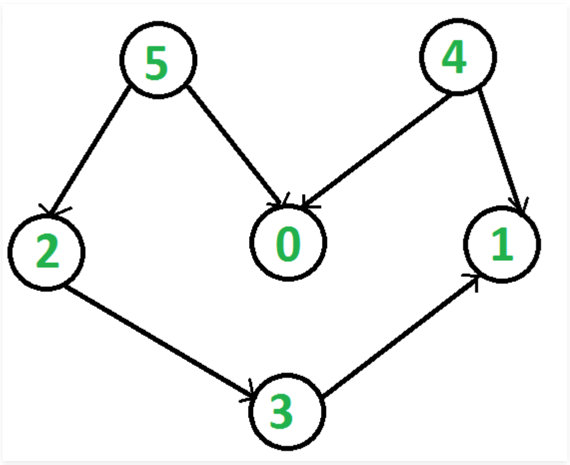
Topological Sorting

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. Topological Sorting for a graph is not possible if the graph is not a DAG.

3.1

For example, a topological sorting of the following 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 following graph is “4 5 2 3 1 0”. The first vertex in topological sorting is always a vertex with in-degree as 0 (a vertex with no in-coming edges).


$$\begin{tikzcd}
 & 5 \arrow[dl] \arrow[dr] & & 4 \arrow[dl] \arrow[dr] & \\
2 \arrow[drr] & & 0 & & 1 \\
 & & 3 \arrow[urr] & &
\end{tikzcd}$$

Topological Sorting vs Depth First Traversal (DFS): In DFS, we print a vertex and then recursively call DFS for its adjacent vertices. In topological sorting, we need to print a vertex before its adjacent vertices. For example, in the given graph, the vertex ‘5’ should be printed before vertex ‘0’, but unlike DFS, the vertex ‘4’ should also be printed before vertex ‘0’. So Topological sorting is different from DFS. For example, a DFS of the shown graph is “5 2 3 1 0 4”, but it is not a topological sorting

Recommended: Please solve it on “*PRACTICE*” first, before moving on to the solution.

Algorithm to find Topological Sorting:
We recommend to first see implementation of DFS here. We can modify DFS to find Topological Sorting of a graph. In DFS, we start from a vertex, we first print it and then recursively call DFS for its adjacent vertices. In topological sorting, we use a temporary stack. We don’t print the vertex immediately, we first recursively call topological sorting for all its adjacent vertices, then push it to a stack. Finally, print contents of stack. Note that a vertex is pushed to stack only when all of its adjacent vertices (and their adjacent vertices and so on) are already in stack.

Following are C++ and Java implementations of topological sorting. Please see the code for Depth First Traversal for a disconnected Graph and note the differences between the second code given there and the below code.

C++

Java

Python

```
#Python program to print topological sorting of a
from collections import defaultdict

#Class to represent a graph
class Graph:
    def __init__(self,vertices):
        self.graph = defaultdict(list) #dictionar
        self.V = vertices #No. of vertices

    # function to add an edge to graph
    def addEdge(self,u,v):
        self.graph[u].append(v)

    # A recursive function used by topologicalSor
    def topologicalSortUtil(self,v,visited,stack)

        # Mark the current node as visited.
        visited[v] = True

        # Recur for all the vertices adjacent to
        for i in self.graph[v]:
            if visited[i] == False:
                self.topologicalSortUtil(i,visite

        # Push current vertex to stack which stor
        stack.insert(0,v)

    # The function to do Topological Sort. It use
    # topologicalSortUtil()
    def topologicalSort(self):
        # Mark all the vertices as not visited
        visited = [False]*self.V
        stack =[]

        # Call the recursive helper function to s
        # Sort starting from all vertices one by
        for i in range(self.V):
            if visited[i] == False:
                self.topologicalSortUtil(i,visite

        # Print contents of stack
        print stack

g= Graph(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);

print "Following is a Topological Sort of the giv
g.topologicalSort()
#This code is contributed by Neelam Yadav
```

Run on IDE

Output:

```
Following is a Topological Sort of the given graph
5 4 2 3 1 0
```

Time Complexity: The above algorithm is simply DFS with an extra stack. So time complexity is same as DFS which is O(V+E).

Applications:

Topological Sorting is mainly used for scheduling jobs from the given dependencies among jobs. In computer science, applications of this type arise in instruction scheduling, ordering of formula cell evaluation when recomputing formula values in spreadsheets, logic synthesis, determining the order of compilation tasks to perform in makefiles, data serialization, and resolving symbol dependencies in linkers [2].

Related Articles:

Kahn's algorithm for Topological Sorting : Another O(V + E) algorithm.
All Topological Sorts of a Directed Acyclic Graph

References:

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/topoSort.htm
http://en.wikipedia.org/wiki/Topological_sorting