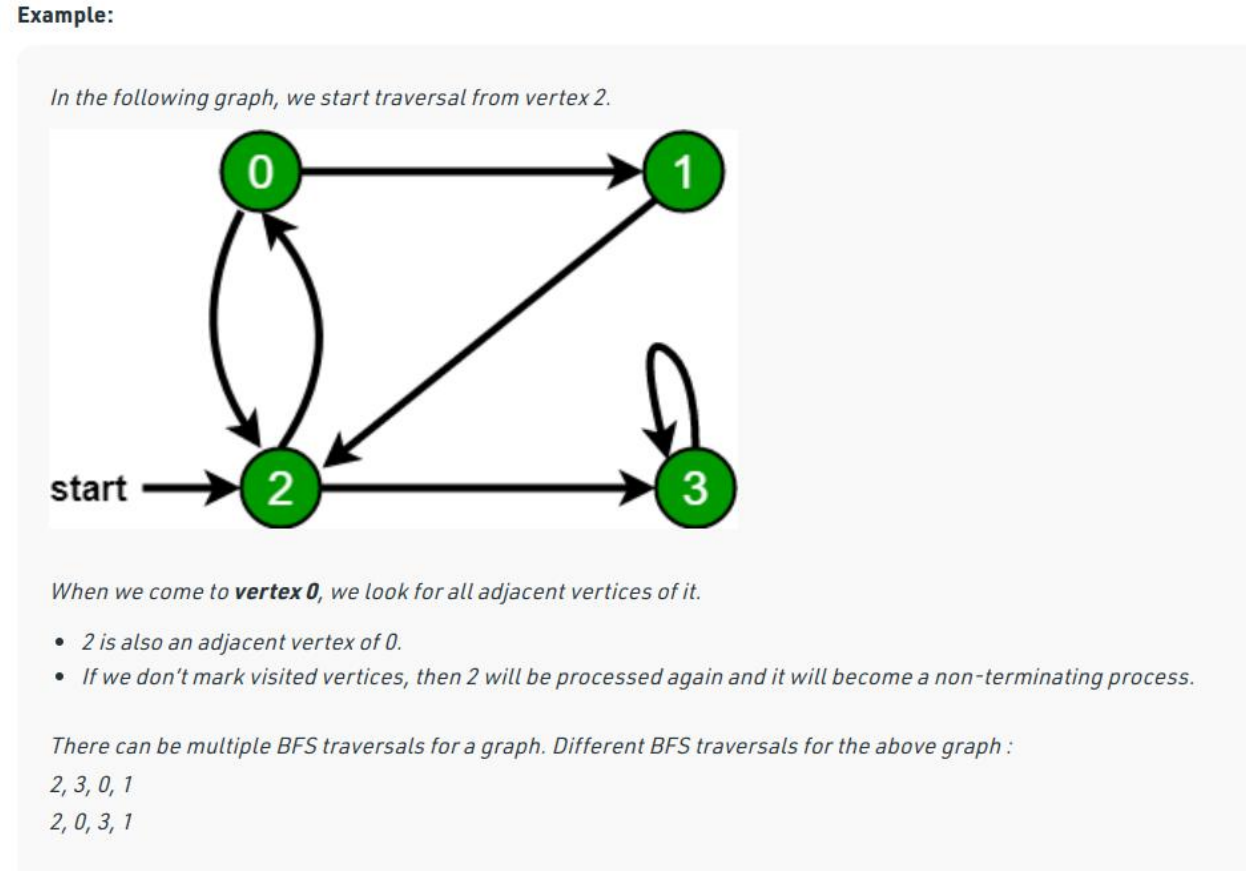
<u>Assignment-62</u>

Graph

1. Breadth First Search or BFS for a Graph.



Example:

In the following graph, we start traversal from vertex 2.

When we come to ***vertex 0****, we look for all adjacent vertices of it.*

- *2 is also an adjacent vertex of 0.*
- *If we don't mark visited vertices, then 2 will be processed again and it will become a non-terminating process.*

There can be multiple BFS traversals for a graph. Different BFS traversals for the above graph :

2, 3, 0, 1

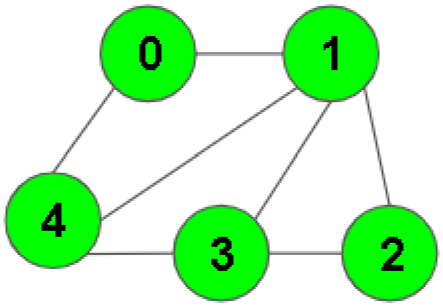
2, 0, 3, 1

Output
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1

2. Given the adjacency list of a bidirectional graph. Your task is to copy/clone the adjacency list for each vertex and return a new list.

