Topological Sort

Kyunghan Lee
Networked Computing Lab (NXC Lab)
Department of Electrical and Computer Engineering
Seoul National University
https://nxc.snu.ac.kr
kyunghanlee@snu.ac.kr

Topological Sort

- In this topic, we will discuss:
  - Motivations
  - Review the definition of a directed acyclic graph (DAG)
  - Describe a topological sort and applications
  - Kahn’s algorithm

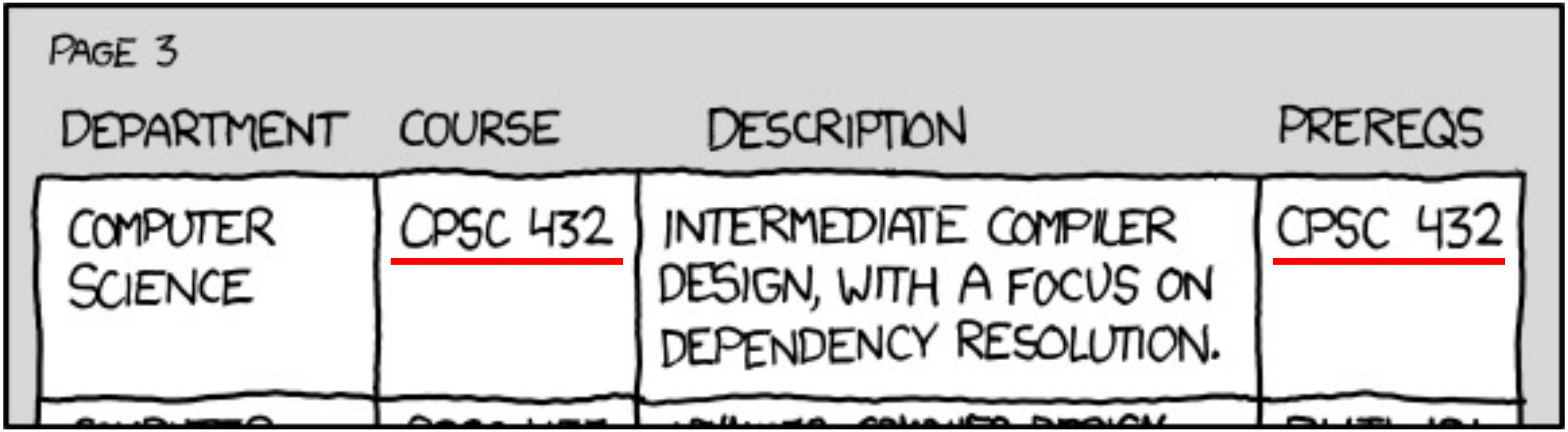
Motivation

- □ Given a set of tasks with dependencies, is there an order in which we can complete the tasks?

- □ Dependencies form a partial ordering
  - ▪ A partial ordering on a finite number of objects can be represented as a directed acyclic graph (DAG)

Motivation

□ Cycles in dependencies can cause issues…



PAGE 3

DEPARTMENT	COURSE	DESCRIPTION	PREREQS
COMPUTER SCIENCE	CPSC 432	INTERMEDIATE COMPILER DESIGN, WITH A FOCUS ON DEPENDENCY RESOLUTION.	CPSC 432

http://xkcd.com/754/

Restriction of paths in DAGs

- □ **Claim:**

  In a DAG, given two different vertices v_j and v_k, there cannot both be a path from v_j to v_k and a path from v_k to v_j

- □ **Proof by contradiction:**

  Assume otherwise; thus there exists two paths:

  $(v_j, v_{1,1}, v_{1,2}, v_{1,3}, \cdots, v_k)$

  $(v_k, v_{2,1}, v_{2,2}, v_{2,3}, \cdots, v_j)$

  From this, we can construct the path

  $(v_j, v_{1,1}, v_{1,2}, v_{1,3}, \cdots, v_k, v_{2,1}, v_{2,2}, v_{2,3}, \cdots, v_j)$

