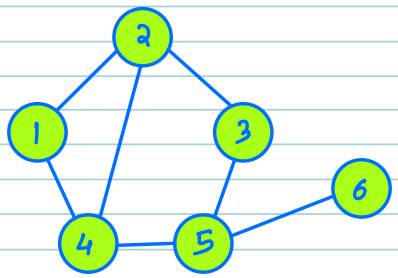
# Graph 2 : BFS, Matrix Questions & Topological Sort

**Question :-** Breadth First Search

Given an undirected graph & Source Node to Destination Node, check if Node can be visited from Source Node.




S D
1 → 6

Q which data structure is used to perform BFS

↳ TREES [ queue ]

**Input :-**

| N | E |
|---|---|
| 6 | 7 |
| 1 — | 2 |
| 1 — | 4 |
| 2 — | 4 |
| 2 — | 3 |
| 3 — | 5 |
| 5 — | 6 |
| 4 — | 5 |

Adj list

0 → { }
1 → { 2, 4 }
2 → { 1, 4, 3 }
3 → { 2, 5 }
4 → { 1, 2, 5 }
5 → { 3, 6, 4 }
6 → { 5 }

**DRY RUN**

queue :- ~~1~~ ~~2~~ ~~4~~ ~~3~~ 5 6

Visited :-

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| // | T | T | T | T | T | T |

IDEA:- ① Do BFS once a node is visited, it should not be added again.

② Delete front element from queue. Add its unvisited neighbour & mark them as visited

③ If in end visited[Destination] is true then possible to reach from source otherwise.

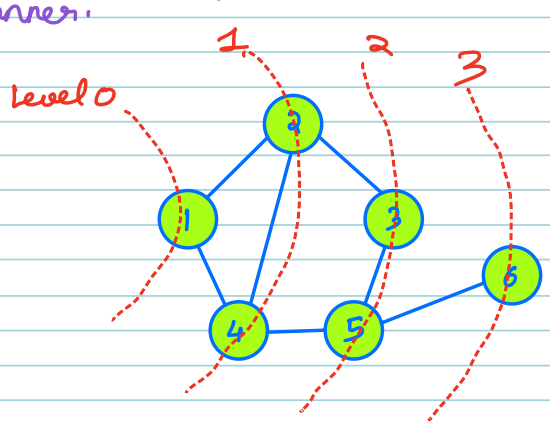
PSEUDO CODE

```
// Given N, M. Create Adj list out of it

bool BFS ( list<int> g[], int S, int D){
    bool [N+1] Vis;
    Queue< int> q;
    q.add(S);
    Vis[S] = T;
    while (q.size()>0){
        int u = q.front();
        q.delete()
        for ( i=0; i<g[u].size(); i++){
            v = g[u].get(i);
            if ( Vis[v] == False){
                Vis[v] = True;
                q.add (v);
            }
        }
    }
    return Vis[d];
}
```

## Imp Observation

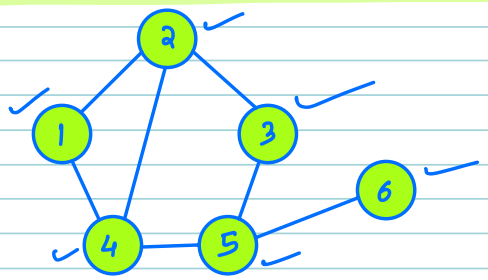
↳ BFS helps to iterate in Level order manner.




1 2 4 3 5 6

★ BFS helps to find Shortest Distance from Source.

DRY RUN




S = 1
D = 6

Queue :- 1 2 4 3 5 6

dis :-

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| // | 0 | 1 | 2 | 1 | 2 | 3 |

Q Does DFS helps to find the shortest Distance

↳ No

1 (2, 4)
2 (1, 4, 3)
4

Q Then why we use DFS?

↳ Easy & Smaller Code to Write, Recursion take care of many task.

↳ If question dont want shortest distance, then why to use BFS.

