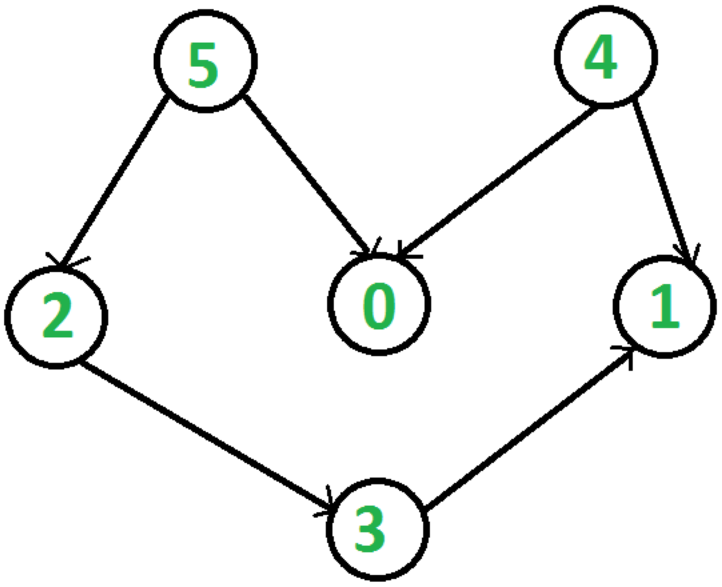
Topological Sorting



Topological Sort doesn't work on cyclic graph or on undirected graph. For checking cycles in topological sort we need some sort of visitedLocal[ ] array. Especially the edges 5->0 & 4->0 will cause the issue and we need to have visitedLocal[ ] array, a global visited[ ] say there is a cycle when there is no cycle in the graph because of the edges 5->0 & 4->0.

```
// A C++ program to print topological
// sorting of a DAG
#include <iostream>
#include <list>
#include <stack>
using namespace std;

// Class to represent a graph
class Graph {
        // No. of vertices'
        int V;

        // Pointer to an array containing adjacency listsList
        list<int>* adj;

        // A function used by topologicalSort
        void topologicalSortUtil(int v, bool visited[],stack<int>& Stack);

public:
        // Constructor
        Graph(int V);

        // function to add an edge to graph
        void addEdge(int v, int w);

        // prints a Topological Sort of
        // the complete graph
        void topologicalSort();
};

Graph::Graph(int V)
{
        this->V = V;
        adj = new list<int>[V];
}

void Graph::addEdge(int v, int w)
{
        // Add w to v’s list.
        adj[v].push_back(w);
}
```

```cpp
// A recursive function used by topologicalSort
void Graph::topologicalSortUtil(int v, bool visited[],stack<int>& Stack)
{
        // Mark the current node as visited.
        visited[v] = true;

        // Recur for all the vertices
        // adjacent to this vertex
        list<int>::iterator i;
        for (i = adj[v].begin(); i != adj[v].end(); ++i)
                if (!visited[*i])
                        topologicalSortUtil(*i, visited, Stack);

        // Push current vertex to stack
        // which stores result
        Stack.push(v);
}

// The function to do Topological Sort.
// It uses recursive topologicalSortUtil()
void Graph::topologicalSort()
{
        stack<int> Stack;

        // Mark all the vertices as not visited
        bool* visited = new bool[V];
        for (int i = 0; i < V; i++)
                visited[i] = false;

        // Call the recursive helper function
        // to store Topological
        // Sort starting from all
        // vertices one by one
        for (int i = 0; i < V; i++)
                if (visited[i] == false)
                        topologicalSortUtil(i, visited, Stack);

        // Print contents of stack
        while (Stack.empty() == false) {
                cout << Stack.top() << " ";
                Stack.pop();
        }
}

// Driver Code
int main()
{
        // Create a graph given in the above diagram
        Graph g(6);
        g.addEdge(5, 2);
        g.addEdge(5, 0);
        g.addEdge(4, 0);
        g.addEdge(4, 1);
        g.addEdge(2, 3);
        g.addEdge(3, 1);

        cout << "Following is a Topological Sort of the given "
                        "graph \n";

        // Function Call
        g.topologicalSort();

        return 0;
}
```