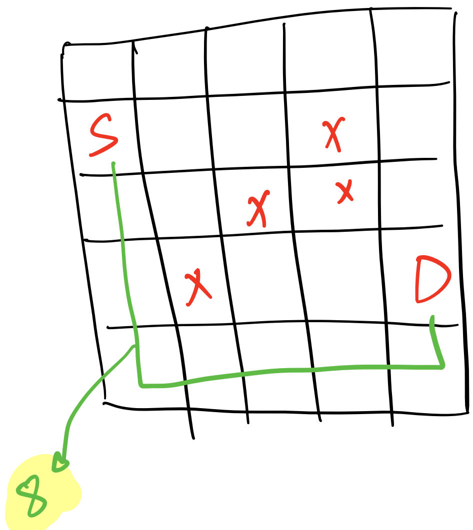
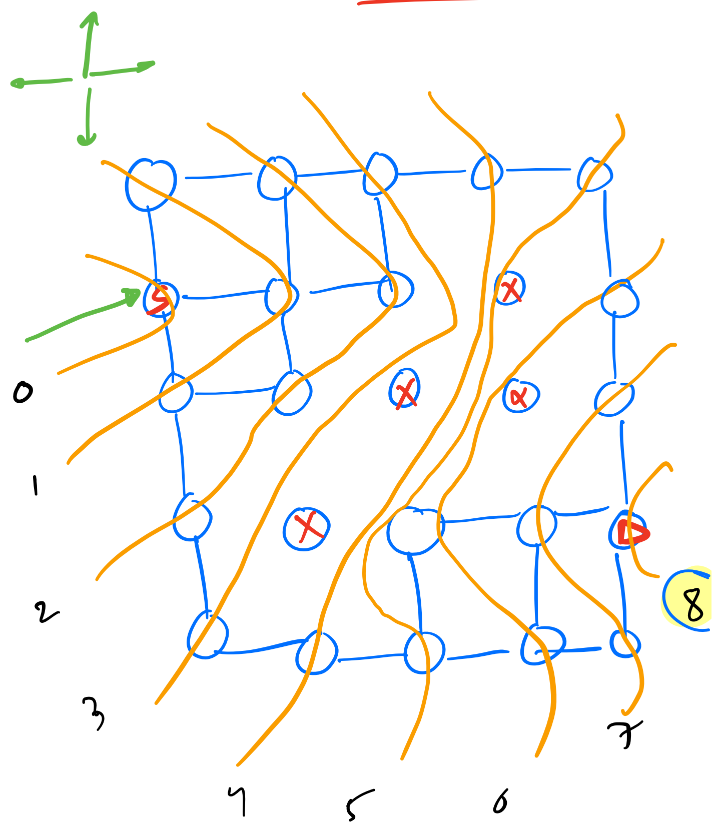
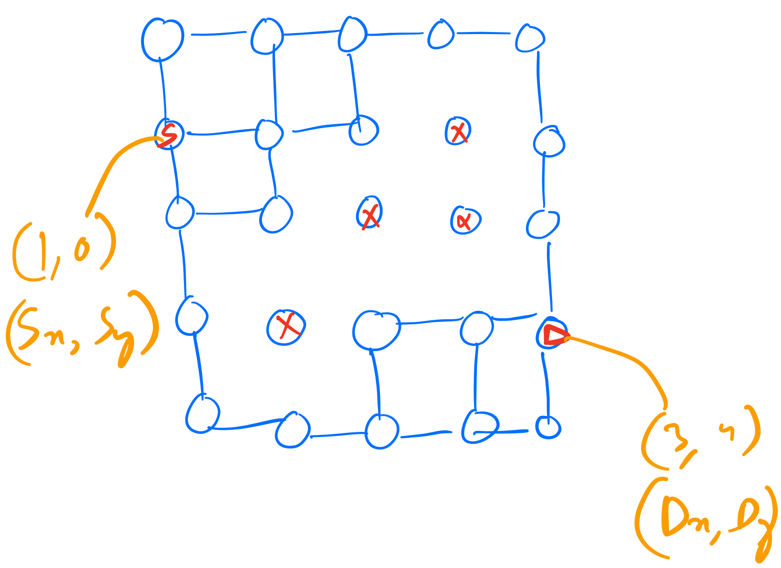
Q. Given $N \times M$ matrix.
Source & Destination.
Find the shortest path b/w them.
X → Cannot visit.

