UNIVERSITY INSTITUTE OF COMPUTING

MASTER OF COMPUTER APPLICATIONS

DESIGN AND ANALYSIS OF ALGORITHMS

24CAT-611

UNIT-2

Outline

- **Decrease and Conquer Approach**
- Topological Sort.

Decrease & Conquer

1. Reduce problem instance to smaller instance of the same problem
2. Solve smaller instance
3. Extend solution of smaller instance to obtain solution to original instance

- Can be implemented either top-down or bottom-up
- Also referred to as *inductive* or *incremental* approach

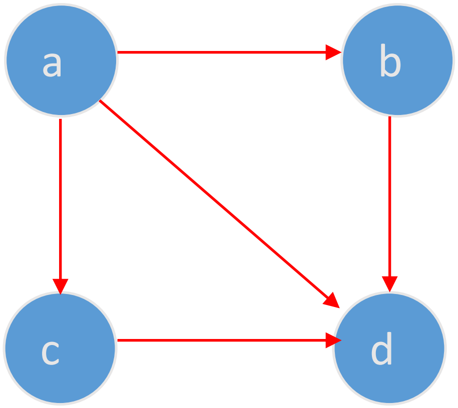
Decrease & Conquer (contd.)

- *Decrease by a constant* (usually by 1):
  - insertion sort
  - graph traversal algorithms (DFS and BFS)
  - topological sorting
  - algorithms for generating permutations, subsets

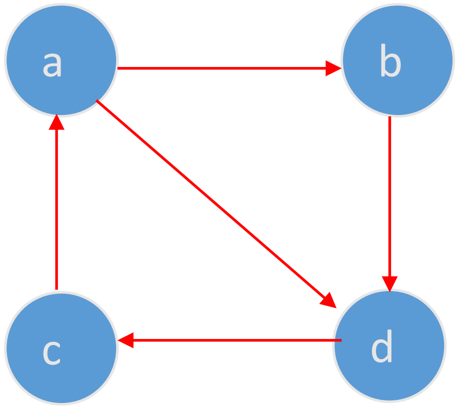
- *Decrease by a constant factor* (usually by half)
  - binary search and bisection method
  - exponentiation by squaring
  - multiplication

Directed Acyclic Graph

- A directed acyclic graph, i.e. a directed graph with no (directed) cycles



a dag



Not a dag

Directed Acyclic Graph (contd.)

- 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

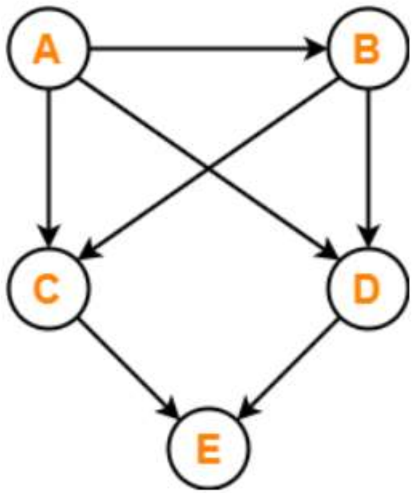
- Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge uv, vertex u comes before v in the ordering. Topological Sorting for a graph is not possible if the graph is not a DAG.
- For example, a topological sorting of the following graph 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 incoming edges).

Topological Sorting (contd.)

- 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.
- 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 incoming edges).

Topological Sorting Example (contd.)

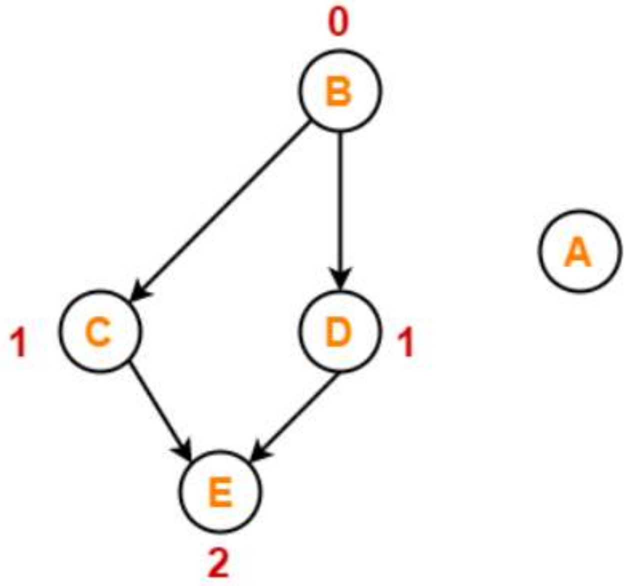
Find the number of different topological orderings possible for the given graph-



Topological Sorting Example (contd.)

