Graph 1

Tuesday, 12 January 2021 8:30 PM

Nodes / Vertices ✓

Edges ✓

Properties

⇒ Directed

Undirected

⇒ Connected

Disconnected

⇒ Weighted

Unweighted

⇒ Cyclic

Acyclic

⇒ Dagger

Indegree / Outdegree

Simple Graph → Graph without self loop or multiedges.

Use of Graph in Code →

⇒ Adjacency Matrix

Space = $O(\text{Nodes}^2)$

	a	b	c
a	0	1	0
b	0	0	1
c	1	0	0

⇒ Adjacency list

Space = $O(V+E)$

a → {b, c}
b → {c}
c → {a, b, d}
d → {c}

O(E)

a → b
b → c
c → a, b, d
d → c

O(V)

Graph Traversals

⇒ DFS (Depth First Search)

a b e f g c d

visit(a) = {a} ✓

```
void dfs(int z) {
    visit[z] = 1;   print(z);
    for (int v : adj[z]) {
        if (!visit[v])  dfs(v);
    }
}
```

d

main

```
for(i = 1; i <= N; i++){
    if (!visit[i])  dfs(i);
}
```

F

Time $O(V+E)$

⇒ BFS (Breadth First Search) – level order traversal

Queue

a b c d e f g h

Q. Path in directed Graph

Given a directed graph check if there is a path from node u to node v.

u = 1
v = 4

1 → 2 → 4

Ans = 1

check BFS/DFS from u & if we reach v ⇒ Ans = 1
else ⇒ Ans = 0

visit[N] = {0}; visited

int dfs (int v) {

    if (v == v) return true; Time = O(E)
    visit[v] = 1;
    for (int x : adj[v]) {
        if (!visit[x]) {
            if (dfs(x)) return true;
        }
    }
    return false;
}

Extra space = O(N)

max E is $O(N^2)$

$N \le 10^5$
$E \le 10^5$

Q. Number of Islands

Given a matrix of integers with 0 & 1.
water land

Find total no. of connected 1's in the matrix.

(0,3,y) (0,2,y) (0,1,y)
(1,3,y) ← (1,2,y) → (1,1,y)
(2,3,y) (2,2,y) (2,1,y)

Ans = 3

visit[n][m] = {0}

the Traversal is Started

max → ans = 0
for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++) {
        if (m[i][j] == 1 && !visit[i][j]) {
            ans++;   dfs(i, j);
        }
    }
}
return ans;

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

void dfs (i, j) {
    visit[i][j] = 1;
    for (int d = 0; d < 8; d++) {
        x = i + dx[d];
        y = j + dy[d];
        if (0 ≤ x ≤ n-1 && 0 ≤ y ≤ m-1 && m[x][y] == 1 && !visit[x][y])
            dfs(x, y);
    }
}

Time = O(n × m)
Space = O(n × m)

O(1 × m)

Q. Rotten Oranges

Given a matrix of integers
0 : empty
1 : fresh orange
2 : rotten orange

Every minute, any fresh orange adjacent to a rotten orange becomes rotten. (x,y-1) ← (x,y) → (x,y+1)
(x-1,y)
(x+1,y)

Find the time when all oranges will become rotten.

If not possible return -1.

Ans = -1

Ans = 2

BFS / BFS

Multisource BFS

Q. Bipartite Graph

Graph in which vertices can be divided into 2 sets such that it there is no edge b/w the nodes in the same set.

✓ Every Tree is a bipartite graph.

Cycle graph with even nodes

$s_1 \to \{a, c, e\}$
$s_2 \to \{b, d\}$

$s_1 \to \{a, c, e\}$
$s_2 \to \{b, d, f\}$

Q. Check if the given graph is bipartite graph.

Time = O(V+E)
Extra space = O(N)

col[v] = -1 ∀ v

col[v] = 0;

bool dfs (v) {
    for (v : adj[v]) {
        if (col[v] == -1) {
            col[v] = 1 - col[v];
            if (!dfs(v)) return false;
        } else if (col[v] == col[v]) {
            return false;
        }
    }
    return true;
}