# VE281
Data Structures and Algorithms

Topological Sorting

Learning Objectives:

- Know what a topological sorting is and why it is useful

- Know the topological sorting algorithm and its runtime complexity

Topological Sorting

- **Topological sorting** : an ordering on nodes of a **directed graph** so that <u>**for each**</u> edge (v_i, v_j) (means: an edge **from** v_i **to** v_j) in the graph, v_i is before v_j in the ordering.
 - Also known as **topological ordering**.
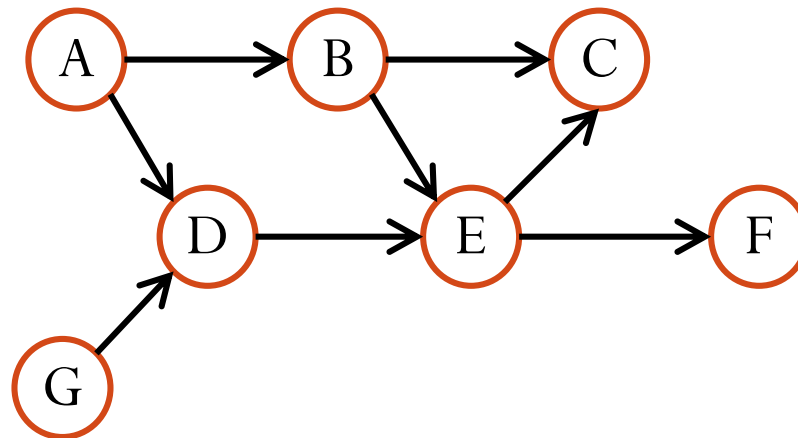
A topological sorting is: A, G, D, B, E, C, F

Which Graph Has Topological Sorting?

- Is there any "topological sorting" for directed graph **with cycles**?

  - In other words, can we order the nodes so that for each edge (v_i, v_j), v_i is before v_j in the ordering?

  - **<u>Answer</u>**: **No!** (Why?)

- How about **directed acyclic graph (DAG)**?

  - Yes! Guarantee to have a topological ordering.

  - Why? There is always a **source node** S in a DAG. Put S first. For the graph without S, again, there is a source node. Put it next …

