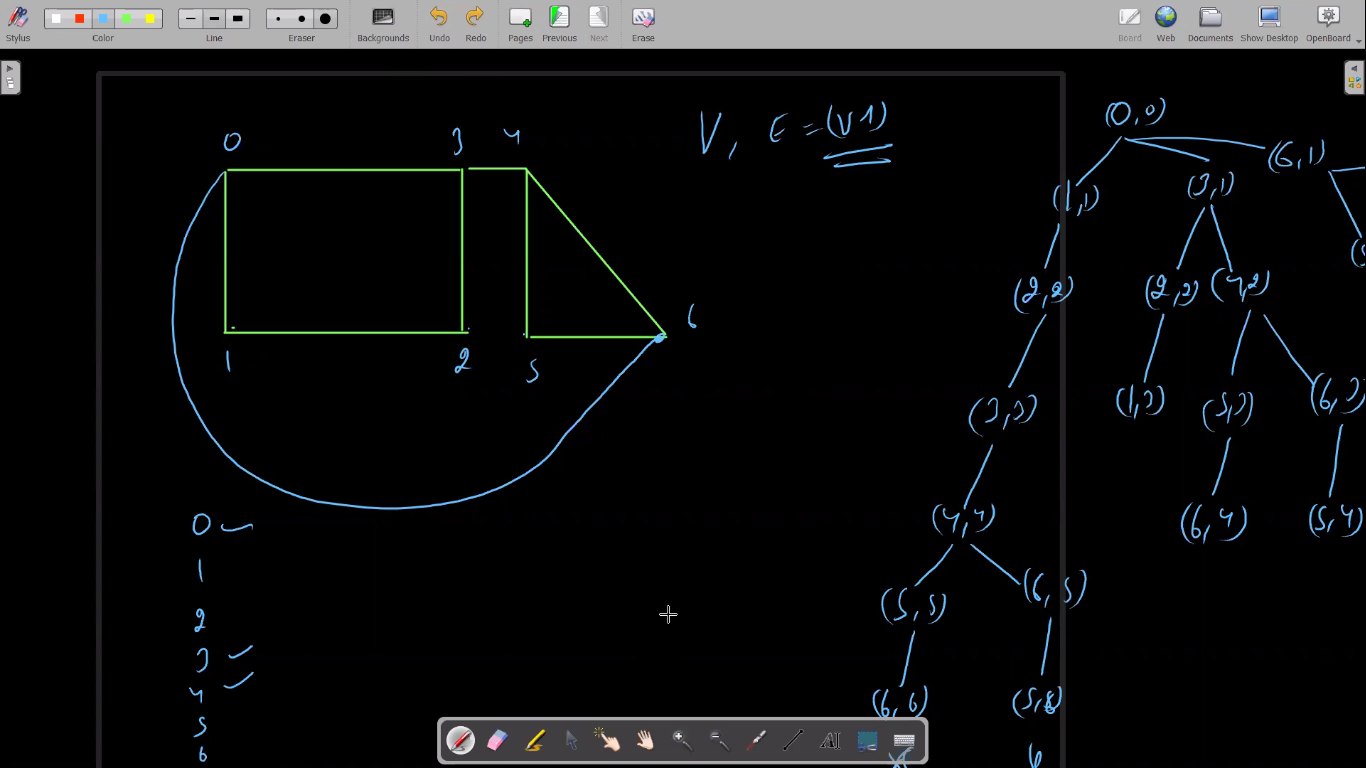
**Class 2 Graph**

1. Hamintonial Path and Cycle

**Hamiltionion Path and cycle in Graph**



 public static int searchVtx(int u,int v){

        for(int i=0;i<graph[u].size();i++){

            Edge e = graph[u].get(i);

            if(e.v == v) return i;

        }

        return -1;

    }

 public static int hamintonianPath(int src,int osrc,boolean[] vis,int edgeCount,String psf){

        if(edgeCount == N - 1){

            psf += src;

            int idx = searchVtx(src,osrc);

            if(idx != -1) System.out.println("Cycle : " + psf);

            else  System.out.println("Path : " + psf);

            return 1;

        }

        vis[src] = true;

        int count = 0;

        for(Edge e: graph[src]){

            if(!vis[e.v])

              count += hamintonianPath(e.v,osrc,vis,edgeCount + 1,psf + src + " ");

