# Directed Acyclic Graphs and Topological Order

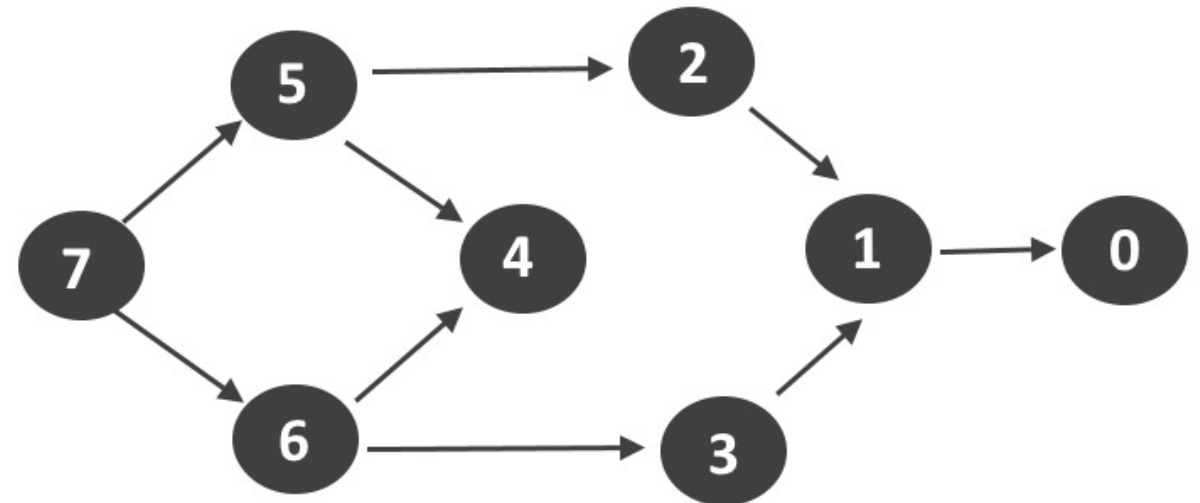Instructor: Meng-Fen Chiang

COMPSCI: WEEK 10.2

THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

# OUTLINE

- Directed Acyclic Graphs (DAGs)

- Topological Orders
  - Definition
  - Illustration

- Topological Sorting
  - Illustrative Examples



Topological Sort : 7 6 5 4 3 2 1 0

# Cycle Detection

- Suppose that there is a cycle in $G$ and let $v$ be the node in the cycle visited first by DFS. If $(u, v)$ is an arc in the cycle then it must be a back arc.

- Conversely if there is a back arc, we must have a cycle.

- Suppose that **DFS** is run on a digraph $G$. Then $G$ is acyclic if and only if $G$ does not contain a back arc.

- A digraph with no cycle is called a directed acyclic graph (DAG).
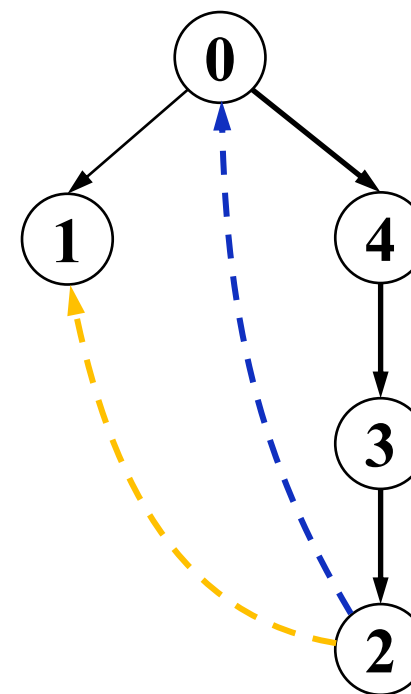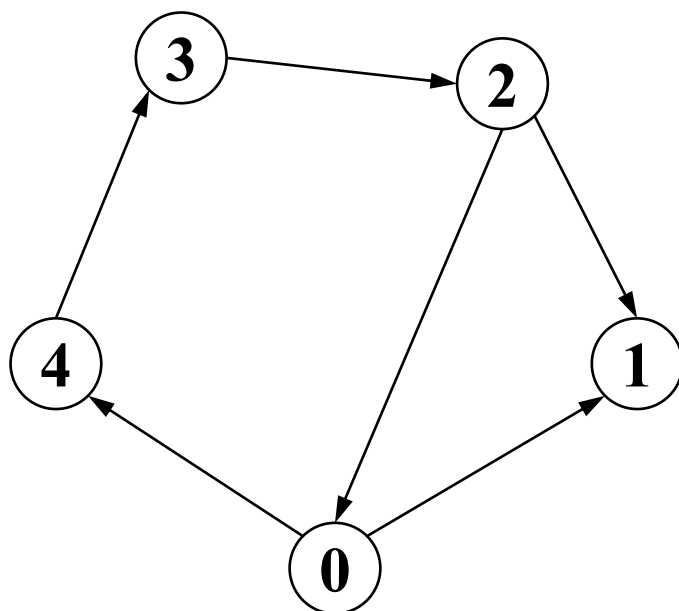
# Example: Directed Acyclic Graphs (DAGs)
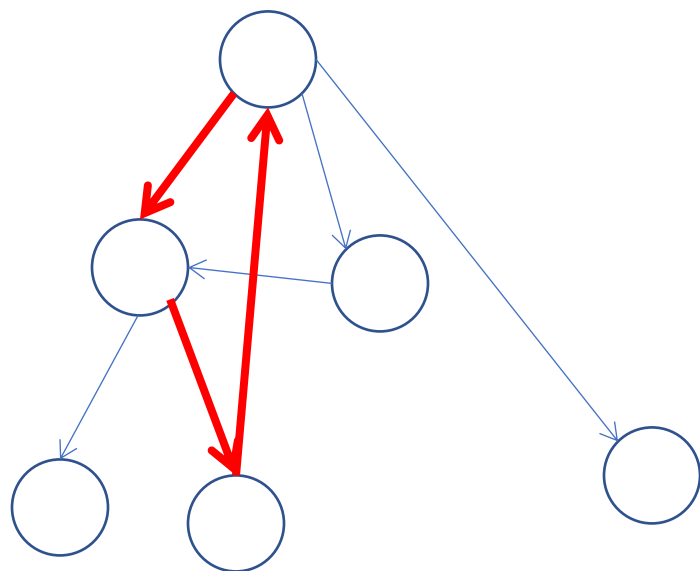


$G$
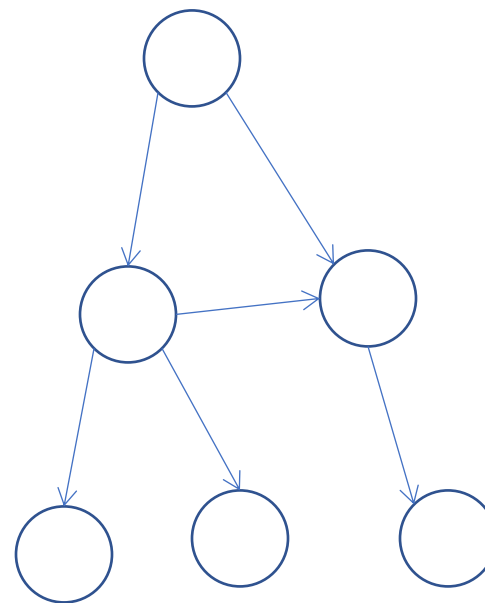
$G'$: Acyclic

# Using DFS to Find Cycles in Digraphs

- Once DFS finds a cycle, the stack contains the nodes that form the cycle

# Properties: Directed Acyclic Graphs (DAGs)

- Any walk on a DAG $G$ is limited in length


- A DAG must have <span style="color:red">at least one source</span> and <span style="color:red">one sink</span> (why?)
    - A graph that has no source or no sink must have a cycle


- Although a DAG has <span style="color:red">no cycles</span> the underlying graph could have cycles.

# Topological Sorting

- A topological sorting of a digraph $G$ is an ordering on its vertices such that, for each arc $(u, v)$ of $G$, $u$ appears before $v$ in the ordering.

- To place nodes of a digraph on a line so all arcs go in one direction.