$1 <= N, M <= 10^3$

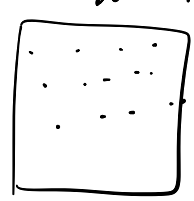




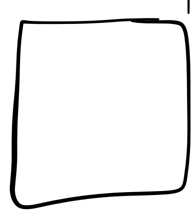


BFS

bool vis[][]



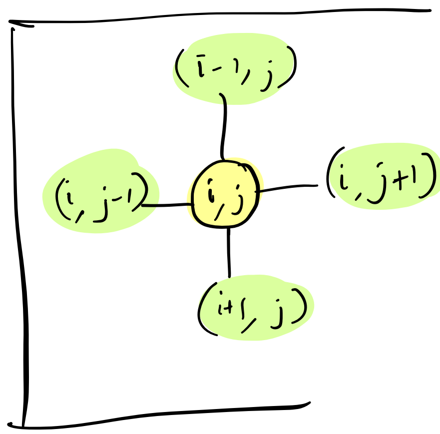
int dis[][]



pair <int, int> p;

↓ p.first ↓ p.second

```
int d[N][M] = {∞};
bool vis[N][M];
// Sx, Sy, Dx, Dy.
Queue < pair<int, int>> Q;
Q.enqueue({Sx, Sy});
d[Sx][Sy] = 0;
vis[Sx][Sy] = true;
while( ! Q.isEmpty()) {
    pair<int, int> p = Q.front();
    Q.dequeue();
    i = p.first, j = p.second;

    f(K=0; K<4; K++) {
        nI = i + dx[K];
        nJ = j + dy[K];
        if(check(nI, nJ) == true) {
            Q.enqueue({nI, nJ});
            d[nI][nJ] = 1 + d[i][j];
            vis[nI][nJ] = true;
        }
    }
} → ret d[Dx][Dy];
```



K

dx[] = {-1, 0, 1, 0}

dy[] = {0, 1, 0, -1}

check (x, y)

1. Inside matrix
2. not visited
3. not 'x'

```
bool check(x, y) {
  if( x>=0 && x<N
      && y>=0 && y<M
      && vis[x][y]==false
      && A[x][y] != 'X')
        ret true;
    ret false;
}
```

Check (x, y)

1. Inside matrix
2. not visited
3. not 'X'

$G = \{V, E\}$

$V = N \times M$

$E = N \times M$

$TC = O(V+E)$

$NM + NM$

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

$SC = O(V)$

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

HW

Dfs on 2D Matrix !!!

Q Given a 2D matrix.

Residence (R)
Hospital (H)
for every (R), find the min. distance to a (H)

R	R	R	H
R	R	H	H
R	H	H	R

$\longrightarrow$

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

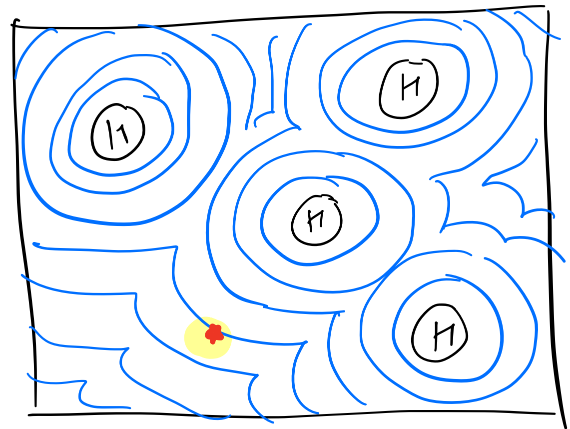
1) $\underbrace{\forall R}_{N \times M}$ $\underbrace{\text{do bfs to find the nearest (H)}}_{N \times M}$