  This path is a cycle, but it is assumed a DAG

  ∴ contradiction

Definition of topological sorting

- A topological sorting of the vertices in a DAG is an ordering

$$v_1, v_2, v_3, \ldots, v_{|V|}$$

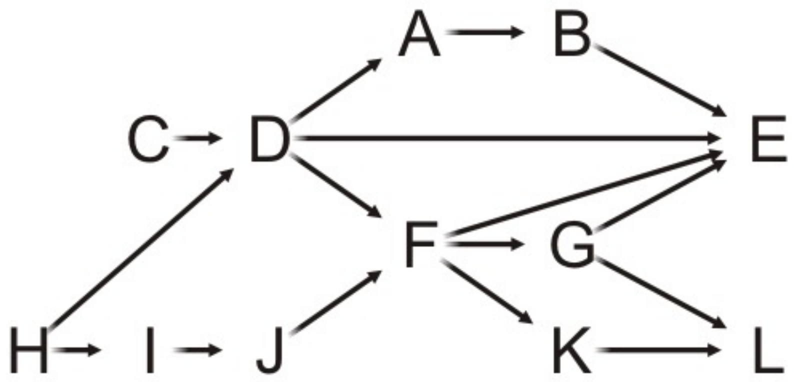
such that v_j should appear before v_k

if there is a path from v_j to v_k

Definition of topological sorting

- □ Given this DAG, a topological sort is

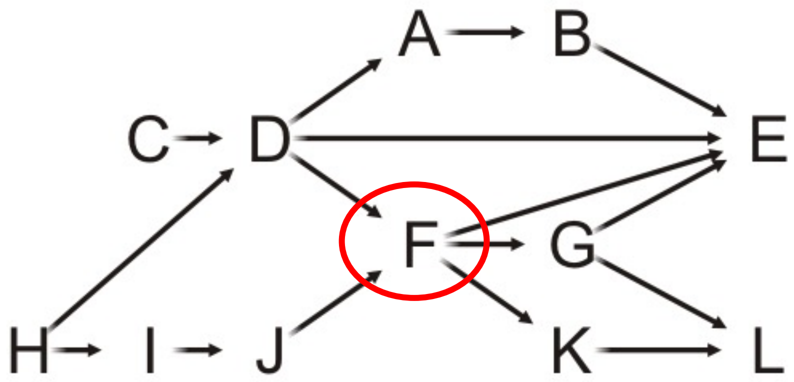
H, C, I, D, J, A, F, B, G, K, E, L



Example

- □ For example, there are paths from H, C, I, D, and J to F, so all these must come before F in a topological sort

H, C, I, D, J, A, F, B, G, K, E, L



- □ Clearly, this sorting need not be unique

Applications

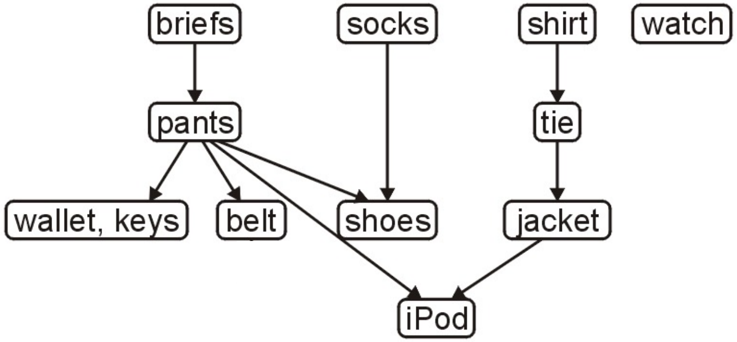
- Consider the course instructor getting ready for a dinner out

- He must wear the following:
  - jacket, shirt, briefs, socks, tie, etc.

- There are certain constraints:
  - the pants really should go on after the briefs,
  - socks are put on before shoes

Applications

- The following is a task graph for getting dressed:



- One topological sort is:
  - briefs, pants, wallet, keys, belt, socks, shoes, shirt, tie, jacket, iPod, watch

