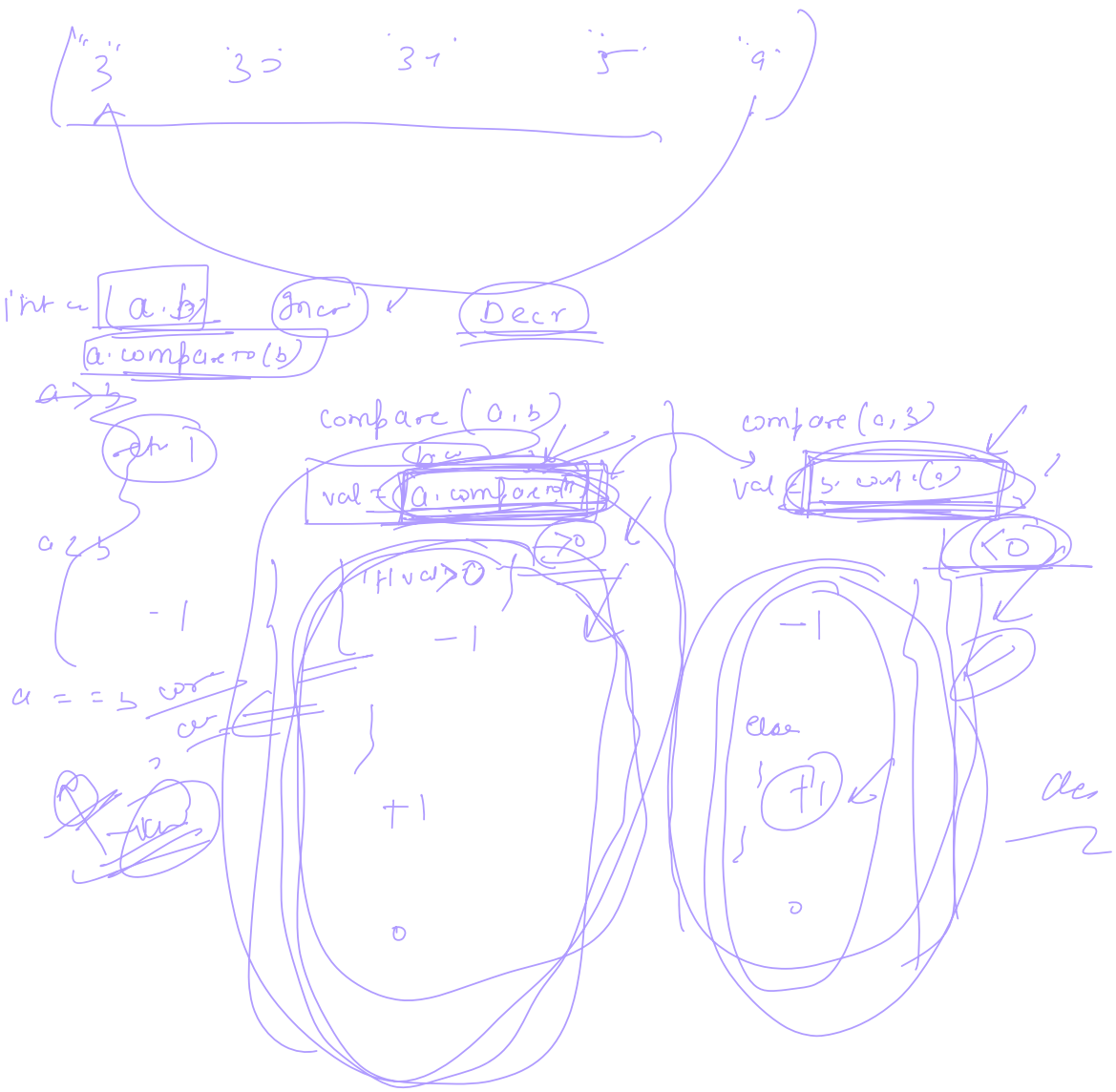
n=15 → 7 kill → 8



code it out ??

Doubt


a="100"
b="99"
a.compareTo(b)
b.compareTo(a)
-ve
+ve


3
3
3
5
9
int a
a.b
a.compareTo(b)
Incr
Decr
compare(a,b)
compare(a,3)
val = a.compareTo(b)
val = 5.compareTo(3)
>0
<0
if val>0
-1
+1
0
else
a==b