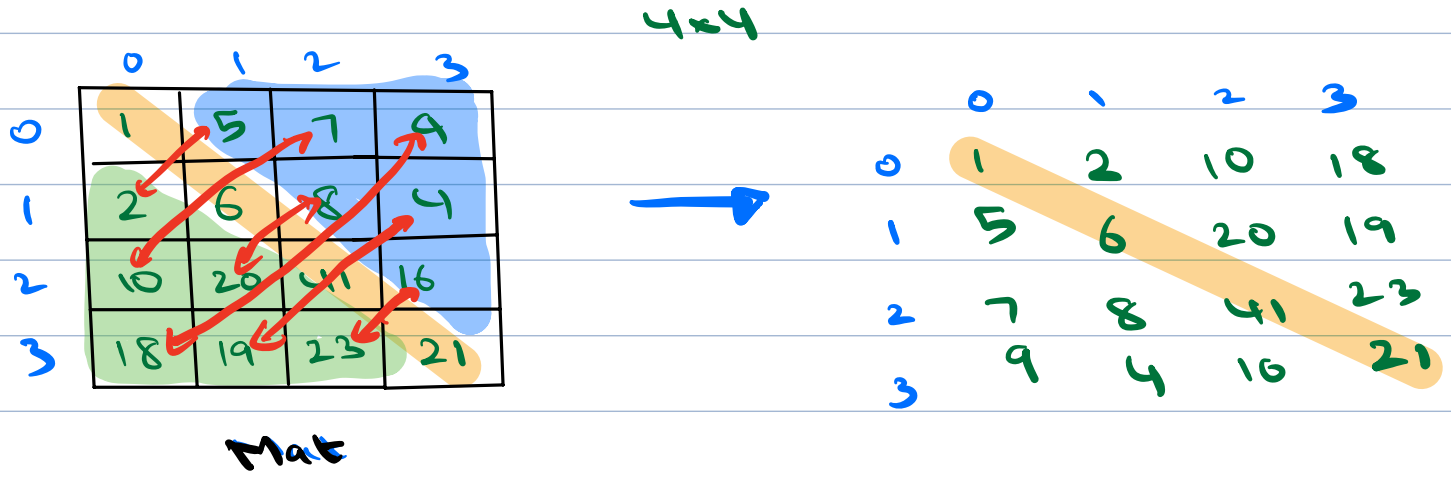
Matrix Transpose

Rotate Matrix by 90 degrees

Length of longest consecutive ones

Reverse String Word By Word

Q. Given a square mat[N][N], find transpose (SC : O(1))



Transpose: interchanging rows and cols of matrix

① All cells where i = j, they're not changing position

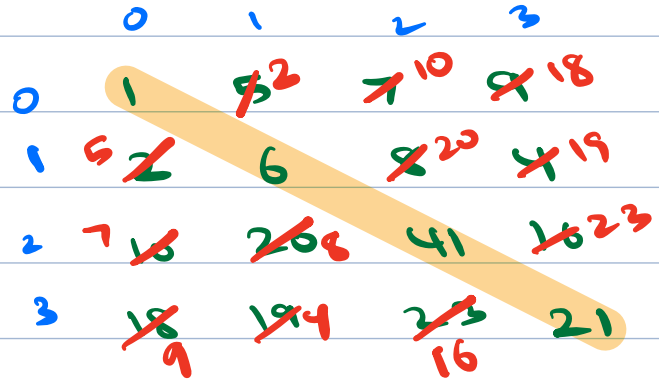
② Swap elements of upper △ or lower △

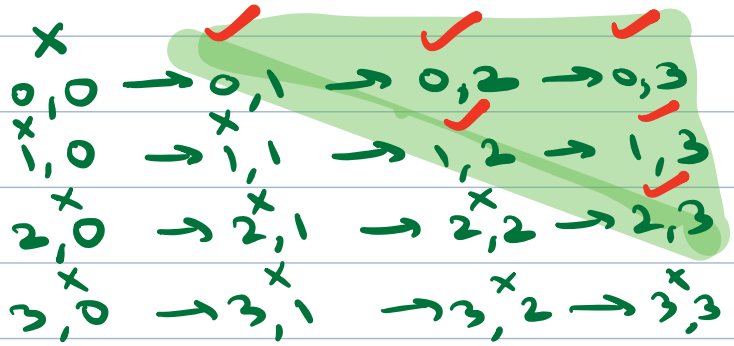
5 ⇆ 2
0,1 ⇆ 1,0

1 ⇆ 10
0,2 ⇆ 2,0

9 ⇆ 18
0,3 ⇆ 3,0

i,j ⇆ j,i





row 0 → col 1 to 3
row 1 → col 2 to 3
row 2 → col 3 to 3

```
for(row=0 ; row<N-1 ; row++){
  for (col=row+1 ; col<N ; col++){
    swap(mat[row][col],
         mat[col][row])
  }
}
```

//mat[N][N]

Total el = N × N = N^2

el apart from diagonal = $N^2 - N$

el in upper $\Delta = \frac{N^2 - N}{2}$

TC: $O(N^2)$ SC: $O(1)$

mat[3][3]

	0	1	2
0			
1			
2			

trans[3][3]

mat[3][4]

	0	1	2	3
0				
1				
2				

trans[4][3]

	0	1	2
0			
1			
2			
3			

6. Given mat[N][N], rotate by 90° clockwise

SC: O(1)

1	5	7	9
2	6	8	4
10	20	41	16
18	19	23	21

