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.A topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a directed acyclic graph (DAG). Any DAG has at least one topological ordering.

Algorithm using Depth First Search

Here we are implementing topological sort using Depth First Search.

- **Step 1**: Create a temporary stack.
- **Step 2**: Recursively call topological sorting for all its adjacent vertices, then push it to the stack (when all adjacent vertices are on stack). Note this step is same as Depth First Search in a recursive way.
- **Step 3**: Atlast, print contents of stack.

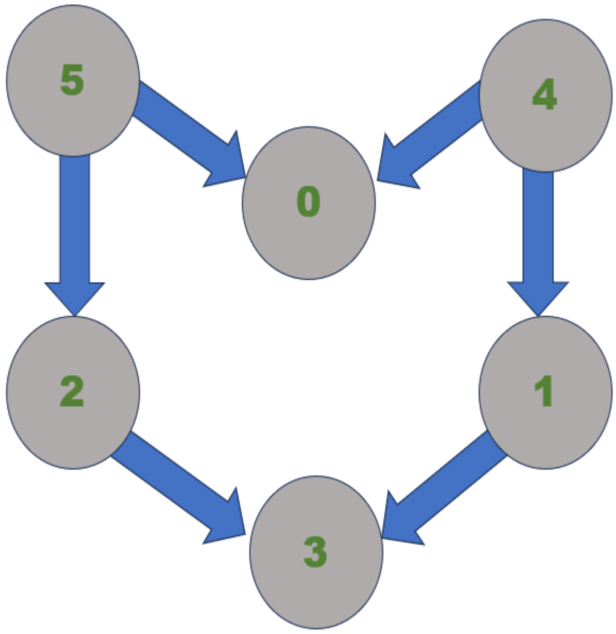
Note: A vertex is pushed to stack only when all of its adjacent vertices (and their adjacent vertices and so on) are already in stack

Pseudocode

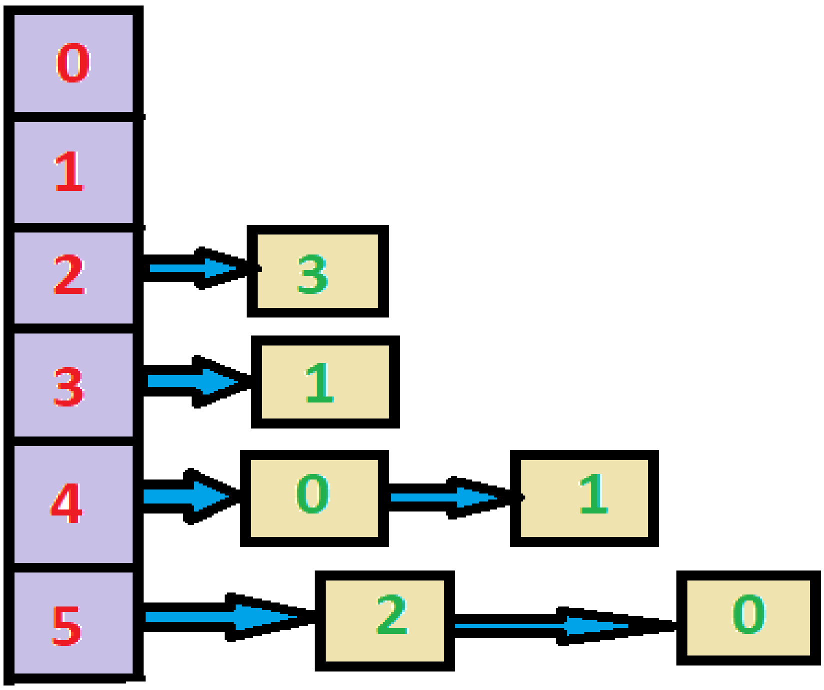
The pseudocode of topological sort is:

- **Step 1:** Create the graph by calling addEdge(a,b).
- **Step 2:** Call the topologicalSort( )
    - **Step 2.1:** Create a stack and a boolean array named as visited[ ];
    - **Step 2.2:** Mark all the vertices as not visited i.e. initialize visited[ ] with 'false' value.
    - **Step 2.3:** Call the recursive helper function topologicalSortUtil() to store Topological Sort starting from all vertices one by one.
- **Step 3:** def topologicalSortUtil(int v, bool visited[],stack<int> &Stack):
    - **Step 3.1:** Mark the current node as visited.
    - **Step 3.2:** Recur for all the vertices adjacent to this vertex.
    - **Step 3.3:** Push current vertex to stack which stores result.
- **Step 4:** Atlast after return from the utility function, print contents of stack.

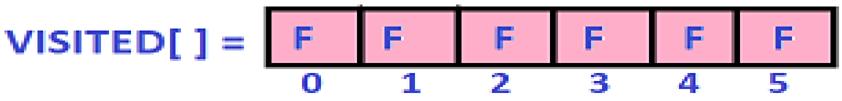
- **Example**
- Consider the following graph:



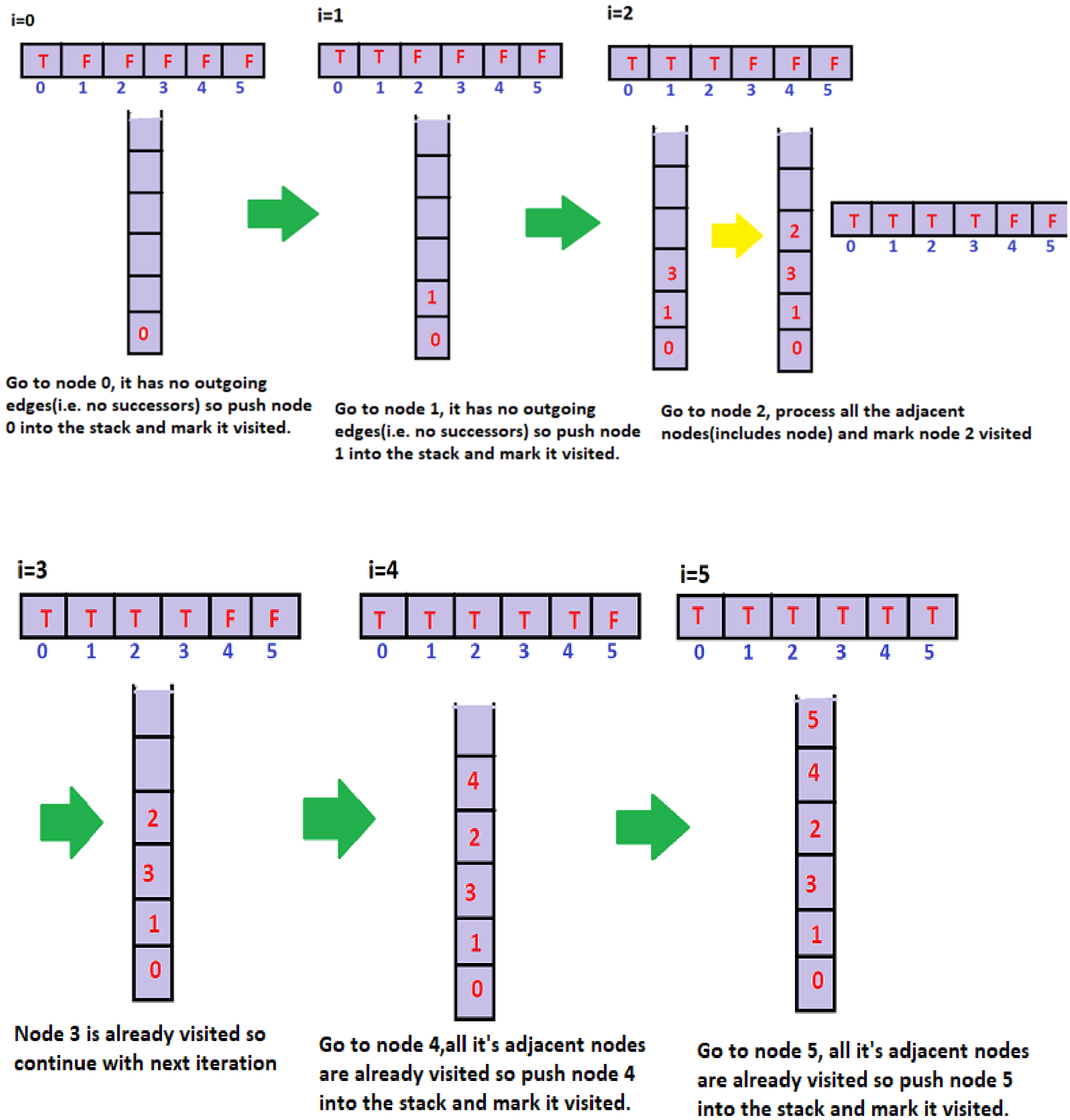
Following is the adjacency list of the given graph:



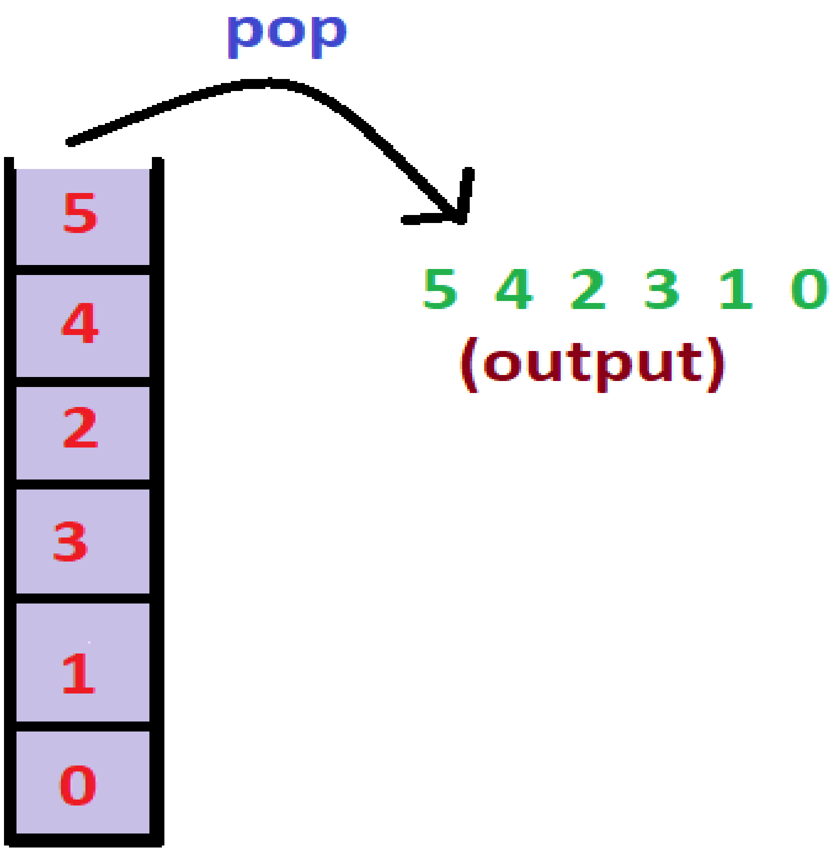
When topological() is called:



Stepwise demonstration of the stack after each iteration of the loop(topologicalSort()):



The contents of the stack are:



So the topological sorting of the above 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 above graph is “4 5 2 3 1 0”. Both of them are correct!

Complexity

Worst case time complexity:**Θ(|V|+|E|)**
Average case time complexity:**Θ(|V|+|E|)**
Best case time complexity:**Θ(|V|+|E|)**
Space complexity:**Θ(|V|)**