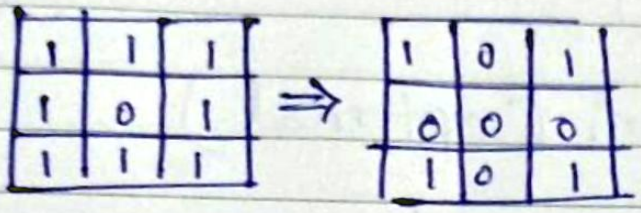
Notes

Date__/__/_____

Day-1

Q-1 Problem Name: Set Matrix Zeros



// if there is 0 in a row or col then make entire row and col to zero.

//

// first approach (Brute force)

⇒ TC: O(N×M) · O(N+M)
SC: O(1)

⇒ // Second approach

- Make two dummy array and traverse in matrix if mat[i][j]==0 then set dummy1[i]=0 and dummy2[j]=0
  ↳ row ↓ for col

// Now traverse again dummy[i][j] == 0 ||

Remarks

dummy2[j] == 0 then
dummy1' mat[i][j] = 0;

// TC: O(N*M + N + M)
SC: O(N)

⇒ // Third approach (optimal)

// Instead of taking dummy array we use first row and first col^m and checking particular row or col^m has 0 or not.

// now traverse the remaining matrix and take variable col0 = 0 value ↳ in reverse dir^n.

```
for (int i = row-1 ; i >= 0 ; i--)
{
    for (int j = col-1 ; j >= 1 ; j--)
    {
        if (mat[i][0] == 0 ||
            mat[0][j] == 0)
        mat[i][j] = 0;
    }
```

Remarks

```
if (col==0)
{
    mat[i][0]=0;
}
```

Problem Name : Pascal's triangle.

// logic : corner value of row start and row end is one and mat[i-1][j-1] + mat[i-1][j] is sum up for next row remaining value.

```
vector<vector<int>> r(numRows)
int n= numRows;
for (int i=0; i<numRows; i++) {
    r[i].resize(i+1);
    r[i][0] = r[i][i] = 1;
    for (int j=1; j<i; j++)
    {
        r[i][j] = r[i-1][j-1] + r[i-1][j];
    }
return r;
```

Remarks

Date__/__/_____ Note

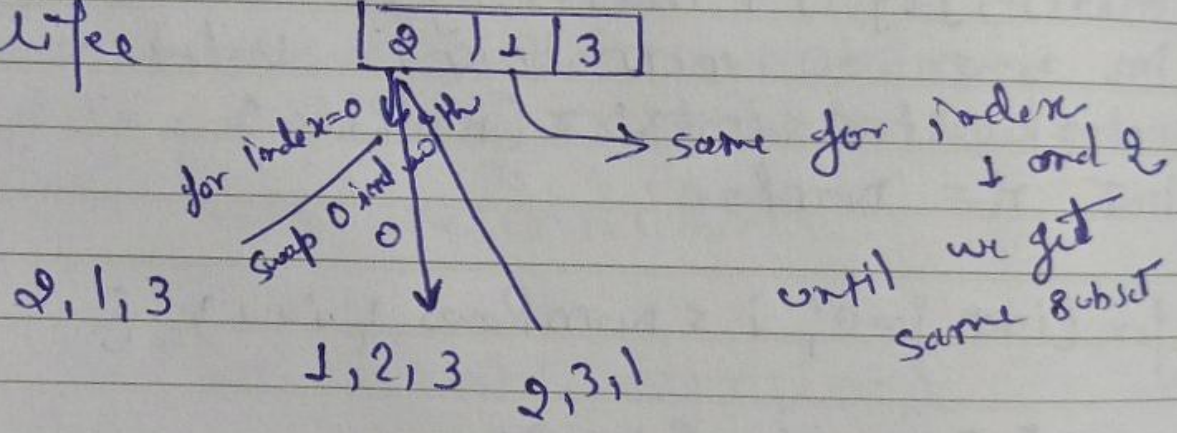
// TC: $O(numRows^2)$
SC: $O(numRows^2)$

// creating 2D- Array.

Problem Name: Next_Permutation

// First approach using recursion
find all subset

like



// TC: $O(N! \times N)$
SC: $O(1)$

Remarks

// Second approach:
// Using inbuilt Stl fn

Notes

Date__/__/_____

next_permutation(nums.begin(), nums.end())

// TC: O(NlogN)

// SC: O(1)

// Third approach (optimal)

// find break point in a given array.

int k = n-2;

// if break point (num[k] < num[k+1])
    break;

// if (k < 0)
    reverse(nums.begin(), nums.end());

// else {

    if (num[i] > num[k])
        break;

Remarks

else
    swap(num[k], num[i])
    reverse(nums.begin() + k + 1, nums.end())

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

Date__/__/_____ Note:

Problem Name: Kadane's Algorithm

// Find contiguous subarray.

// arr[] = {-2, 1, -3, 4, -1, 2, 1, -5, 4}

// output: 6

// logic: create a variable sum = 0;
traverse whole array.

like

```
sum += arr[i];
```

// if (sum < 0)
sum = 0;

{base}

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

Remarks // Brute force using loop (3 pointer)

i = 0 - n
j = 0 - n-1
k = i to j

O(N^3) //