PSEUDO CODE

```
// given input N, M
    ↳ Construct adjacency List

bool bfs( list<int>[] g, int s, int d){

    bool vis[N+1];
    int dis[N+1];

    queue<int> q;
    q.add(s);
    dis[s] = 0

    vis[s] = T;

    while( q.size()>0){

        int u = q.front();
        q.delete();

        // Add unvisited node to q
        for(i:0; i<g[u].size(); i++){

            dis[v] = dis[u]+1;

            v = g[u].get(i);
            if( vis[v] == F){

                vis[v] = T
                q.add(v);
            }
        }
    }
    return visited[d]; / dis[d]
}
```

Nodes ↑ ↗ Edges

TC → O(N+E)

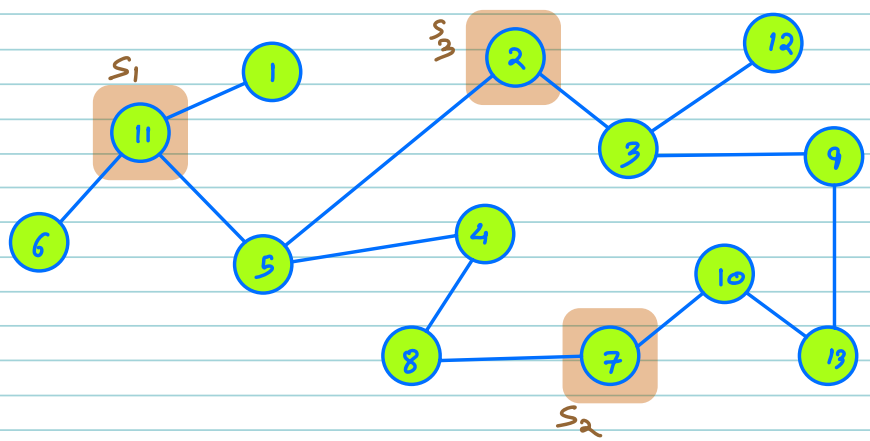
SC → O(N+E) + N + N + N

Adj List, dis, vis, queue

↳ O(N+E)

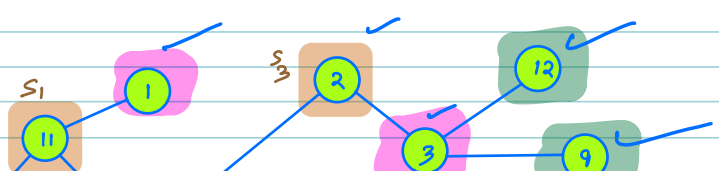
Question :- Multi Source BFS

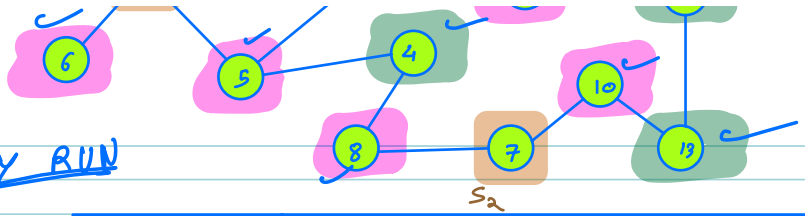
There are N number of nodes & multisource (S1, S2, S3), we need to find the length of shortest path for given destination node to any one of the source node (S1, S2, S3).




Real Life Problem

↳ [Instamart, zepto, blinkit ... ]






DRY RUN

queue :- (S3) (S2) (S1)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| dist :- | 1 | 0 | 1 | 2 | 1 | 1 | 0 | 1 | 2 | 1 | 0 | 2 | 2 |

(Very Very Important)

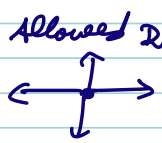
**Question** :- Given a matrix of Integer

A[i][j] → 0 Empty
→ 1 Fresh orange
→ 2 Rotten orange

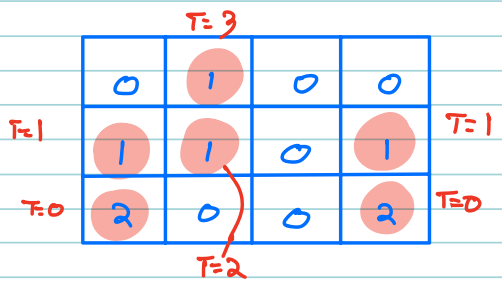
Every minute a fresh orange, adjacent to a rotten orange becomes rotten. Find the time when all the oranges become rotten.

If not possible return -1.