- Next, we will focus on topological sorting on **DAG**.
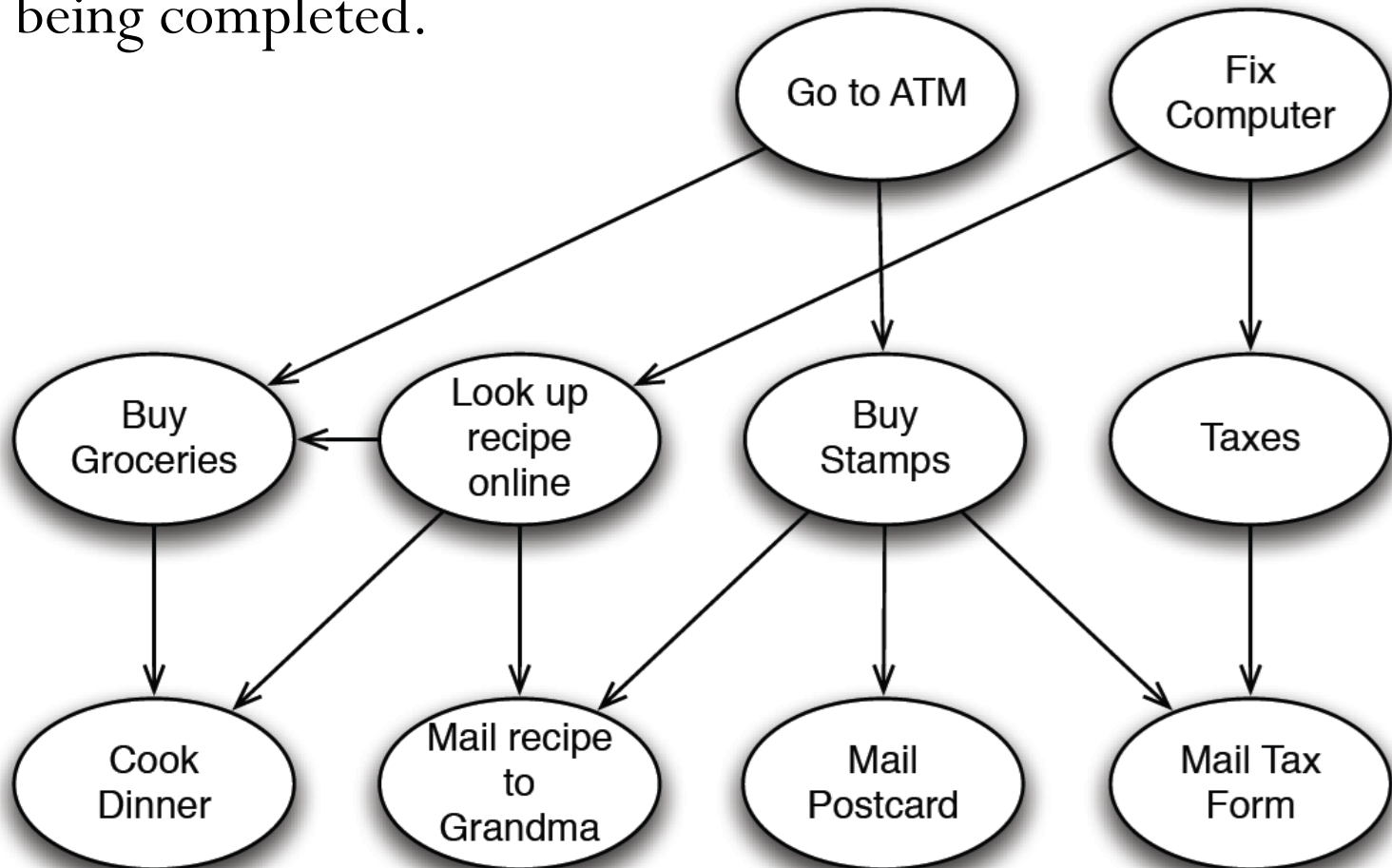
Topological Sorting
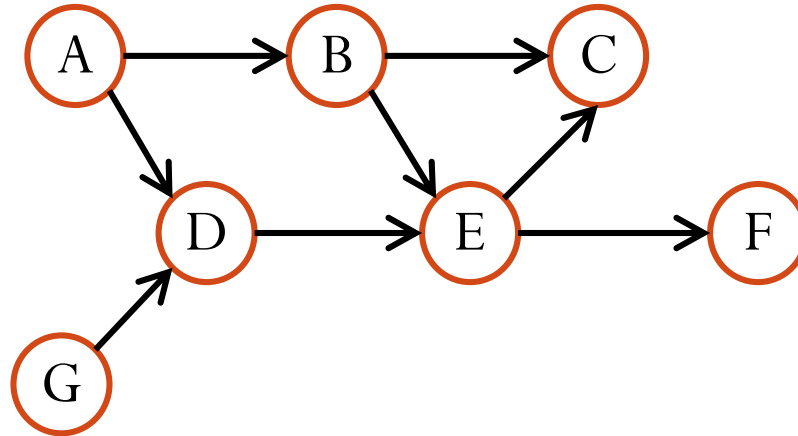
- Topological sorting is not necessarily **unique**:
  - A, G, D, B, E, C, F and A, B, G, D, E, F, C