DSE 2256 DESIGN & ANALYSIS OF ALGORITHMS

Lecture 16

Decrease-and-Conquer:

Topological Sorting

Instructors:

Dr. Savitha G,
Assistant Professor, DSCA, MIT, Manipal

Dr. Abhilash K. Pai,
Assistant Professor, DSCA, MIT, Manipal

Courtesy : www.alamy.com

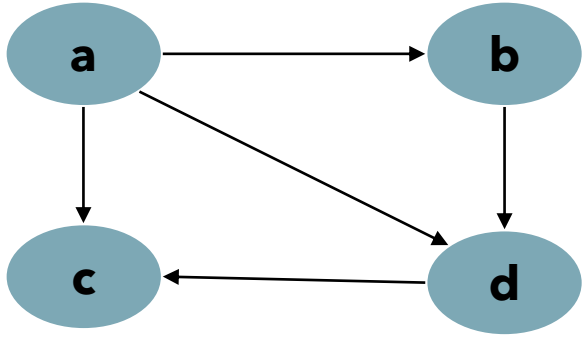
Recap of L14 & L15

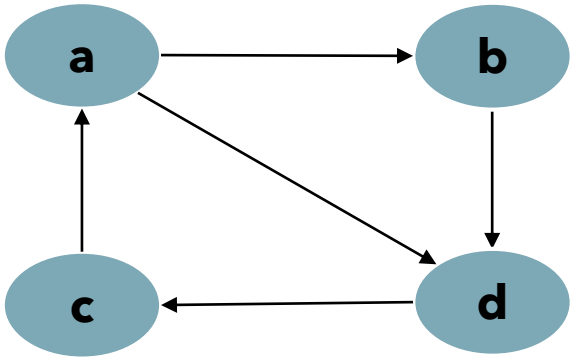
- Decrease-and-Conquer Introduction
  - Types of decrease-and conquer techniques
- Insertion Sort
- Depth First Search
- Breadth First Search

Directed Acyclic Graph (DAG)

A *dag*: a directed acyclic graph, i.e. a directed graph with no (directed) cycles

a DAG





not a DAG

- Arise in modeling many problems that involve prerequisite constraints (construction projects, document version control)
- **Vertices of a dag can be linearly ordered so that for every edge, its starting vertex is listed before its ending vertex (*topological sorting*).**
- Being a dag is also a necessary condition for topological sorting to be possible.

Topological Sorting

Vertices of a DAG can be linearly ordered so that for every edge its starting vertex is listed before its ending vertex is called topological sorting.

- **Two methods to implement: DFS based topological sorting, Source removal algorithm**

Example:

{C1,C2,C3,C4,C5} – are set of five courses a part-time student has to take in the degree program

Courses should meet the following prerequisites:

- C1 and C2 have no prerequisites
- C3 requires C1 and C2
- C4 requires C3
- C5 requires C3 and C4
- Student can take only one course per term.

In which order should the student take the courses?

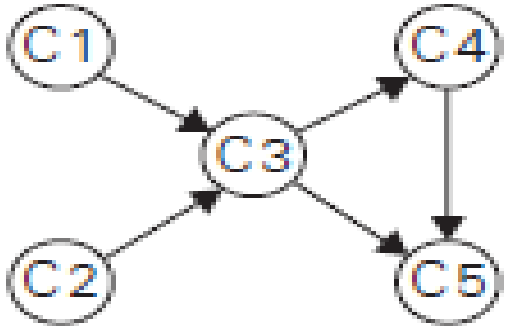
DFS-based algorithm for topological sorting

Solution:

- **Perform DFS traversal, noting the order of vertices, popped off the traversal stack**
- **Reverse order solves topological sorting problem**
- **Back edges encountered?→ NOT a dag!**

Prerequisites:

- C1 and C2 have no prerequisites
- C3 requires C1 and C2
- C4 requires C3
- C5 requires C3 and C4

