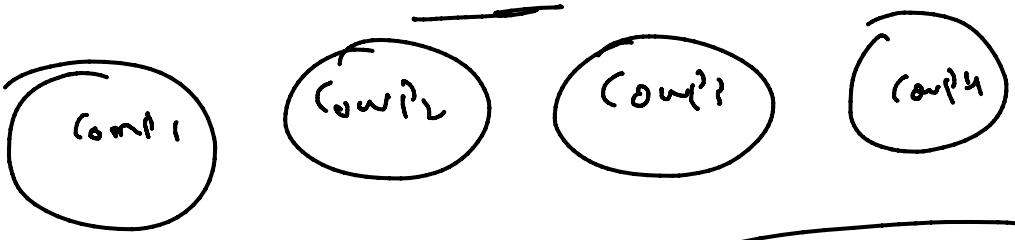
Class 95

27 May 2023 11:08

Class 95

Graphs

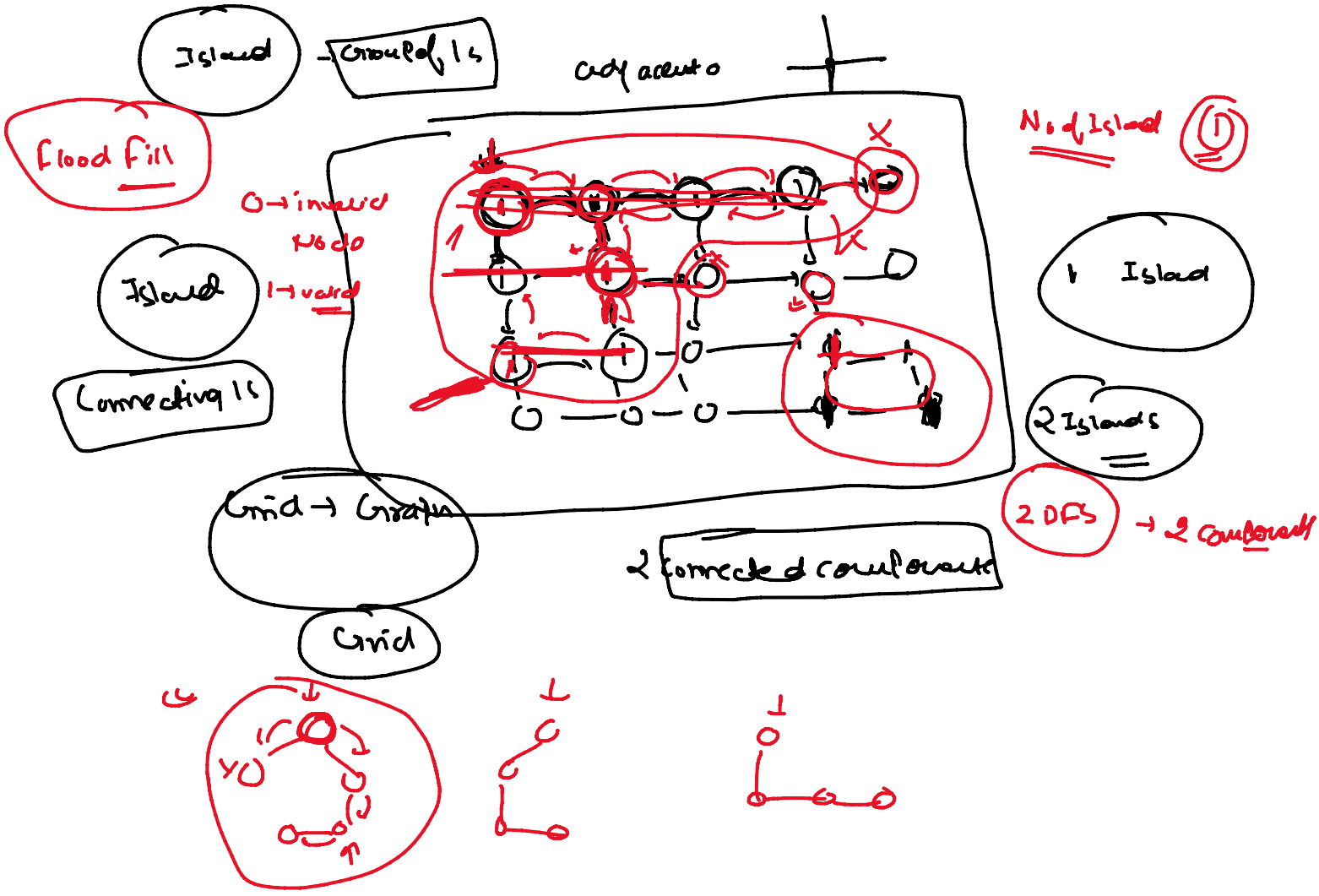


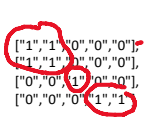
1 → land

0 → water

```
["1","1","1","1","0"],
["1","1","0","1","0"],
["1","1","0","0","0"],
["0","0","0","0","0"]
```

From <https://leetcode.com/problems/number-of-islands/>





3 connected components

From <https://leetcode.com/problems/number-of-islands/>

No of Islands = No of Connected components

```
int numIslands(
{
    int ans = 0;
    For(i=0; i<n; i++)
    {
        For(j=0; j<m; j++)
        {
            if (grid[i][j] == '1')
            {
                bfs(grid, n, m, i, j);
                ans++;
            }
        }
    }
    return ans;
int bfs(vector<vector<char>>& grid, int n, int m, int i, int j)
{
    queue<pair<int,int>> q;
    q.push({i, j});
    grid[i][j] = '0';
    int dx[4] = {-1, 1, 0, 0};
    int dy[4] = {0, 0, -1, 1};
    while(!q.empty())
    {
```

Debugging

Node → 0

'0'

int i, int j

```
while ( q.empty() )
{
    auto node = q.front();
    q.pop();

    int x = node.first;
    int y = node.second;

    for ( i=0; i<4; i++ )
    {
        int nx = x + dx[i];
        int ny = y + dy[i];

        if ( nx >= 0 and nx < n and ny >= 0 and ny < m and
             grid[nx][ny] == '1' )
        {
            q.push( {nx, ny} );
            grid[nx][ny] = '0';
        }
    }
}
return ;
```

```
void dfs ( vector<vector<char>> &grid, int i, int j, int n, int m,
           int &area )
{
    grid[i][j] = '0';
    area++;

    for ( i=0; i<4; i++ )
    {
        int nx = i + dx[i];
        int ny = j + dy[i];

        if ( nx >= 0 and nx < n and ny >= 0 and ny < m and
             grid[nx][ny] == '1' )
```

dfs (grid, nx, ny, n, m);

No of components

Size of each component.

