Topological Sort

Topological sorting only exists in Directed Acyclic Graph (DAG). If the nodes of a graph are connected through directed edges and the graph does not contain a cycle, it is called a directed acyclic graph(DAG).

The topological sorting of a directed acyclic graph is nothing but the linear ordering of vertices such that if there is an edge between node u and v(u -> v), node u appears before v in that ordering.

Topological Sort

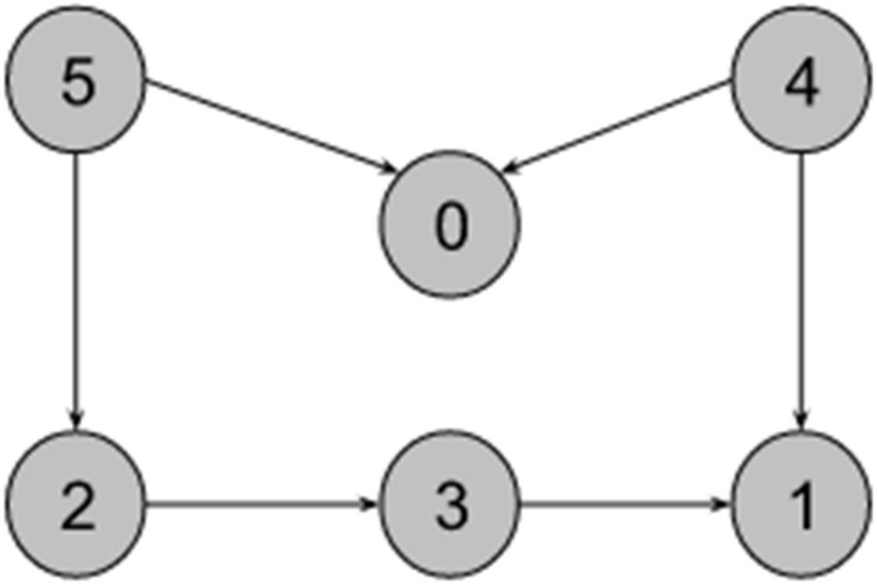
Why topological sort only exists in DAG:

Case 1 (If the edges are undirected): If there is an undirected edge between node u and v, it signifies that there is an edge from node u to v(u -> v) as well as there is an edge from node v to u(v -> u). But according to the definition of topological sorting, it is practically impossible to write such ordering where u appears before v and v appears before u simultaneously. So, it is only possible for directed edges.

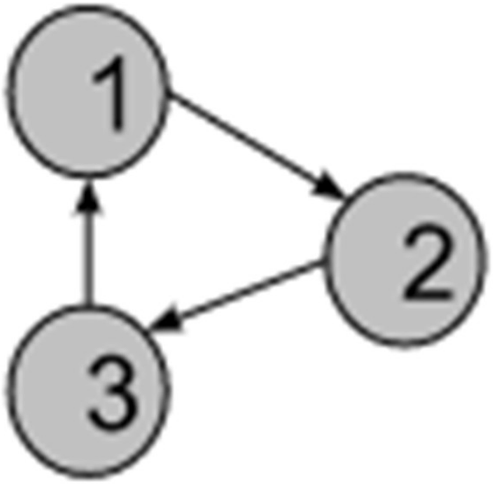
Case 2(If the directed graph contains a cycle): The following directed graph contains a cycle: If we try to get topological sorting of this cyclic graph, for edge 1->2, node 1 must appear before 2, for edge 2->3, node 2 must appear before 3, and for edge 3->1, node 3 must appear before 1 in the linear ordering. But such ordering is not possible as there exists a cyclic dependency in the graph. Thereby, topological sorting is only possible for a directed acyclic graph.

