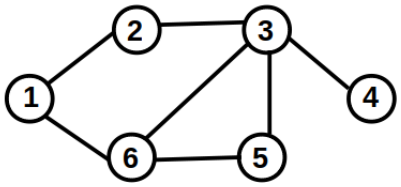
Algorithms Laboratory (CS29203)
Assignment 8: Graph Algorithms
Department of CSE, IIT Kharagpur

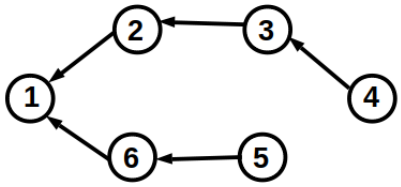
10th November 2022

Question-1 (50 points)

Consider a given number of cities in a country which are connected by water pipes. You can consider this as a connected and undirected graph, where the nodes represent the cities and the edges represent the water pipes. Let the cities are labelled as integer numbers from 1 to n. We are interested in designing a water drainage system among these cities. We are provided with an information that one particular city will be chosen as the *"sink"* where the water of all other cities will be drained through the water pipes. That is, basically we have to construct a directed graph from the given undirected graph such that any path in the directed graph leads to that particular *"sink"* vertex. For example consider the following connected and undirected graph where the cities are numbered from 1 to 6. Consider the vertex lavelled '1' as the *"sink"* vertex.



Then the following graph converts the above graph to a directed graph satisfying the requirement:



Your task is to write a program to solve the problem (*Hint: use BFS/DFS*). The complexity of the algorithm should not be worse than $O(V + E)$ where V and E are the number of vertices and number of edges in the graph, respectively. Note that the solution may not be unique. You may hard code the graph in your implementation like the following (you may use other convention if you want, this is just for reference):

```
struct Graph* graph = createGraph(6);
addEdge(graph, 1, 2); addEdge(graph, 2, 3); addEdge(graph, 3, 4); addEdge(graph, 6, 3);
addEdge(graph, 1, 6); addEdge(graph, 6, 5); addEdge(graph, 5, 3);
```

Example: (considering the same input as before)

Output:

```
Path from vertex 2 to sink is: 2->1
Path from vertex 3 to sink is: 3->2->1
Path from vertex 4 to sink is: 4->3->2->1
Path from vertex 5 to sink is: 5->6->1
Path from vertex 6 to sink is: 6->1
```

Question-2 (50 points)

You have n gardens, labeled from 1 to n, and an array *'paths'* where $paths[i] = [xi, yi]$ describes a bidirectional path between garden xi to garden yi. In each garden, you want to plant **one of the** 4 **colors** of flowers. All gardens have **at most 3** paths coming into or leaving it. Your task is to choose a flower color for each garden such that, for any two gardens connected by a path, they have different colors of flowers.

Return any such a choice as an array *mychoice*, where $mychoice[i]$ is the color of flower planted in the $(i + 1)$-th garden (considering that array index starts from 0). The flower colors are denoted $1, 2, 3$, or 4. It is guaranteed an answer exists.

For example, consider $n = 3$, and $paths = [[1, 2], [2, 3], [3, 1]]$. Then $[1, 2, 3]$ is a valid output due to the following reason:

- Gardens 1 and 2 have different colors.
- Gardens 2 and 3 have different colors.
- Gardens 3 and 1 have different colors.

Other valid answers include $[1, 2, 4]$, $[1, 4, 2]$, and $[3, 2, 1]$.

Hint: First, build the graph with the given pairs. Set the node index to start from 0, while the color index starts from 1 (0 represents the unvisited node). Then iterate through the node and collect the colors of its neighbors. Then, iterate through all the possible colors. There are only four possible colors at most since there are only three edges at most in each node. If the neighbor uses the color, skip it and record the color of the current position.

Example 1:

```
Input: n = 4, paths = [[1,2],[3,4]]
Output: [1,2,1,2]
```

Example 2:

```
Input: n = 4, paths = [[1,2],[2,3],[3,4],[4,1],[1,3],[2,4]]
Output: [1,2,3,4]
```