Mat

→

18	10	2	1
19	20	6	5
23	41	8	7
21	16	4	9

1	5	7	9
2	6	8	4
10	20	41	16
18	19	23	21

Mat
4x4

→

18 10 2 1
19 20 6 5
23 41 8 7
21 16 4 9

Rotated
4x4

	0	1	2	3
0	1	2	10	18
1	5	6	20	19
2	7	8	41	23
3	9	4	16	21

Transpose

Rev →
Rev →
Rev →
Rev →

18 10 2 1
19 20 6 5
23 41 8 7
21 16 4 9

Rotated
4x4

① Get transpose

② Reverse each row

① transpose(mat) → N^2

②
```
for (row = 0 ; row < N ; row++) {
    // reverse a row
    int i = 0 , j = N-1
    while (i < j) {
        swap(mat[row][i], mat[row][j])
        i++   j--
    }
}
```

Reverse 1 row → N/2

Reverse N rows → N × N/2
= $N^2/2$

TC: $O(N^2)$ SC: $O(1)$

MS, LinkedIn, Amazon

3. Given a binary array (all elements are 0/1), we are allowed to replace atmost one 0 with a 1.
Find the max length of consecutive 1s.

Approach: Try replacing every 0, and find max consecutive ones because of replacement.

If A[i] = 0

① Count consecutive 1s on left side
(l → i-1 to 0)

② Count consecutive 1s on right side
(r → i+1 to N-1)

l ..... 0 ..... r

cnt = l + r + 1

ans = max (ans, cnt)