Topological Sort

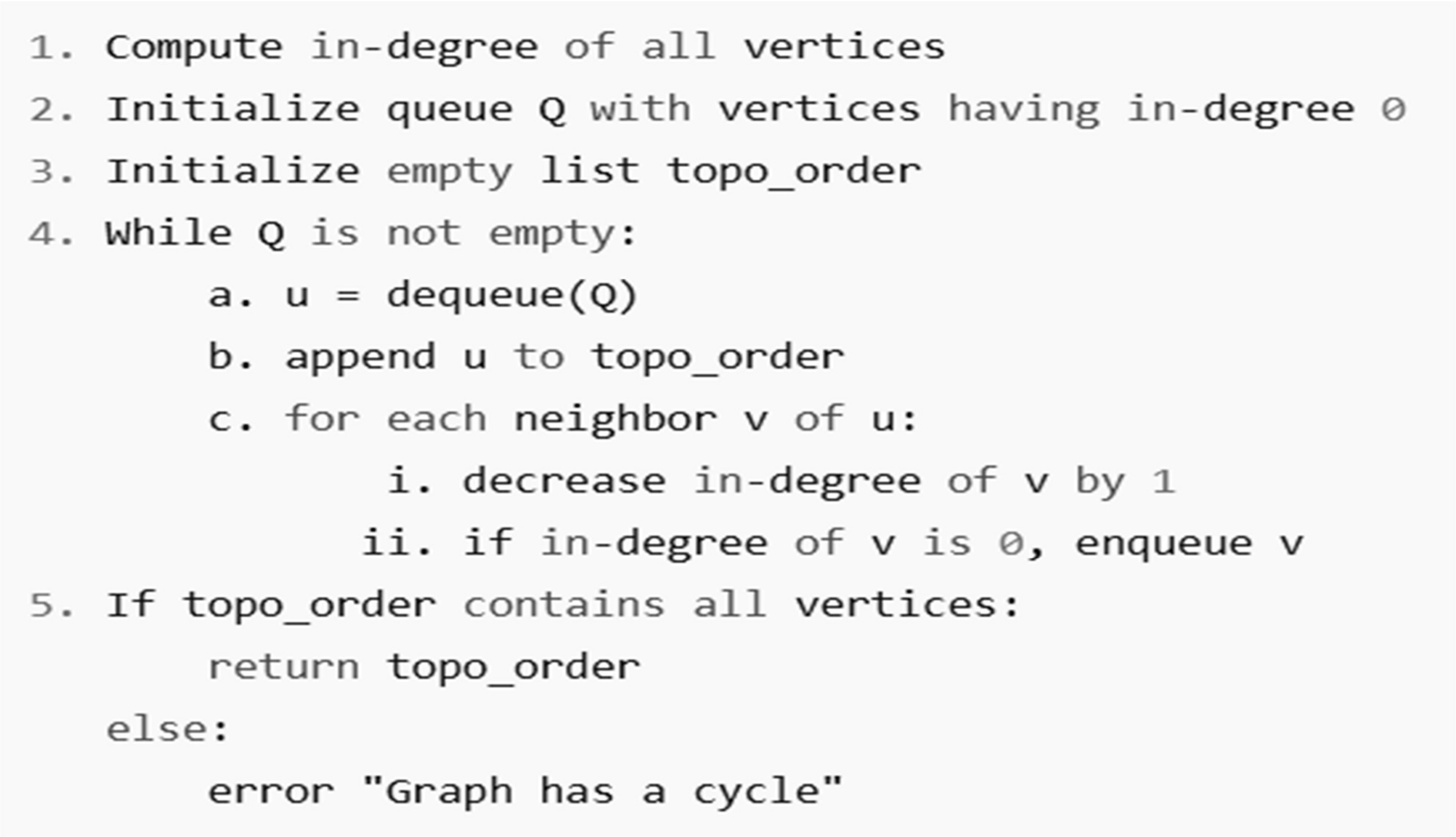
Directed Acyclic Graph



Directed Cyclic Graph

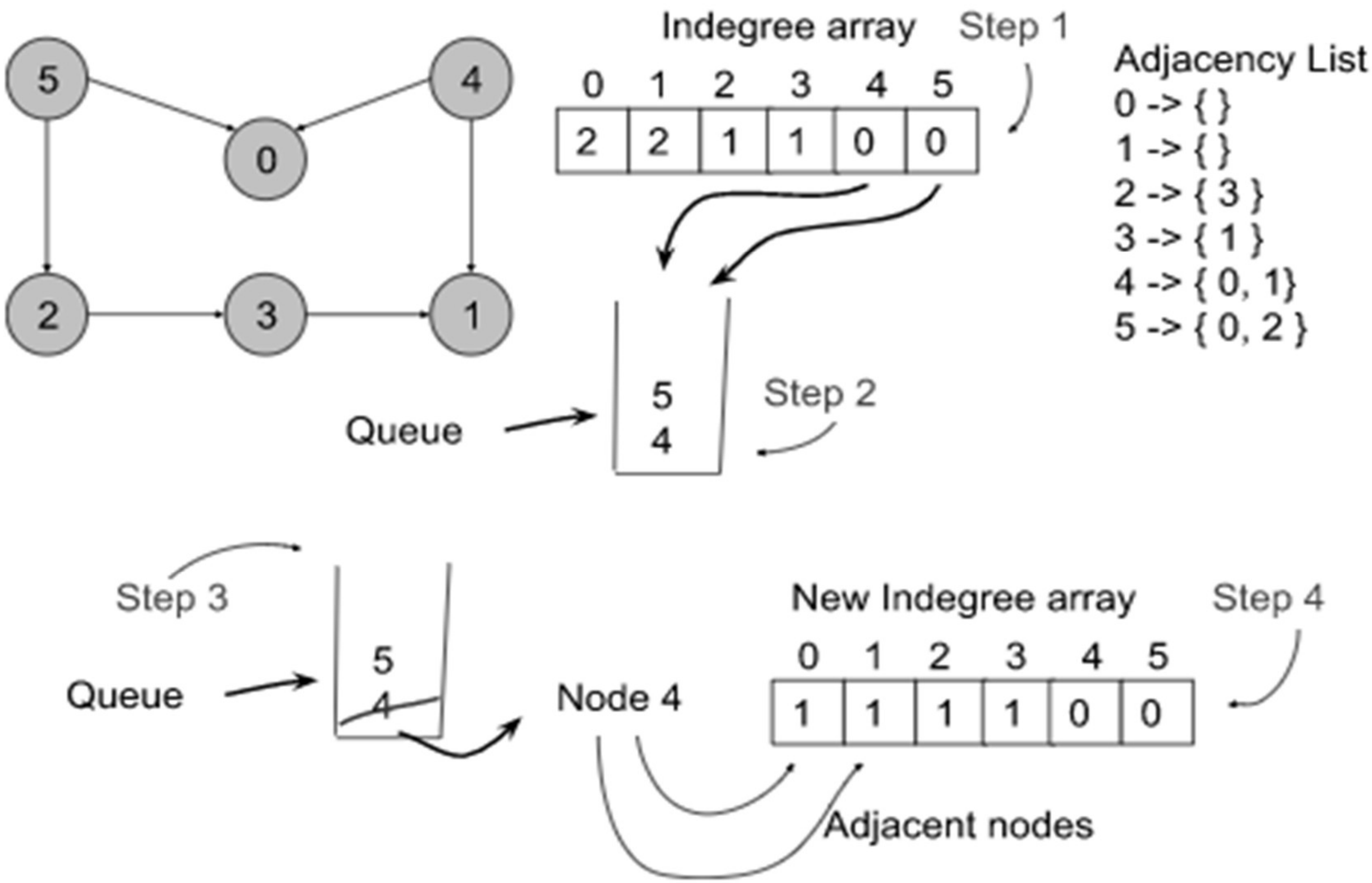


Kahn’s Algorithm



```
1. Compute in-degree of all vertices
2. Initialize queue Q with vertices having in-degree 0
3. Initialize empty list topo_order
4. While Q is not empty:
       a. u = dequeue(Q)
       b. append u to topo_order
       c. for each neighbor v of u:
              i. decrease in-degree of v by 1
             ii. if in-degree of v is 0, enqueue v
5. If topo_order contains all vertices:
       return topo_order
   else:
       error "Graph has a cycle"
```

Explanation



Indegree array

0	1	2	3	4	5
2	2	1	1	0	0

Adjacency List
0 -> { }
1 -> { }
2 -> { 3 }
3 -> { 1 }
4 -> { 0, 1}
5 -> { 0, 2 }

New Indegree array

0	1	2	3	4	5
1	1	1	1	0	0

Thank You