- Main application: scheduling events (arithmetic expressions, university prerequisites, etc).
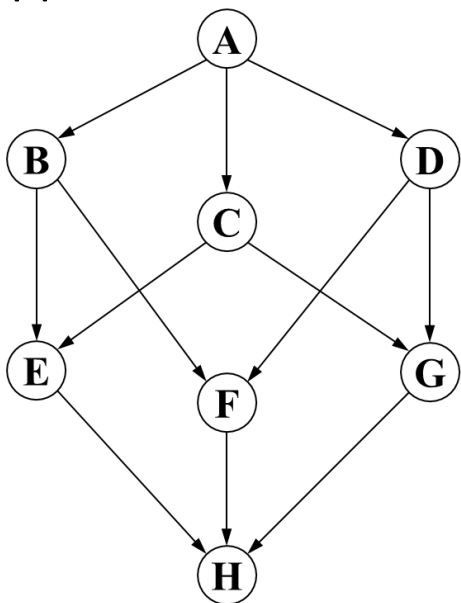


- A D B E C F G H is NOT a topological sorting, because there is an arc (C,E), but E comes prior to C in the ordering.

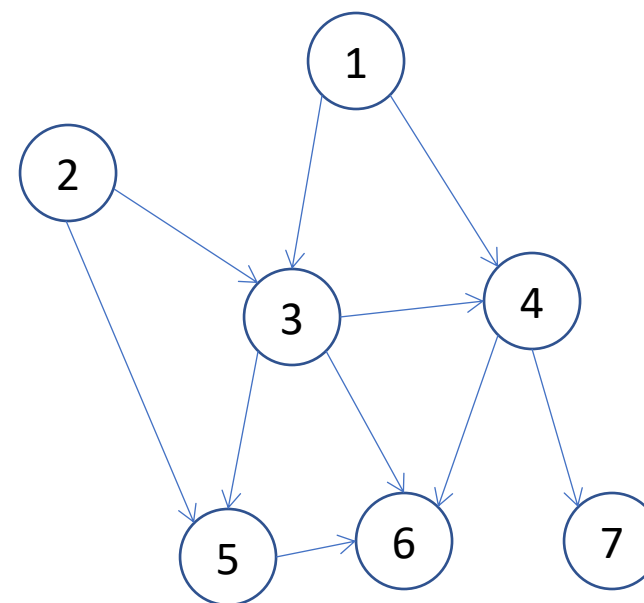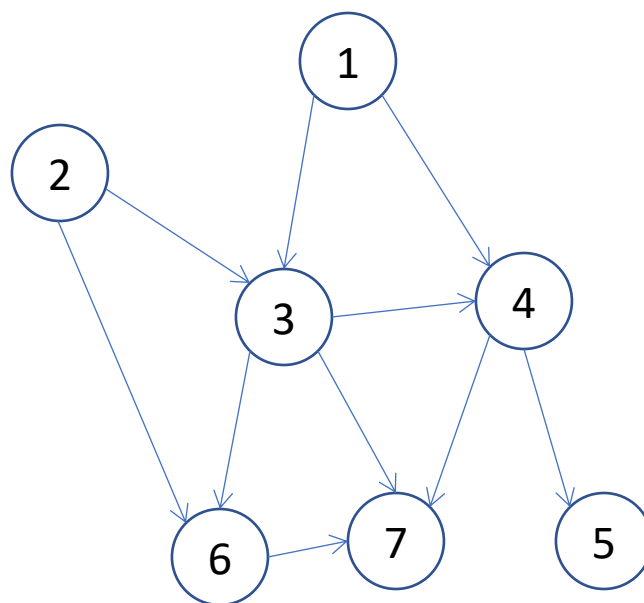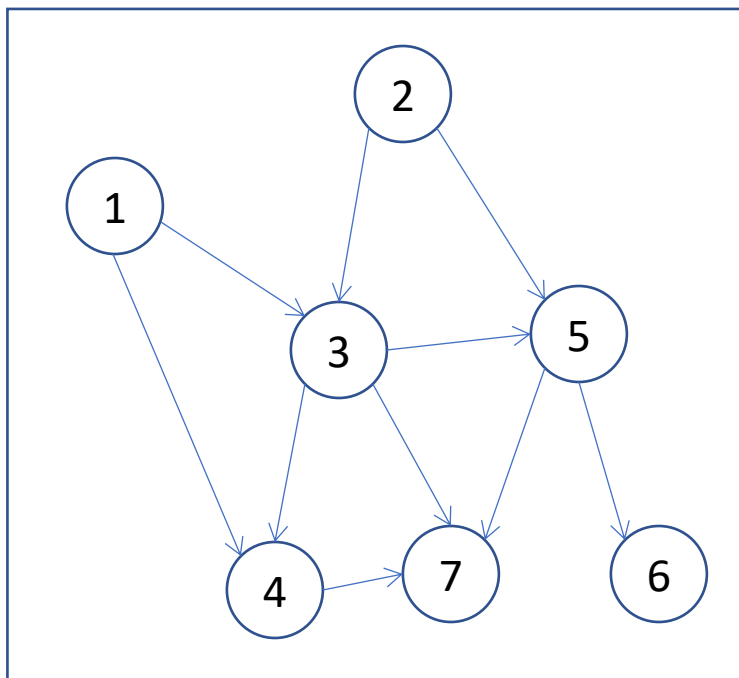- A D B C E F G H is a topological sorting.

# Topological Sorting (Contd.)

- If G is a DAG, then it is possible to find a <span style="color:red">topological order</span> of the vertices.

- A topological order is a numbering of the vertices such that an arc (u,v) in the digraph means that <span style="color:red">u has a smaller number than v</span>.

- Only DAGs have a topological ordering
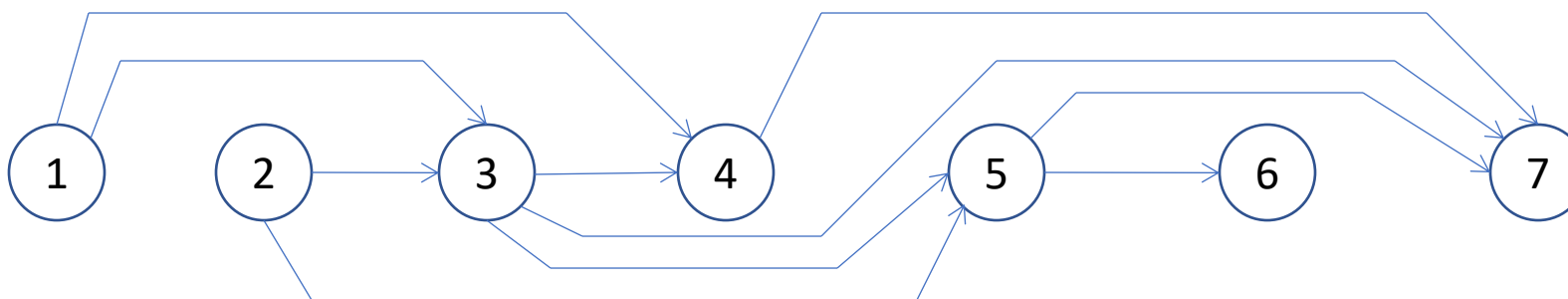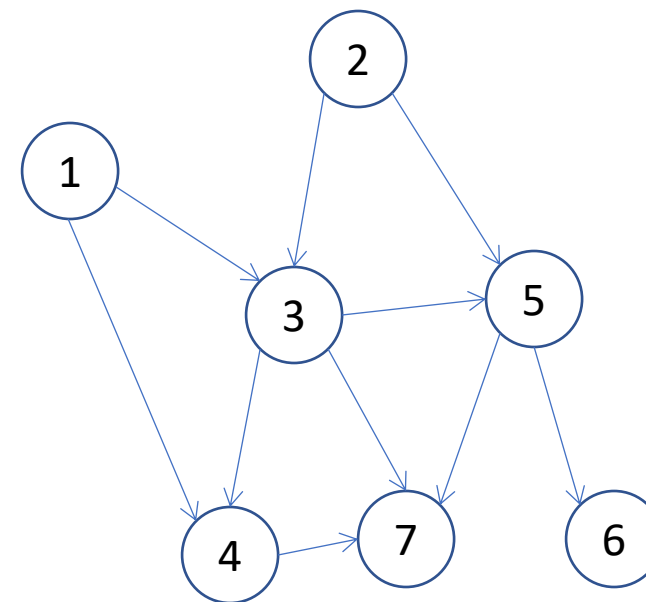
# Topological Orders (Contd.)

