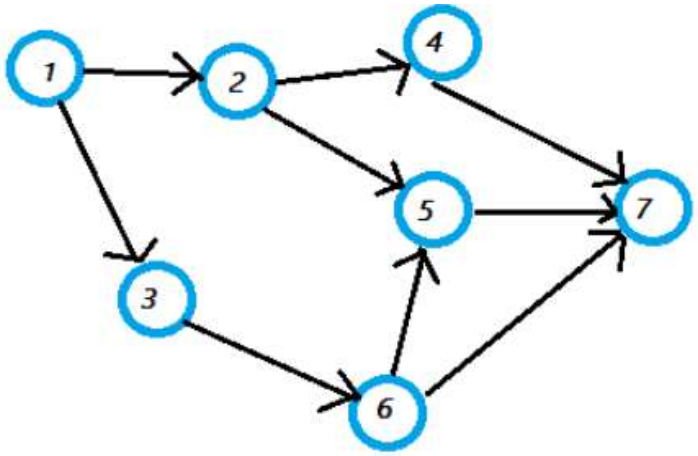
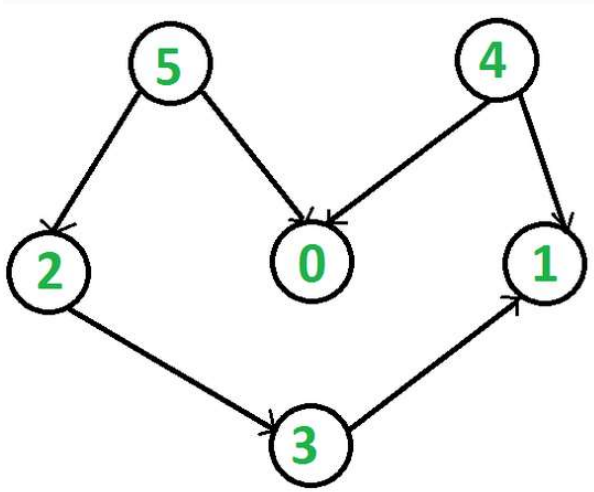
UNIT 2: Decrease and Conquer

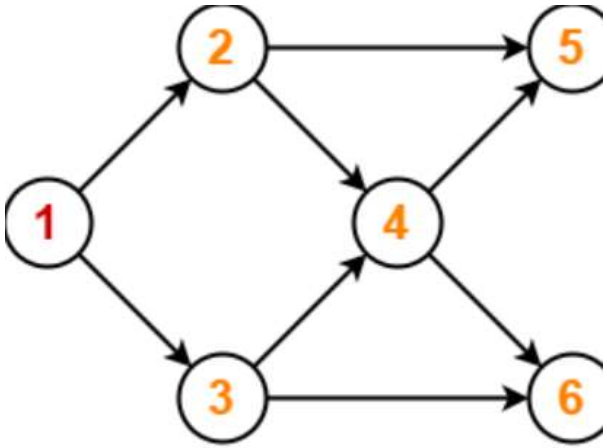
Topological sorting

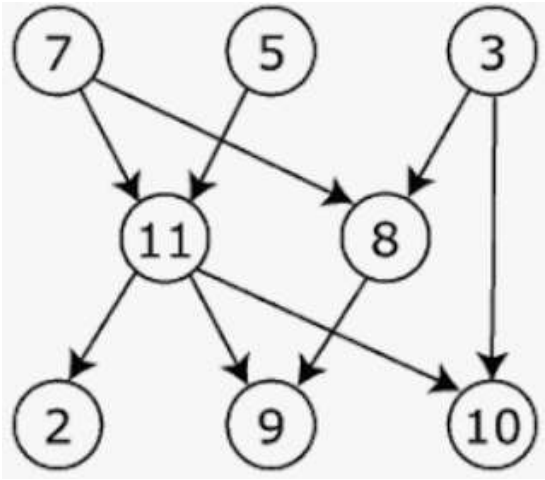




Directed Acyclic Graph (DAG)