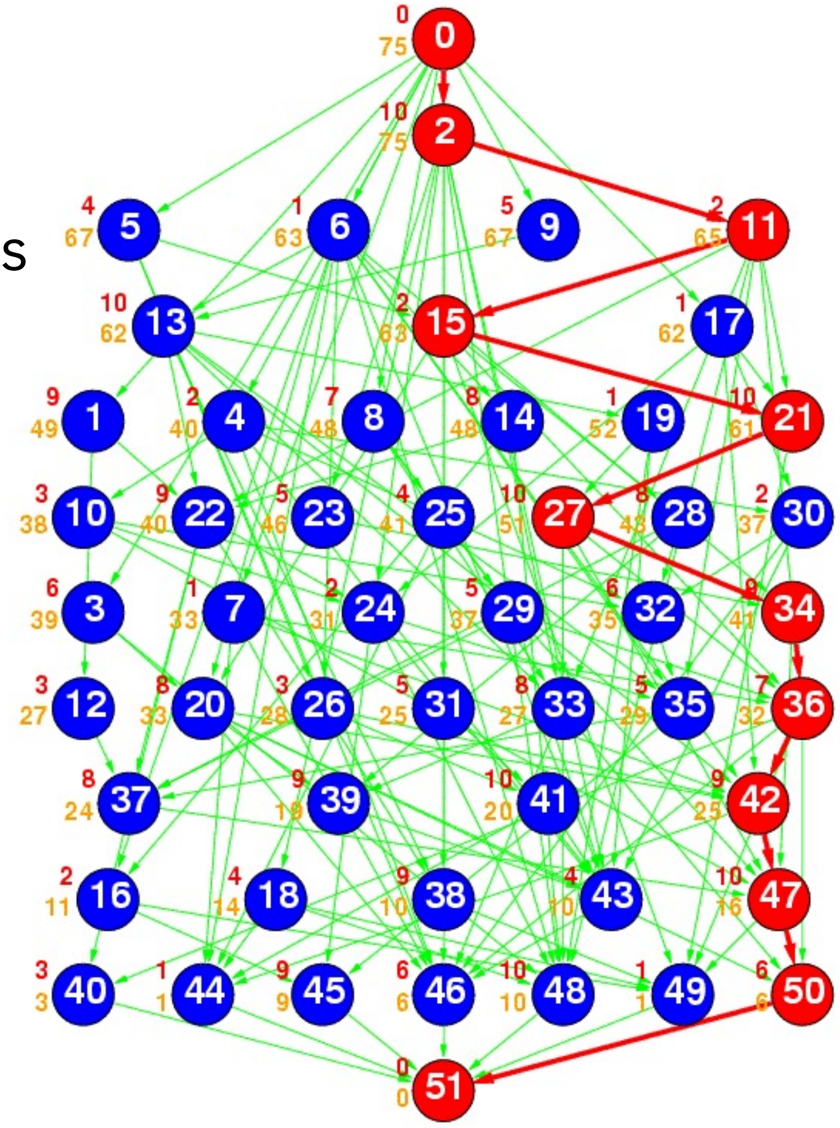
- A more reasonable topological sort is:
  - briefs, socks, pants, shirt, belt, tie, jacket, wallet, keys, iPod, watch, shoes

Applications

- □ C++ header and source files have #include statements
  - A change to an included file requires a recompilation of the current file
  - On a large project, it is desirable to recompile only those source files that depended on those files which changed
  - For large software projects, full compilations may take hours

Applications

- The following is a DAG representing a number of tasks
  - The green arrows represent dependencies
  - The numbering indicates a topological sort of the tasks



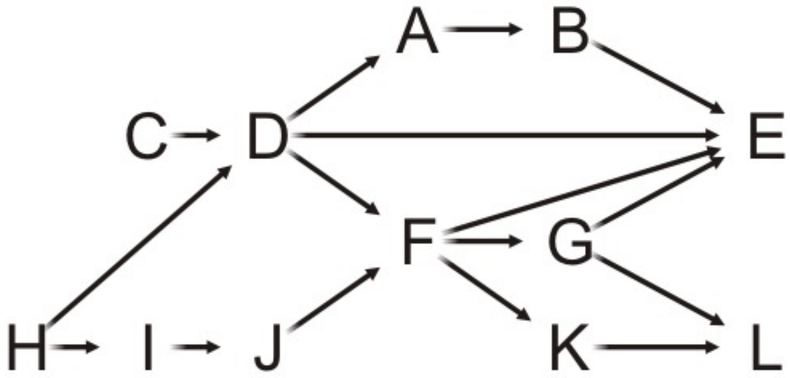
Ref: The Standard Task Graph
http://www.kasahara.elec.waseda.ac.jp/schedule/

Idea: How to Solve Topological Sort

- Idea:
  - Given a DAG *V*, make a copy *W* and iterate:
    - Find a vertex *v* in *W* with in-degree zero
    - Let *v* be the next vertex in the topological sort
    - Continue iterating with the vertex-induced sub-graph *W* \ {*v*}