Allowed Directions



Ex1




Ex2

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 2 |
| 2 | 0 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 2 | 1 | 1 |

→ Time to get rotten

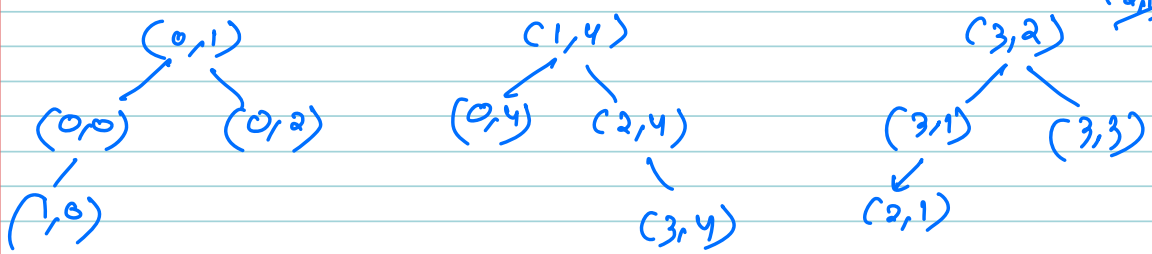
# DRY RUN

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 2 |
| 2 | 0 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 2 | 1 | 1 |

(Time array)

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | ~~1~~ | 0 | ~~1~~ | ∞ | ~~1~~ |
| 1 | ~~2~~ | ∞ | ∞ | ∞ | 0 |
| 2 | ∞ | ~~2~~ | ∞ | ∞ | ~~1~~ |
| 3 | ∞ | ~~1~~ | 0 | ~~1~~ | ~~2~~ |

queue :- ~~(0,1) (1,4) (3,2) (0,0) (0,2) (0,4) (2,4) (1,0) (3,1) (3,3) (3,4)~~ (2,1)




Q How will you figure out whether their is any fresh orange Remaining.

↳ If ∞ is present corresponding to Fresh orange (1)

CONCLUSION :- if ∞ corresponding to any 1 :

return -1

else

Max of Time corresponding to all 1's

## PSEUDO CODE

**Step 1 :-** Insert all source inside queue & update 0 corresponding to all sources in time array.

**Step 2 :-** Do BFS & update the time array

**Step 3 :-** After BFS :-

if all oranges rotten
return max time

else
return -1

Q Do we need Adjacency list in this question?

↳ No

TC → O(M * N)

SC → O(M * N)

## Alternate way of asking

**Flipkart** Grocery has several warehouses spread across the country and in order to minimize the delivery cost, whenever an order is placed we try to deliver the order from the nearest warehouse.

Therefore, each Warehouse is responsible for a certain number of localities which are closest to it for deliveries, this **minimizes the overall cost for deliveries**, effectively managing the distribution workload and minimizing the overall delivery expenses.

OR

You are given a 2D matrix **A** of size **NxM** representing the map, where each cell is marked with either a **0 or a 1**. Here, a **0** denotes a locality, and a **1** signifies a warehouse. The objective is to calculate a new **2D matrix** of the same dimensions as **A**.

In this new matrix, the value of each cell will represent the minimum distance to the nearest warehouse. For the purpose of distance calculation, you are allowed to move to any of the **eight adjacent cells** directly surrounding a given cell.

Ex

```
0 0 0 0 1 0 0 1 0
0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1
```

**Answer**

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

(Very Imp)

**Question :-** Given N courses with pre-requisites, check if it is possible to finish all courses.

**Ex:-** N=5

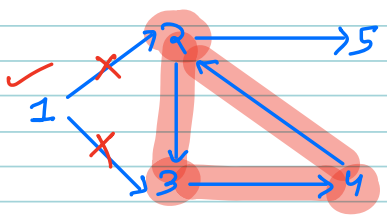
x is pre-requisite of y.

1 ⟶ 2 & 3

2 ⟶ 3 & 5

3 ⟶ 4

4 ⟶ 2

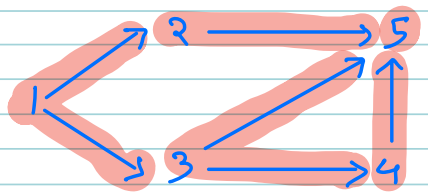



**CONCLUSION :-**

```
If ( cyclic Graph)
    return False
} else {
    return True; // Task
}
```

★ To Solve this question :- Directed Acyclic Graph is given

Ex :-




⇒ check which is a valid order to do Courses.

