- Transpose of a Square Matrix
- Rotate Matrix
- Maximum Consecutive 1's
  - → Atmost 1 Replace
  - → Atmost 1 Swap
- Reverse String

Notes

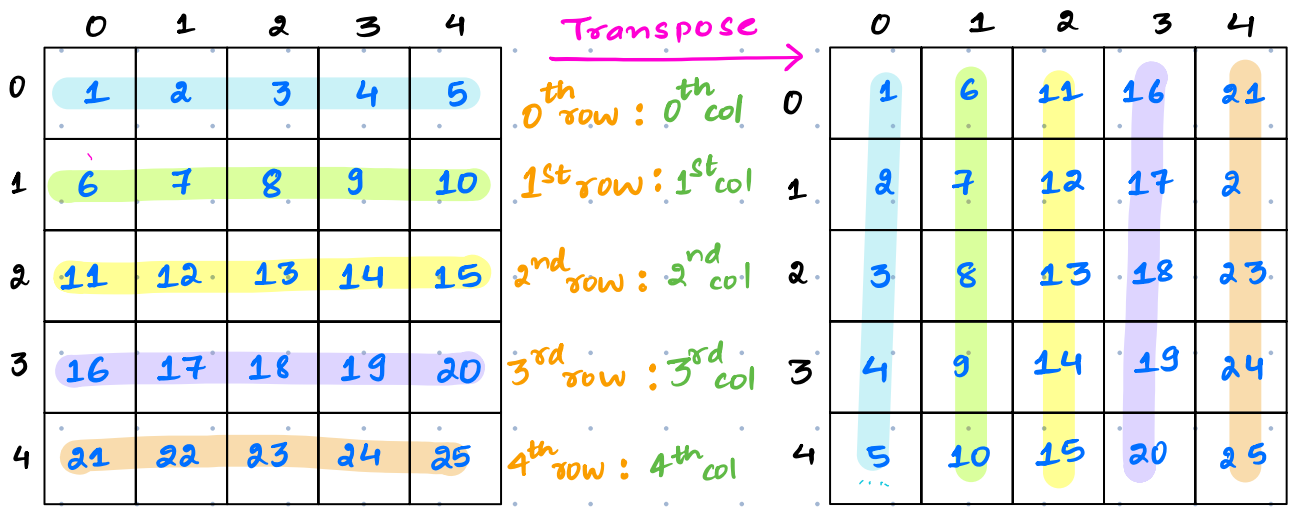
NO OF ROWS = NO OF COL.

