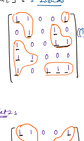
Q1 No of Islands (Medium) (Matrix)

Given a matrix of '1' & '0' find no of Islands.

Islands one present.

Islands surrounded by 0 directions

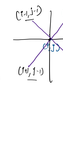
mat[2] = { Island }



Sol 2:

→ Algo: 2 (Adj & Visited)

Idea: Non-Connected components



All is: Ω: (all have one node to finish only)

Connect: How many connected components?

mat[n][m] : visited matrix

for x[] = {-1, 0, +1, 0}

for y[] = {0, -1, 0, +1}

void DFS(for i, for j, for n, for m){

if (i<0 || i>=n || j>=m || j<0 || mat[i][j]==0) return;

// mark (i,j) as visited

mat[i][j] = 0;

// Apply DFS to all nodes (i,j) can reach

DFS (i-1, j, n, m)

DFS (i+1, j, n, m)

DFS (i, j-1, n, m)

DFS (i, j+1, n, m)

}

Conn 3:

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

DFS (i, j, n, m)

}

Conn 4:

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

DFS (i, j, n, m)

}

for mat[n][m]

for c=0

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

for (int j=0; j<m; j++){

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

cnt++

DFS(i, j, n, m)

}

}

}

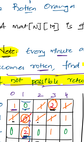
TC: $O(N^2m)$ SC: $O(N^2m)$

Q2 Rotten Oranges (Medium) (BFS)

A mat[n][m] is given → 0 → empty, 1 → fresh orange, 2 → rotten orange

Note: Every minute any fresh orange adjacent to a rotten orange becomes rotten. Find min time when all oranges become rotten.

Q1. not possible return → -1

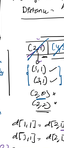


Idea:

If single rotten orange

Oranges are factors

Distance → Ans



Idea:

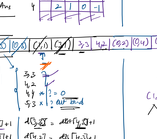
Take all rotten oranges as a source & apply BFS

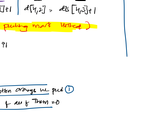
After pushing them in queue, mark

Then → mat[i,j] = 0

mat[i,j] = 0

If a mark is 0 we don't push it





Mat[n][m], dist[n][m], =

Queue < pair<int,int> q;

for x[] = {-1, 0, +1, 0}

for y[] = {0, +1, 0, -1}

while(q.size() > 0){

pair<int, int> cur = q.front();

q.pop();

int i = cur.first, int j = cur.second

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

int xi = i + x[i], int yi = j + y[i] (Adjacent nodes) (no bounds)

if(xi<=n || yi<=n || xi>=0 || yi>=0 || mat[xi][yi]==2 || mat[xi][yi]==0){continue}

dist[xi][yi] = dist[i][j]+1

mat[xi][yi] = 2

q.push({xi, yi})

}

}

// Iterate on mat[i][j] if '1' is present

→ An fresh orange still present return -1

// Iterate & get max dist

TC: $O(N^2m)$

SC: $O(N^2m)$?

S_{dist} + S_{queue}

Note: We can take the distance dist[]

→ I have to count no of level

→ After every one level distance keeping and level

Distance: (level-1) → Indicating end of level

q.size()>>1 } being the distance

Q3 Capture regions on board (Medium)

Given a 2D Mat[] board A & flip all the 'X' converting 'X' to 'O'.

Capture are regions surrounded by 'X'.

→ Closed components

Idea:

→ We can go all components

If a particular 'O' surrounded or not?

(Note): If a 'O' single node & component, it will be boundary. Entire component cannot be surrounded

Idea:

Step: Along the boundary nodes of mat[i][j] = 0, Apply DFS(i,j) from (i,j), all mat[i][j]=0 what can be visited we mark them as mat[i][j] = 1. They cannot be captured

Step: Iterate on mat[i][j]

If mat[i][j] == 'o'

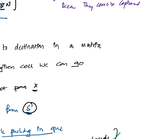
: It can be captured

: mat[i][j] = 'X'

If mat[i][j] == 'o'

: It cannot be captured

: mat[i][j] = 1

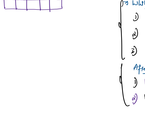


Q4 Shortest distance from source to destination in a matrix

Note: From a given cell we can go

Note: You cannot pass X

Idea: Apply BFS from (2)



While pushing in queue

1) Node cannot go out of bounds

2) Node could not be X

3) Node cannot be already visited

After pushing

1) We need to update the dist

2) We need to mark it has visited

Q5 Valid Path

Given a rectangle with left bottom as (0,0) & right top as (x,y).

Given N circles & their centers & their radius. All circles without touching

Check if it is possible to move from (0,0) → (x,y) along circle

Note: From any point we can go to all 8 directions

Ex1:

Input

$\{x,y\} = \{3,3\}$

N=1, R=1

$C_1 = \{2,2\}$

Ex2:

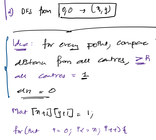
Input

$\{x,y\} = \{3,5\}$

N=3, R=1

$A[] = \{1, 0, 3\}$

$B[] = \{3, 2, 1\}$



	0	1	2	3	4	5
0	1	0	0	0	1	1
1	1	1	0	0	0	1
2	1	0	1	0	1	1
3	0	0	0	1	1	1
4	1	0	1	1	1	1

→ (0,0) → (4,5) / Using DFS

Idea: At every point, if its within circle X touched by circle. Mark it → 0

→ DFS from (0,0) → (x,y)

Idea: for every point, compute its distance from all centers ≥ R from all centers = 1

else = 0

Mat[x+1][y+1] = 1;

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

for(int j=0; j<=y; j++){

// (i,j) We need to get our distance from all centers.

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

for x = A[k], y = B[k]

if ((i-x)^2 + (j-y)^2 <= R^2){

(i,j) is in circle

mat[i][j] = 0; break;

}

}

}

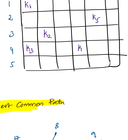
}

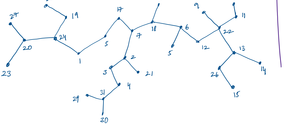
→ DFS from (0,0) to (x,y)

Overall TC: $(x^2 \cdot y^2 \cdot N) + O(xy)$ SC: x^2y

Q6 No: of Arrangements

Knight movement



Will discuss in lab session

Q7: Find length of longest common path