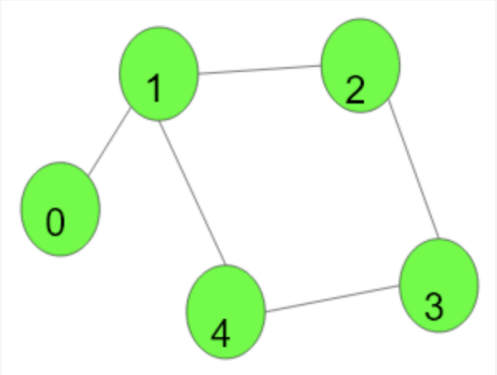
AGENDA

- Cycle in an Undirected Graph
- Cycle in a Directed Graph
- Topological Sort
    - DFS-based
    - In-degree based
- Rotten Oranges
- Dijkstra’s Algo

Detect Cycle in Undirected Graphs:

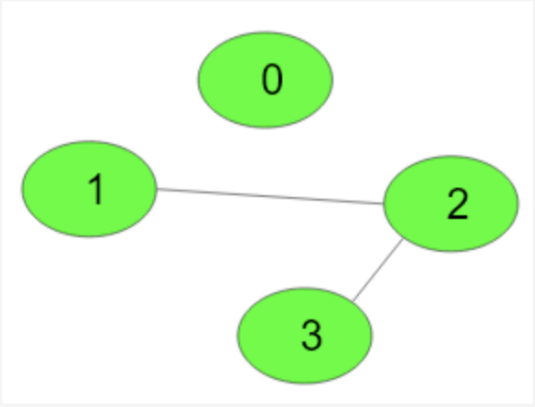
Given an undirected graph with V vertices and E edges, check whether it contains any cycle or not.

Input:



Output: 1
Explanation: 1->2->3->4->1 is a cycle.

Input:

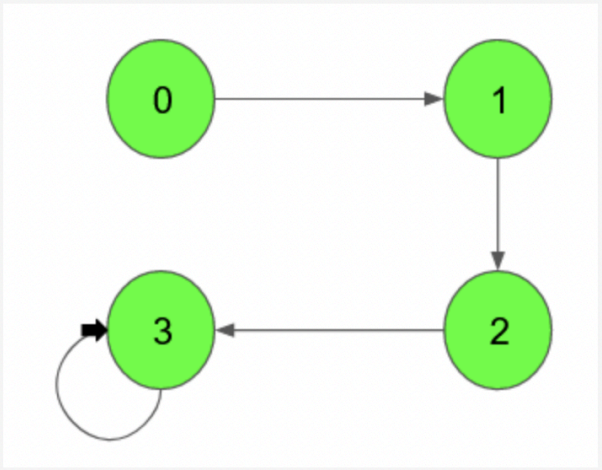


Output: 0
Explanation: No cycle in the graph.

Detect Cycle in a Directed Graph:

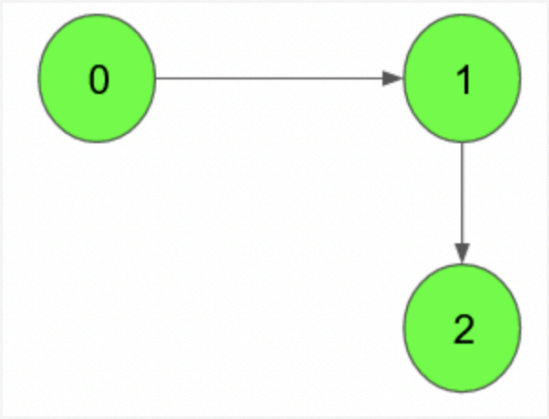
Given a Directed Graph with **V** vertices (Numbered from **0** to **V-1**) and **E** edges, check whether it contains any cycle or not.

Input:



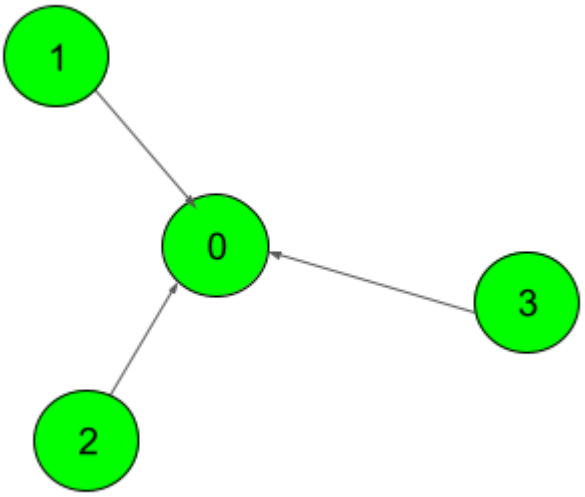
Output: 1
Explanation: 3 -> 3 is a cycle