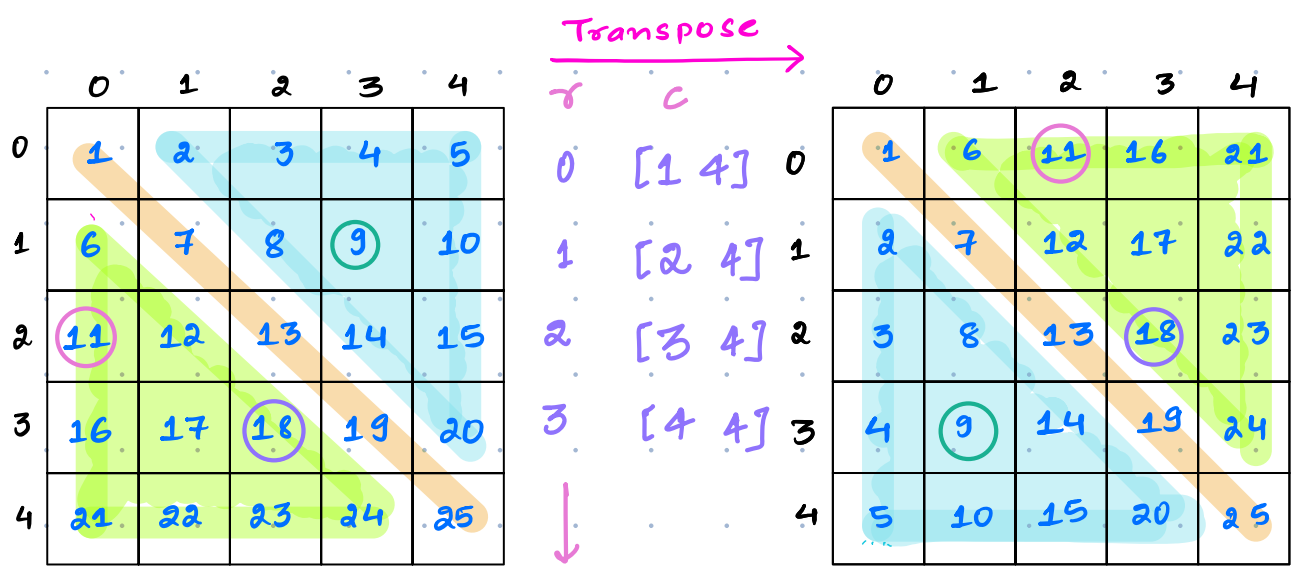
Transpose of a Square Matrix

Given a square 2D matrix mat[N][N], find transpose.

The transpose of a matrix is a new matrix obtained by interchanging the rows and columns of the original matrix.

Do in const space.





Observations:

Principal Diagonal is same in both matrices.

arr[i][j] $\xrightarrow{\text{Transpose}}$ arr[j][i]

Approach: Swap elements lying above the Principal diagonal with elements lying below the Principal diagonal.

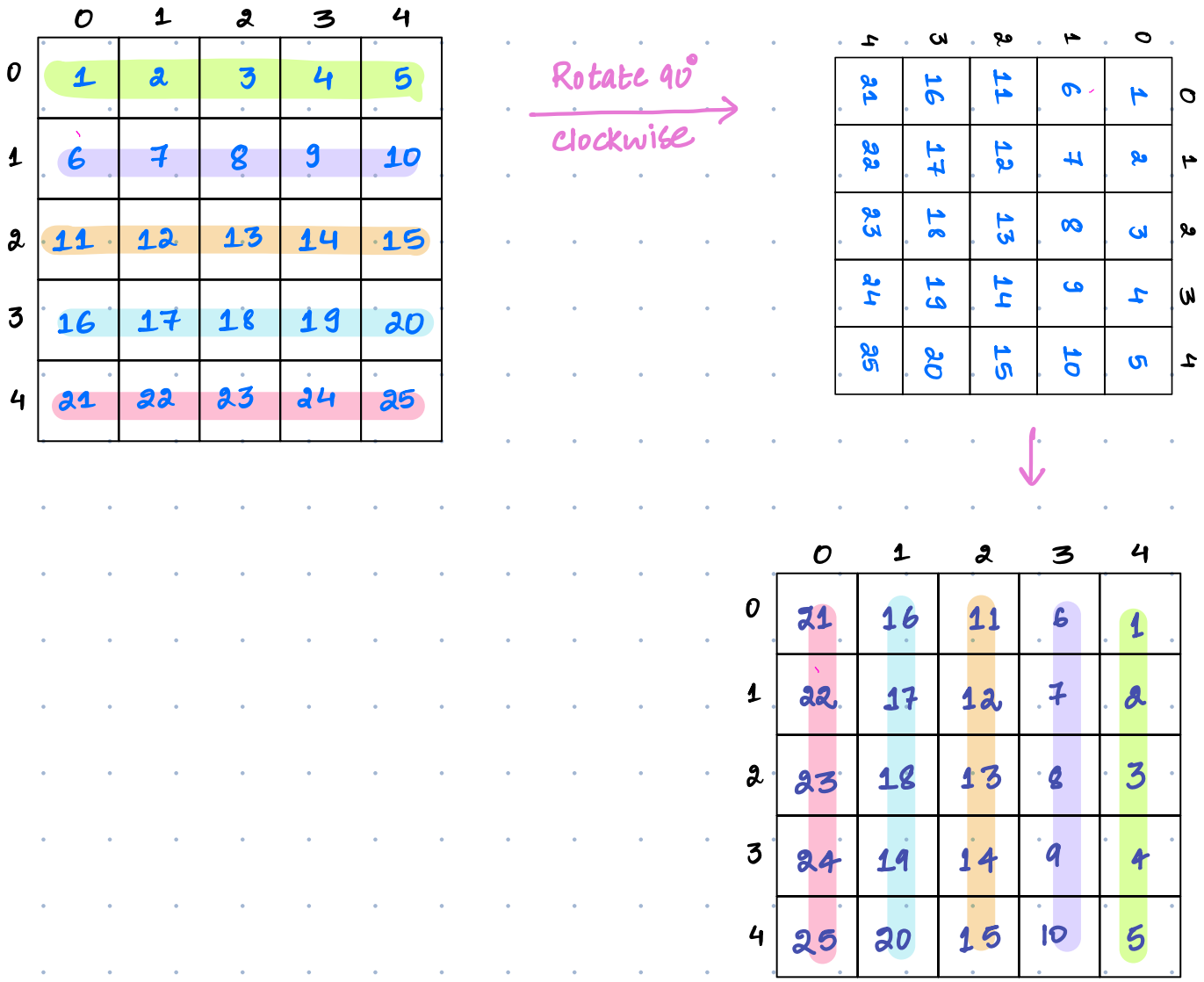
```
void Transpose(mat[][], int N){

    // Iterate on Upper triangle.
    for(r=0; r<=N-2; r++){
        for(c=r+1; c<=(N-1); c++){
            swap(mat[r][c], mat[c][r]);
        }
    }

}
```