- There is often more than one topological order for a digraph (see examples on previous slides)

- A topological order is also called a <span style="color:red">topological sort</span> or a <span style="color:red">linear order</span>.

- Why "linear" order? Because you can order the vertices in a line and make all arcs point the same way!

# Example: Topological Sorting 1

# Topological Order: to Linear

These two digraphs
are exactly the same!

In linear sort order, all arcs run from left to right!

# Topological Sorting

- Topological sorting is possible if and only if digraph is a DAG.
  - If there is a topological order of a digraph, then there is no cycle
    - Suppose there is a cycle $u \to \cdots \to v \to \cdots \to u$, $u$ and $v$ are both ancestors of each other
  - A DAG always has at least one topological order
    - Every DAG has a source

      Suppose there is no source. Let $u_1, u_2, \ldots, u_n$ be a directed path of maximal length.
      Then there is no longer directed path that contains $u_1, u_2, \ldots, u_n$. But then $u_1$ is a source.
      $\rightarrow$ A contradiction.

    - By removing the source and any out-arcs from the source we still have a DAG.
    - The order we remove the sources forms a topological order

$u_1$

$\downarrow$

$u_2$

$\downarrow$

$\vdots$

$\downarrow$

$u_n$

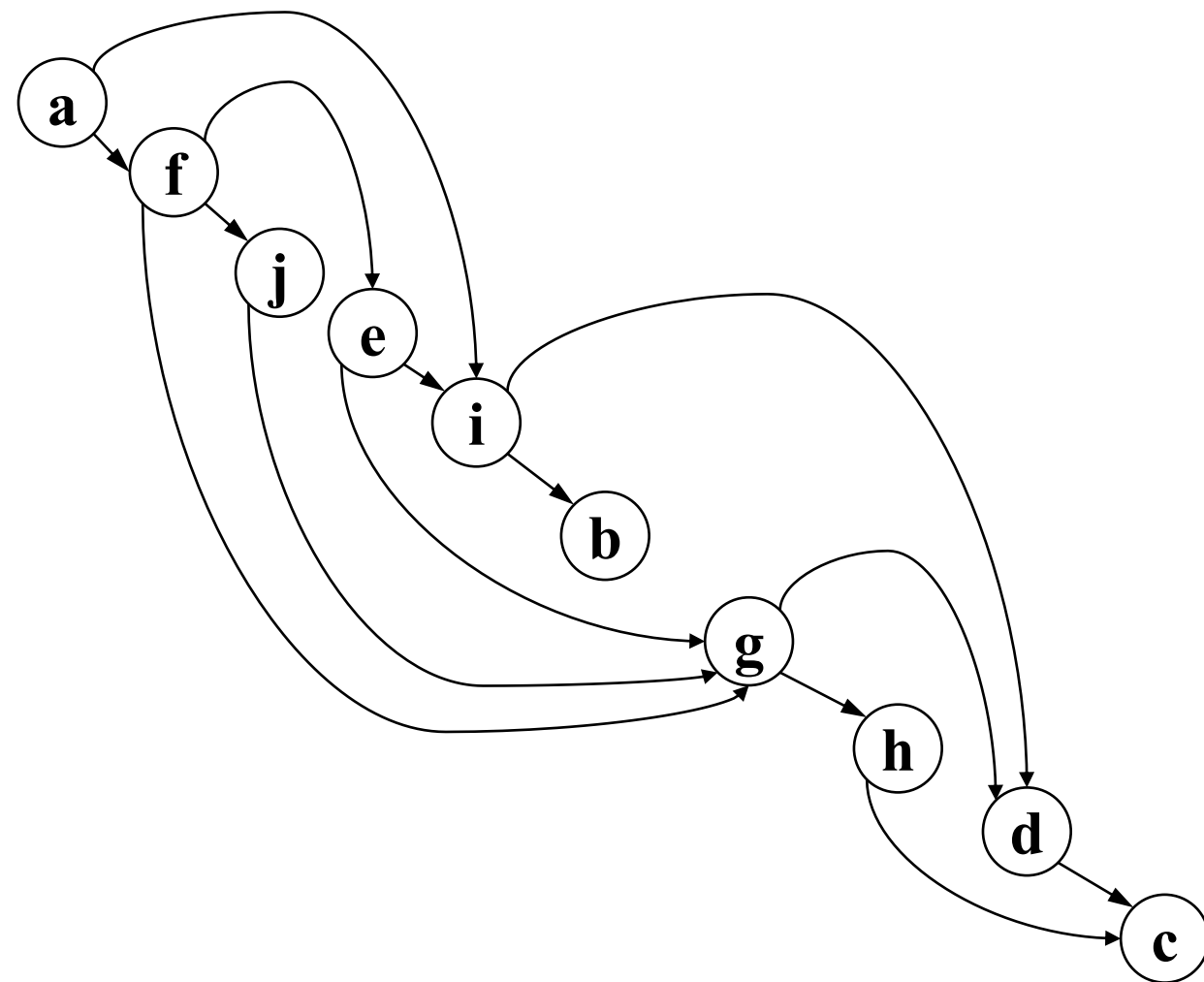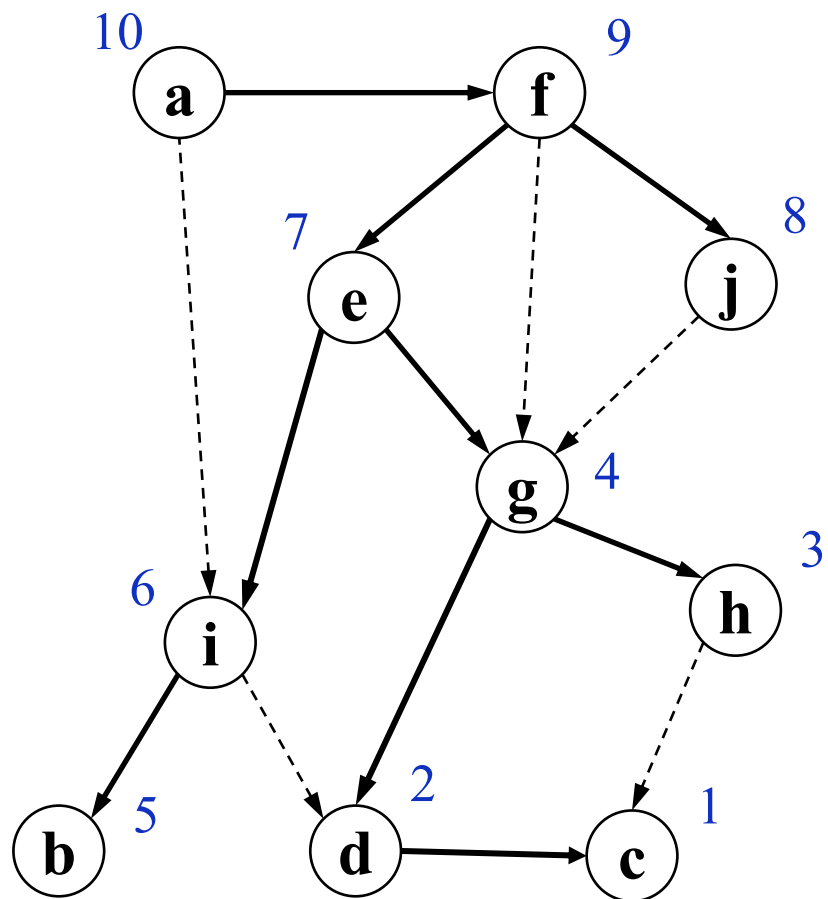# Topological Sorting

- Two solutions:

1. List of finishing times by DFS, in reverse order (since there are no back arcs, each node finishes before anything pointing to it).