are both topological sorting.
- Are the following orderings topological sorting?
  - A, B, E, G, D, C, F
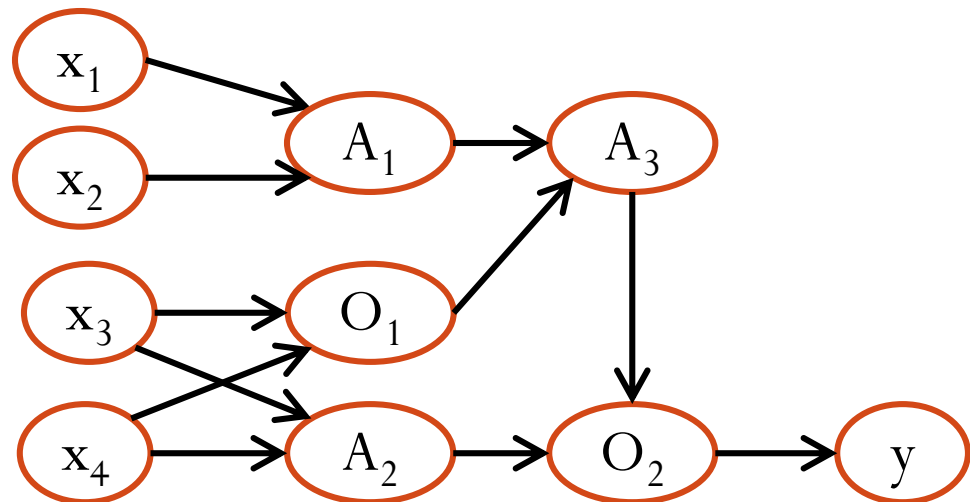 - A, G, B, D, E, F, C
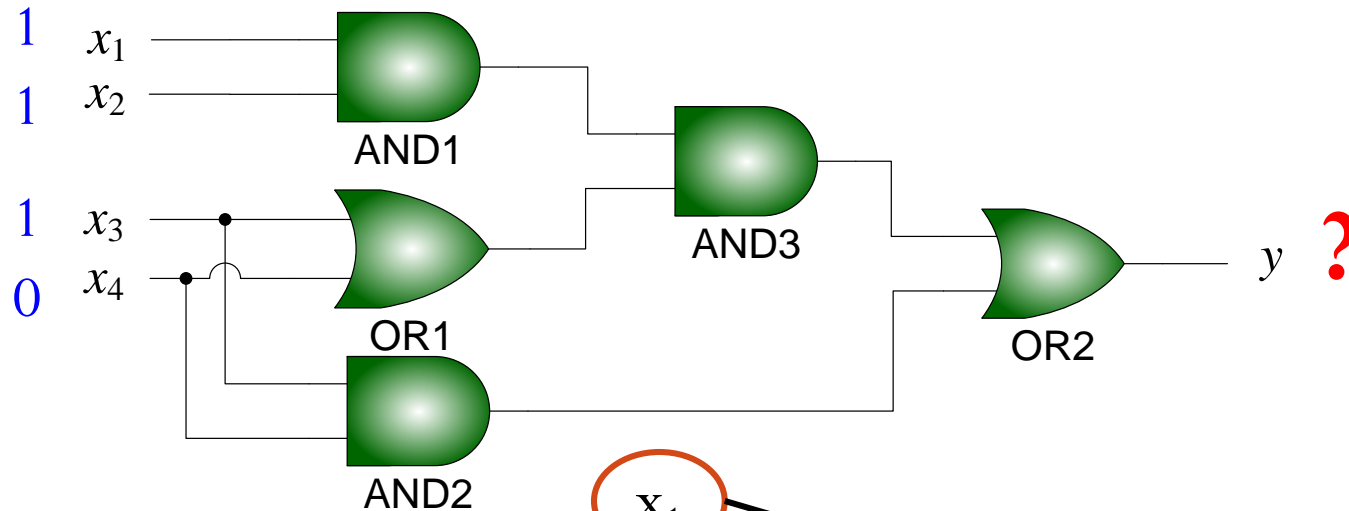
Topological Sorting
Applications