Input:

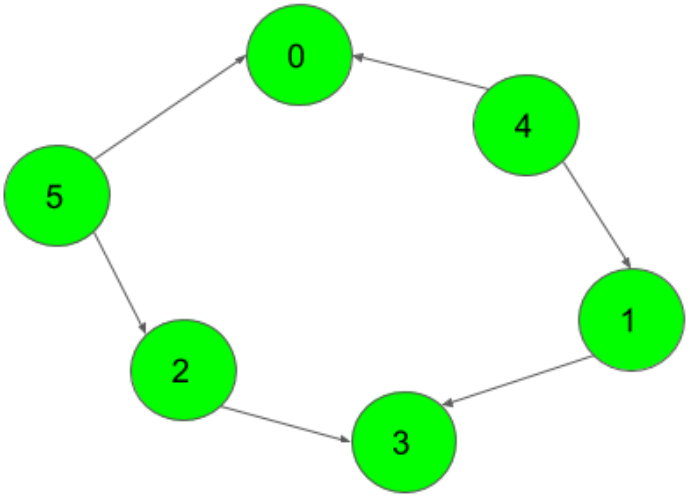


Output: 0
Explanation: no cycle in the graph

Topological sort:
Given a Directed Acyclic Graph (DAG) with V vertices and E edges, Find any Topological Sorting of that Graph.
Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge u v, vertex u comes before v in the ordering.



A correct output can be: [3, 2, 1, 0] or [3, 1, 2, 0] etc.



A correct output can be: [5, 4, 2, 1, 3, 0]

Rotten Oranges

Given a matrix where each cell in the matrix can have values 0, 1 or 2 which has the following meaning:
0 : Empty cell
1 : Cells have fresh oranges
2 : Cells have rotten oranges

We have to determine what is the earliest time after which all the oranges are rotten. A rotten orange at index [i, j] can rot other fresh orange at indexes (i-1, j), (i+1, j), (i, j-1), (i,j+1) (**up**, **down**, **left** and **right**) in unit time.

Note: Your task is to return the minimum time to rot all the fresh oranges. If not possible returns **-1.**

Examples:

Input: mat[][] =
[[0, 1, 2],
[0, 1, 2],
[2, 1, 1]]
Output: 1
Explanation: Oranges at positions (0,2), (1,2), (2,0) will rot oranges at (0,1), (1,1), (2,2) and (2,1) in unit time.

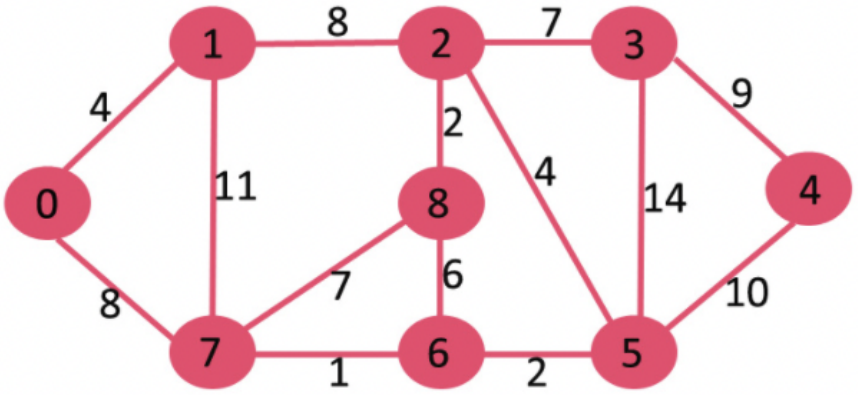
Input: mat[][] = [[2, 2, 0, 1]]
Output: -1
Explanation: Oranges at (0,0) and (0,1) can't rot orange at (0,3).

Input: mat[][] =
[[2, 2, 2],
[0, 2, 0]]
Output: 0
Explanation: There is no fresh orange.

Dijkstra Algorithm:

Dijkstra is an SSSP (Single Source Shortest Path) algorithm used to find the shortest distance from a source node to every other node in a weighted graph.

Note: It works only for a graph that does not have negatively weighted edges.



If source = 0, then:

Vertex	Distance from Source
0	0
1	4
2	12
3	19
4	21
5	11
6	9
7	8
8	14