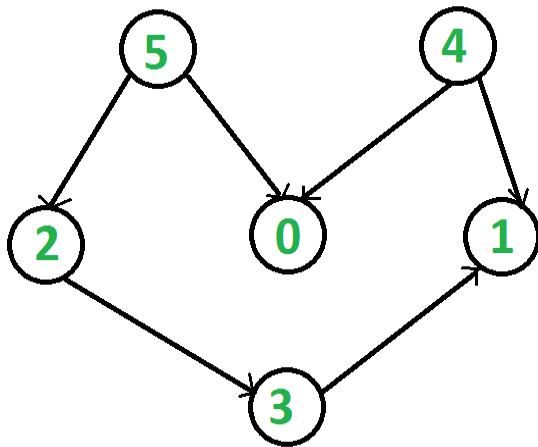


# Topological Sorting

Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge  $uv$ , vertex  $u$  comes before  $v$  in the ordering. Topological Sorting for a graph is not possible if the graph is not a DAG.

For example, 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. For example, another topological sorting of the following graph is “4 5 2 3 1 0”. The first vertex in topological sorting is always a vertex with in-degree as 0 (a vertex with no in-coming edges).



We can modify DFS (depth first search) to find Topological Sorting of a graph. In DFS, we start from a vertex, we first print it and then recursively call DFS for its adjacent vertices. In topological sorting, we use a temporary stack. We don't print the vertex immediately, we first recursively call topological sorting for all its adjacent vertices, then push it to a stack. Finally, print contents of stack. Note that a vertex is pushed to stack only when all of its adjacent vertices (and their adjacent vertices and so on) are already in stack. So when printing vertices from the stack, last inserted vertex will be printed first. Thus any vertex will be printed before its adjacent vertices.

**Time Complexity:** The above algorithm is simply DFS with an extra stack. So time complexity is same as DFS which is  $O(V+E)$ .

Implementation of this algorithm can be found at the link provided below.

<http://www.geeksforgeeks.org/topological-sorting/>