T.C : $O(N^2)$
S.C : $O(1)$

mat[3][4] $\xrightarrow{\text{Transpose}}$ mat[4][3]

Rotate a mat[N][N]

Given a matrix mat[N][N], rotate it to 90 degree clockwise.



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

↓ Transpose

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

Reverse Row → Reverse Row → Reverse Row → Reverse Row → Reverse Row

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

```
void Rotate(mat[][], int N) {
    // Find Transpose
    Transpose(mat, N);
    // Reverse all the rows
    for(r = 0; r < N; r++) {
        ReverseRows(mat[r]);
    }
}
```

T.C : $O(N^2)$

S.C : $O(1)$

< *Question* > : Given a binary array [ ]. We can almost replace a single 0 with 1. Find the maximum consecutive 1's we can get in the array[ ] after the replacement.

$1 \le N \le 10^3$

[0 1 1 1 0 1 1 0 1 1 0]

1 1 1 1 0 1 1 0 1 1 0 → 4

0 1 1 1 1 1 1 0 1 1 0 → 6

0 1 1 1 0 1 1 1 1 1 0 → 5

0 1 1 1 0 1 1 0 1 1 1 → 3

ans : 6

QUIZ

[1 1 0 1 1 0 1 1 1]

1 1 1 1 1 0 1 1 1 → 5

1 1 0 1 1 1 1 1 1 → 6

ans : 6

[1 1 1 1 1 1 1]

ans : 7

[0 0 0 0 0]

1

ans : 1

[1 1 1 0 1 1 0 1 1 1 1 0 0 1 1 0 1]

3 2 4 2 1

3+2+1 = 6

4+0+1 = 5

2+4+1 = 7

0+2+1 = 3

2+1+1 = 4

i

ans = 0

Approach: For every 0's in the array :

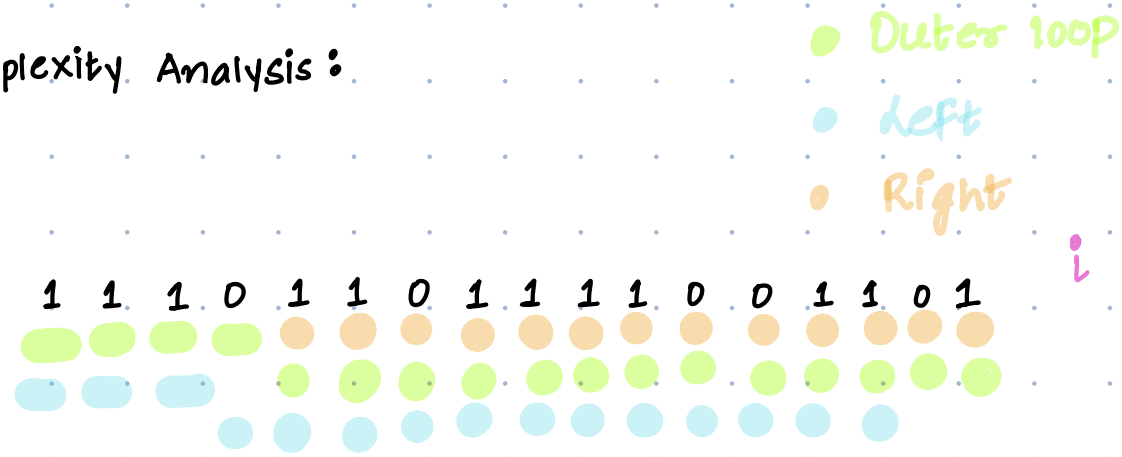
- Count NO of 1's on its Left side : l
- Count NO of 1's on its Right side : r