A directed acyclic graph is a directed graph that has **no cycles**





Topological sorting

First studied in the early 1960s in the context of the **PERT (Program Evaluation and Review Technique)** for scheduling in project management

Topological sorting

- linear ordering of its vertices such that for every directed edge (u,v) from vertex u to vertex v, u comes before v in the ordering.
- Two algorithms to solve topological ordering problem:

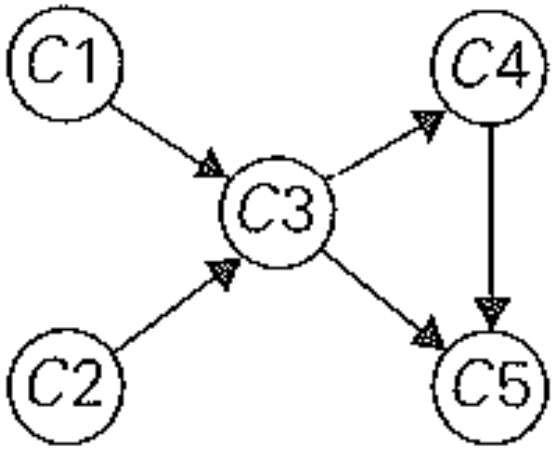
  Depth First Search

  Source Removal method

NOTE:

For topological sorting to be possible, a digraph must be a DAG

Directed Acyclic Graph



Topological ordering:

C1, C2, C3, C4, C5

C2, C1, C3, C4, C5

Solving topological ordering using DFS

- perform a DFS traversal on the DAG and note the order in which vertices become dead ends (i.e., are popped off the traversal stack).

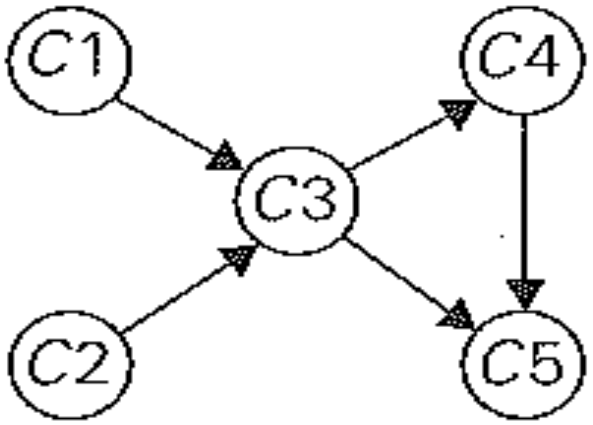
- **Reversing popping order yields a solution to the topological sorting problem.**

Why does DFS algorithm work in solving topological ordering problem?

- When a vertex v is popped off a DFS stack, no vertex u with an edge from u to v can be among the vertices popped off before v.

  (Otherwise, (u, v) would have been a back edge.)

- Hence, any such vertex u will be listed after v in the popped-off order list, and before v in the reversed list.



(a) Digraph

$C5_1$
$C4_2$
$C3_3$
$C1_4$ $C2_5$

(b) DFS traversal stack

The popping-off order:

*C*5, *C*4, *C*3, *C*1, *C*2

The topologically sorted list:

(c) Solution to the problem.