Example 1:

Input:



Output:

```
0->1 2
1->4 0 3 2
2->3 1 4
3->4 1
4->0
```

3. Given a binary 2D matrix, find the number of islands. A group of connected 1s forms an island. For example, the below matrix contains 5 islands.

Example:

```
Input: mat[][] = {{1, 1, 0, 0, 0},
                  {0, 1, 0, 0, 1},
                  {1, 0, 0, 1, 1},
                 {0, 0, 0, 0, 0},
                {1, 0, 1, 0, 0}}
```

Output: 5

4. Given an undirected graph, The task is to check if there is a cycle in the given graph.

Example:

Input: N = 4, E = 4

Detect cycle in an undirected graph 1

Output: Yes

Explanation: The diagram clearly shows a cycle 0 to 2 to 1 to 0

Input: N = 4, E = 3 , 0 1, 1 2, 2 3

Detect cycle in an undirected graph 2

Output: No

Explanation: There is no cycle in the given graph

5. Find whether it is possible to finish all tasks or not from given dependencies

Examples:

Input: 2, [[1, 0]]

Output: true

Explanation: There are a total of 2 tasks to pick. To pick task 1 you should have finished task 0. So it is possible.

Input: 2, [[1, 0], [0, 1]]

Output: false

Explanation: There are a total of 2 tasks to pick. To pick task 1 you should have finished task 0, and to pick task 0 you should also have finished task 1. So it is impossible.

Input: 3, [[1, 0], [2, 1], [3, 2]]

Output: true

Explanation: There are a total of 3 tasks to pick. To pick tasks 1 you should have finished task 0, and to pick task 2 you should have finished task 1 and to pick task 3 you should have finished task 2. So it is possible.

6. Find the ordering of tasks from given dependencies
There are a total of n tasks you have to pick, labeled from 0 to n-1. Some tasks may have prerequisites tasks, for example to pick task 0 you have to first finish tasks 1, which is expressed as a pair: [0, 1] Given the total number of tasks and a list of prerequisite pairs, return the ordering of tasks you should pick to finish all tasks. There may be multiple correct orders, you just need to return one of them. If it is impossible to finish all tasks, return an empty array. Examples:

Input: 2, [[1, 0]]
Output: [0, 1]
Explanation: There are a total of 2 tasks to pick. To pick task 1 you should have finished task 0. So the correct task order is [0, 1] .
Input: 4, [[1, 0], [2, 0], [3, 1], [3, 2]]
Output: [0, 1, 2, 3] or [0, 2, 1, 3]
Explanation: There are a total of 4 tasks to pick. To pick task 3 you should have finished both tasks 1 and 2. Both tasks 1 and 2 should be pick after you finished task 0. So one correct task order is [0, 1, 2, 3]. Another correct ordering is [0, 2, 1, 3].

7. Given a snake and ladder board, find the minimum number of dice throws required to reach the destination or last cell from the source or 1st cell. Basically, the player has total control over the outcome of the dice throw and wants to find out the minimum number of throws required to reach the last cell. If the player reaches a cell which is the base of a ladder, the player has to climb up that ladder and if reaches a cell is the mouth of the snake, and has to go down to the tail of the snake without a dice throw.

8. Count the total number of ways or paths that exist between two vertices in a directed graph. These paths don't contain a cycle, the simple enough reason is that a cycle contains an infinite number of paths and hence they create a problem.

