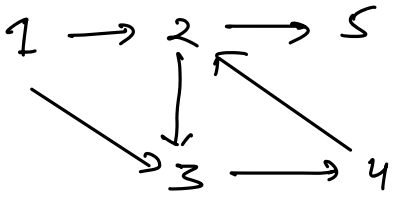
Welcome ☺

Agenda : BFS traversal
questions
Topological Sort

Breadth First Search (BFS) Traversal



O/P ⇒ 1 2 3 5 4

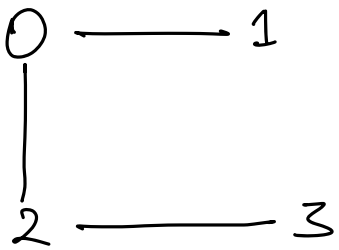
```
Code   ∀i vst[i] = false
       for( i → 1 to N)
       {
          if( ! vst[i] ) bfs(i)
       }

       void bfs ( node u)
       {
          vst[u] = True
          q.enqueue(u)
          while ( ! q.isEmpty())
          {
             y = q.dequeue()
             print(y)
             for( z : adj[y])
             {
                if( ! vst[z] )
                {
                   q.enqueue(z)
```

T.C = O(N+E)

S.C = O(N)

```
            vst[z] = True
        }
    }
}
```

Quiz



O/P 0 1 2 3

0 2 1 3

Q Rotten Oranges

Given a matrix of integers

0 → empty

1 → fresh orange

2 → rotten orange

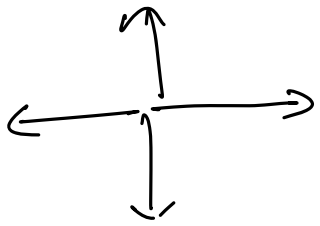
Every minute any fresh orange which is adjacent to rotten orange become rotten

In how many minutes, will all oranges become rotten? If not possible, then return -1

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

T = 0

Ans = 2



Multi-source BFS

~~(0,1)~~ ~~(1,4)~~ ~~(3,2)~~ ~~(0,0)~~ ~~(0,2)~~ ~~(0,4)~~ ~~(2,4)~~ ~~(3,1)~~ ~~(3,3)~~ ~~(1,0)~~ ~~(3,4)~~ ~~(2,1)~~

Code

```
freshO = 0
min = 0

// 1. Traverse and insert rotten oranges into a
      queue. Also keep count of fresh oranges.
if( freshO == 0) {
        return 0
}

while ( !q.isEmpty())
{
    cell = q.dequeue()
    x = cell[0]
    y = cell[1]
    min = cell[2]

    for( i -> 0 to 3)
    {
        newX = x + dx[i]
```

```
dx [-1  1  0  0]
dy [ 0  0 -1  1]
```

```
        newy = y + dy[i]
        if( newx >= 0 && newx < N &&
            newy >= 0 && newy < N &&
                        grid[newx][newy] == 1)
        {
            grid[newx][newy] == 2
            freshO --
            q.enqueue({ newx, newy, min+1 })
        }
    }
}
return (freshO == 0) ? min : -1
```

Q Flipkart delivery. Find min. distance from warehouse. for each pincode.

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

Same as previous question

Q. Possibility of finishing course

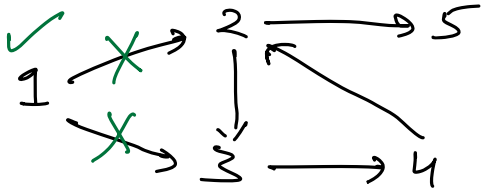
N = 5

1 → 2 & 3 (1 is pre-requisite for 2 & 3)

2 → 3 & 5

3 → 4

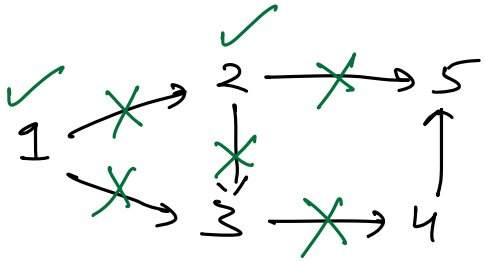
4 → 2



NO

⇒ If there is a cycle in graph, return false. else true.

1 → 2 → 3 → 4 → 5



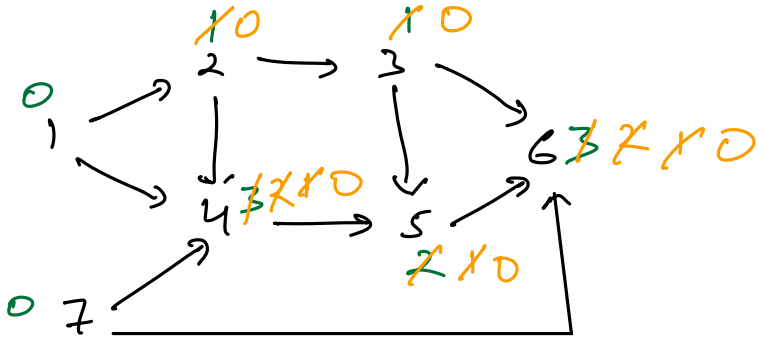
Topological Order ⟶ Directed Acyclic Graph

⇒ linear ordering of nodes s.t if there is an edge from node i → j, then i will be on the left of j

⇒ Multiple topological order can exist for graphs.

1 3 2

3 1 2



Code

1	7	2	3	4	5	6

O/P ⇒ 1 7 2 3 4 5 6

Code

Compute indegree

1. Insert all nodes in queue with indegree 0
2. ∀ dequeue, visit all neighbour and update indegree (reduce by 1)
3. If updated indegree of any node becomes zero, insert it into queue and repeat step 2 & 3 till queue is empty.

T.C = O(N+E)

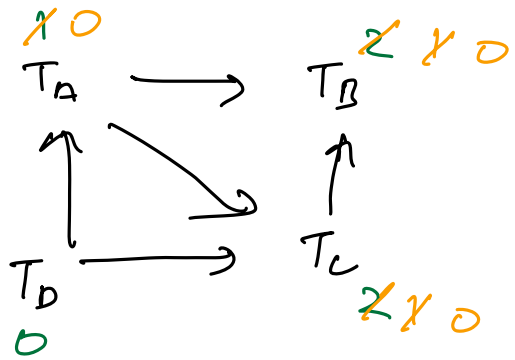
S.C = O(N)

Topological sort (Right to left)

Topological sort can end if outdegree = 0 for that node.

→ length of adj. list.

⇒ print it in reverse order



T_D T_A T_C T_B

⇒ BFS leads to the shortest path to destination.