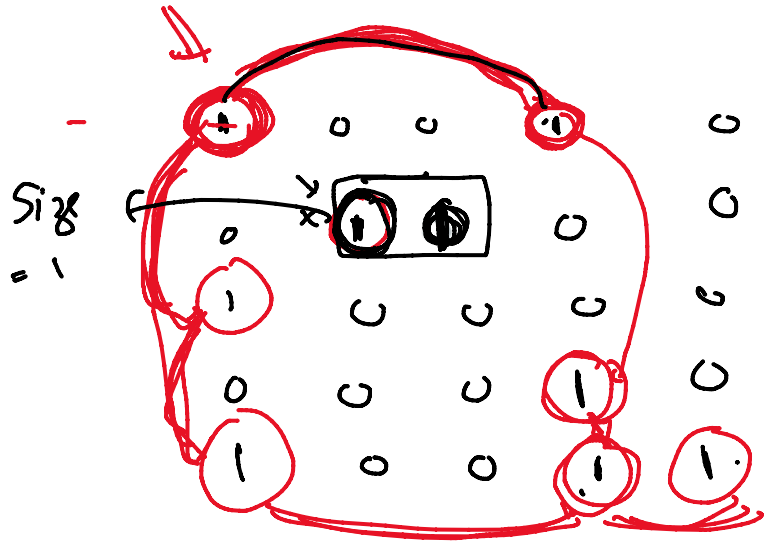
```
int bfs(int i,int j,vector<vector<int>>&grid)
    {
        queue<pair<int,int>>q;
        grid[i][j]=0;
        int ans=0;
        int n,m;
        n=grid.size();
        m=grid[0].size();
        q.push({i,j});
        int dx[4]={-1,1,0,0};
        int dy[4]={0,0,-1,1};
        while(!q.empty())
        {
            auto tp=q.front();
            q.pop();
            int x=tp.first;
            int y=tp.second;
            ans++;
            for(int l=0;l<4;l++)
            {
                int nx=x+dx[l];
                int ny=y+dy[l];
                if(nx>=0 and nx<n and ny>=0 and ny<m and grid[nx][ny]!=0)
                {
                    q.push({nx,ny});
                    grid[nx][ny]=0;
                }
            }

        }
        return ans;
    }
    int maxAreaOfIsland(vector<vector<int>>& grid) {
        int i,j,k,n,m;
        n=grid.size();
        m=grid[0].size();
        int ans=0;
        for(i=0;i<n;i++)
        {
            for(j=0;j<m;j++)
            {
                if(grid[i][j]==1)
                {
                  ans=max(ans,bfs(i,j,grid));
                }
            }
        }
        return ans;
```

count total No of Nodes

```
            ans=max(ans,bfs(i,j,grid));
        }
    }
}
return ans;
}
```

Multisource BFS

Count islands that connected



Size = 1

2 BFS call

Single component

3 + 2 = 9

int countIslands ( )

{

int ans = 0;

For (i = 0; i < n; i++)