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

```
int MaxConsecutiveOnes(int arr[]){
    int total_Ones = 0;
    for(i=0; i< arr.length; i++){
        if(arr[i]==1)
            total_Ones++;
    }
    if(total_Ones == arr.length)
        return arr.length;
    int ans = 0;
    for(i=0; i< arr.length; i++){
        if(arr[i]==0){
            // Count NO of 1's on left
            int l=0, j=i-1;
            while(j>=0 && arr[j]==1){
                l++;
                j--;
            }
            // Count NO of 1's on right
            int r = 0, j = i+1;
            while(j<N && arr[j]==1){
                r++;
                j++;
            }
            ans = Math.Max(ans, l+r+1);
        }
    }
    return ans;
}
```

QUIZ

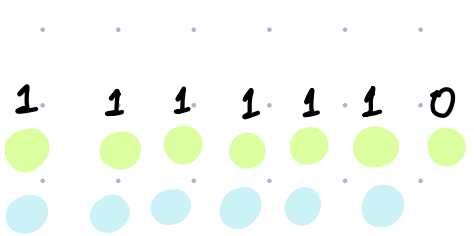
Time Complexity Analysis :



Each element is accessed atMax 3 times.

T.C : O(3N) = O(N)

S.C : O(1)



< Question > : Given a binary array [ ]. We can swap a single 0 with 1. Find the maximum consecutive 1's we can get in the array[ ] after atmost 1 swap.

```
                0  1  2  3  4  5  6  7  8
arr[ ] → [      1  1  0  1  1  0  1  1  1 ]

<2,8>    1  1  1  1  1  0  1  1  0  → 5
                                          ans : 6
<0,5>    0  1  0  1  1  1  1  1  1  → 6
```

```
                0  1  2  3  4  5
arr[ ] → [      1  1  0  1  1  1 ]

<2,5>       1  1  1  1  1  0
                                   ans : 5
```

Approach :

ans = 0

For every 0's in the array :

- Count NO of 1's on its Left side : l
- Count NO of 1's on its Right side : r

```
IF (l + r == total_ones)
    count_of_1's = l + r
else
    count_of_1's = l + r + 1

ans = max(ans, count of 1's)
```

