DESIGN AND ANALYSIS OF ALGORITHMS

Surabhi Narayan

Department of Computer Science & Engineering

DESIGN AND ANALYSIS OF ALGORITHMS

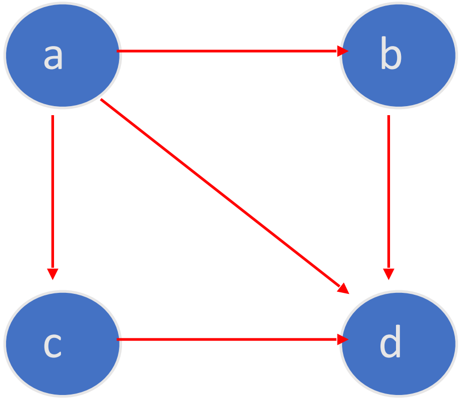
DECREASE AND CONQUER

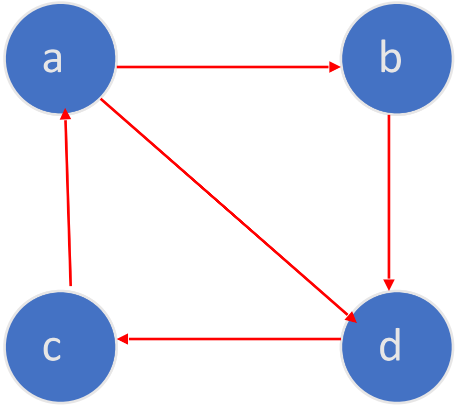
Surabhi Narayan

Department of Computer Science & Engineering

DAGs and Topological Sorting

DAG: a directed acyclic graph, i.e. a directed graph with no (directed) cycles





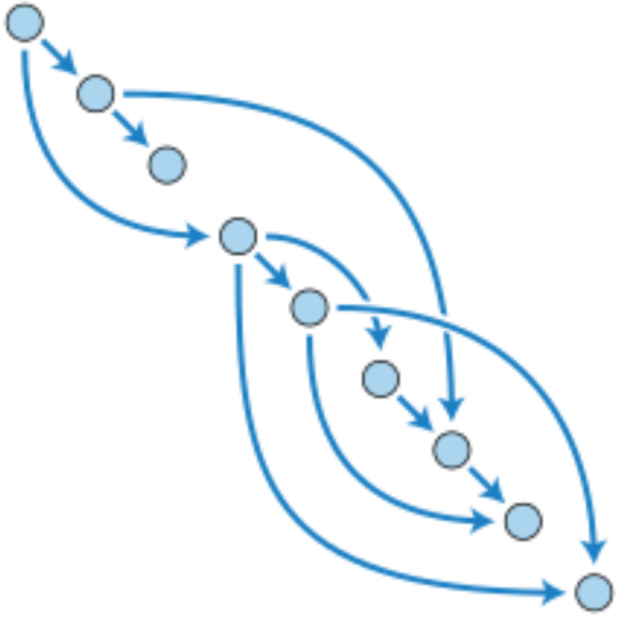
Arise in modeling many problems that involve prerequisite constraints (construction projects, document version control)

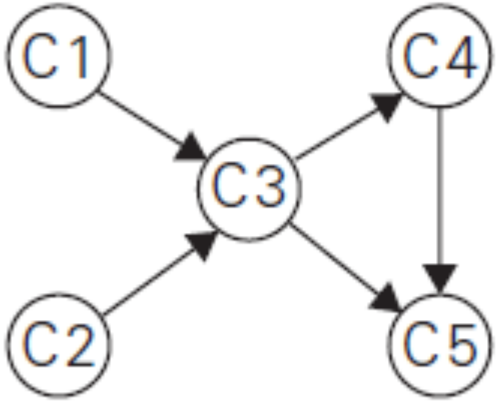
Vertices of a dag can be linearly ordered so that for every edge its starting vertex is listed before its ending vertex (*topological sorting*). Being a dag is also a necessary condition for topological sorting to be possible.