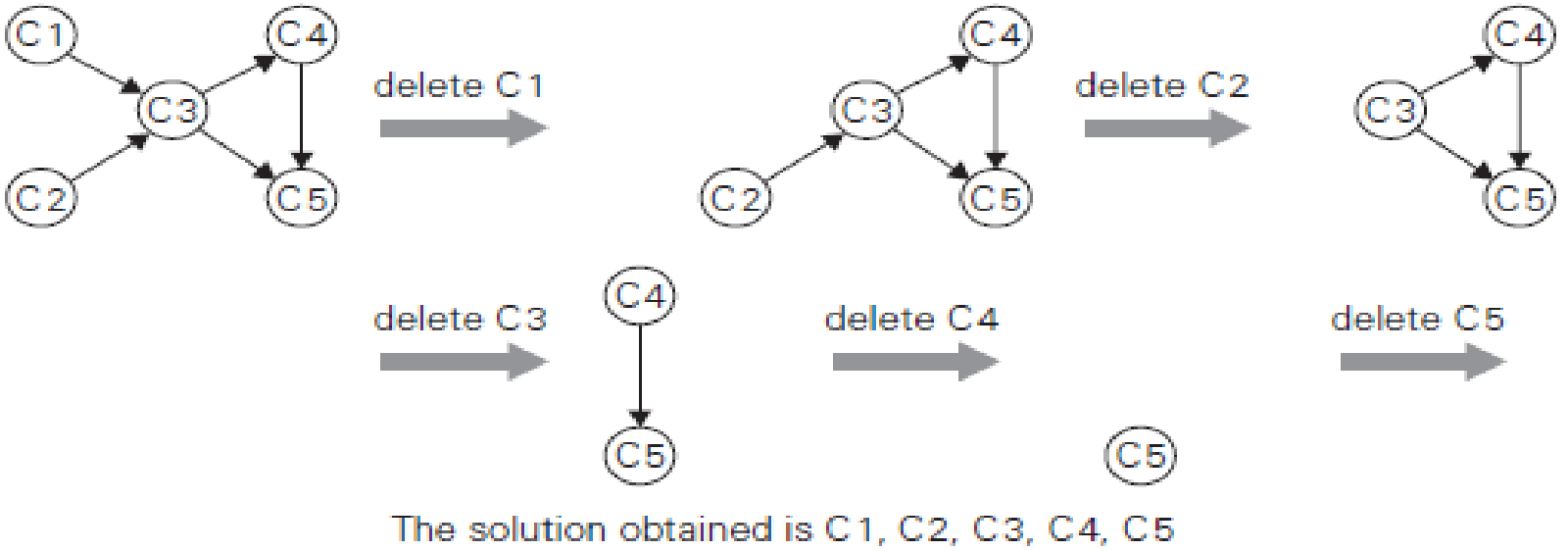


$C5_1$
$C4_2$
$C3_3$
$C1_4$ $C2_5$

The popping-off order:
C5, C4, C3, C1, C2
The topologically sorted list:

Source removal algorithm for Topological Sorting

- **Repeatedly identify and remove a *source* (a vertex with no incoming edges) and all the edges incident to it until either no vertex is left or there is no source among the remaining vertices (not a dag)**



Notes on Topological Sorting

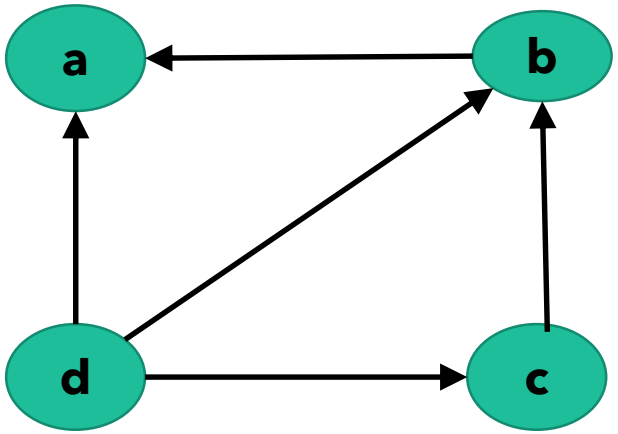
- Algorithm for directed graph or digraph
- A directed graph should be DAG (Directed Acyclic Graph)

- Topological sorting implemented by two methods
  - DFS based algorithm
  - Source removal algorithm

- Topological sorting may have several alternative solutions.
  - Solution obtained by the DFA based algorithm and source removal are different.

Problem

Apply the DFS-based algorithm and Source removal algorithm to solve the topological sorting problem for the following digraph.



Thank you!

Any queries?