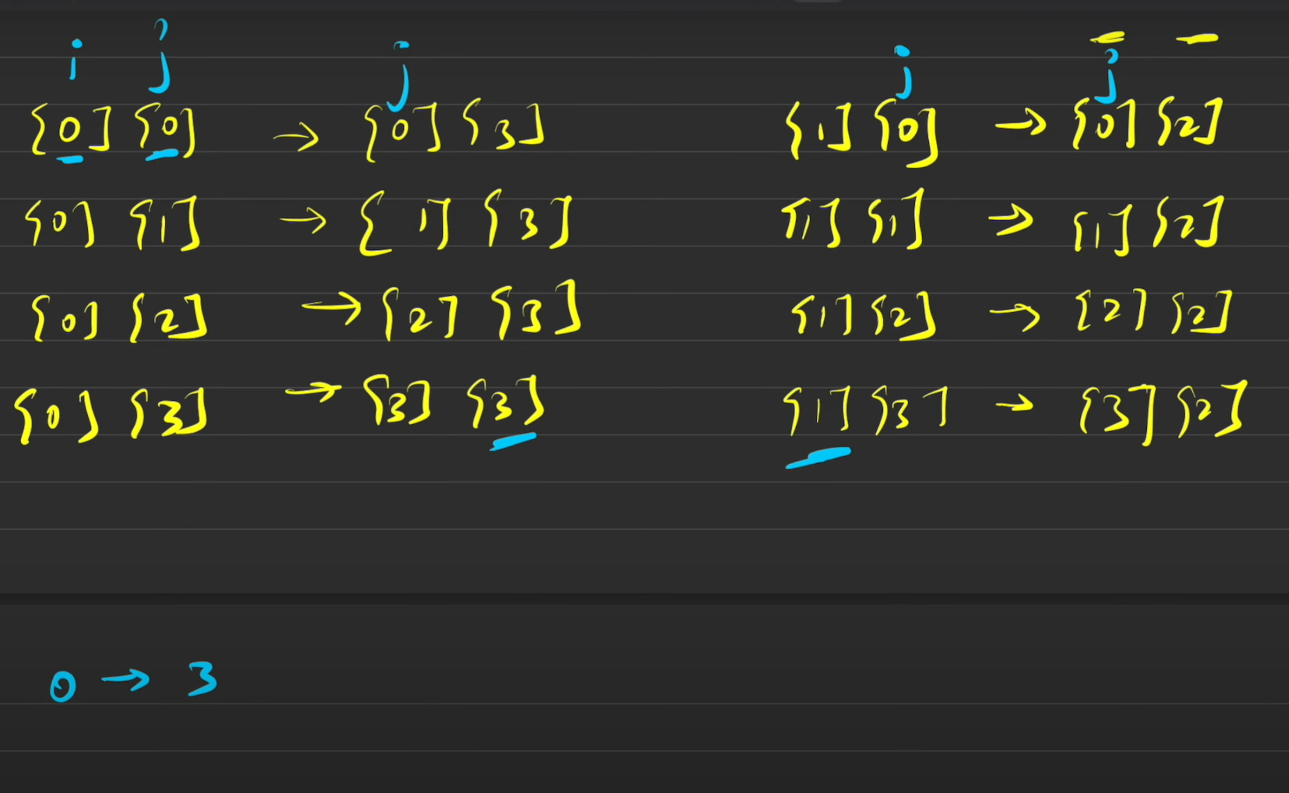
**LC#48:ROTATE IMAGE**

**APP1:BRUTE O(N^2)**

* **From this we infer that from if 1st row element is in last row**
* **Matrix[j][n-1-i] = res[i][j]**

****

class Solution {

    public void rotate(int[][] matrix) {

        int n = matrix.length;

        int res[][] = new int[n][n];

        for(int i = 0 ; i < n ; i++)

               for(int j=0 ; j< n ; j++)

                res[j][n-1-i] = matrix[i][j];

        for(int i = 0 ; i< n; i++)

        {

            for(int j = 0 ;j < n ;j++)

            {

                matrix[i][j] = res[i][j];

            }

        }

    }}

**APPROACH 2:**

class Solution {

    public void rotate(int[][] m) {

        int n = m.length;

        for(int i = 0 ; i<n;i++)

        {

            for(int j = i+1; j <n;j++)**//TRANSPOSE**

            {**SWAPPING BASED ON XOR**

                m[i][j] = m[j][i]^m[i][j];

                m[j][i] = m[j][i]^m[i][j];

                m[i][j] = m[j][i]^m[i][j];

            }

        }

        for(int i= 0 ;i<n;i++)

        {   int left = 0 , right = m[i].length-1;

            while(left<right)**//BINARY SEARCH SWAP**

            {**//perform normal swap xor for uniqueness**

                m[i][left] ^=m[i][right];

                m[i][right] ^=m[i][left];

                m[i][left] ^=m[i][right];

                left++;

                right--;

            }

}

    }

}

**LC#189 : ROTATE ARRAY BY k places**

**Example 1:**

**Input: nums = [1,2,3,4,5,6,7], k = 3**

**Output: [5,6,7,1,2,3,4]**

**Explanation:**

**rotate 1 steps to the right: [7,1,2,3,4,5,6]**

**rotate 2 steps to the right: [6,7,1,2,3,4,5]**

**rotate 3 steps to the right: [5,6,7,1,2,3,4]**

**APP1 : USING THREE LOOPS SEP AND AN ARRAY**

class Solution {

    public void rotate(int[] nums, int k) {

        int n = nums.length;

        int[] a = new int[n];

        k = k % n;

        int j = 0;

        for (int i = n - k; i < n; i++)

            a[j++] = nums[i];//after k elements

        for (int i = 0; i < n - k; i++)

            a[j++] = nums[i];//stores the remaining from start

        for (int i = 0; i < n; i++)

            nums[i] = a[i];//copying back

    }

}

**APP2:USING MODULUS AND AN EXTRA ARRAY**

class Solution {

public void rotate(int[] nums, int k) {

int n = nums.length;

k = k % n;

int[] temp = new int[n];

for (int i = 0; i < n; i++) {

temp[(i + k) % n] = nums[i];//rgets the element correctly (i+k)%n

}

for (int i = 0; i < n; i++) {

nums[i] = temp[i];

}

}

}

**APP3: OPTIMAL**

* **Reverse the array**
* **Reverse first k in the reversed array**
* **Reverse the remaning elements from k to n**

class Solution {

    public void rotate(int[] nums, int k) {

        k = k%nums.length;

        int n = nums.length;

        reverse(0,n-1,nums);

        reverse(0,k-1,nums);

        reverse(k,n-1,nums);

    }

    void reverse(int left , int right , int[] nums)

    {

        while(left<right)

        {

            nums[left] = nums[left]^nums[right];

            nums[right] = nums[left]^ nums[right];

            nums[left] = nums[left] ^ nums[right];

            left++;

            right--;

        }

    }

}