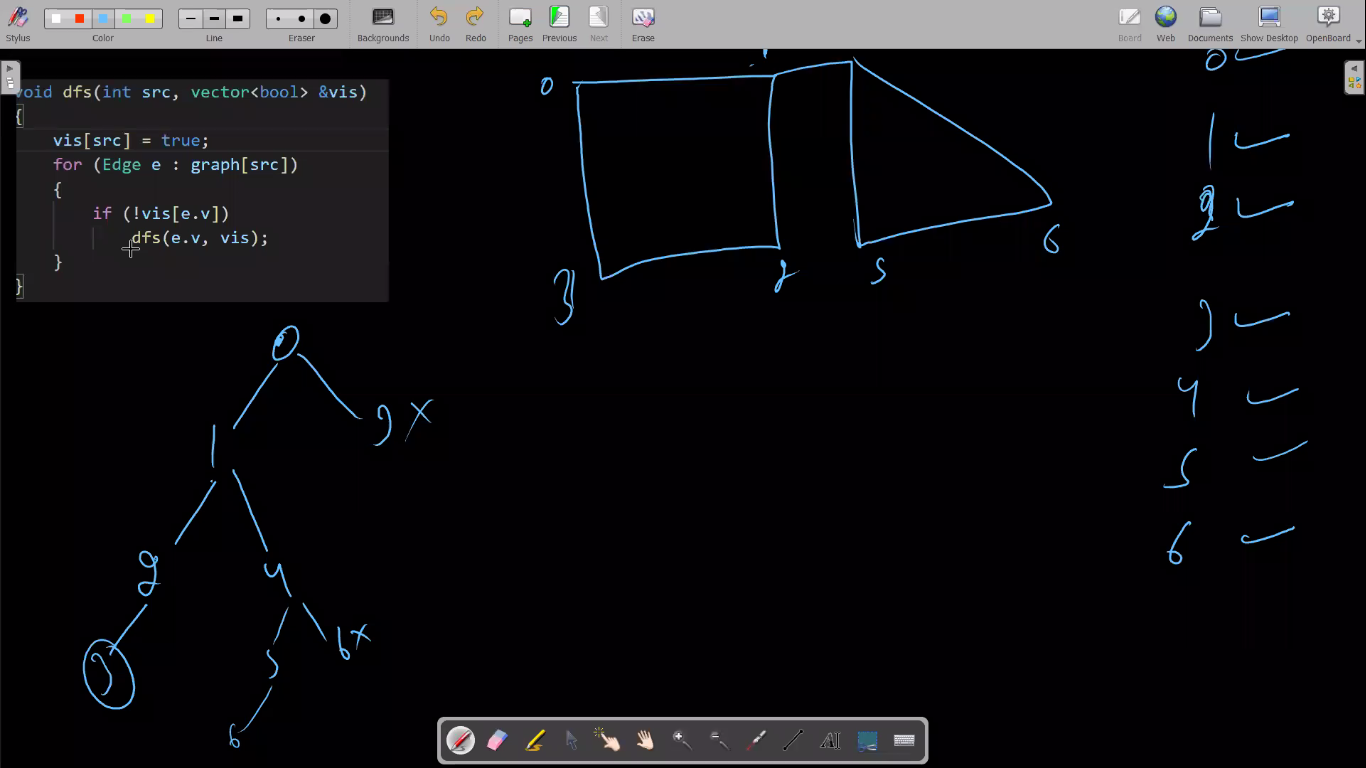
        }

        vis[src] = false;

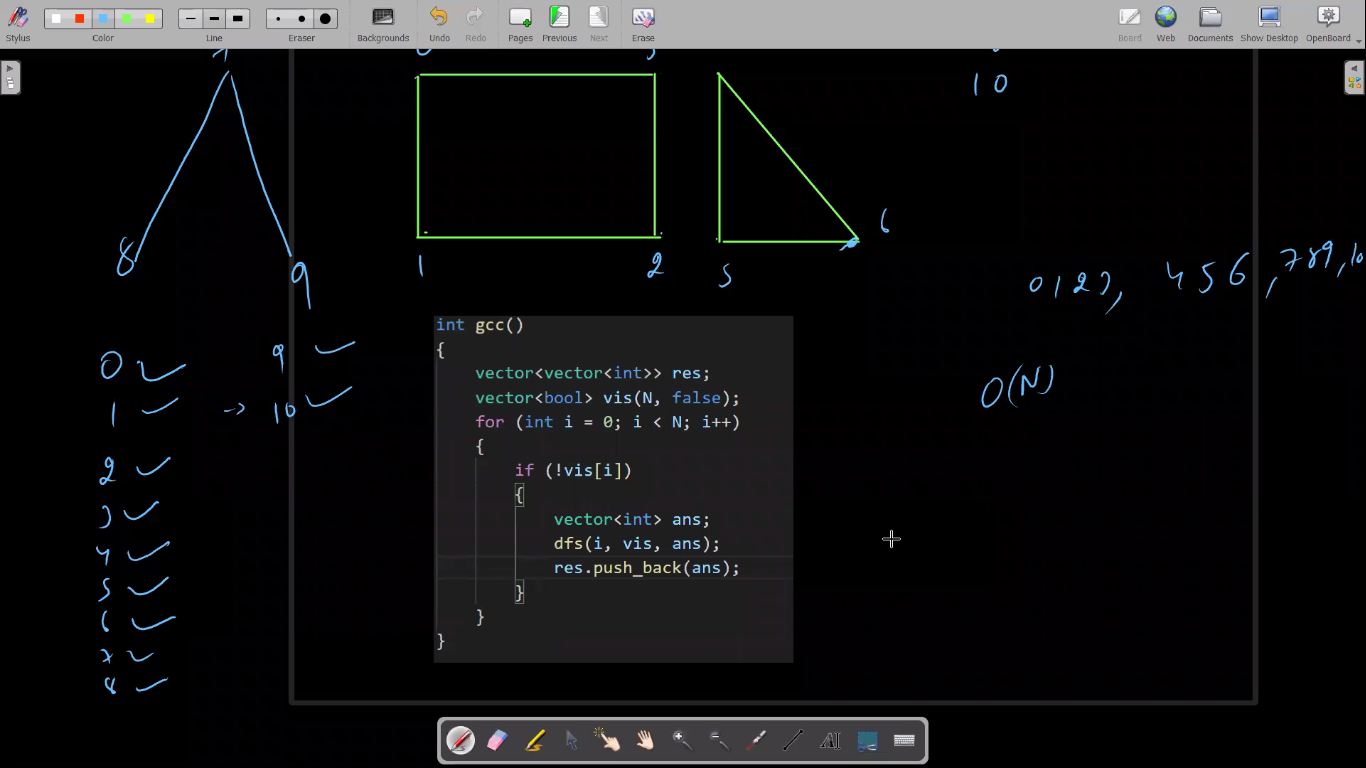
        return count;