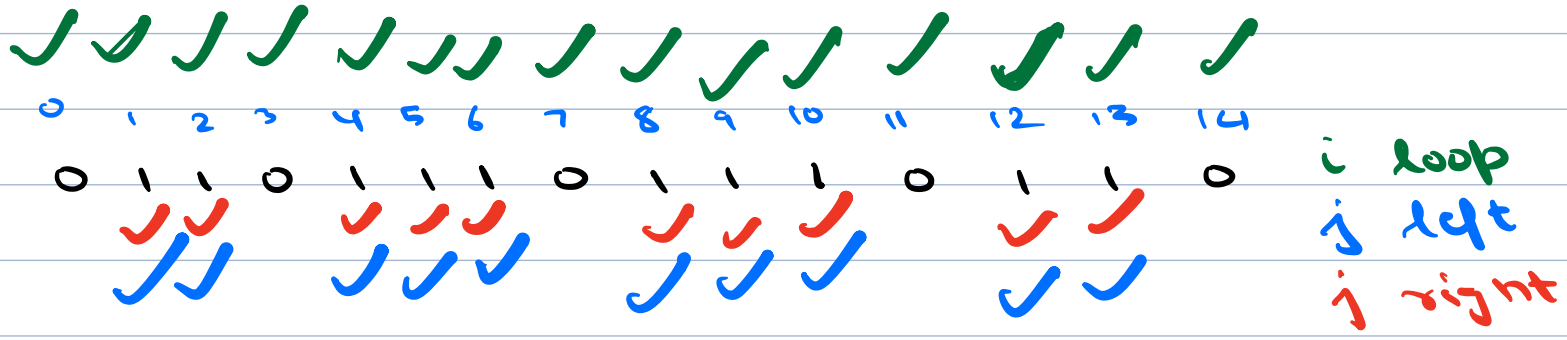
```
int findMaxConsecutiveOnes(int nums[]){

    int n = nums.size()
    int ans = 0

    for(i=0 ; i<N ; i++){
        if(nums[i] == 0){
            // replace 0 with 1

            int j = i-1, cnt_left = 0
            while(j>=0 && nums[j] == 1){
                cnt_left++  j--
            }

                j = i+1, cnt_right = 0
            while(j<N && nums[j] == 1){
                cnt_right++   j++
            }

            int cnt = cnt_left + cnt_right + 1
            ans = max(ans, cnt)
        }
    }

    if(ans == 0)
        return N
    return ans
}
```

A = [1 1 1 1]

ans = 4

A = [0 0 0 0]

ans = 1



1 dc → 3 itr
N dc → 3N itr

ans = ~~0~~ ~~1~~ ~~2~~ 7

TC : O(N)
SC : O(1)

1 dc → N itr
N dc → N² itr

ans remains 0 in 2 cases

① empty array → [ ]
loop won't run

② array containing all 1s
if condition is always false

actual ans = N

N=0 ① → 0

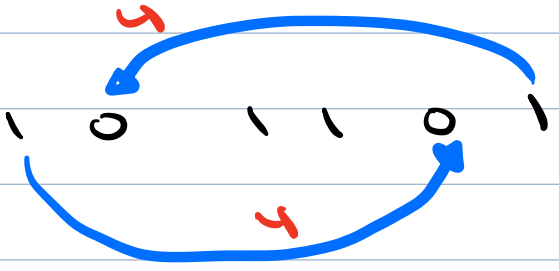
② N=5 [11111] → 5

10:30

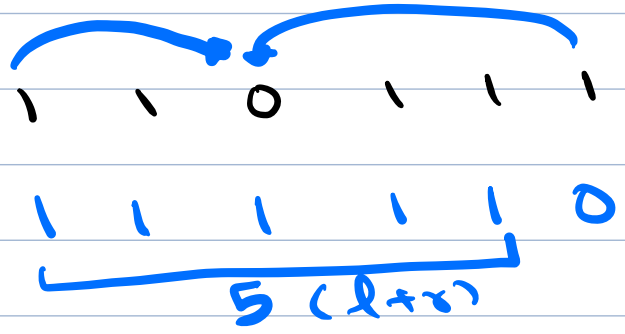
4. Modified version of max consecutive 1s

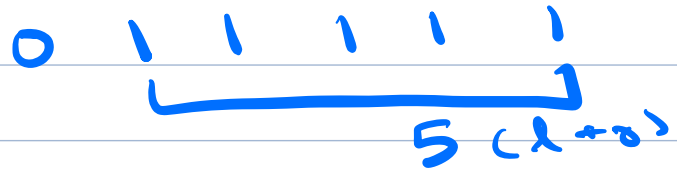
Given a binary array (all elements are 0/1), we are allowed to swap atmost one 0 with 1 present in array itself.
Find the max length of consecutive 1s.