2. Zero in-degree sorting – Find a node of in-degree zero, delete it and repeat until all nodes listed.
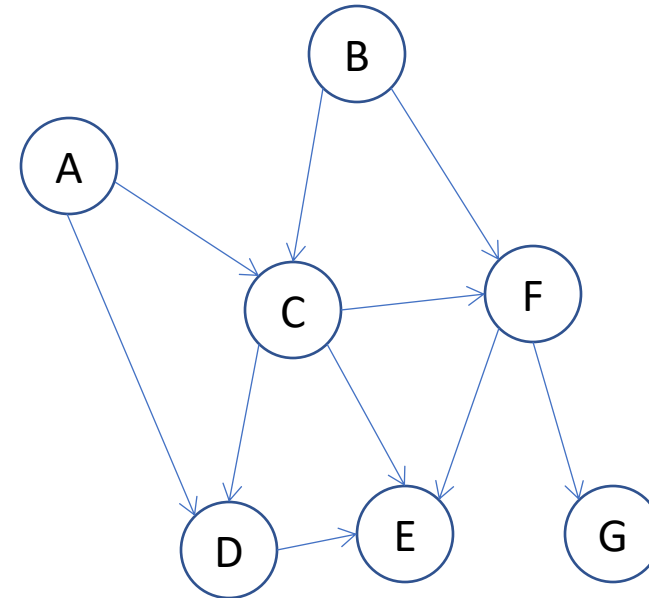
# Example: Topological Order
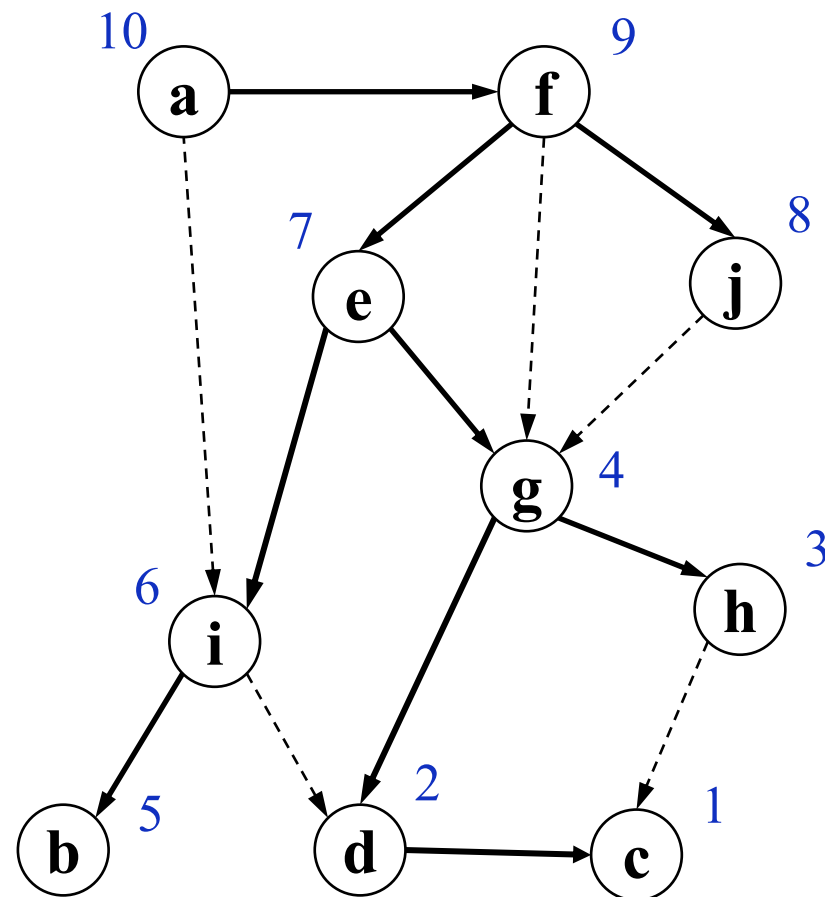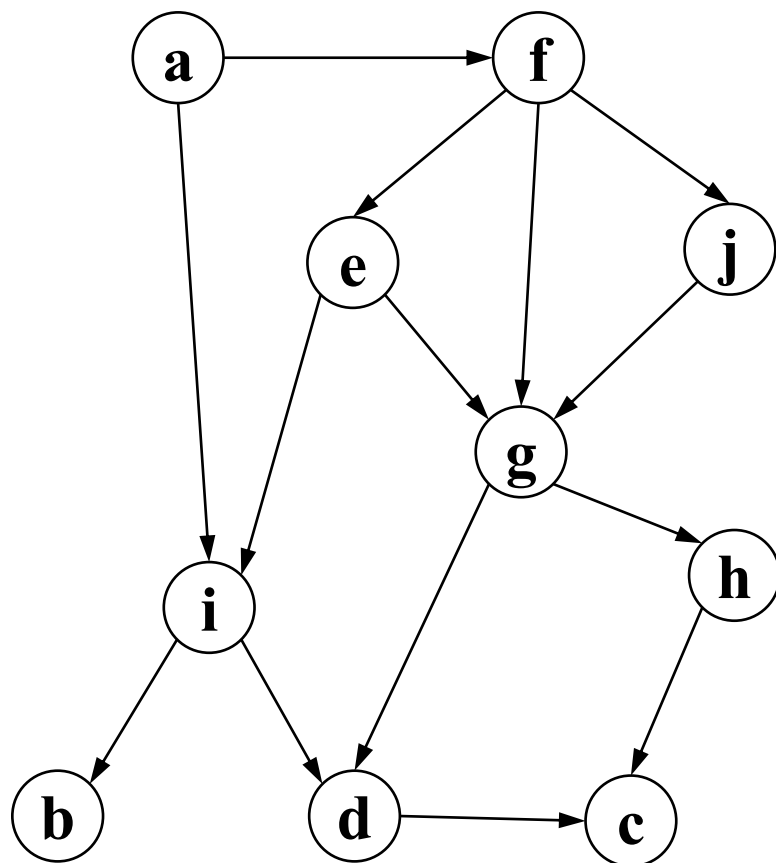
# Example: Topological Order
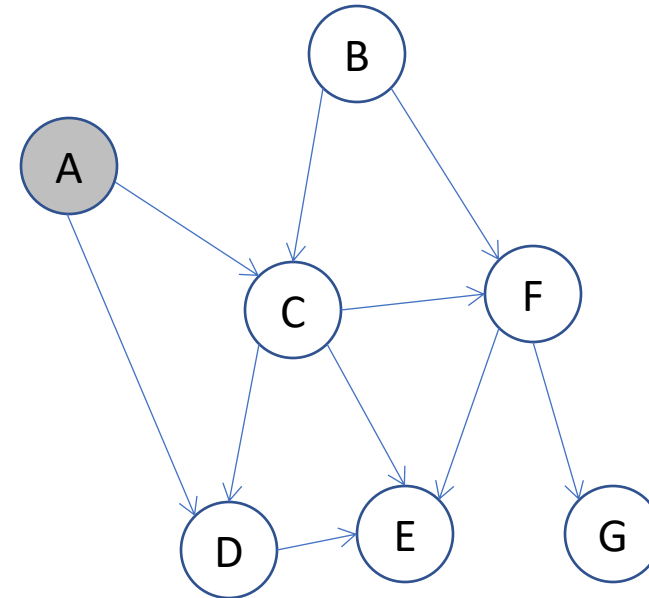
# Example: Topological Order by DFS

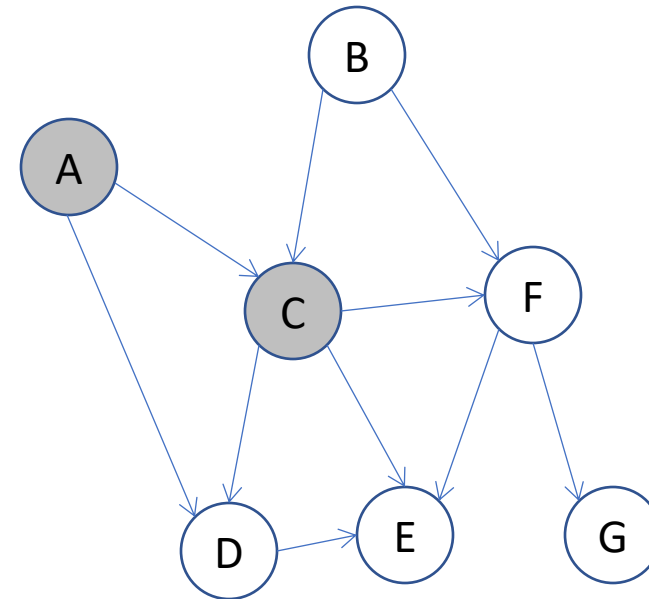- Starting at the node A:

# Example: Topological Order by DFS
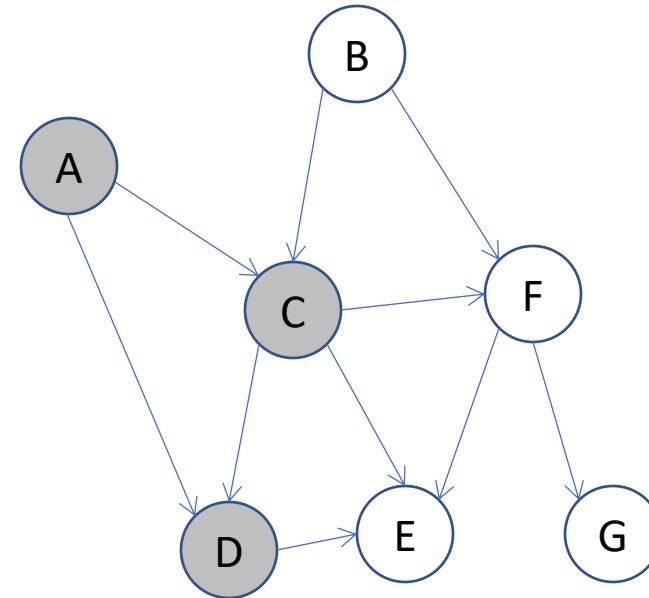
- Starting at the node A:



- push A

# Example: Topological Order by DFS

- Starting at the node A:
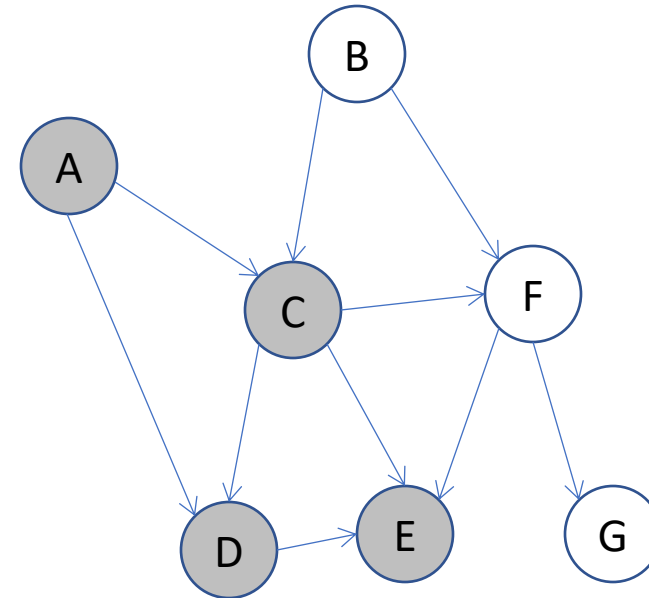

- push A, push C

# Example: Topological Order by DFS

- Starting at the node A:

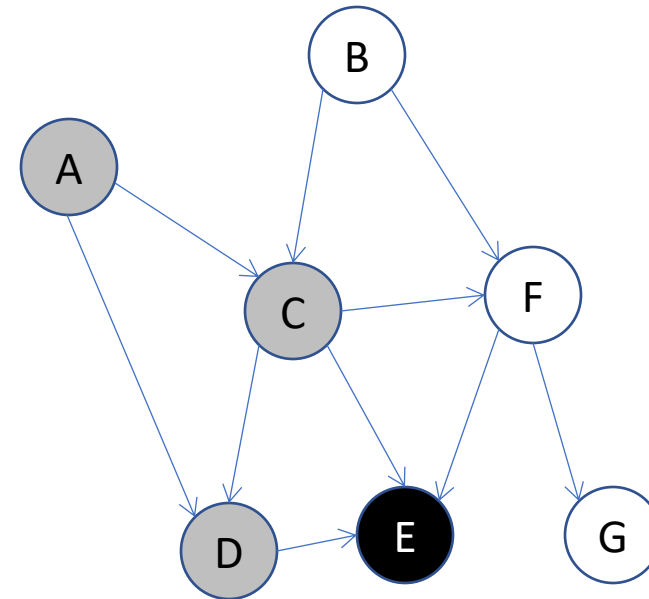- push A, push C, push D

# Example: Topological Order by DFS

- Starting at the node A:

- push A, push C, push D, push E

# Example: Topological Order by DFS

- Starting at the node A:

- push A, push C, push D, push E,
- <u>pop E</u>,

- And store E in the last position in the array



| | | | | | | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:

- push A, push C, push D, push E,
- pop E, pop D
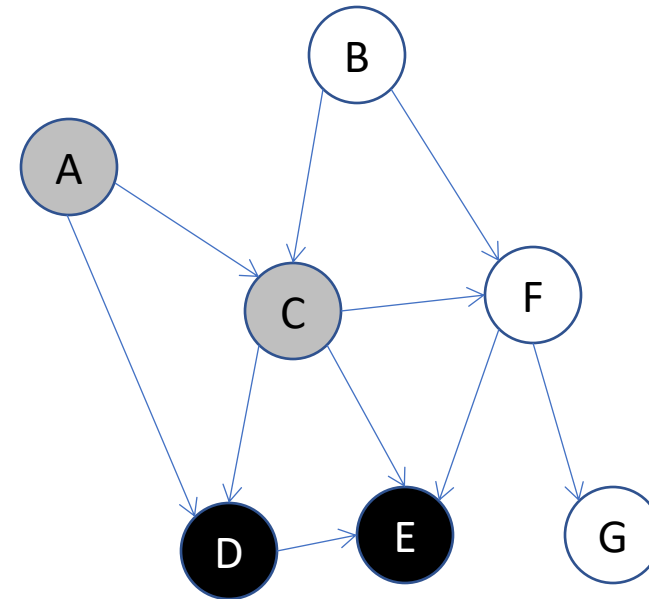
- And store D in position array.length()-2

| | | | | | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:


- push A, push C, push D, push E,
- <u>pop E</u>, <u>pop D</u>, push F



| | | | | | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:


- push A, push C, push D, push E,

- pop E, pop D, push F, push G



| | | | | | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:
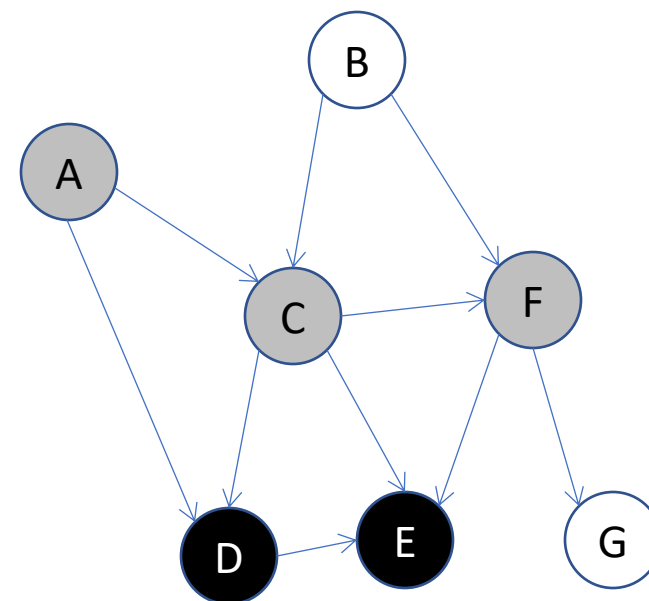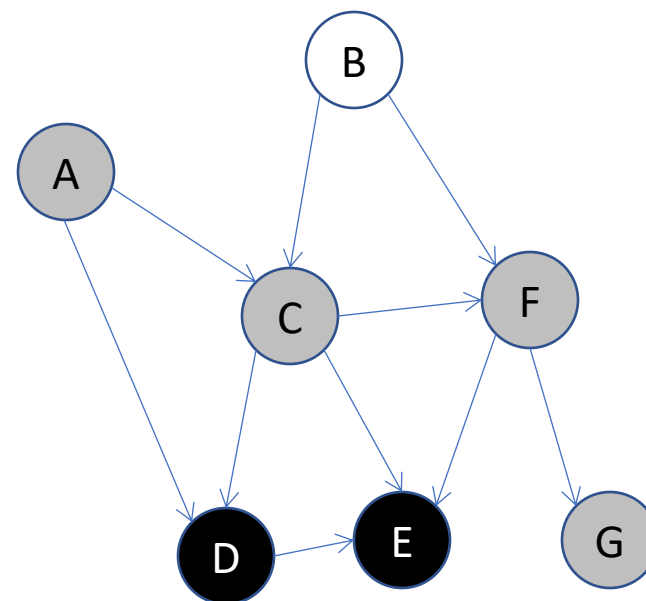
- push A, push C, push D, push E,
- pop E, pop D, push F, push G, pop G