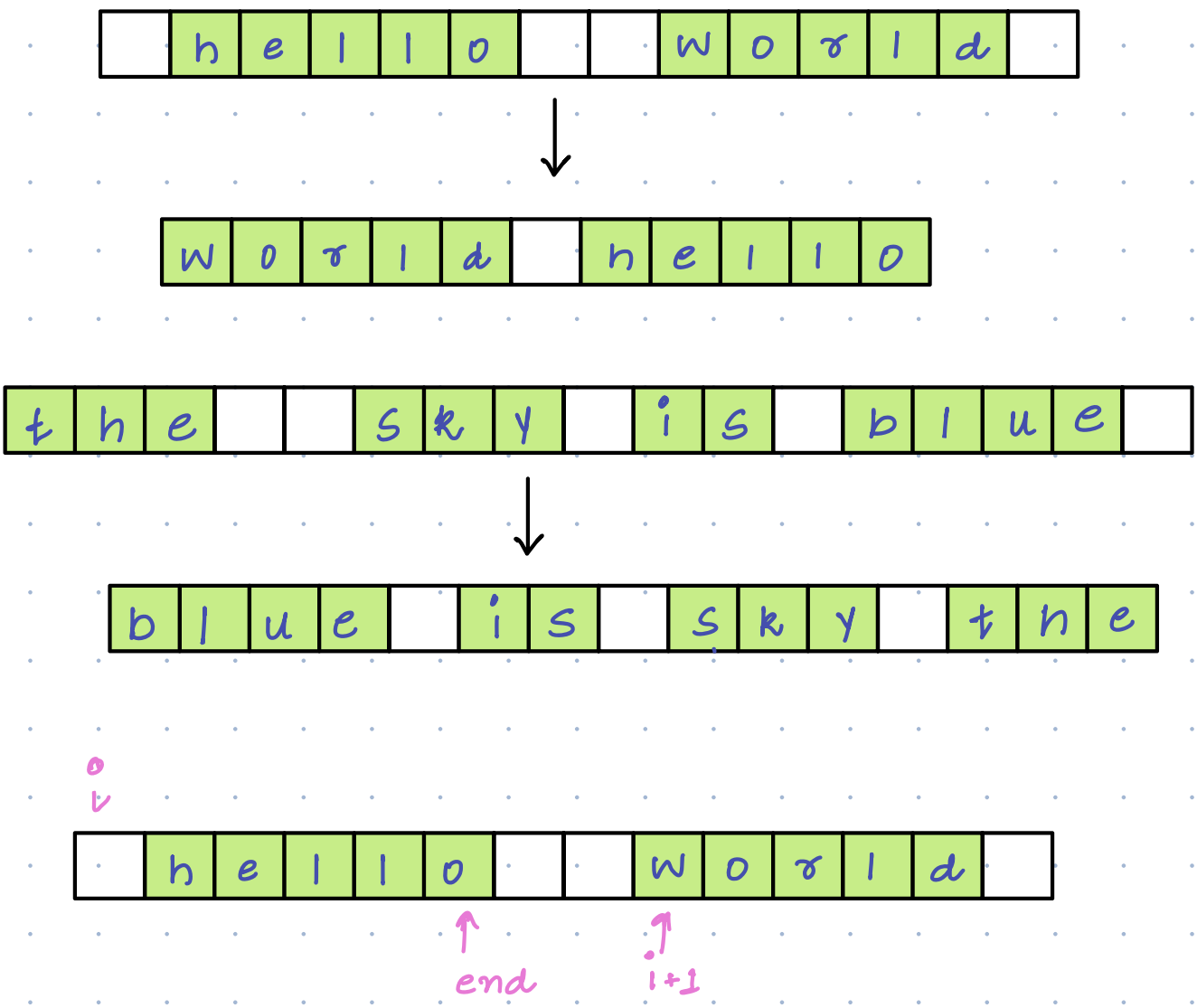
```
int MaxConsecutiveOnes(int arr[]){
    int total_ones = 0;
    for(i=0; i< arr.length; i++){
        if(arr[i]==1)
            total_ones++;
    }
    if(total_ones == arr.length)
        return arr.length;
    int ans = 0;
    for(i=0; i< arr.length; i++){
        if(arr[i]==0){
            // Count No of 1's on left
            int l=0, j=i-1;
            while(j>=0 && arr[j]==1){
                l++;
                j--;
            }
            // Count No of 1's on right
            int r = 0, j=i+1;
            while(j<N && arr[j]==1){
                r++;
                j++;
            }
            int count = 0;
            if(l+r==total_ones)
                count = l+r;
            else
                count = l+r+1;
            ans = Math.Max(ans, count);
        }
    }
}
```

T.C : O(3N) = O(N)

S.C : O(1)

< *Question* > : Given an input string s, reverse the order of the words. A word is defined as a sequence of non-space characters. The words in "s" are separated by at least one space.

Return a string of the words in reverse order, concatenated by a single space. The returned string should only have a single space separating the words, even if there were multiple spaces in the input string. Leading or trailing spaces should be removed.



```
String ReverseWords (String s) {
    string ans = "  ";
    int n = s.size();
    int i = n-1;

    while(i >=0) {
        // skip trailing spaces.
        while(i >=0 && s[i] == ' ')
            i--;

        int end=i;
        while(i >=0 && s[i] != ' ')
            i--;

        if(end >=0) {
            if (! ans.empty())
                ans += ' ';

            ans += s.substr(i+1, end-i );
                                 start  length
        }
    }
    return ans;
}
```

T.C : O(N)
S.C : O(1)