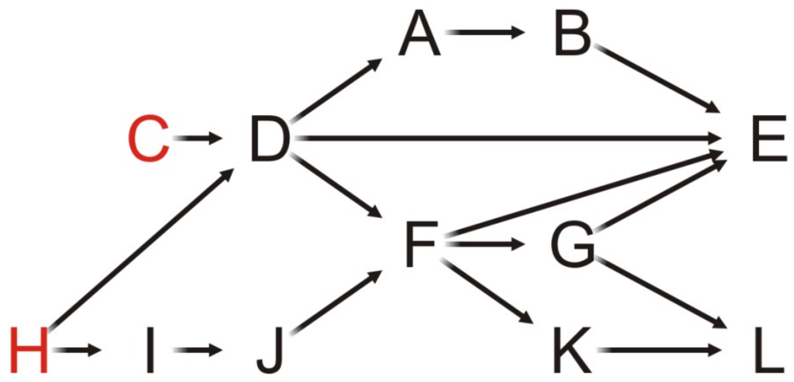
Idea: Example

- On this graph, iterate the following |*V*| = 12 times
  - Choose a vertex *v* that has in-degree zero
  - Let *v* be the next vertex in our topological sort
  - Remove *v* and all edges connected to it



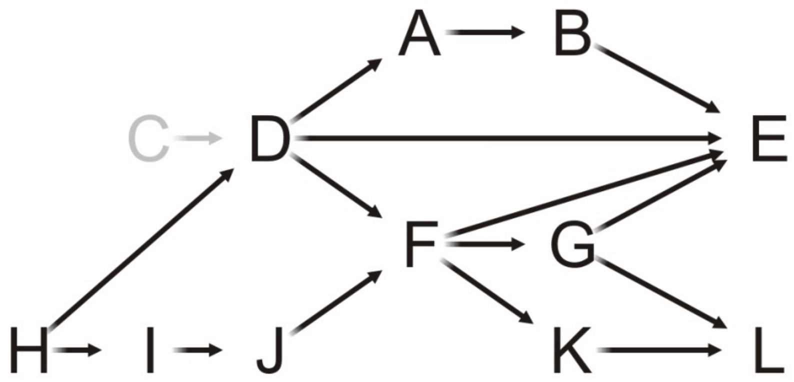
Idea: Example

- Let's step through this algorithm with this example
  - Which task can we start with?
  - Of Tasks C or H, choose Task C



Idea: Example

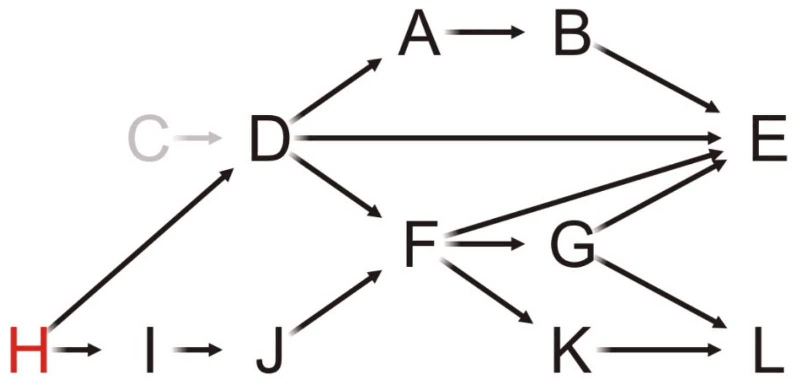
- □ Having completed Task C, which vertices have in-degree zero?



C

Idea: Example

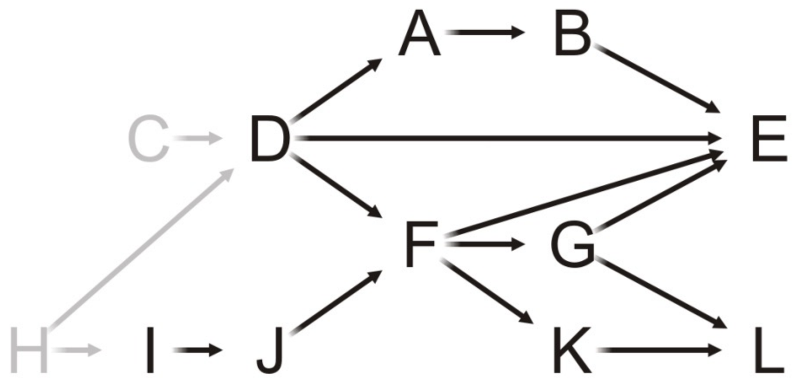
- □ Only Task H can be completed, so we choose it



