```
#include <iostream>
#include <vector>
#include <stack>

using namespace std;

class Graph {
  int V; // num of vertices
  vector<vector<int> > adj;

public:
  Graph(int V);
  void addEdge(int v, int w);
  void topoSortDFS(int v, vector<bool>& visited, stack<int>& Stack);
};

Graph::Graph(int V) {
  this->V = V;
  adj.resize(V);

}

void Graph::addEdge(int v, int w) {
  adj[v].push_back(w);
}

void Graph::topoSortDFS(int v, vector<bool>& visited, stack<int>&
Stack) {
  visited[v] = true;

  for (int i : adj[v]) {
    if (!visited[i]) {
      topoSortDFS(i);
    }
  }
  Stack.push(v); // storing in the reverse order
}

void Graph::topologicalSort() {
  stack<int> Stack;
  vector<bool> visited(V, false);

  for (int i = 0; i < V; i++) {
    if (!visited[i]) {
      topoSortDFS(i, visited, Stack);
    }
  }

  while (!Stack.empty()) {
    cout << Stack.top() << " ";
    Stack.pop();
  }
}
```

```
int main() {
  // Example usage:
  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 << "Topological Sort: ";
  g.topologicalSort();

  return 0;
}
```