This ordering is known as TOPOLOGICAL ORDER

- 1 2 3 4 5 ✓
- 1 3 2 4 5 ✓
- 1 3 4 2 5 ✓

// TOPOLOGICAL SORTING

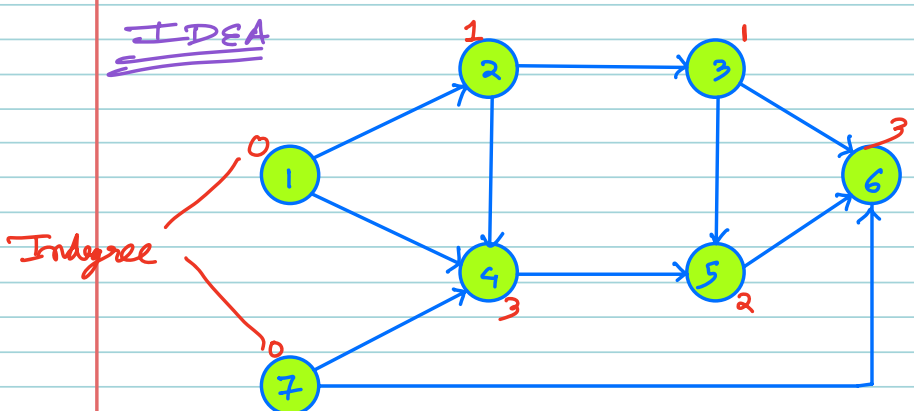
Real Life Example

↳ [ Schooling , Softwares ]

## DEFINATION

**Topological sort** is a linear ordering of the vertices (nodes) in a directed acyclic graph (DAG) such that for every directed edge (u, v), vertex u comes before vertex v in the ordering. In other words, it arranges the nodes in such a way that if there is a directed edge from node A to node B, then node A comes before node B in the sorted order.

IDEA

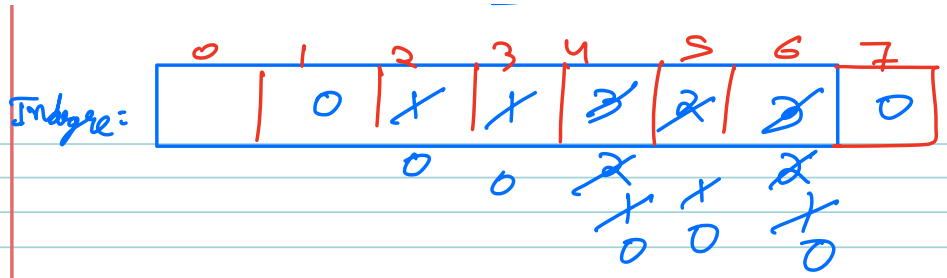



Indegree ↓

(No. of incoming edges)

queue :- ~~7~~ 1 ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

Print :- 7 1 2 3 4 5 6




## PSEUDO CODE

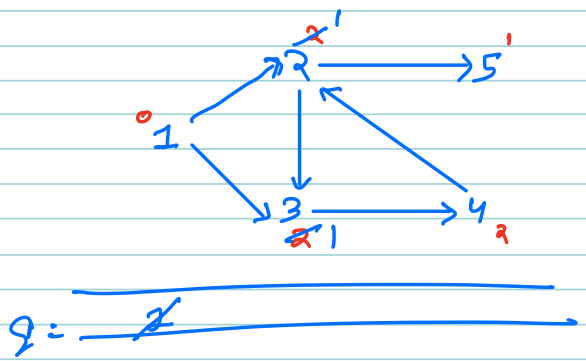
```
list <int> g[N+1]
indegree[N+1];
for( i=0; i<m; i++){
        g[u].add(v);
        Indegree[v]++
}

q <int>;
for i=1; i<=N; i++){
        if (indegree[i]==0){
            q.add(i);
        }
}

while( q.size()>0){
        u = q.front();
        q.remove();
        Print(u);
        for( i=0; i<g[u].size();i++){
            v = g[u].get(i);
            Indegree[v]--;
            if( Indegree[v]==0){
                q.add(v);
            }
        }
}
```

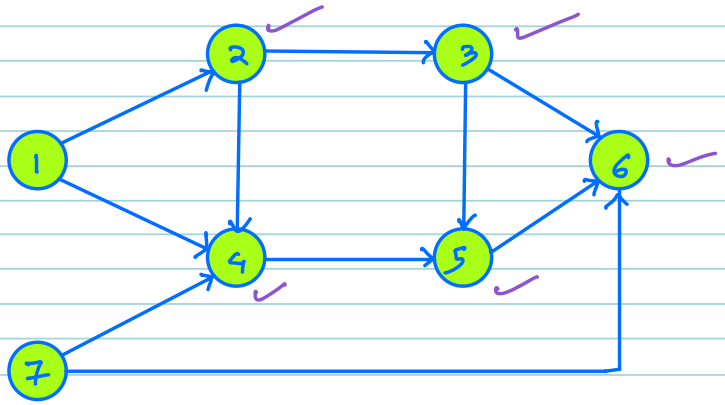
Q what is in the end if indegree is not 0?

