Q Given N x M matrix
Source & destination
Find length of shortest path b/w them.

. → empty
s → source
d → destination
x → blocked

Directions of movement

Q Check for a cycle in an undirected graph

⇒ → DFS
→ check if neighbour is already visited (exclude previous from which we came)

* observation:
If $E > V-1$ ⇒ cycle!

Q Detect cycle in directed graph

⇒ Prev idea of undirected graph won't work.

Recitation - Mon 30 Nov 1:10 PM

Q Given a matrix and source & destination
Find shortest distance b/w S & D.
∵ → empty 'D' → destination
'S' → source 'x' → blocked

⇒

- BFS is NOT useful to find shortest distance
- DFS doesn't work alone due to cycle dependency

- Since we want to traverse level wise, BFS is the best choice.

⇒ • Use IsValid() function to check bounds | presence of 'x' | vis
• For generating neighbour cells,
$d_x = [-1 \quad +1 \quad 0 \quad 0]$
$d_y = [0 \quad 0 \quad -1 \quad +1]$
for k in range(4):
new i := i + d_x[k]
new j := j + d_y[k]
if IsValid(new i, new j):
~ Queue logic ~

• Use vis & dist matrices to keep track.

TC: $\boxed{V = mn}$

E every node has 4 neighbours (adj nodes)

Total nodes = 4mn ✓

∴ Total no of edges = 2mn

$\boxed{E = 2mn}$

TC of BFS = $O(V+E)$ = $O(2mn+2mn)$

$\boxed{TC = O(mn)}$

SC = Q + vis + dist
= V + V + V

$\boxed{SC = O(mn)}$

Q Given a 2D matrix
Residues - R
Hospital - H
Find residues, find dist to all H

⇒

R	R	R	H
R		H	H
R	R	R	R

3	2	1	0
2	1	0	0
3	2	1	1

my idea is to start BFS from hospitals and mark dist as 0. If you encounter:
H → reset dist to 0
R → increment dist by 1

• Idea: Start BFS from all H at once.
How? Put all H in Queue & start BFS
All H are at level 0.
And they are already marked as vis

This technique of starting BFS from multiple nodes is known as

$\boxed{\text{Multi Source BFS}}$

V = mn
E = 2mn
undirected graph.

TC = $O(V+E)$
TC = $O(mn)$
SC = $O(V) = O(mn)$

Q Given an undirected graph
Check if a cycle is present or not

⇒ DFS → If a visited node is reached again, from there in a cycle forms.

→ make sure to not visit the node you just came from.

Q 2 already visited

```
def (u, prev):
    vis[u] = T
    f( u: adj[u]):
        if (u == prev):
            if not vis[u]:
                dfs(u, v)
            else:
                isCycle = True
```

• Graph may have more than 1 connected components

for i in range(1,5): → Try all nodes
if not vis[i]: → in the graph
dfs(i, -1) → Effectively this is only called twice (for every CC)

• In 1 CC in a graph, there are exactly V-1 edges that connected V vertices. If there is just 1 more edge, we can declare that it has a cycle.

```
dfs(v)
    vis[v] = T
    cnt += 1
    sum$rc += len(adj[v])
    for u in adj[v]:
        if not vis[u]
            dfs(u)

for i in range(1, n+1):
    if not vis[i]:
        cnt = 0
        sum$rc = 0
        dfs(i)
        v = cnt
        E = sum$rc/2
        if E > v-1:
            return True → cycle
```

TC = $O(V+E)$ SC = $O(V)$

Q Given a directed graph. Detect Cycle

→ The same logic as that of undirected graphs doesn't work here.

0 - unvisited
1 - visited
2 - visited + gone back

```
vis = [0] * n
dfs(v):
    vis[v] = 1
    for u in adj[v]:
        if vis[u] == 0:
            dfs(u)
        elif vis[u] == 1:
            isCycle = True → cycle
    vis[v] = 2
for i in range(1, n+1):
    if vis[i] == 0:
        dfs(i)
```

TC = $O(V+E)$
SC = $O(V)$

HW Q Find if there is a path from Source & Destination.
Use DFS