- Scheduling tasks when some tasks depend on other tasks being completed.

Topological Sorting
Applications

- Evaluating a combination logic circuit given a set of inputs.

Topological Sorting: Algorithm

- Based on a **queue**.

- Algorithm:　　　　　　　入度
    1. Compute the in-degrees of all nodes. (**in-degree**: number of **incoming** edges of a node.)
    2. **Enqueue** all in-degree 0 nodes into a queue.
    3. While queue is not empty
        1. **Dequeue** a node v from the queue and visit it.
        2. Decrement in-degrees of node v's neighbors.
        3. If any neighbor's in-degree becomes 0, **enqueue** it into the queue.

Topological Sorting Algorithm
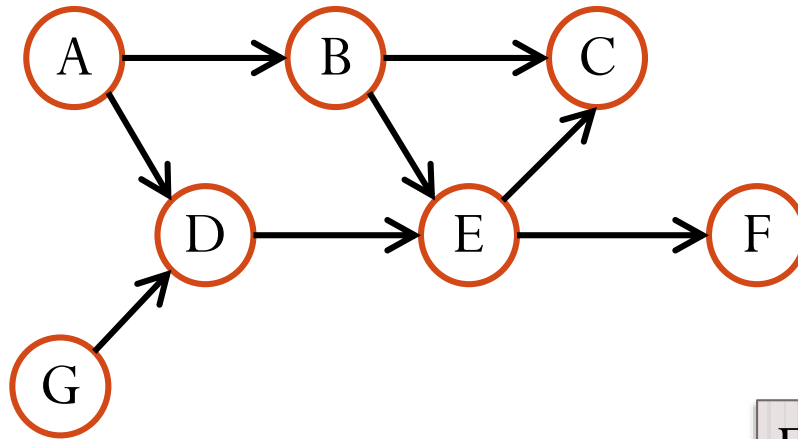
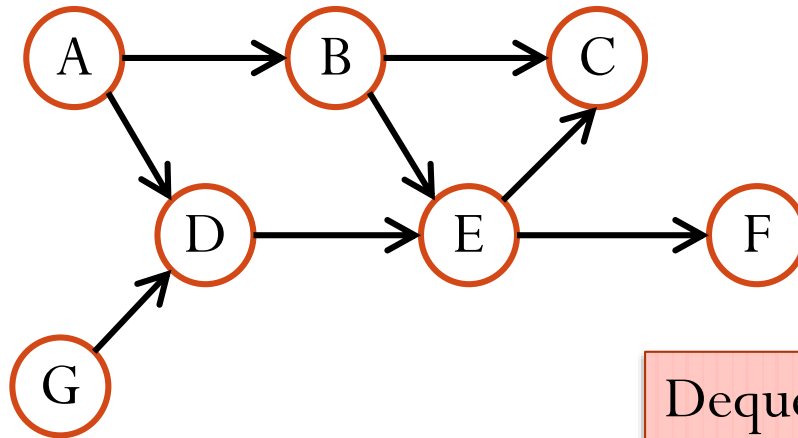*Example*

Queue

Enqueue A and G

In-degrees

A	B	C	D	E	F	G
0	1	2	2	2	1	0

Order

Topological Sorting Algorithm
Example

Queue

A
G

Dequeue A, visit A, and decrement in-degrees of A's neighbors.