- And store G in position array.length()-3



| | | | | G | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:



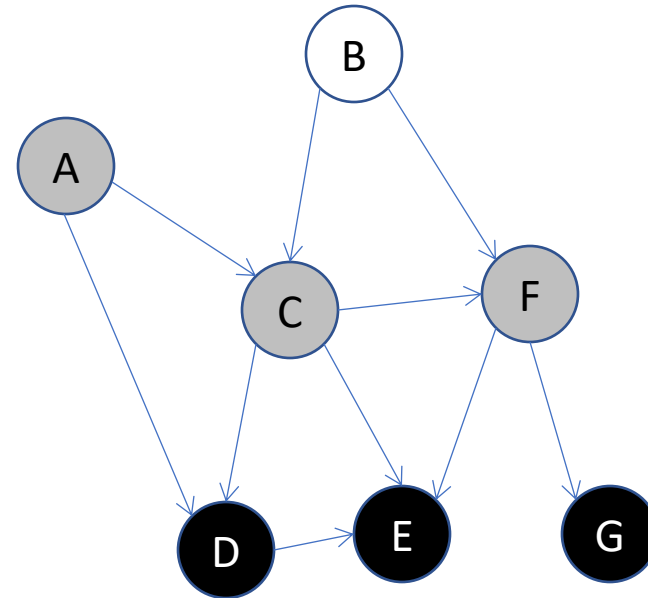- push A, push C, push D, push E,
- pop E, pop D, push F, push G, pop G
- pop F



- And store F in position array.length()-4

| | | | F | G | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:

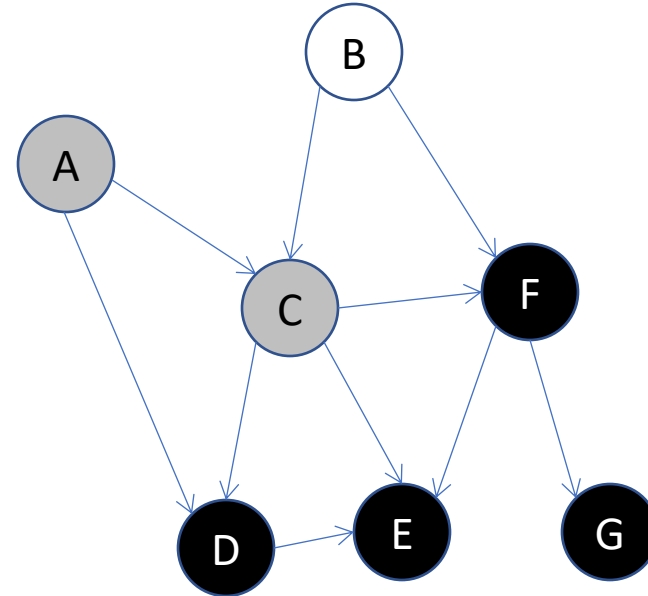- push A, push C, push D, push E,
- pop E, pop D, push F, push G, pop G
- pop F, pop C



- And store C in position array.length()-5
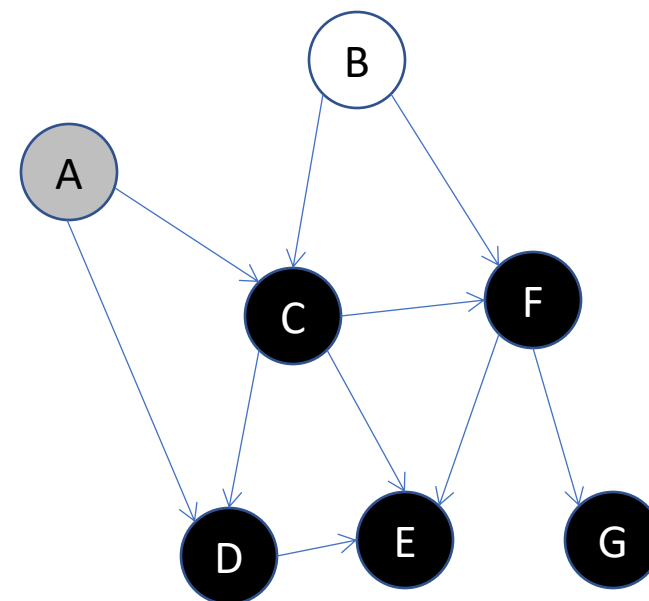
| | | C | F | G | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:

- push A, push C, push D, push E,
- pop E, pop D, push F, push G, pop G
- pop F, pop C, pop A,

- And store A in position array.length()-6

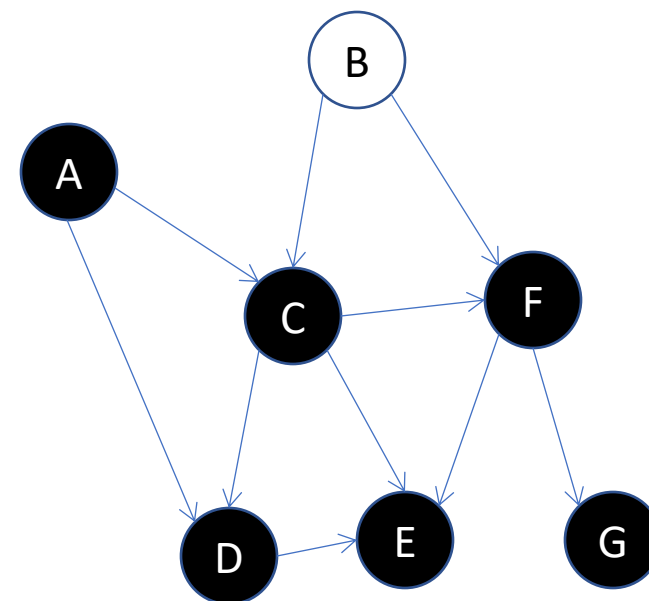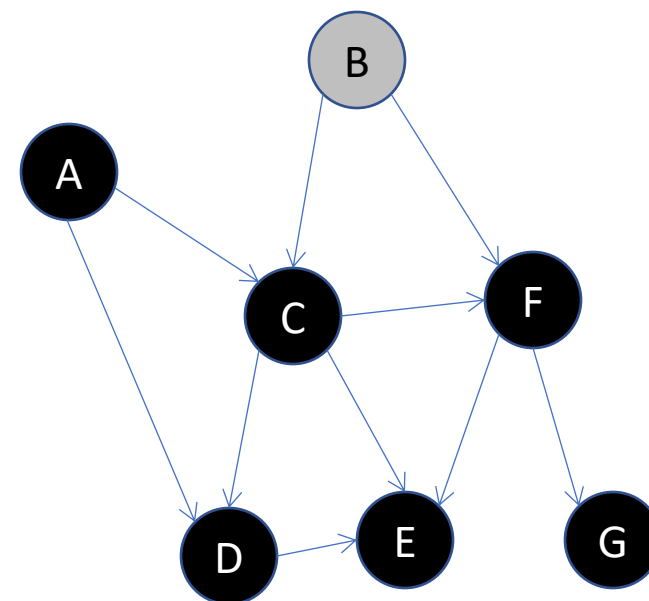| | A | C | F | G | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:


- push A, push C, push D, push E,
- pop E, pop D, push F, push G, pop G
- pop F, pop C, pop A, push B



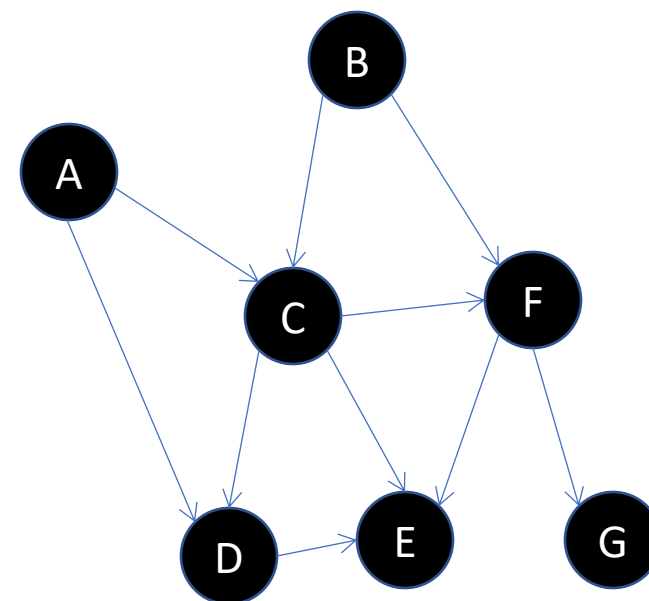| | A | C | F | G | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:



- push A, push C, push D, push E,
- pop E, pop D, push F, push G, pop G
- pop F, pop C, pop A, push B, pop B

- And store B in position array.length()-7

| B | A | C | F | G | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order by DFS

- Starting at the node A:

- push A, push C, push D, push E,
- pop E, pop D, push F, push G, pop G
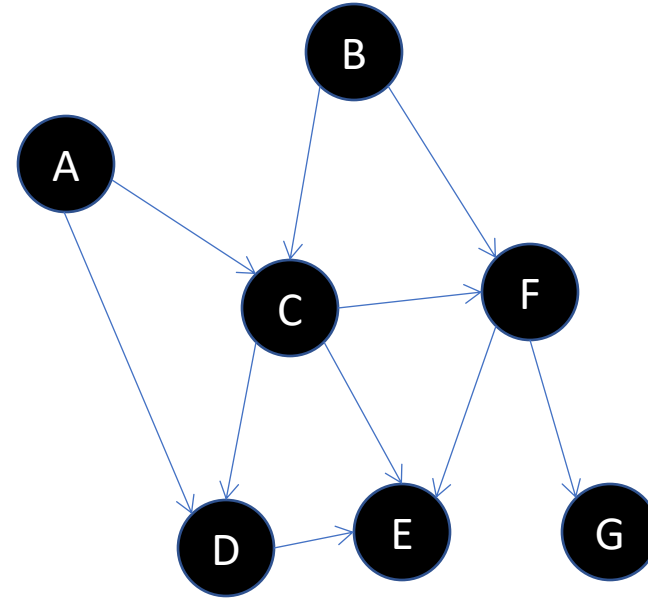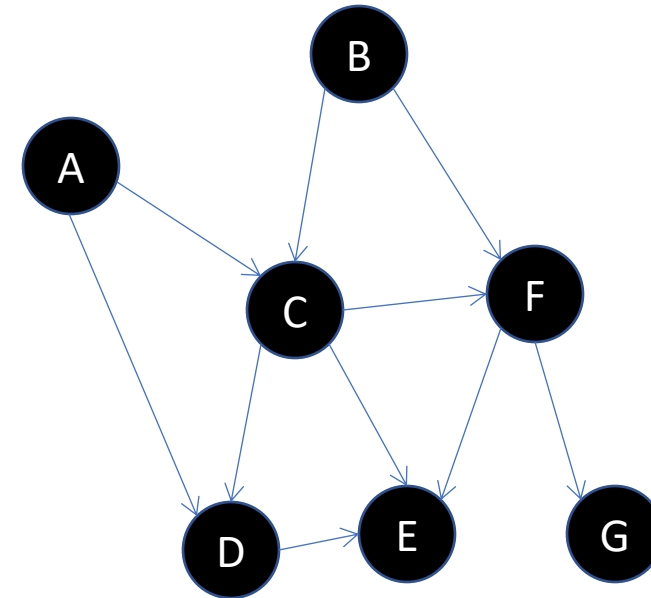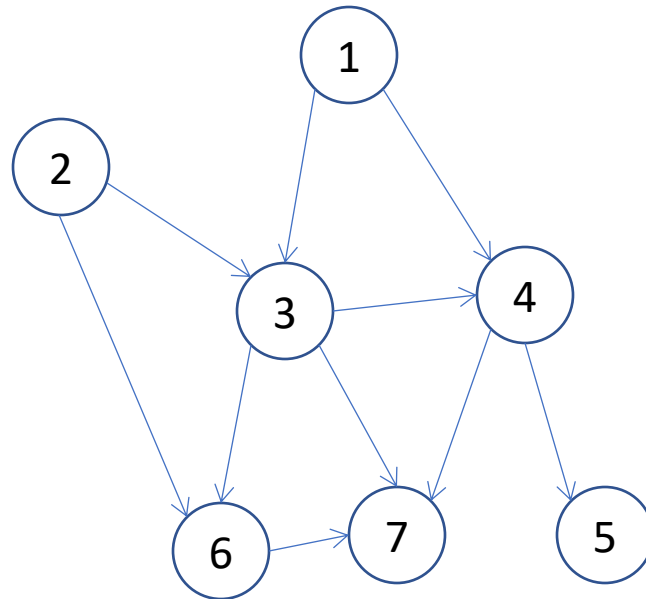- pop F, pop C, pop A, push B, pop B

- So, we get B=1, A=2, C=3, F=4, G=5, D=6, E=7

| B | A | C | F | G | D | E |
|---|---|---|---|---|---|---|

# Example: Topological Order 2



B=1, A=2, C=3, F=4, G=5, D=6, E=7

# Zero in-degree Sorting

- **Zero indegree sorting**: Start with DAG, say $G$. Initialize the order list $L$ as empty.

1. Find a source vertex $v$.

2. Delete $v$ and all its out-going arcs. Append node $v$ to the ordered list $L$. Since we only delete arcs and vertices, this process does not create a cycle. Hence, the resulting graph is a DAG and has at least a source node.

3. Repeat 1 and 2 under all the vertices have been deleted. Then, we will get a topological order $L$ of $G$.

# Zero in-degree Sorting

- What is the running time of a naive implementation of zero-indegree sorting where a source is found and then removed at each step? How could this idea be made more efficient?

- For each iteration, we need to find zero indegree vertex $v$, delete $v$ and its out-going arcs. We calculate the in-degree over every node each time, it takes

  Adjacency Matrix: $O(n^2)$             Adjacency List: $O(m)$

- For all n iterations:

  Adjacency Matrix: $O(n^3)$             Adjacency List: $O(nm)$

- We don't have to calculate the in-degree every time, we can use an array to track the in-degree of every node!