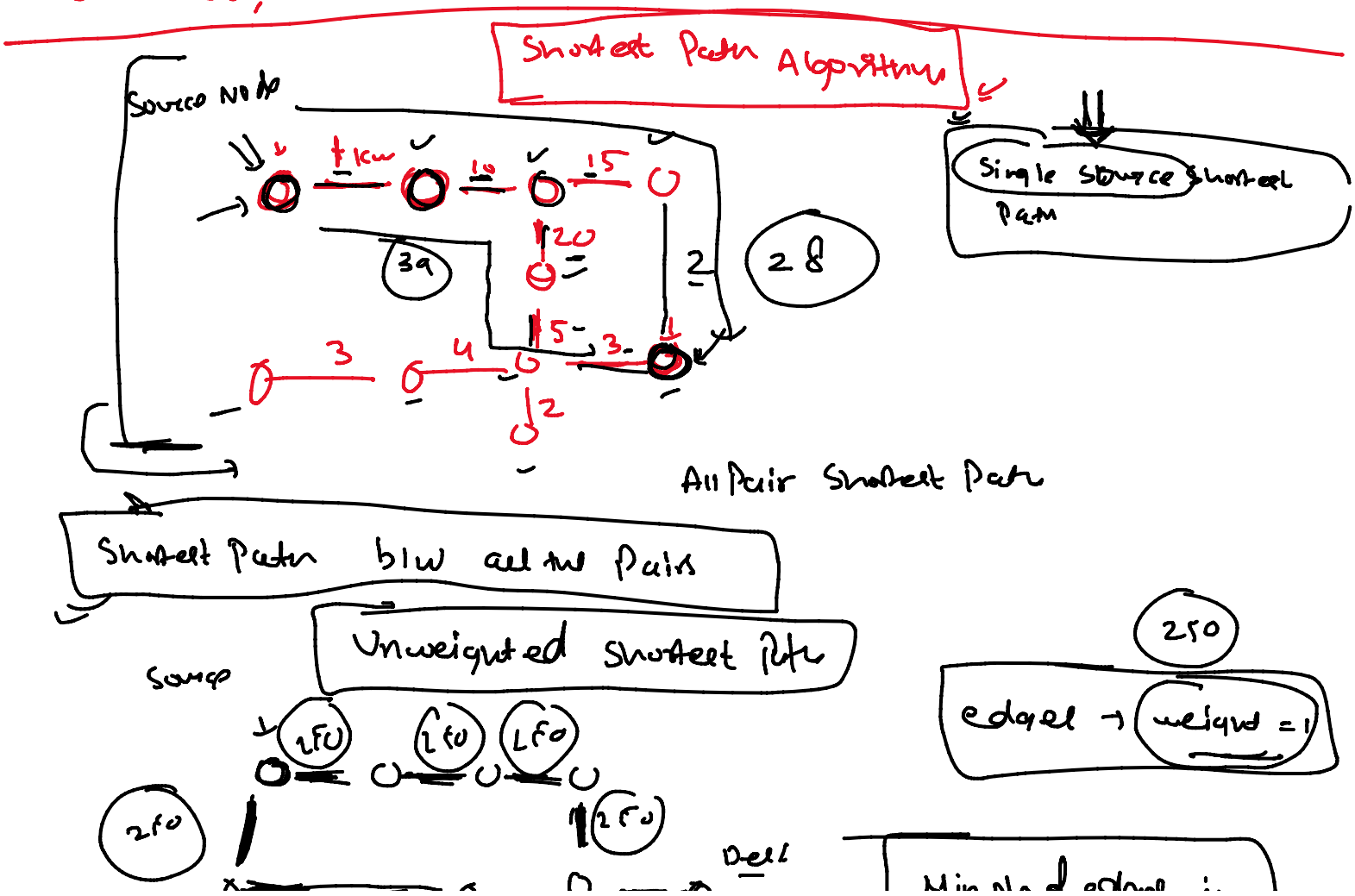
{ For (j = 0; j < m; j++)

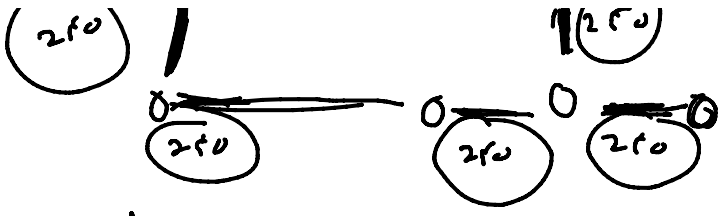
{ if (grid[i][j] == 1)

{ int temp = bfs (grid, i, j);

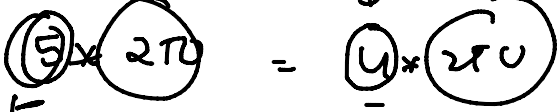
```
if( temp >= 2)
    ans += temp;
}
}
return ans;

int bfs (vector<vector<int>> grid, int i, int j)
{ queue<pair< int ,int >> q;

    int m, n;

    q.push ({i,j});
    grid[i][j]= 0;

  int ans = 1.

  while (!q.empty())
{ int x = q.front().first;
   int y=    q.front().second;

    q.pop();
   ans++;
// adjlist
// save row and save  column
// Save column:
   for( k= 0; k<n;k++
{ if ( grid[x][y] == 1)
```

```
        q.push({i,j});
        grid[i][j] = 0;
    }
}

// same row
for(i=0; i<m; i++)
{
    if(grid[i][j] == 1)
    {
        q.push({i,j});
        grid[i][j] = 0;
    }
}
}
return ans;
```





Min No of edges in the Path



Minimize = No of edges

unweighted → BFS

weighted → Dijkstra
Bellman Ford
Floyd Warshall