In-degrees

A	B	C	D	E	F	G
0	1	2	2	2	1	0

Order

Topological Sorting Algorithm
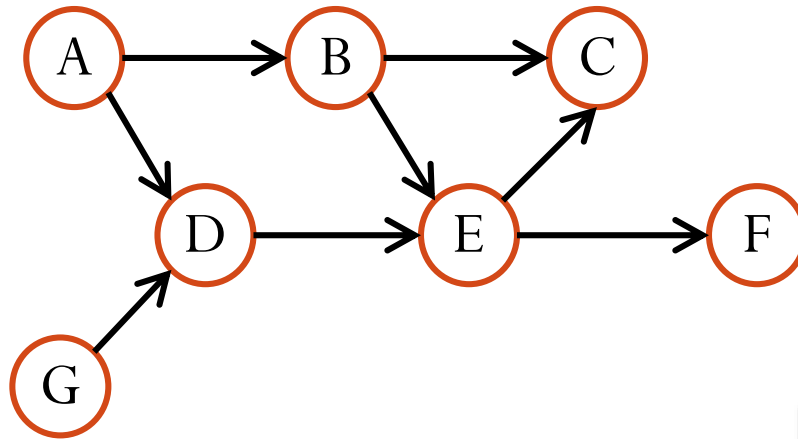
Example

Queue

G

Enqueue B

In-degrees

A	B	C	D	E	F	G
0	~~1~~ 0	2	~~2~~ 1	2	1	0

Order

A						

Topological Sorting Algorithm
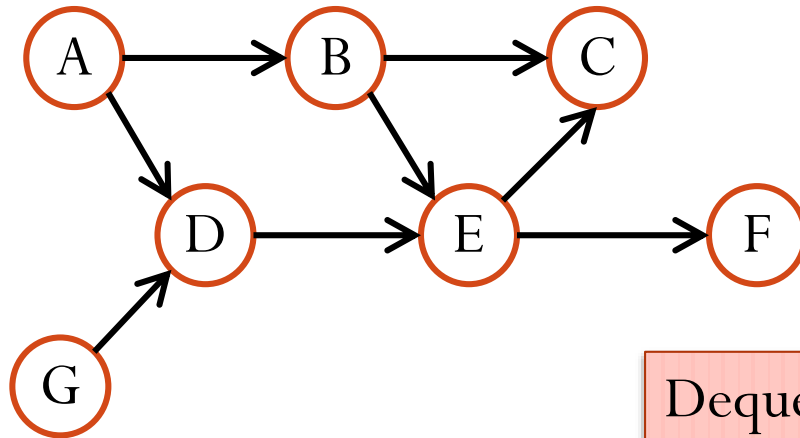
Example

Queue

G
B

Dequeue G, visit G, and decrement in-degrees of G's neighbors.