Topological Sorting

Topological Sorting: is listing vertices of a directed graph in such an order that for every edge in the graph, the vertex where the edge starts is listed before the vertex where the edge ends.

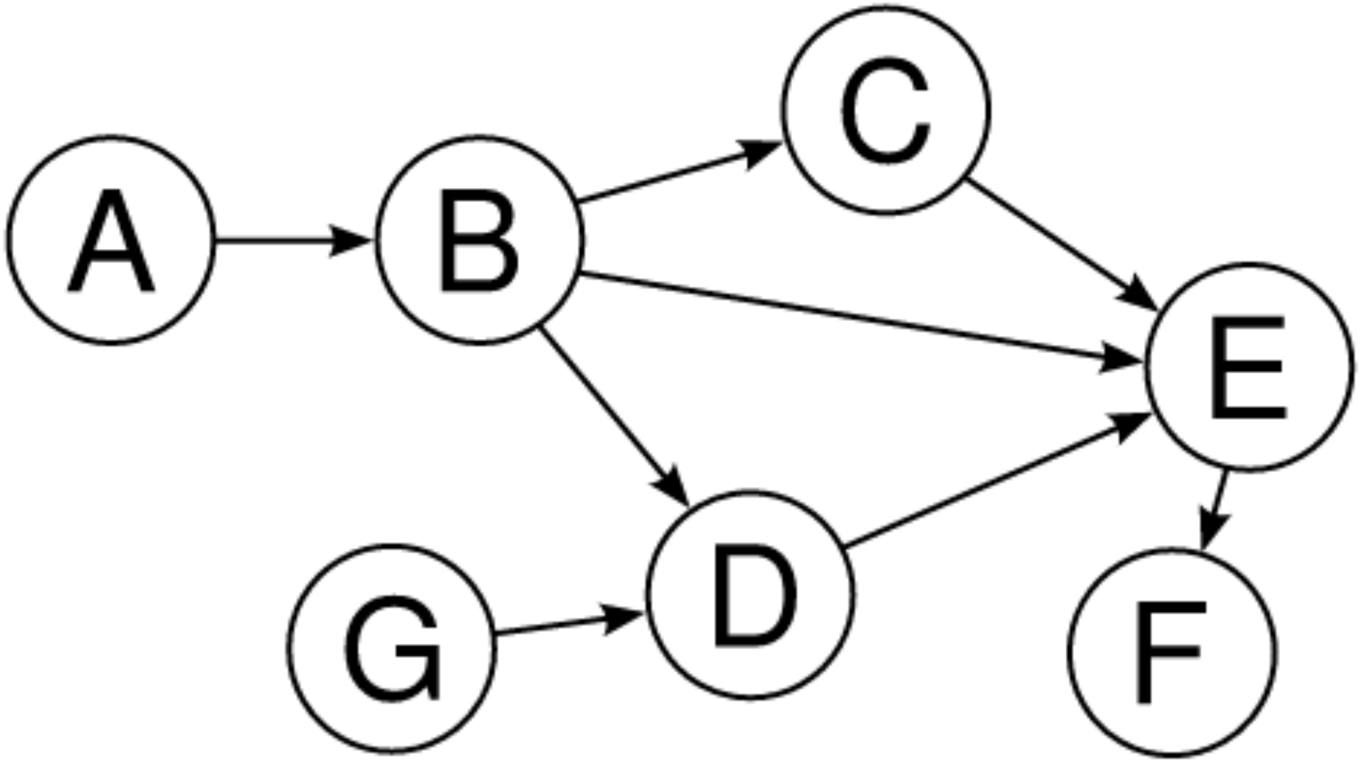
A digraph has a topological sorting iff it is a **dag**.