Step-02:

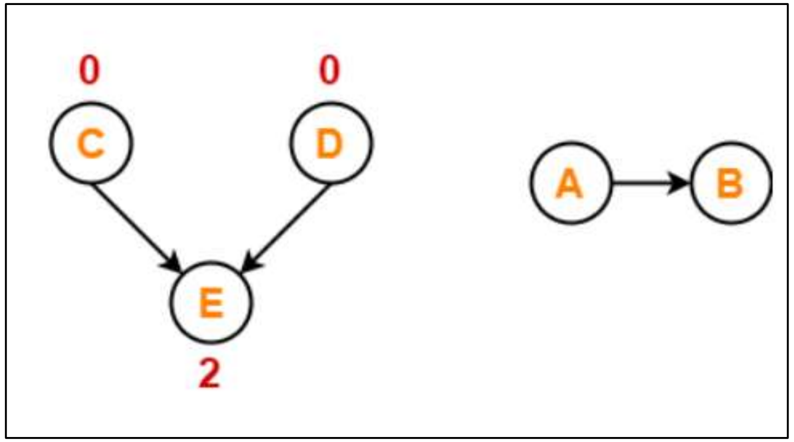
- Vertex-A has the least in-degree.
- So, remove vertex-A and its associated edges.
- Now, update the in-degree of other vertices.



Topological Sorting Example (contd.)

Step-03:

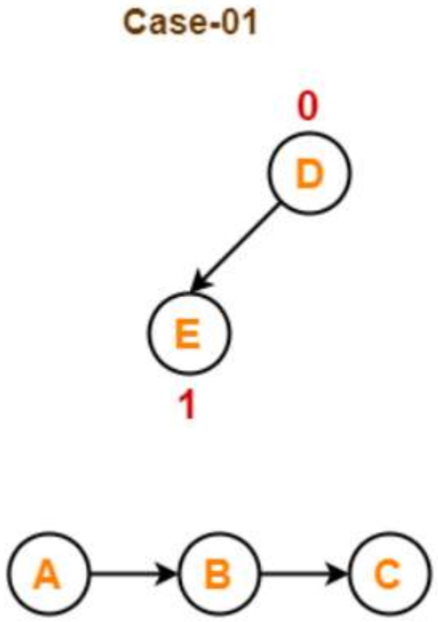
- Vertex-B has the least in-degree.
- So, remove vertex-B and its associated edges.
- Now, update the in-degree of other vertices.



Topological Sorting Example (contd.)

Step-04:

There are two vertices with the least in-degree. So, following 2 cases are possible-
In case-01,

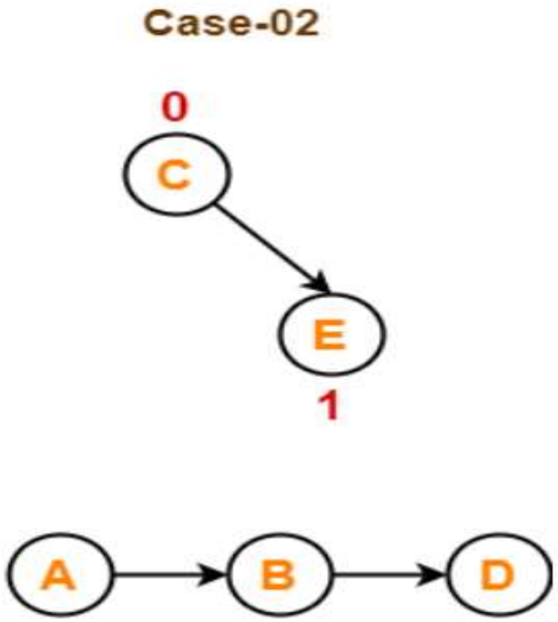


- Remove vertex-C and its associated edges.
- Then, update the in-degree of other vertices.

Topological Sorting Example (contd.)

In case-02,

- Remove vertex-D and its associated edges.
- Then, update the in-degree of other vertices.



Topological Sorting Example (contd.)

- **Step-05:**
- Now, the above two cases are continued separately in the similar manner.

In case-01,

  - Remove vertex-D since it has the least in-degree.
  - Then, remove the remaining vertex-E.

In case-02,

- Remove vertex-C since it has the least in-degree.
- Then, remove the remaining vertex-E.

Topological Sorting Example (contd.)

For the given graph, following **2** different topological orderings are possible-

- **A B C D E**
- **A B D C E**

Algorithm of Topological Sorting

topoSort(u, visited, stack)

- **Input –** The start vertex u, An array to keep track of which node is visited or not.

  A stack to store nodes.

Output – Sorting the vertices in topological sequence in the stack.

```
Begin
mark u as visited
for all vertices v which is adjacent with u,
do
if v is not visited, then
topoSort (c, visited, stack)
done
push u into a stack End
```

Algorithm of Topological Sorting (contd.)

Perform Topological Sorting(Graph)

- **Input –** The given directed acyclic graph.
- **Output –** Sequence of nodes.

```
Begin
initially mark all nodes as unvisited
for all nodes v of the graph,
do
if v is not visited,then
topoSort(i, visited, stack)
done
pop and print all elements from the stack
End.
```

Application of Topological Sorting

- Scheduling jobs from the given dependencies among jobs
- Instruction Scheduling
- Determining the order of compilation tasks to perform in makefiles
- Data Serialization

Frequently asked questions

- What do you understand by topological sort?
- Define complexity of topological sort.

REFERENCES

1) https://www.gatevidyalay.com/topological-sort-topological-sorting/

2) **Data Structures and Algorithms made easy** By *Narasimha Karumanchi*.

5) The Algorithm Design Manual, 2nd Edition by Steven S Skiena

6) **Fundamentals of Computer Algorithms - Horowitz and Sahani**

THANK YOU