    }

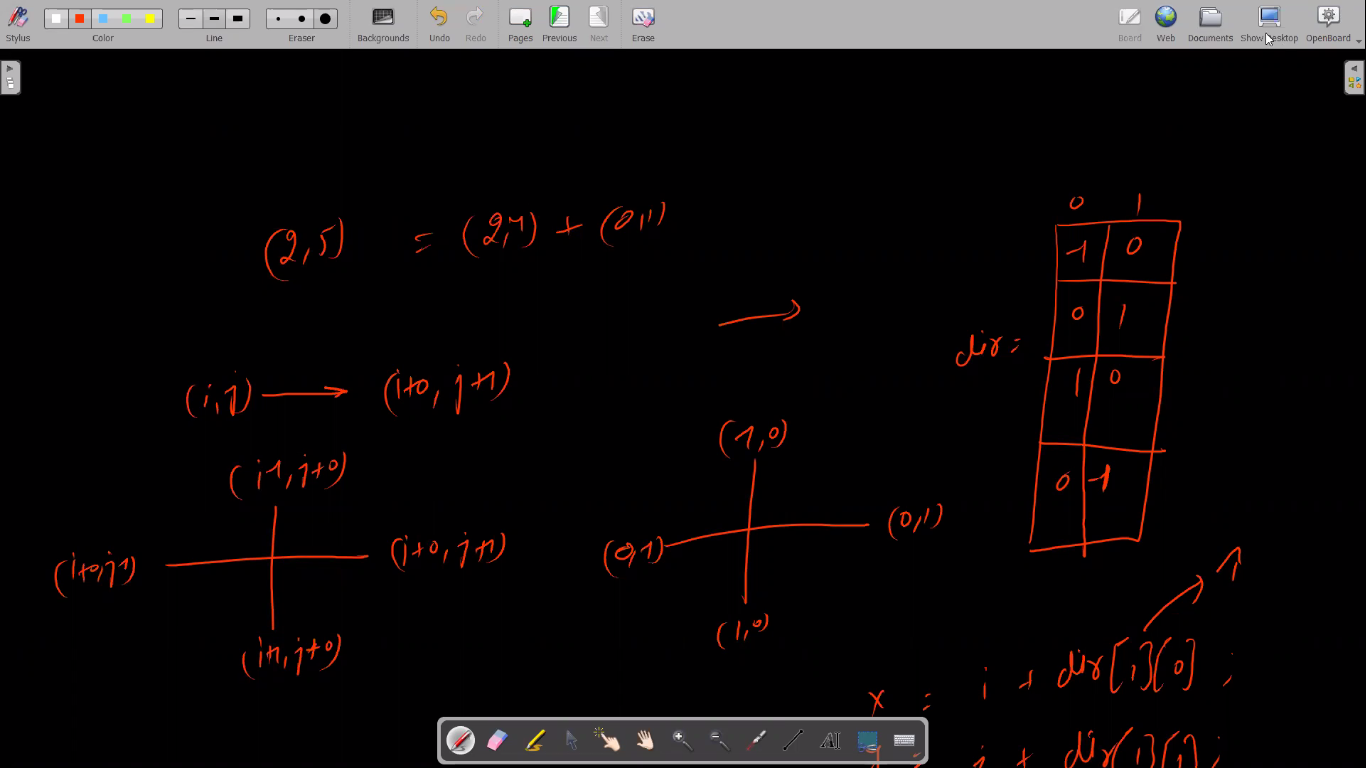
**DFS in a Graph**

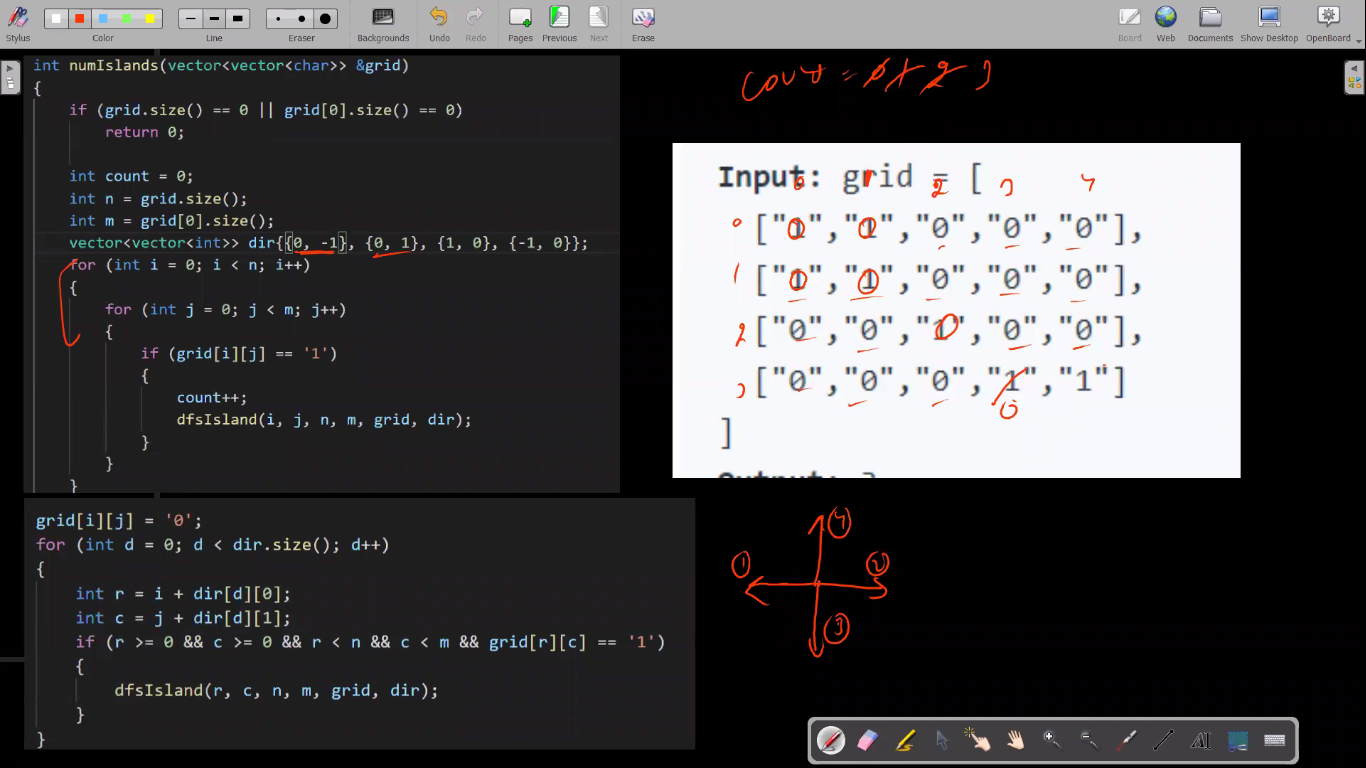


**Get connected components in Graph using DFS**

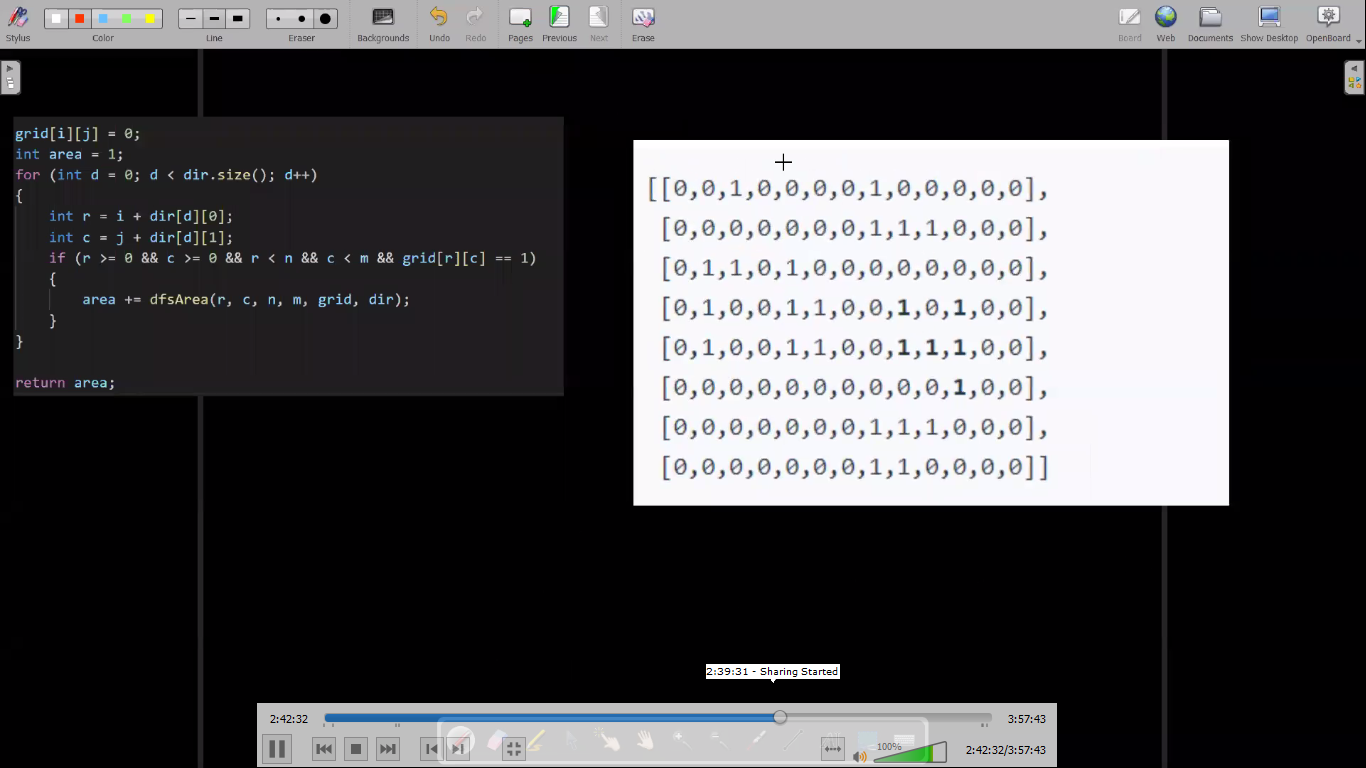


**LeetCode 200 (find number of Islands)**





**LeetCode 695 (Max area of Island)**



   //Leetcode 695

    public int maxAreaOfIslandDFS(int[][] grid,int r,int c,int[][] dir){

        grid[r][c] = 0;

        int count = 0;

        for(int d = 0; d < dir.length ; d++){

            int x = r + dir[d][0];

            int y = c + dir[d][1];

            if( x >= 0 && y >= 0 && x < grid.length && y < grid[0].length && grid[x][y] == 1){

                count += maxAreaOfIslandDFS(grid,x,y,dir);

            }

        }

        return count + 1;

    }

    public int maxAreaOfIsland(int[][] grid) {

        int[][] dir = {{0,1},{0,-1},{1,0},{-1,0}};

        int maxArea = 0;

        for(int i = 0; i < grid.length;i++){

            for(int j = 0; j < grid[0].length; j++){

                if(grid[i][j] == 1){

                    maxArea = Math.max(maxArea,maxAreaOfIslandDFS(grid,i,j,dir));