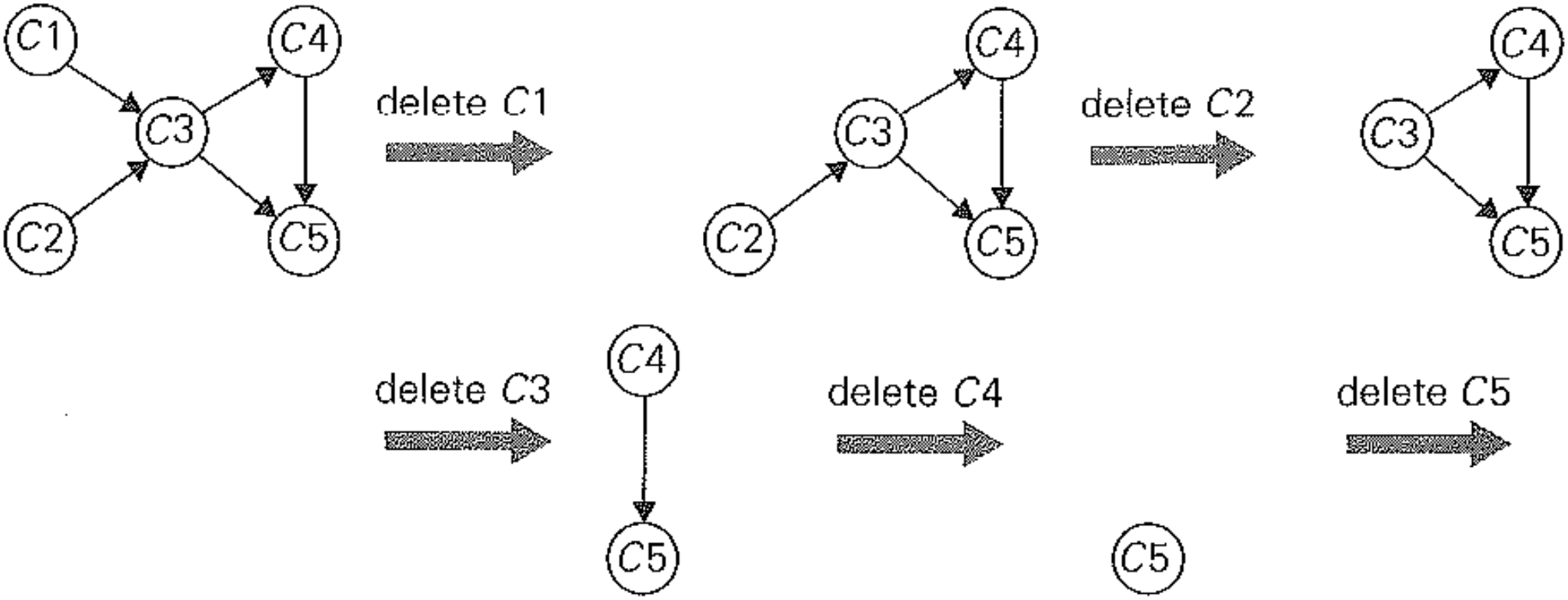
Solving topological ordering using "source removal method"

- direct implementation of the decrease (by one)-and-conquer technique

- Repeatedly, identify in a remaining digraph a source, which is a vertex with no incoming edges, and delete it along with all the edges outgoing from it. (If there are several sources, break the tie arbitrarily. If there is none, stop because the problem cannot be solved.

The order in which the vertices are deleted yields a solution to the topological sorting problem

source-removal algorithm for the topological sorting



The solution obtained is $C1, C2, C3, C4, C5$

NOTE

- topological sorting problem may have several alternative solutions.

- solution obtained by the source-removal algorithm may be different from the one obtained by the DFS-based algorithm.