$N \times M \quad \times \quad N \times M$

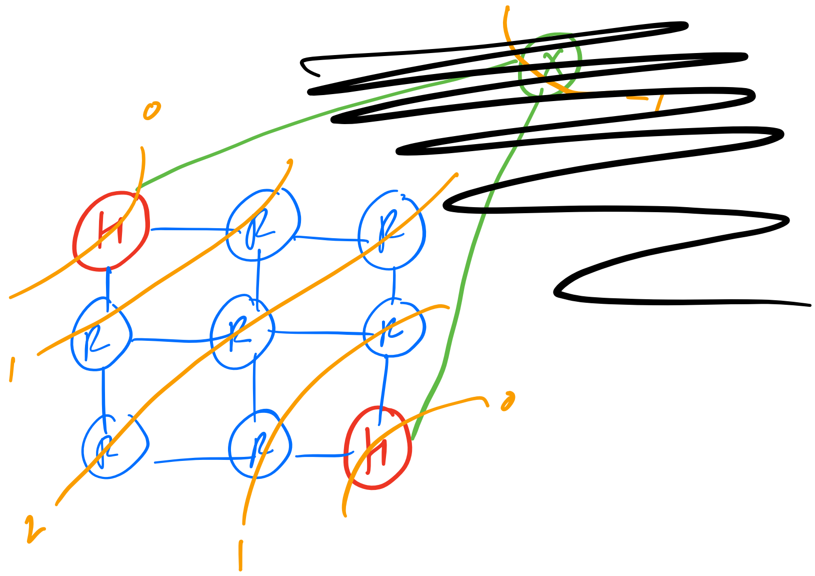
TC $\sim N^2 M^2$

2) $\underbrace{\forall H}_{N \times M}$ $\underbrace{\text{do bfs \& update the shortest dist of every (R)}}_{N \times M}$

TC $\sim N^2 M^2$



H	R	R
R	R	R
R	R	H



Initial :

q. enque (All the H coordinates)

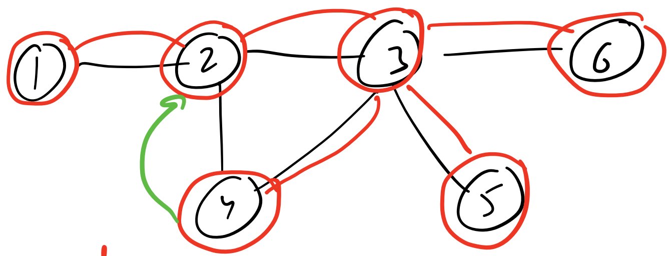
→ dis [ ] [ ] = 0 (∀ H)

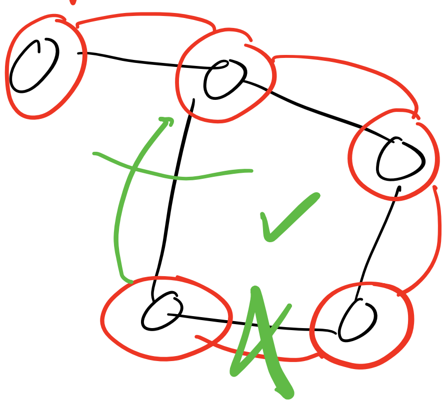
→ vis [ ] [ ] = true (∀ H)

start while (! q. isEmpty ( )) {

}

Q Given an undirected graph [1 – N].
Check if a cycle is present or not!

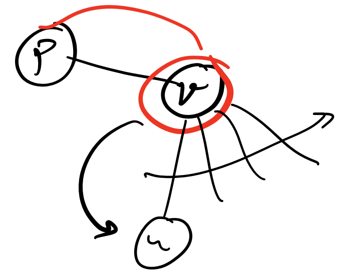




Idea: If a node has any neighbour other than the node you came from!
which is already visited

$\Rightarrow$ cycle!

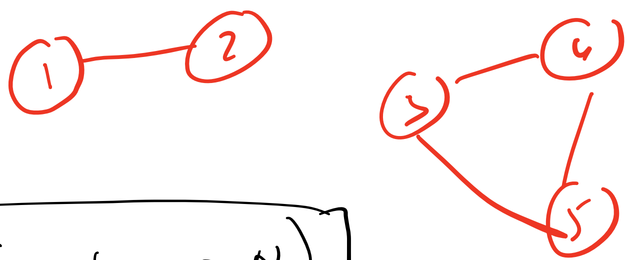
```
bool isCycle = false;
void dfs(int v, int p){
    vis[v] = true;

    f(u: adj[v]){
        if(u != p){
            if(vis[u] == false){
                dfs(u, v);
            }
            else{
                isCycle = true;
            }
        }
    }
}
```



$\boxed{TC = O(V+E)}$

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

dfs(1, 0);




```
f(i=1 ──> N)
    if(!vis[i])
        dfs(i,0)
```

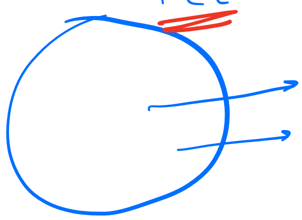
NOTE:

REMOVE BIAS

GRAPH MIGHT NOT ALWAYS BE CONNECTED!