In-degrees

A	B	C	D	E	F	G
0	0	2	1	2	1	0

Order

A						

Topological Sorting Algorithm
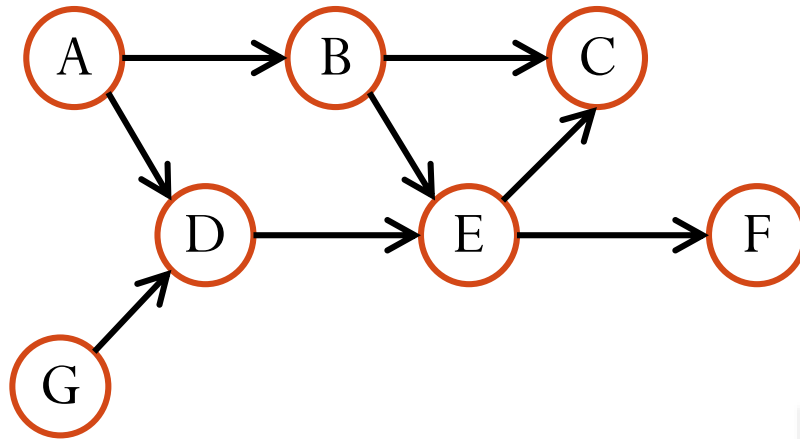
Example

Queue

B

Enqueue D

In-degrees

A	B	C	D	E	F	G
0	0	2	~~1~~ 0	2	1	0

Order

A	**G**					

Topological Sorting Algorithm
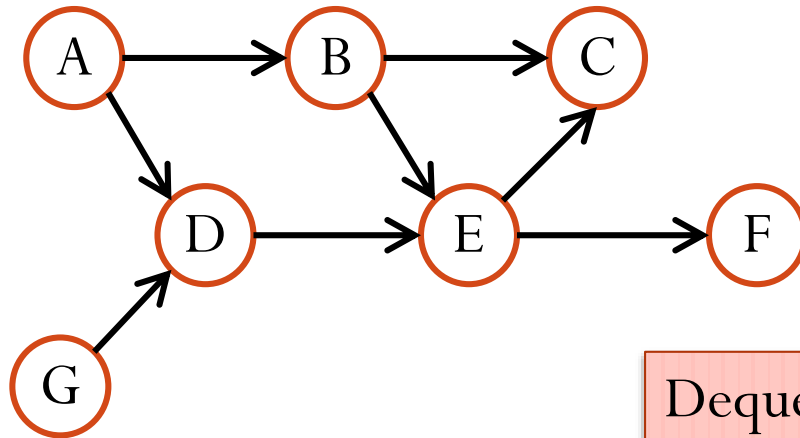
Example

Queue

B
D

Dequeue B, visit B, and decrement in-degrees of B's neighbors.

In-degrees

A	B	C	D	E	F	G
0	0	2	0	2	1	0

Order

A	G					

Topological Sorting Algorithm
Example

Queue

D

Dequeue D, visit D, and decrement in-degrees of D's neighbors.
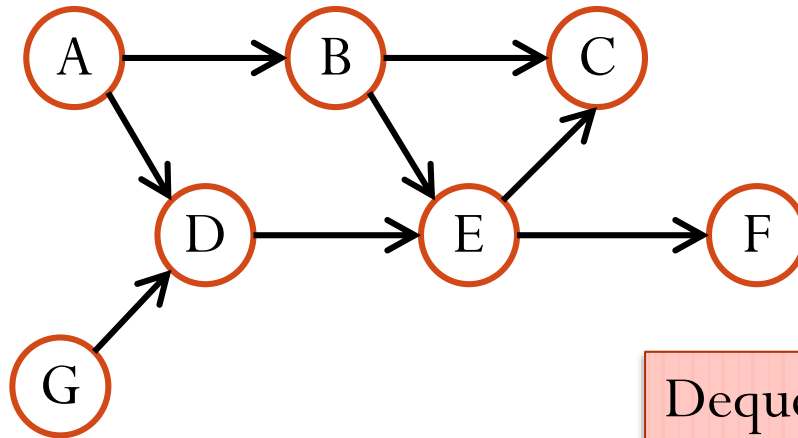
In-degrees