C

Idea: Example

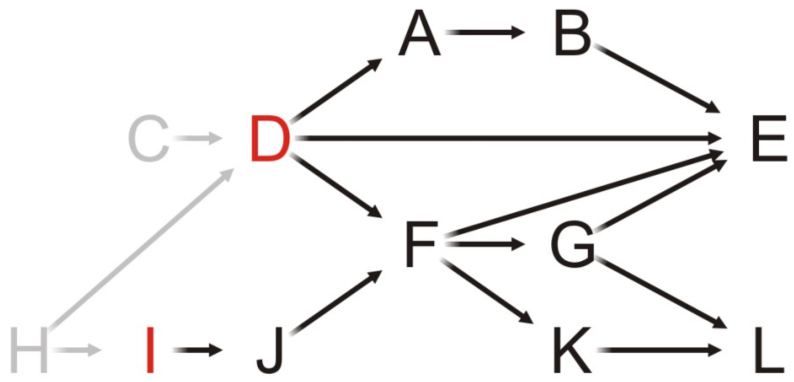
- Having removed H, what is next?



C, H

Idea: Example

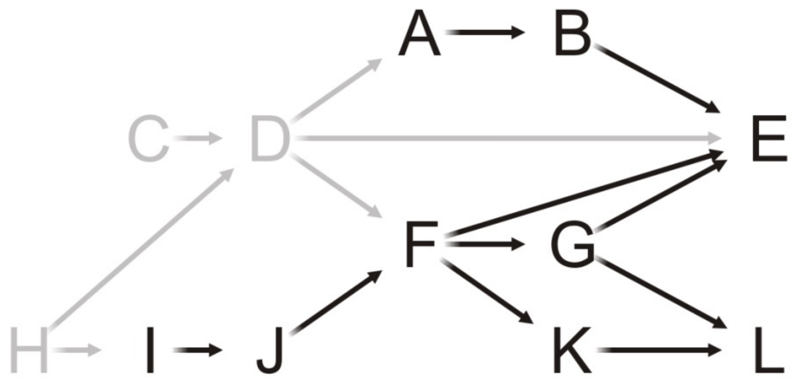
- Both Tasks D and I have in-degree zero
  - Let us choose Task D



C, H

Idea: Example

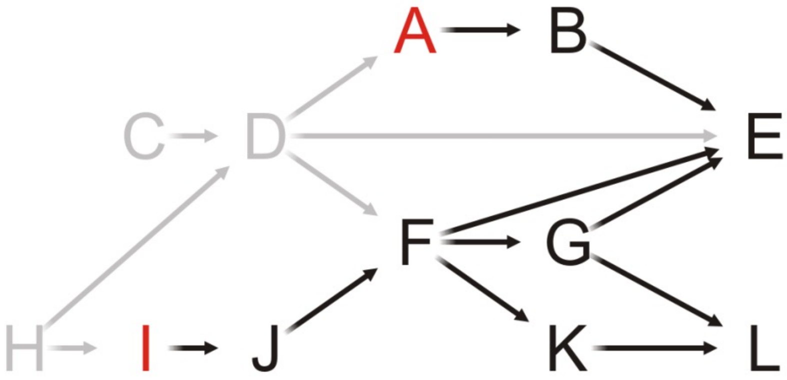
- We remove Task D, and now?



C, H, D

Idea: Example

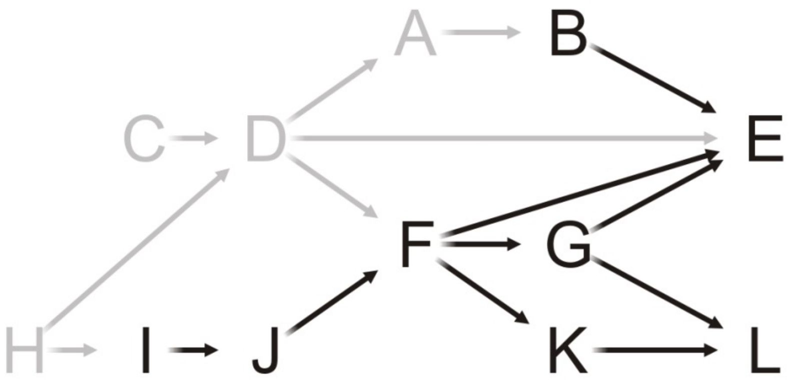
- □ Both Tasks A and I have in-degree zero
  - ▪ Let's choose Task A