                }

            }

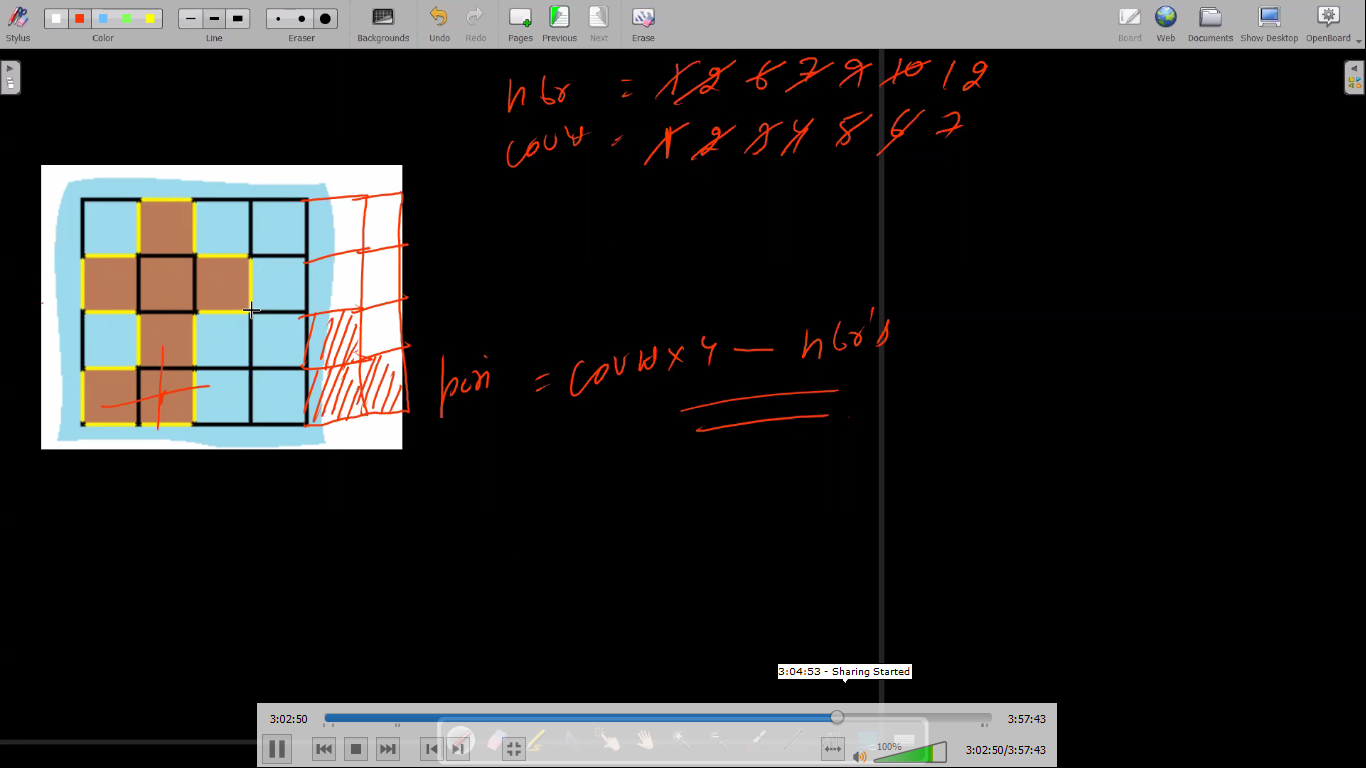
        }

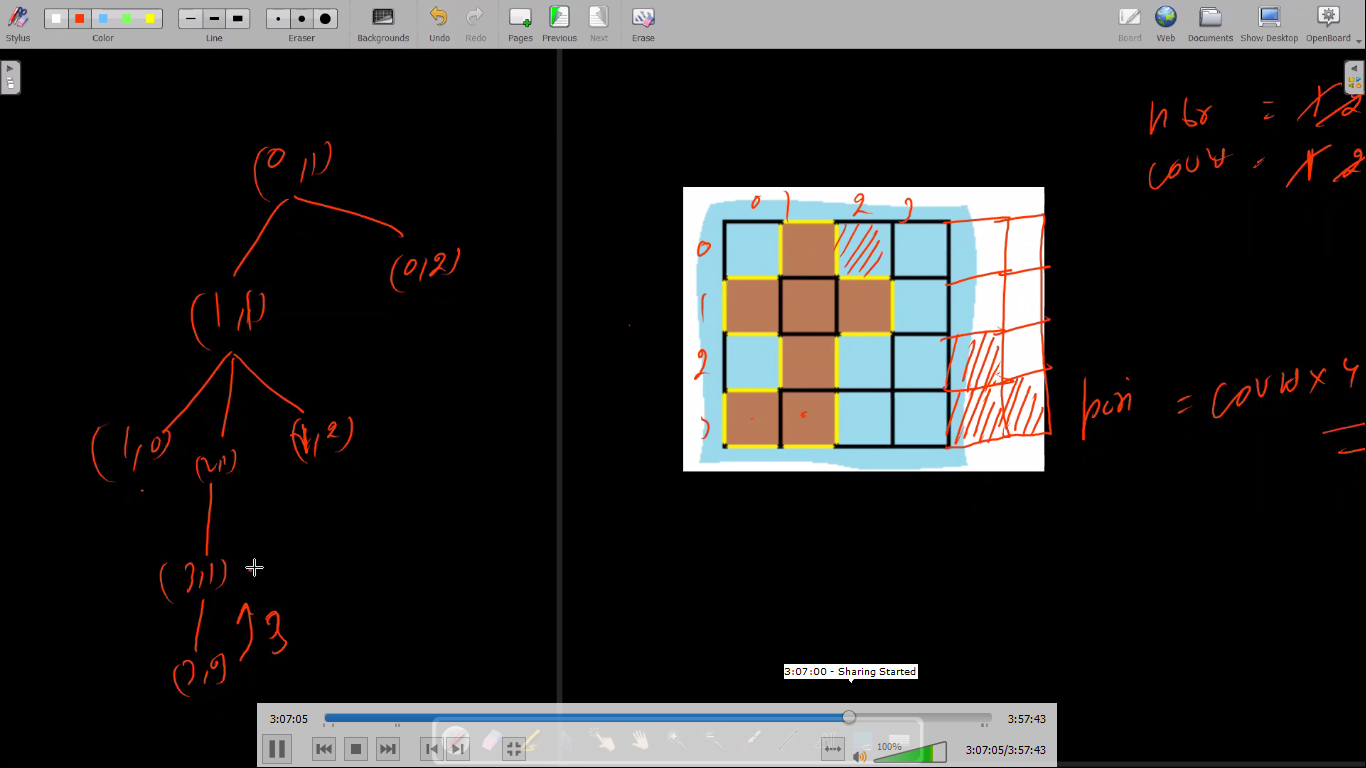
        return maxArea;

    }

**Complexity – 2(n\*m)**

**Leetcode 463 IsLand Perimeter**





    //Leetcode 463

    public int islandPerimeter(int[][] grid) {

        int[][] dir = {{0,1},{0,-1},{1,0},{-1,0}};

        int countOnes = 0;

        int CountNbrs =0;

        for(int i = 0; i < grid.length;i++){

            for(int j = 0; j < grid[0].length; j++){

                if(grid[i][j] == 1){

                    countOnes++;

                    for(int d = 0 ;d<4;d++){

                        int r = i + dir[d][0];

                        int c = j + dir[d][1];

                        if( r>=0 && c>=0 && r<grid.length && c< grid[0].length && grid[r][c] == 1)