Finding a **Topological Sorting** of the vertices of a dag:

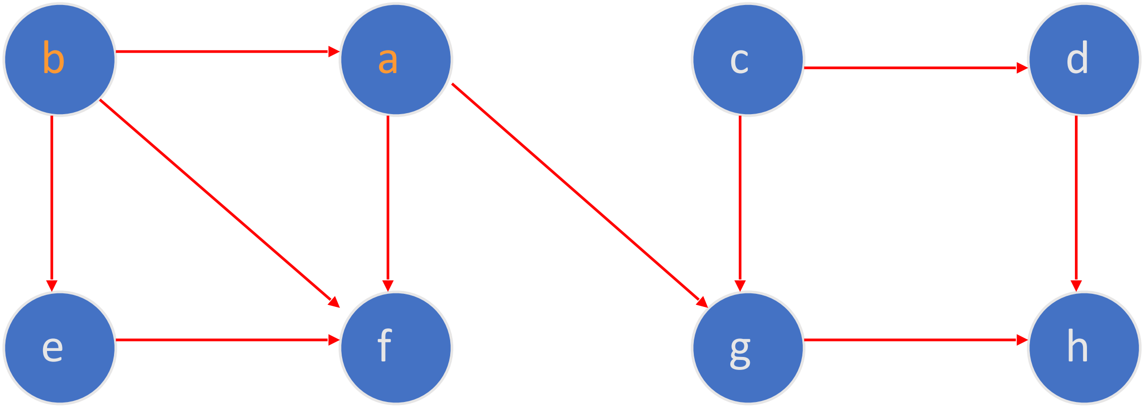
- **DFS-based** algorithm
- **Source-removal** algorithm



DFS-based algorithm for topological sorting

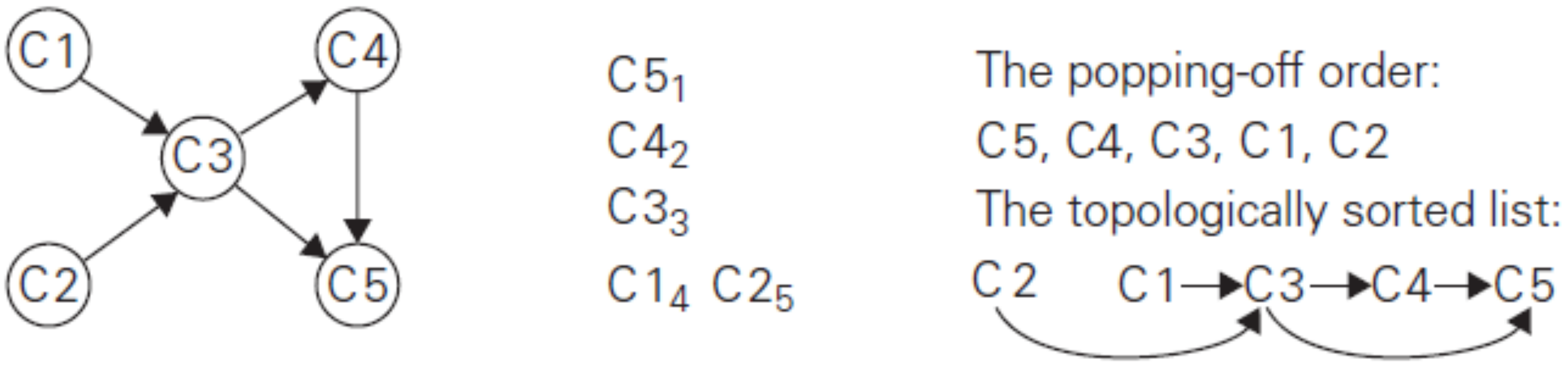
- Perform DFS traversal, noting the order vertices are popped off the traversal stack
- Reverse order solves topological sorting problem
- Back edges encountered?→ NOT a dag!

Example:



Efficiency: The same as that of DFS.