# Zero In-degree Sorting: A Faster Algorithm

---

**Algorithm 1** TopSort.

---

1: **function** TopSort(digraph $G$)

2:     **for** $u \in V(G)$ **do**

3:         $indegree[u] \leftarrow$ indegree of $u$

4:       queue $Q, count \leftarrow 0, order = []$

5:     **for** $u \in V(G)$ **do**

6:         **if** $indegree[u] = 0$ **then** $Q.enqueue(u)$

7:     **while** $Q$ is not empty **do**

8:         $u \leftarrow Q.dequeue(), \quad order.append(u), \quad count \leftarrow count + 1$

9:         **for** $v$ as out-neighbor of $u$ **do**

10:             $indegree[v] \leftarrow indegree[v] - 1$

11:             **if** $indegree[v] = 0$ **then** $Q.enqueue(v)$

12:     **if** $count \mathrel{!=} |V(G)|$ **then**

13:         **return** $NULL$

14:     **else**

15:         **return** $order$

# Exercise: Topological Orders

- Topological orders are not unique. List all possible topological orders of the following digraph. We can use zero in-degree sorting.
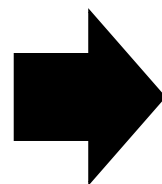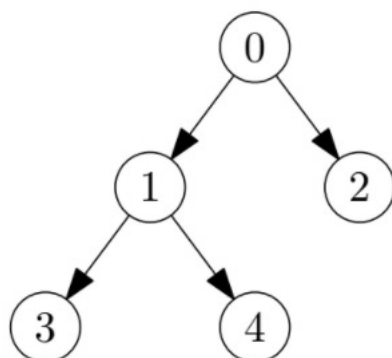
8 topological orders

Ordering: 0 1 2 3 4
Ordering: 0 1 2 4 3
Ordering: 0 1 3 2 4
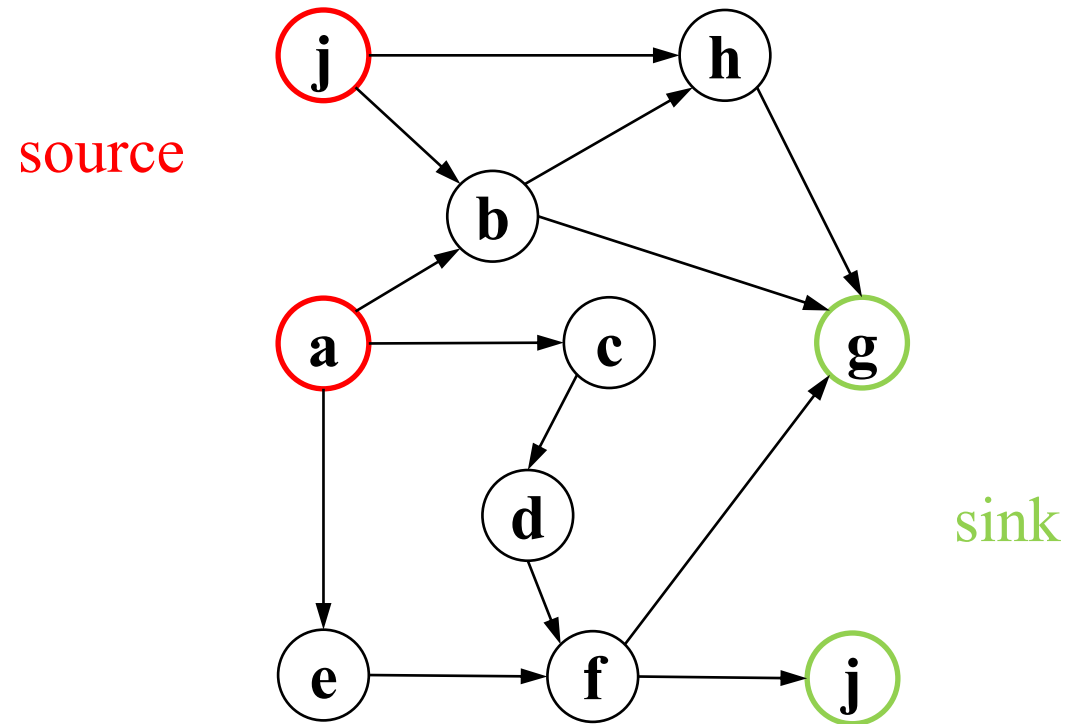Ordering: 0 1 3 4 2
Ordering: 0 1 4 2 3
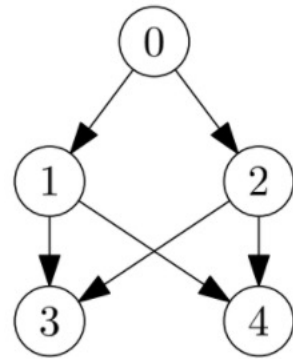Ordering: 0 1 4 3 2
Ordering: 0 2 1 4 3
Ordering: 0 2 1 3 4

# Exercise: Topological Sorting

For each arc $(u, v)$, $u$ appears before $v$ in a topological sorting.

- a e j b c d f i h g.
- Usually not unique.



source

sink

**Example 25.3.** A digraph with all possible topological orders and drawn with topological order 0, 1, 2, 3, 4. Draw the digraph for the topological order 0, 2, 1, 4, 3.
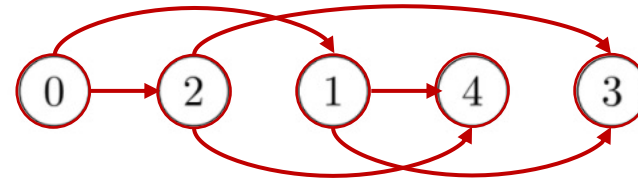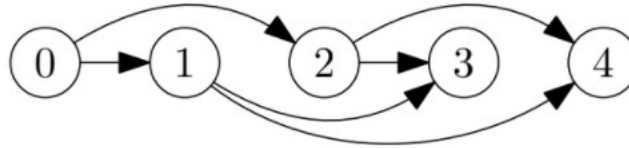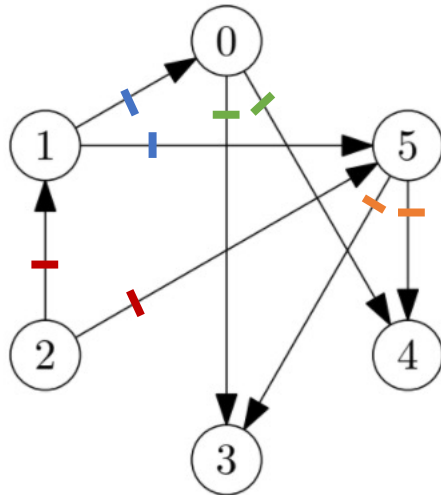


$0, 1, 2, 3, 4$

$0, 1, 2, 4, 3$
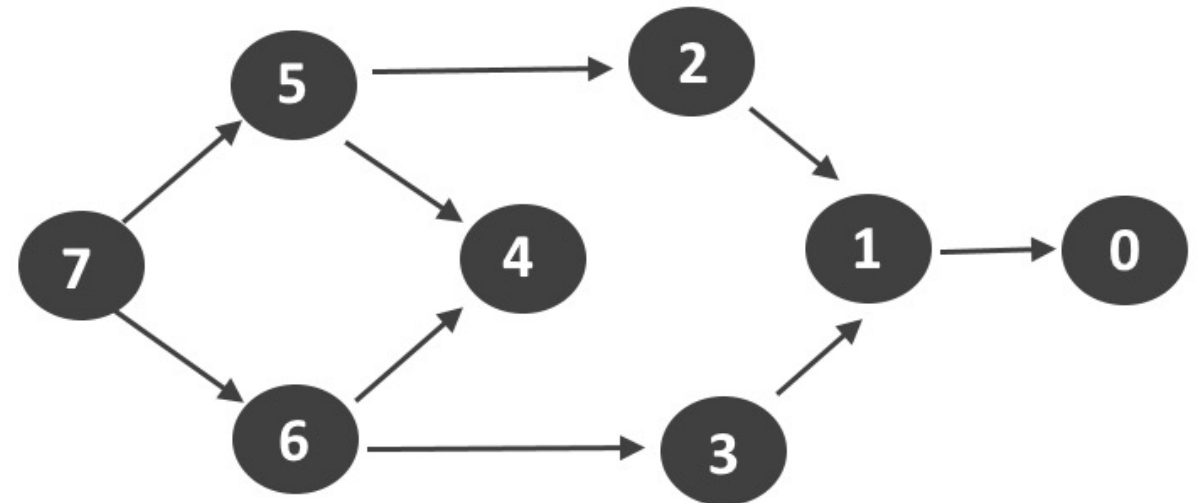
$0, 2, 1, 3, 4$

$0, 2, 1, 4, 3$

Find a topological order of the following digraph and draw it.

# SUMMARY

- Directed Acyclic Graphs (DAGs)

- Topological Orders
  - Illustrative Examples

- Topological Sorting
  - DFS
  - Zero-indegree Sorting



Topological Sort : 7 6 5 4 3 2 1 0