                           CountNbrs++;`

                    }

                }

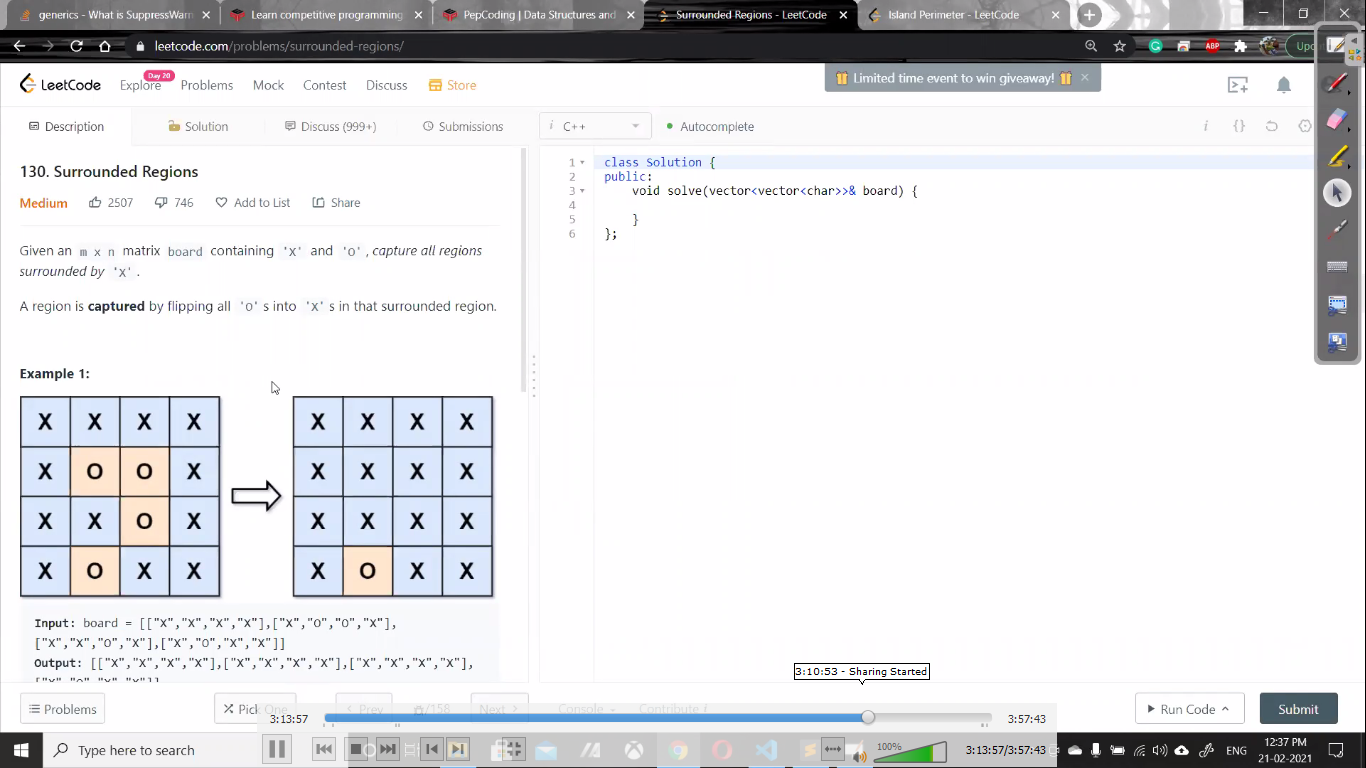
            }

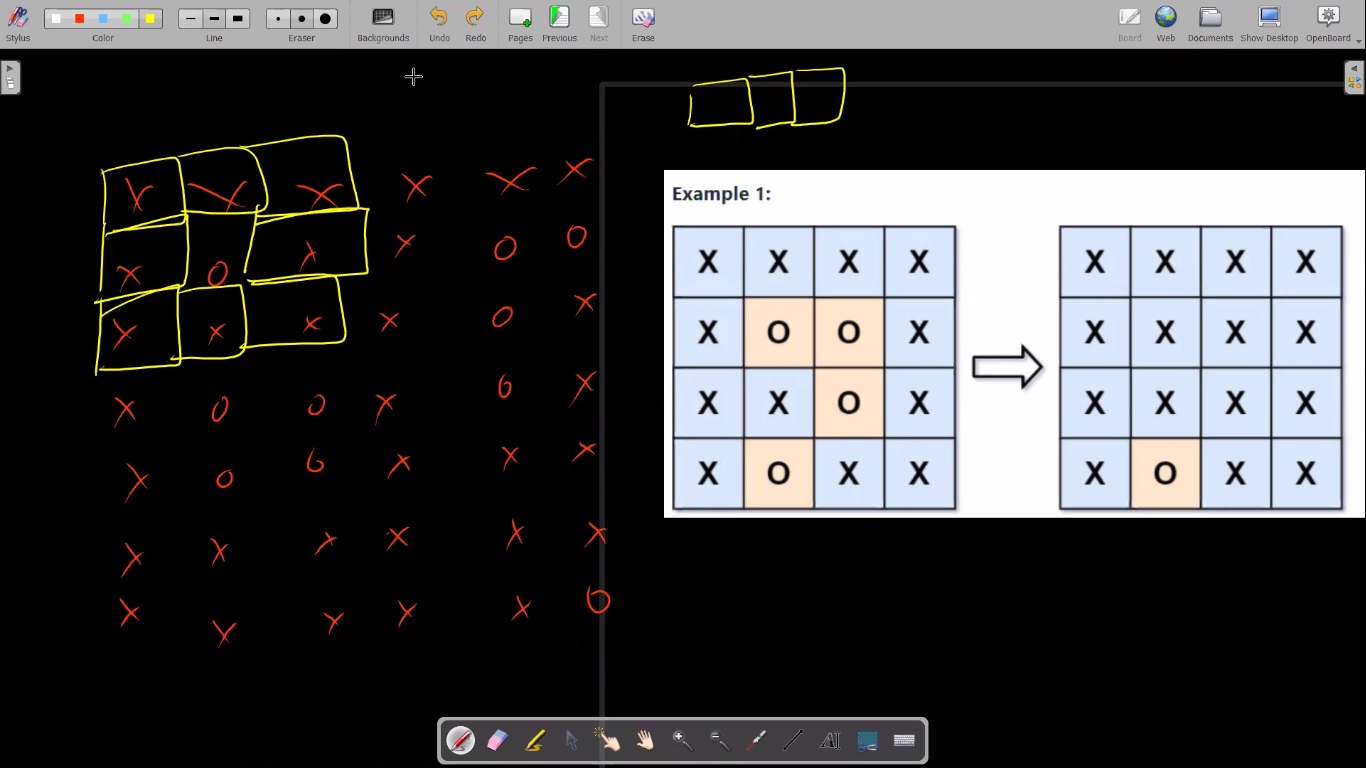
        }

        return 4 \* countOnes - CountNbrs;

    }

**LeetCode 130 Surrounded Regions**





    //Leetcode 130

    public void SurroundedRegionDFS(char[][] grid,int r,int c,int[][] dir){

        grid[r][c] = '$';

        for(int d = 0 ;d < 4 ; d++){

            int x = r + dir[d][0];

            int y = c + dir[d][1];

            if( x>=0 && y>=0 && x < grid.length && y < grid[0].length && grid[x][y] == 'O')

               SurroundedRegionDFS(grid,x,y,dir);

        }

    }

    public void solve(char[][] grid) {

        int[][] dir = {{0,1},{0,-1},{1,0},{-1,0}};

        for(int i = 0; i < grid.length;i++){

            for(int j = 0; j < grid[0].length; j++){

                if(i ==0 || j == 0 || i == grid.length-1 || j == grid[0].length - 1){

                    if(grid[i][j] == 'O')

                       SurroundedRegionDFS(grid,i,j,dir);

