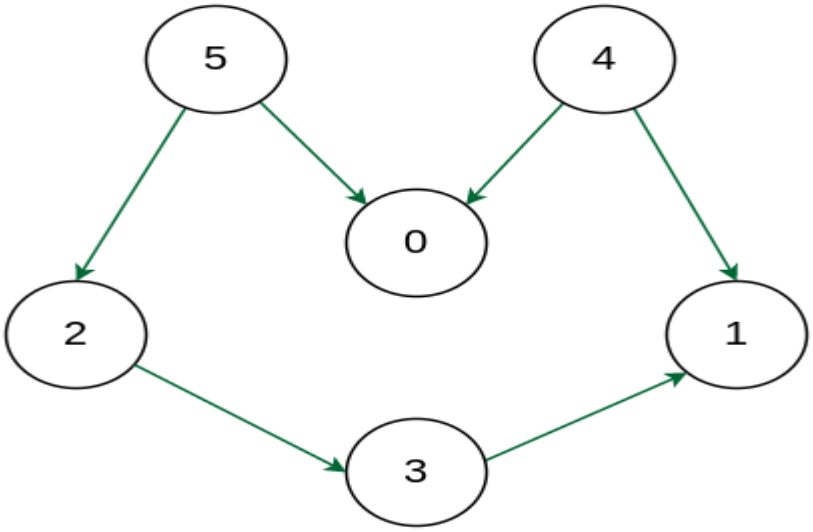
Unit 5 Topological Sorting

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.
Note: Topological Sorting for a graph is not possible if the graph is not a **DAG**.
Example:
Input: *Graph :*



Output: *5 4 2 3 1 0*

Explanation: *The first vertex in topological sorting is always a vertex with an in-degree of 0 (a vertex with no incoming edges). A topological sorting of the following graph is "5 4 2 3 1 0". There can be more than one topological sorting for a graph. Another topological sorting of the following graph is "4 5 2 3 1 0".*

Topological Sorting vs Depth First Traversal (DFS):

In DFS, we print a vertex and then recursively call DFS for its adjacent vertices. In topological sorting, we need to print a vertex before its adjacent vertices.

For example, In the above given graph, the vertex '5' should be printed before vertex '0', but unlike DFS, the vertex '4' should also be printed before vertex '0'. So Topological sorting is different from DFS. For example, a DFS of the shown graph is "5 2 3 1 0 4", but it is not a topological sorting.

Topological order may not be Unique:

Topological sorting *is a dependency problem in which completion of one task depends upon the completion of several other tasks whose order can vary*

Algorithm for Topological Sorting using DFS:

Here's a step-by-step algorithm for topological sorting using Depth First Search (DFS):

- Create a graph with **n** vertices and **m**-directed edges.
- Initialize a stack and a visited array of size **n**.
- For each unvisited vertex in the graph, do the following:
    - Call the DFS function with the vertex as the parameter.
    - In the DFS function, mark the vertex as visited and recursively call the DFS function for all unvisited neighbors of the vertex.
    - Once all the neighbors have been visited, push the vertex onto the stack.
- After all, vertices have been visited, pop elements from the stack and append them to the output list until the stack is empty.
- The resulting list is the topologically sorted order of the graph.

Illustration Topological Sorting Algorithm:

Below image is an illustration of the above approach