DFS-based algorithm for finding **Topological Sorting**



The popping-off order:

C5, C4, C3, C1, C2

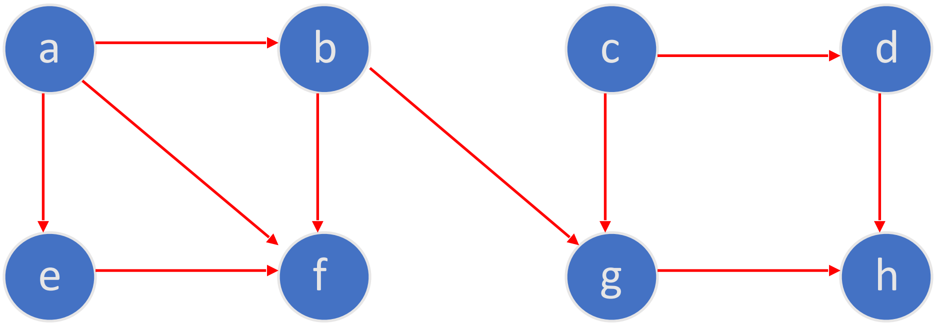
The topologically sorted list:

(a) (b) (c)

Source removal algorithm

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)

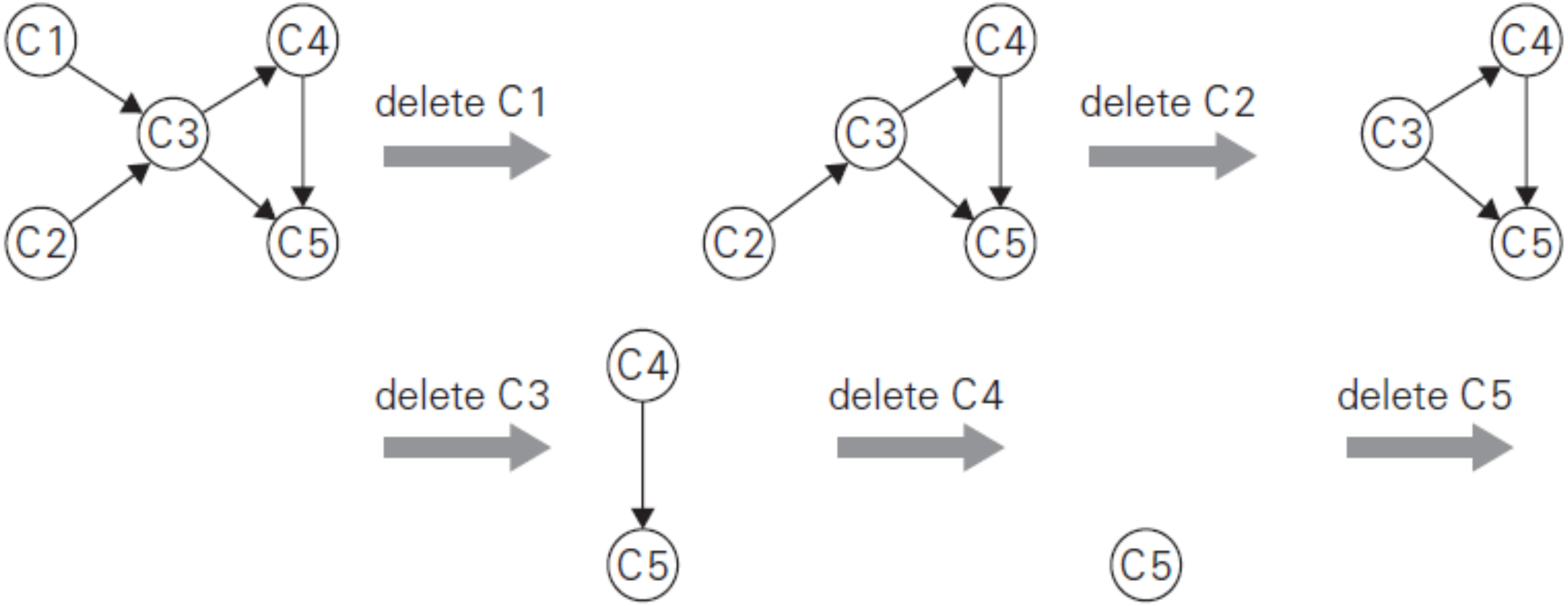
Example:



"Invert" the adjacency lists for each vertex to count the number of incoming edges by going thru each adjacency list and counting the number of times that each vertex appears in these lists. To remove a source, decrement the count of each of its neighbors by one.

