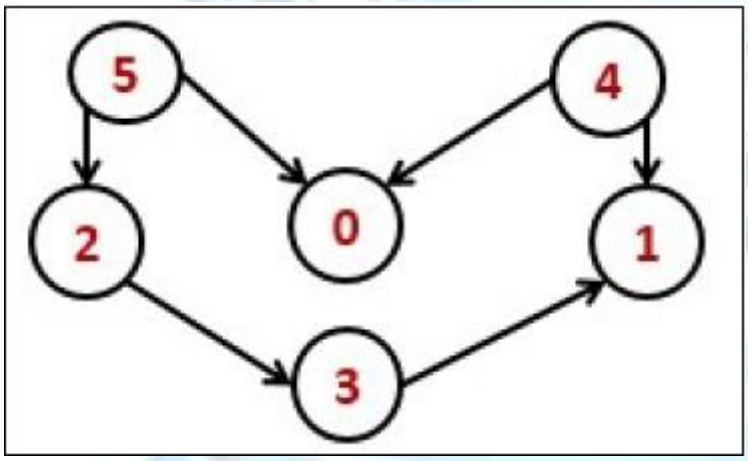
Topological Sorting

Topological sorting is a way of arranging the nodes of a Directed Acyclic Graph (DAG) in a line, making sure that for every directed edge from u to v, node u comes before v. If the graph has cycles, topological sorting isn't possible.

In topological sorting, if theres an edge from u to v, u should come before v in the order. Its different from DFS, where we visit a node and then explore its neighbors. In topological sorting, a node should be printed before any of its connected nodes.

Now, lets understand Directed Acyclic Graphs (DAGs). A directed graph has edges that go from one node to another in a specific direction. It is called acyclic if there are no cyclesmeaning theres no way to start from a node and return to it by following edges.

Topological sorting only works for DAGs. If theres a cycle in the graph, we cant arrange the nodes in a proper order because some nodes will keep pointing back, making a valid linear order impossible.



Topological Sort Algorithm

Topological sorting can be done using Depth First Search (DFS) algorithm. The algorithm follows the following steps:

Algorithm

Following are the steps to perform topological sorting using DFS:

- Call the DFS function for a vertex u.
- In the DFS function, mark the vertex u as visited.
- For every adjacent vertex v of u, if v is not visited, then call the DFS function for vertex v.
- After visiting all the adjacent vertices of vertex u, add the vertex u to the topological order list.

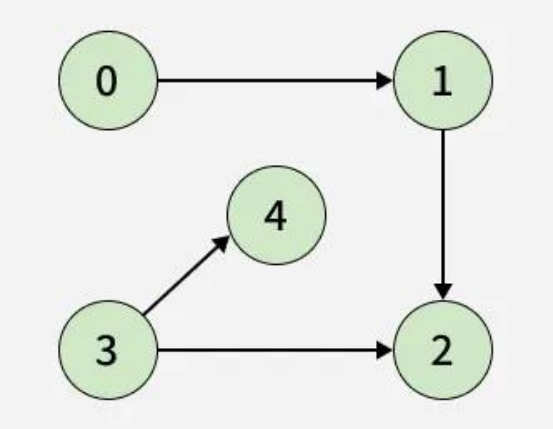
After performing the above steps for all the vertices, the topological order list will have the vertices in topological order.

Applications of Topological Sort

Topological sorting has many applications in various fields. Some of the applications of topological sort are:

- Task Scheduling: Used to schedule tasks based on dependencies. Each task is a node, and dependencies are edges. This ensures tasks are executed in the correct order.
- Course Prerequisites: Helps in deciding the order of taking courses. Courses are nodes, and prerequisites are edges, ensuring prerequisites are completed first.
- Build Systems: Used in software build processes to determine the correct sequence for compiling files based on dependencies.
- Network Routing: Helps in routing packets through a network by determining the correct order of processing based on connections between routers.

Example



Output: *[0, 3, 1, 4, 2]*

Explanation: Since vertices 0 and 3 do not have any incoming edges from any other vertex, they appear first in the topological ordering. Next, vertex 1 depends only on vertex 0, so it comes after 0. Similarly, vertex 4 depends only on vertex 3, placing it after 3. Finally, vertex 2 depends on both vertices 1 and 3, so it appears after both of them in the ordering.