II

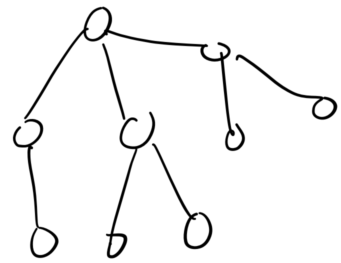
1 CC



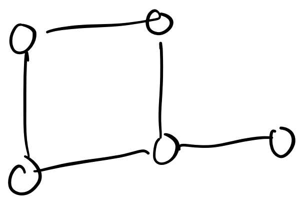
$\boxed{E > V-1}$ → cycle!

tree

V, E

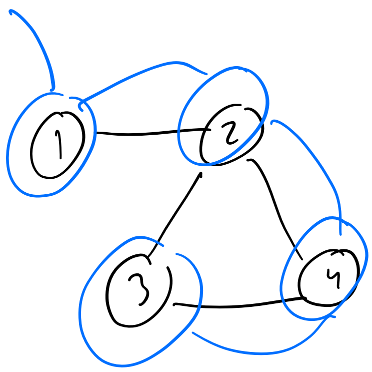


V = 5
E = 5



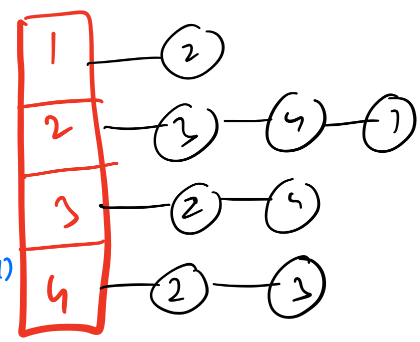
$\boxed{E = V-1}$ No cycle

E > V-1 : cycle!



E

E ?



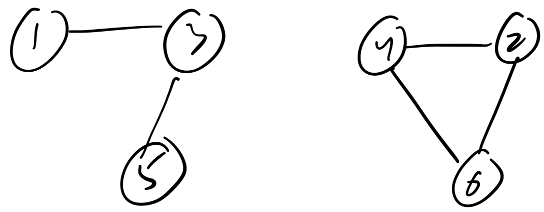
V → dfs : cnt++

E → dfs : $\dfrac{sz \mathrel{+}= adj[v].size()}{2}$

```
dfs(v) {
  vis[v] = true;
  cnt++;
  sz += adj[v].size();
  for(u : adj[v]) {
     if( vis[u] == false )
         dfs(u);
     }
  }
}
```

V = cnt ;
E = sz/2 ;

if(E > V-1)
  → cycle!
else
  → No cycle!

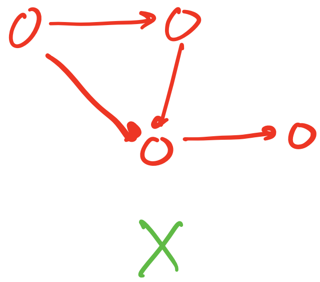


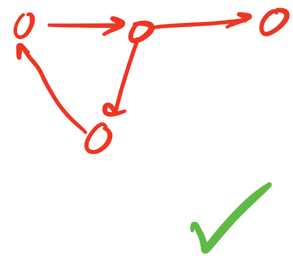

```
f( i=1 → N) {
    if( !vis(i)) {
        cnt = 0; sz = 0;
        dfs(i);

        V = cnt ;
        E = sz/2;

        if( E > V-1)
            → cycle!
    }
}
→ NO cycle!
```

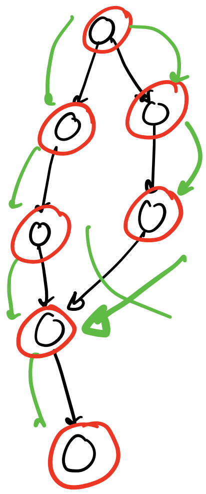
$\boxed{TC = O(V+E)}$

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

Q Given a directed graph. Detect cycle!





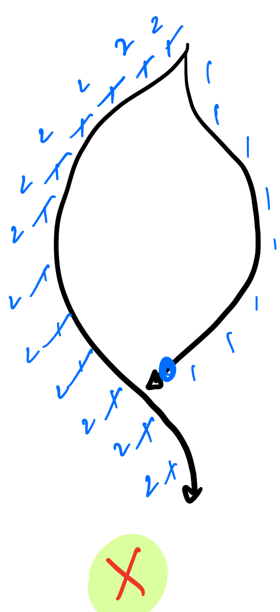


VIS

0 : NOT VIS

1 : VIS & IN STACK

2 : VIS & OUT



```
void dfs(v){
    vis[v]=1;
    f(u: adj[v]){
        if(vis[u]==0){
            dfs(u);
        }
        else if(vis[u]==1){
            isCycle = true;
        }
    }
    vis[v]=2;
}
```

$\boxed{TC = O(V+E)}$

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

$f(i: 1 \rightarrow V)$

// CC