A	B	C	D	E	F	G
0	0	2̶ 1	0	2̶ 1	1	0

Order

A	G	**B**				

Topological Sorting Algorithm
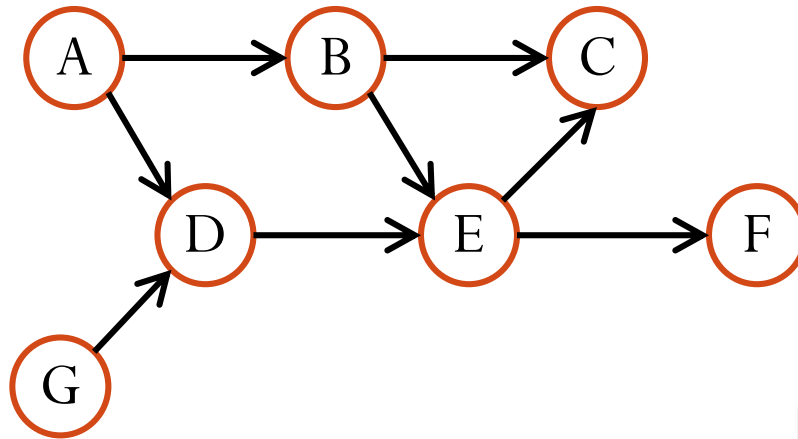
Example

Queue

Enqueue E

In-degrees

A	B	C	D	E	F	G
0	0	1	0	~~1~~ 0	1	0

Order

A	G	B	D			

Topological Sorting Algorithm
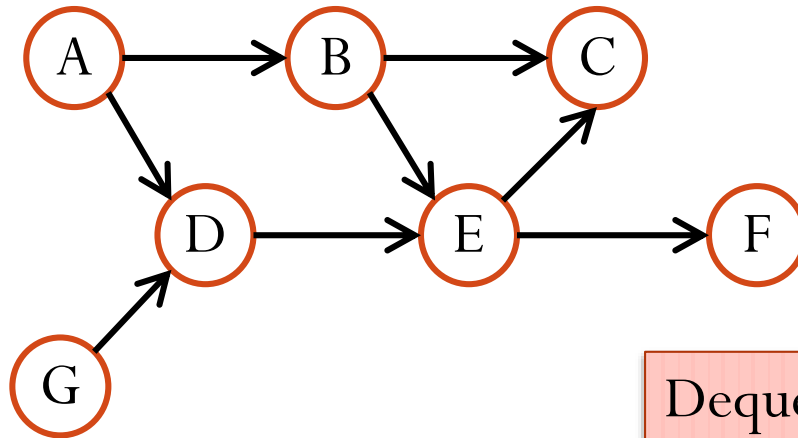
Example

Queue

E

Dequeue E, visit E, and decrement in-degrees of E's neighbors.
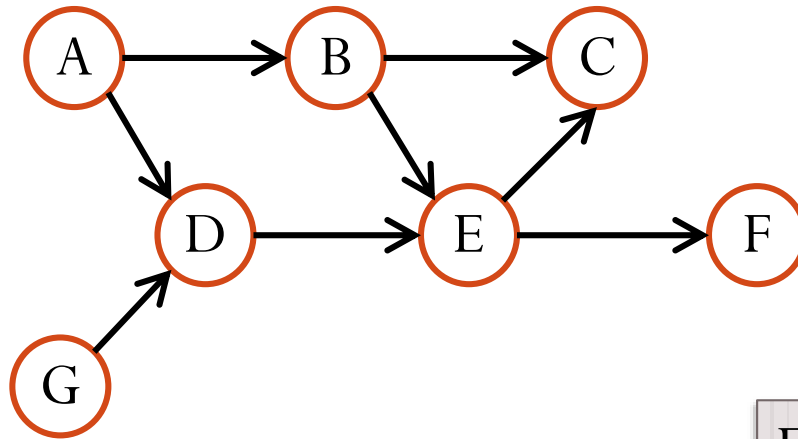
In-degrees

A	B	C	D	E	F	G
0	0	1	0	0	1	0

Order

A	G	B	D			

Topological Sorting Algorithm
Example

Queue

Enqueue C and F

In-degrees

A	B	C	D	E	F	G
0	0	~~1~~ 0	0	0	~~1~~ 0	0

Order

A	G	B	D	E		

Topological Sorting Algorithm
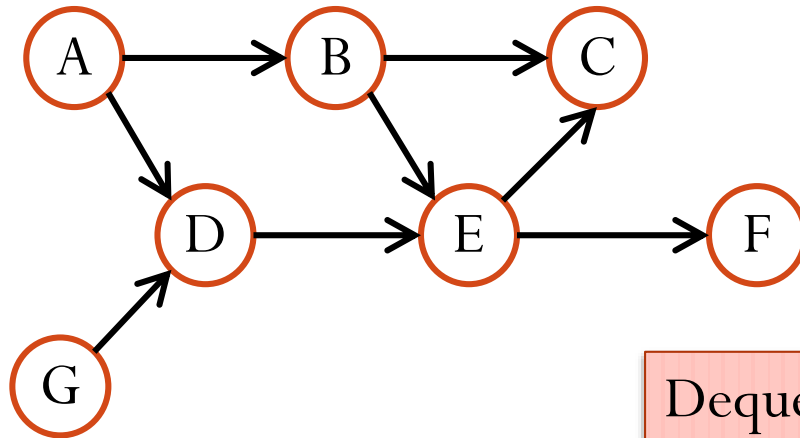
Example

Queue

C
F

Dequeue C, visit C, and decrement in-degrees of C's neighbors.