                }

            }

        }

        for(int i = 0; i < grid.length;i++){

            for(int j = 0; j < grid[0].length; j++){

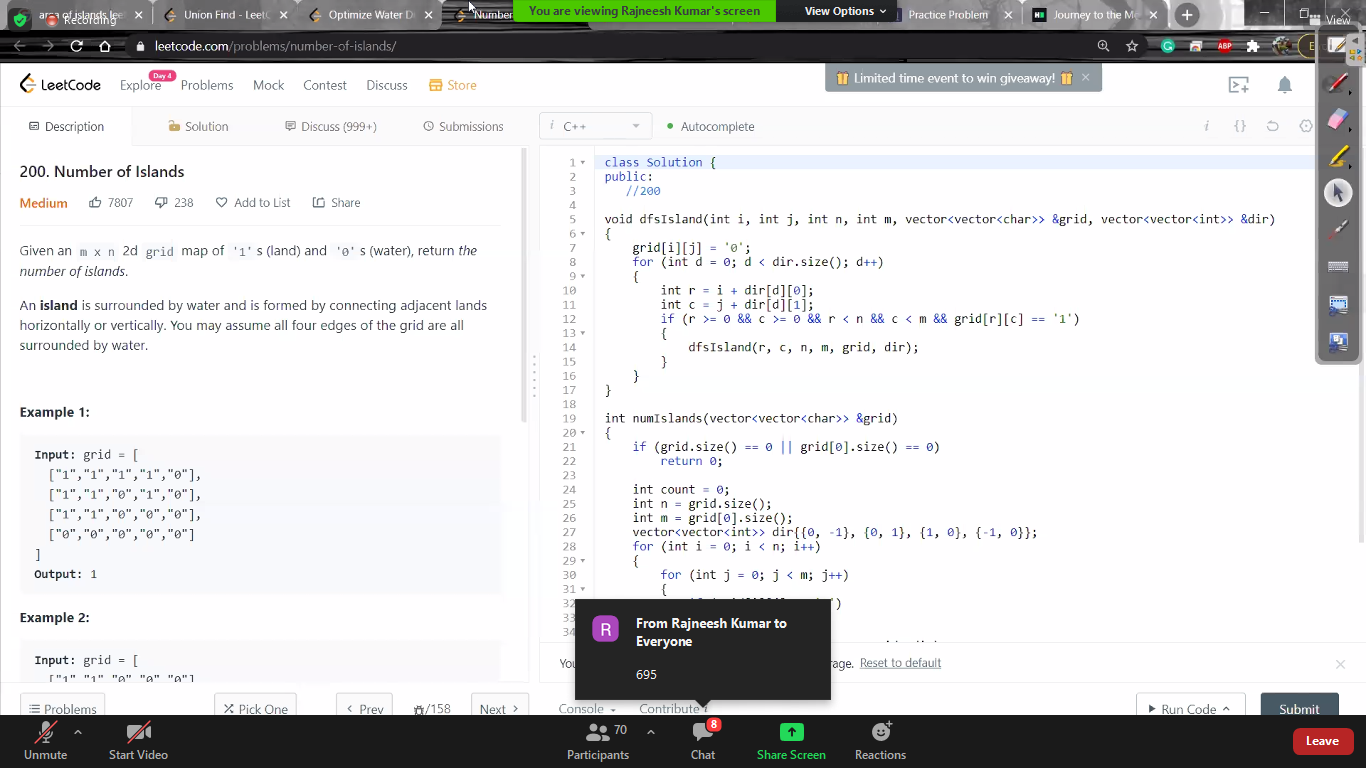
                if(grid[i][j] == 'O') grid[i][j] = 'X';

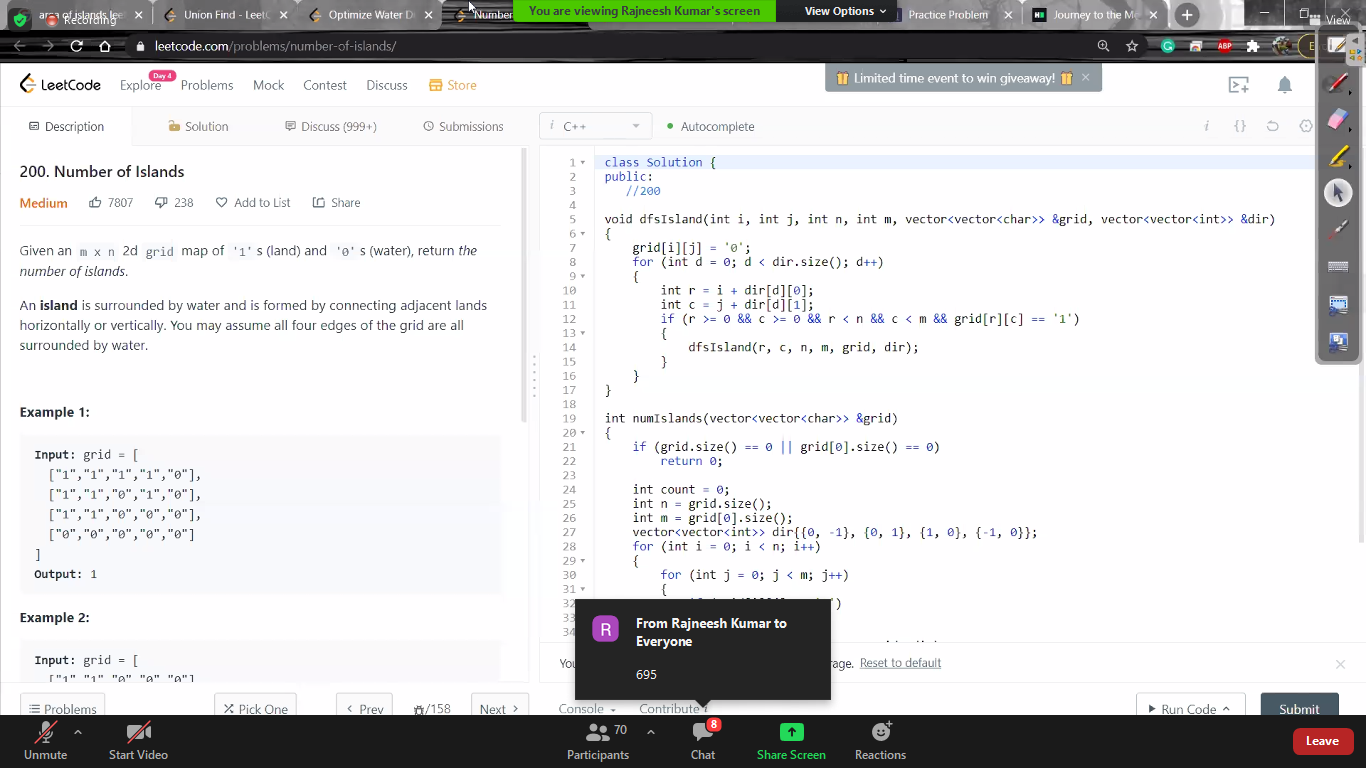
                else if(grid[i][j] == '$') grid[i][j] = 'O';