↳ It means their is a Cycle.

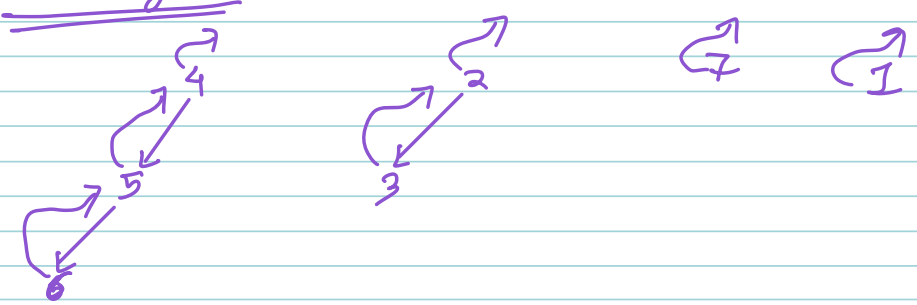



This is another way of finding Cycle in a Directed Graph.

## METHOD 2 [ Using DFS ]




Pick Any No




Print :- 6 5 4 3 2 7 1

Rev → Topological Sort

Q. why this is working?

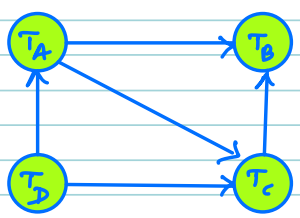
↳ we are printing No. when it is not acting like a dependency to any other No.

# PSEUDO CODE

```
Vis[N+1] = False
for (i -> 1 to N){
    if (! Vis[i]){
        DFS (i, Vis);
    }
}
```

```
void DFS (u, Vis){
    Vis[u] = True;
    for( i -> Neighbours of u){
        if (! Vis[i]){
            DFS (i, Vis);
        }
    }
    Print (u);
}
```

**Ques 1:-** Which of the following is correct Topological order?




(E)

TOPOLOGICAL ORDER $\Rightarrow T_D\ T_A\ T_C\ T_B$ — C

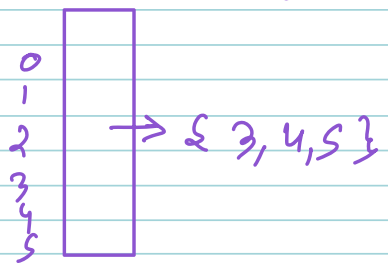
1 2 3 4 5

PROBLEM WITH THIS APPROACH-2

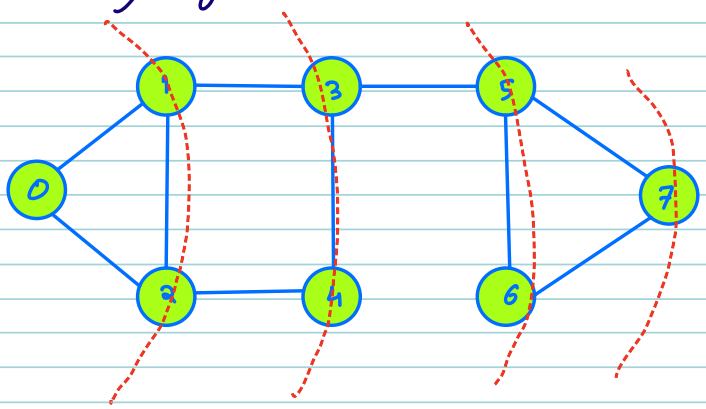
↳ It always gives cycles even for Directed Cyclic graph.

TO RESOLVE :-

(1) -> create Adj. List




Question :- Find the minimum no. of Edges to reach V starting from u in undirected simple graph.




u = 0

v = 7

↳ SIMPLE RESOLVED