In-degrees

A	B	C	D	E	F	G
0	0	0	0	0	0	0

Order

A	G	B	D	E		

Topological Sorting Algorithm

Example

Queue

F

Dequeue F, visit F, and decrement in-degrees of F's neighbors.
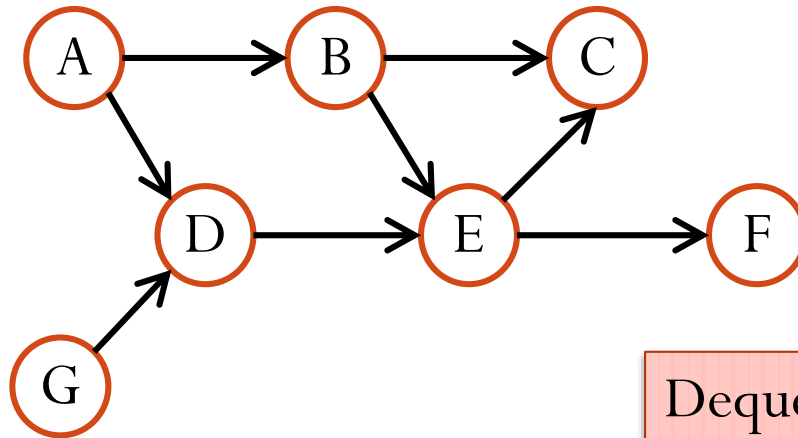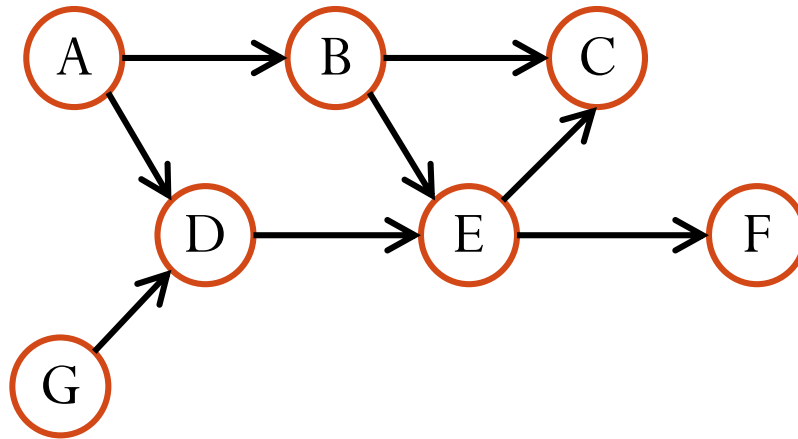
In-degrees

A	B	C	D	E	F	G
0	0	0	0	0	0	0

Order

A	G	B	D	E	C	

Topological Sorting Algorithm

Example

Queue

Queue is now empty. Done!

In-degrees

A	B	C	D	E	F	G
0	0	0	0	0	0	0

Order

A	G	B	D	E	C	F

Topological Sorting Algorithm
Time Complexity

Assume adjacency list representation

1. Compute the in-degrees of all nodes. $O(|V| + |E|)$ in total

2. Enqueue all in-degree 0 nodes into a queue. $O(|V|)$ in total

3. While queue is not empty

    1. Dequeue a node v from the queue and visit it. $O(|V|)$ in total

    2. Decrement in-degrees of node v's neighbors. $O(|E|)$ in total

    3. If any neighbor's in-degree becomes 0 …

        • … place it in the queue. $O(|V|)$ in total

Total running time is $O(|V| + |E|)$.