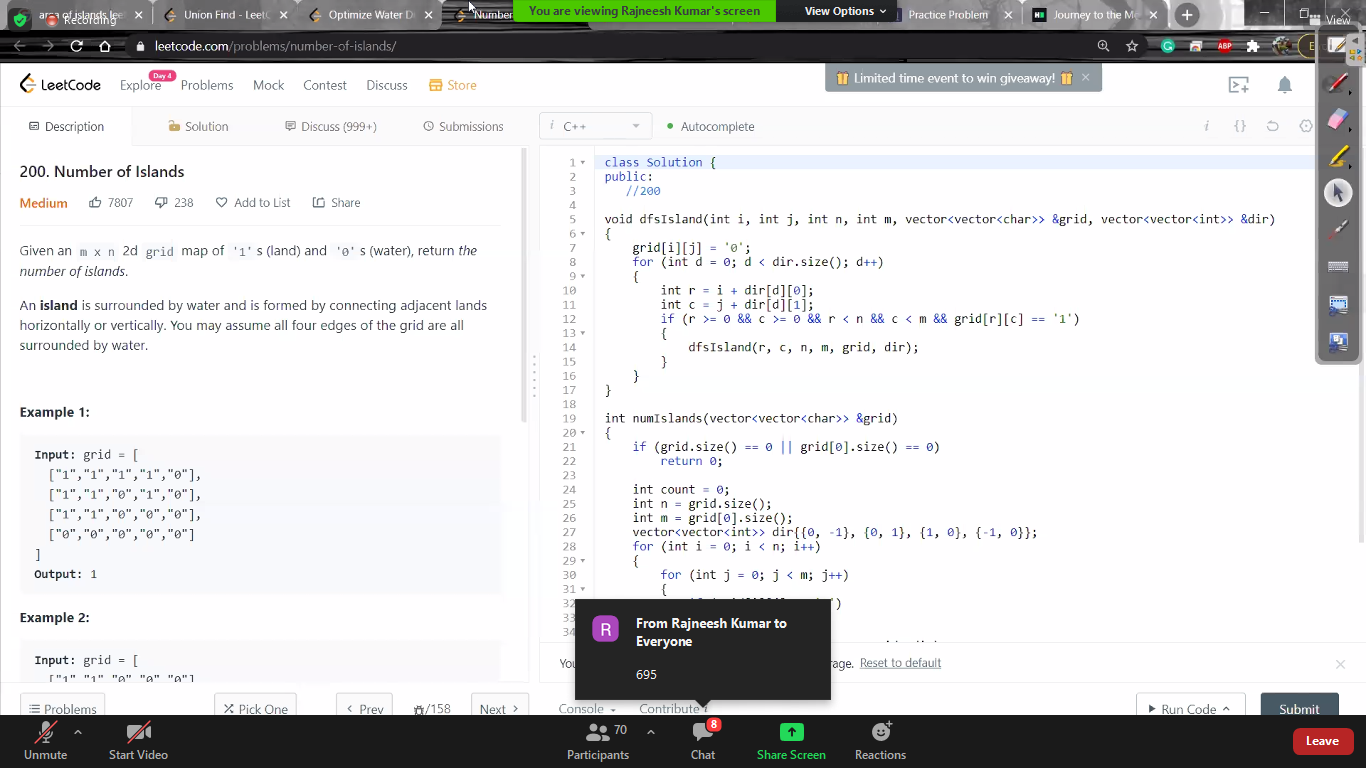
            }

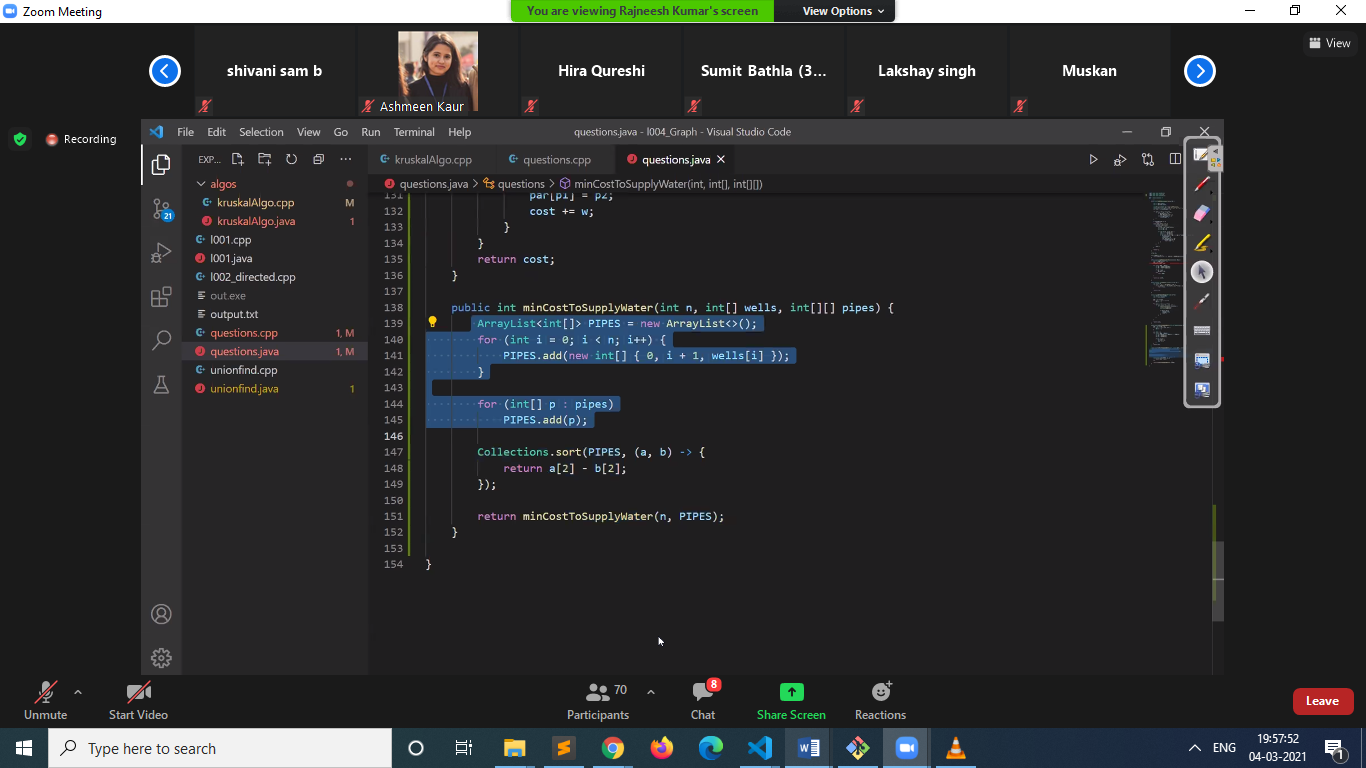
        }

    }









**Solution of Leetcode 200 (find the number of Islands using union find method)**