Ex 1 A = 1 0 1 1 0 1 ans = 4



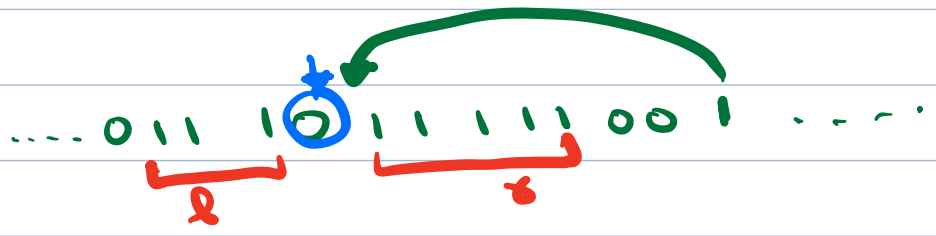
A = 1 1 0 1 1 1





Approach : Swap 0

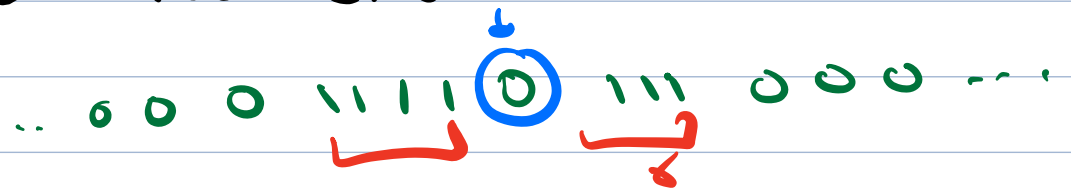
Case 1 :



when there is an extra 1 (apart from l and r)

```
if (l + r < totalOnes) {
    cnt = l + r + 1
}
```

Case 2 : No extra 1



cnt = l + r

```
int findMaxConsecutiveOnes(int nums[]){
    int n = nums.size()
    int ans = 0, totalOnes = 0
    for(i=0; i<N; i++){
        if(nums[i] == 1)
            totalOnes++
    }
    if(totalOnes == N) return N
    for(i=0; i<N; i++){
        if(nums[i] == 0){
            // swap 0 with 1
            int j = i-1, cnt-left = 0
            while(j>=0 && nums[j] == 1){
                cnt-left++ j--
            }
            j = i+1, cnt-right = 0
            while(j<N && nums[j] == 1){
```

```
        cnt = cnt_left + cnt_right
    }
    if (cnt_left + cnt_right <
                        totalOnes) {
        cnt = cnt_left + cnt_right + 1
    }
    else {
        cnt = cnt_left + cnt_right
    }
    ans = max(ans, cnt)
}
return ans
}
```

A = [1 1 1 1]

ans = 4

A = [0 0 0 0]

ans = 1

1 1 1 0 1 1 0 1 1 0 1

3 2 6

ans = 0

totalOnes = 8

0 0 1 1 1 0 1 1 0 0 0

3 2

totalOnes = 5

TC: O(N)

SC: O(1)

5. Given an input string s, reverse order of the words.

s = "scaler is the best"

o/p : best the is scaler

s = "hello world"

o/p : world hello

Approach 1 : 1. Split the words by space

2. Reverse word one by one and append them together.

```
String reverseWords (String s) {

    String[] words = s.split("\\s+")
    StringBuilder sb = new StringBuilder()
    for(i = words.length()-1 ; i ≥ 0 ; i--){
        sb.append(word[i])
        if (i > 0)
            sb.append(" ")

    return sb.toString()
```

```
solve(A):
    A = A.strip()
    A = A.split()
    A = A[::-1]
    return ' '.join(A)
```

Approach 2

① "I like scaler" → Reverse → "relacs ekil I"

② Reverse every word scaler like I