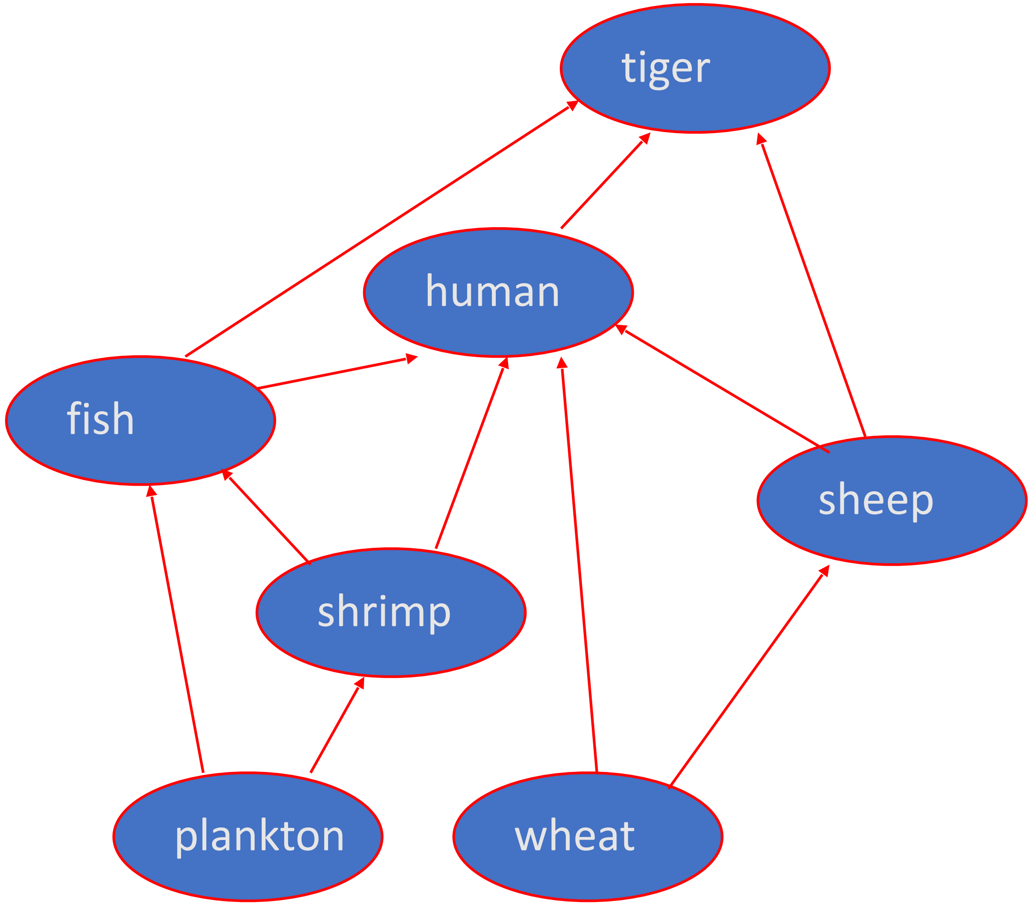
```
Algorithm SourceRemoval_Toposort(V, E)
L ← Empty list that will contain the sorted vertices
S ← Set of all vertices with no incoming edges
while S is non-empty do
    remove a vertex v from S
    add v to tail of L
    for each vertex m with an edge e from v to m do
        remove edge e from the graph
        if m has no other incoming edges then
            insert m into S
if graph has edges then
    return error (not a DAG)
else return L (a topologically sorted order)
```

Decrease and Conquer



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

Order the following items in a food chain



THANK YOU

Surabhi Narayan

Department of Computer Science &Engineering

surabhinarayan@pes.edu