Input: Count paths between A and E
Output: Total paths between A and E are 4
Explanation: The 4 paths between A and E are:

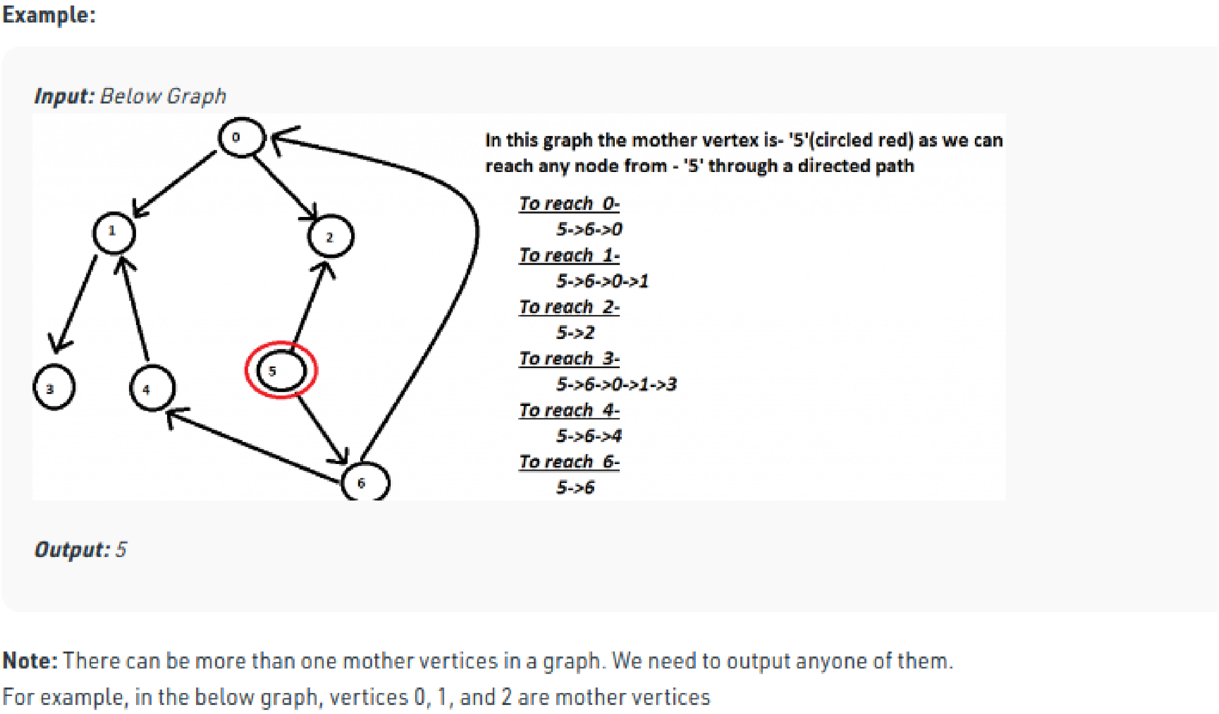
A -> E
A -> B -> E
A -> C -> E
A -> B -> D -> C -> E

Input: Count paths between A and C
Output: Total paths between A and C are 2
Explanation: The 2 paths between A and C are:

A -> C
A -> B -> D -> C

9. Write a function to find a mother vertex in the graph.
What is a Mother Vertex?

A mother vertex in a graph G = (V, E) is a vertex v such that all other vertices in G can be reached by a path from v



Example:

Input: *Below Graph*

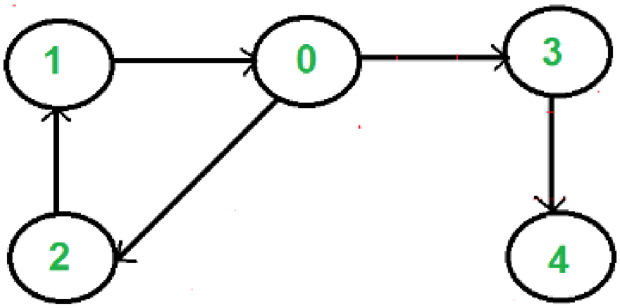
In this graph the mother vertex is- '5'(circled red) as we can reach any node from - '5' through a directed path

To reach 0-
5->6->0
To reach 1-
5->6->0->1
To reach 2-
5->2
To reach 3-
5->6->0->1->3
To reach 4-
5->6->4
To reach 6-
5->6

Output: *5*

Note: There can be more than one mother vertices in a graph. We need to output anyone of them.

For example, in the below graph, vertices 0, 1, and 2 are mother vertices



Mother Vertices : 0, 1 and 2

10. Given n and k, construct a palindrome of size n using a binary number of size k repeating itself to wrap into the palindrome. The palindrome must always begin with 1 and contains maximum number of zeros.

Examples :

Input : n = 5, k = 3
Output : 11011
Explanation : the 3 sized substring is
110 combined twice and trimming the extra
0 in the end to give 11011.