Topological sorting: Visualization

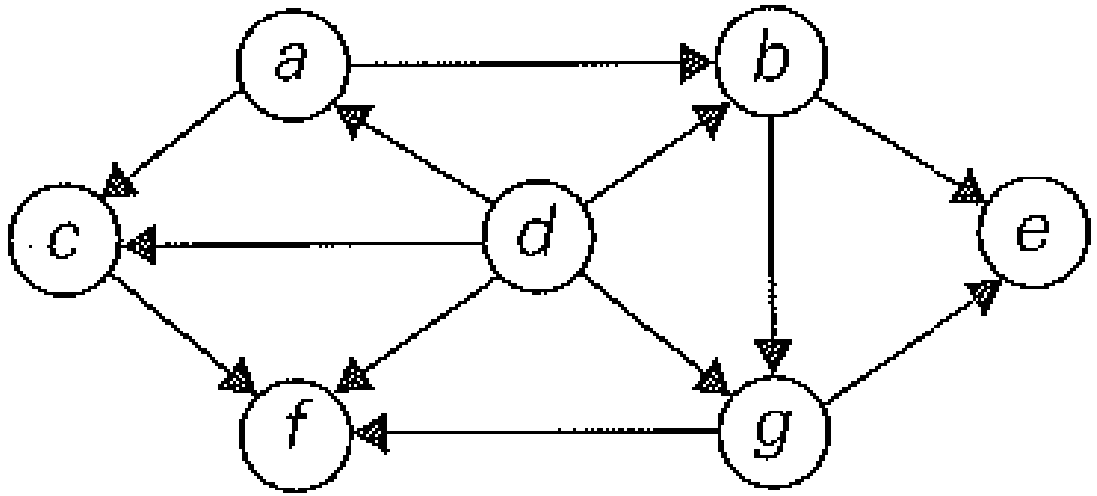
https://visualgo.net/en/dfsbfs

(select topological ordering)

https://www.cs.usfca.edu/~galles/visualization/TopoSortDFS.html

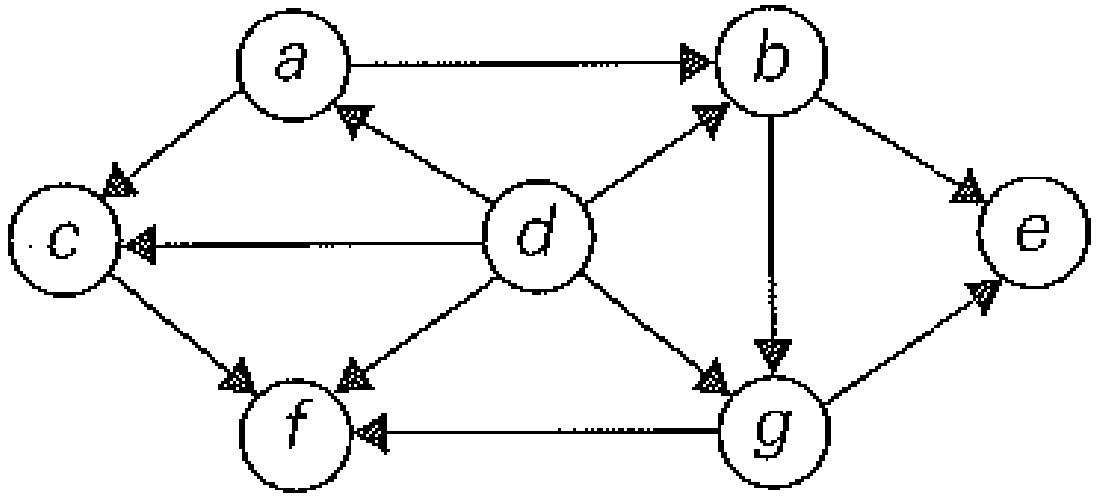
Let's check our understanding…

Apply the DFS-based algorithm to solve the topological sorting problem.



Let's check our understanding…

Apply the source-removal method to solve the topological sorting problem.



Topological sorting: Applications

- scheduling a sequence of jobs or tasks based on their dependencies.
- 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,
- resolving symbol dependencies in linkers
- to decide in which order to load tables with foreign keys in databases.

Algorithms that use Topological sorting as a building block:

- **scheduling in project management**: the vertices of a graph represent the milestones of a project, and the edges represent tasks that must be performed between one milestone and another. Topological sorting used for finding the critical path of the project, a sequence of milestones and tasks that controls the length of the overall project schedule.

- to quickly **compute shortest paths** through a weighted directed acyclic graph

Uniqueness:

- If a topological sort has the property that all pairs of consecutive vertices in the sorted order are connected by edges, then these edges form a **directed Hamiltonian path** in the DAG.
- **If a Hamiltonian path exists, the topological sort order is unique**; no other order respects the edges of the path.
- Conversely, if a topological sort does not form a Hamiltonian path, the DAG will have two or more valid topological orderings, for in this case it is always possible to form a second valid ordering by swapping two consecutive vertices that are not connected by an edge to each other

Coding Challenge

Solve

https://www.hackerearth.com/practice/algorithms/graphs/topological-sort/practice-problems/algorithm/lonelyisland-49054110/