C, H, D

Idea: Example

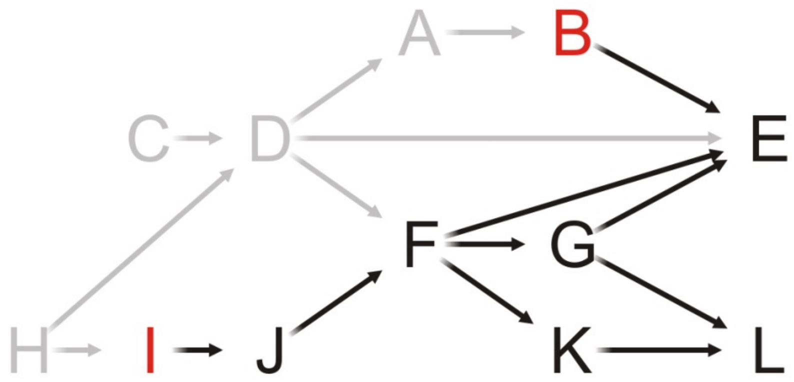
- Having removed A, what now?



C, H, D, A

Idea: Example

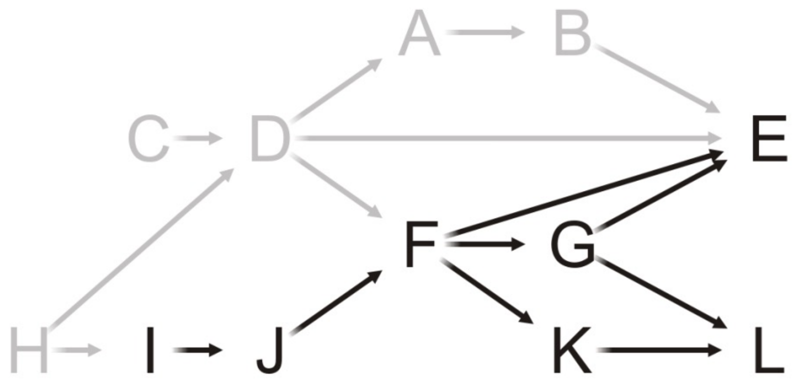
- Both Tasks B and I have in-degree zero
  - Choose Task B



C, H, D, A

Idea: Example

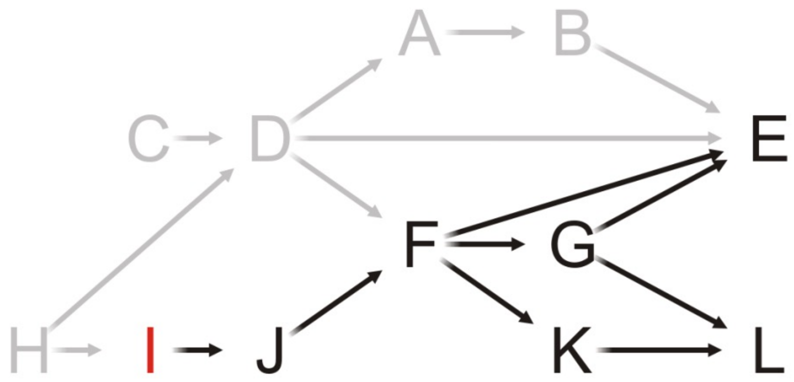
- □ Removing Task B, we note that Task E still has an in-degree of two
  - ▪ Next?



C, H, D, A, B

Idea: Example

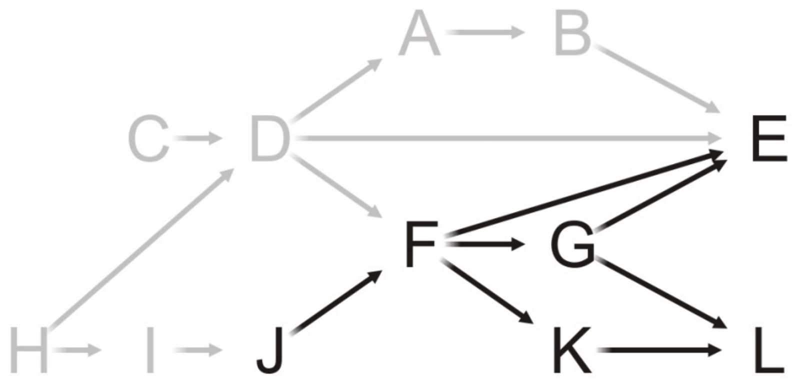
- □ As only Task I has in-degree zero, we choose it



C, H, D, A, B

Idea: Example

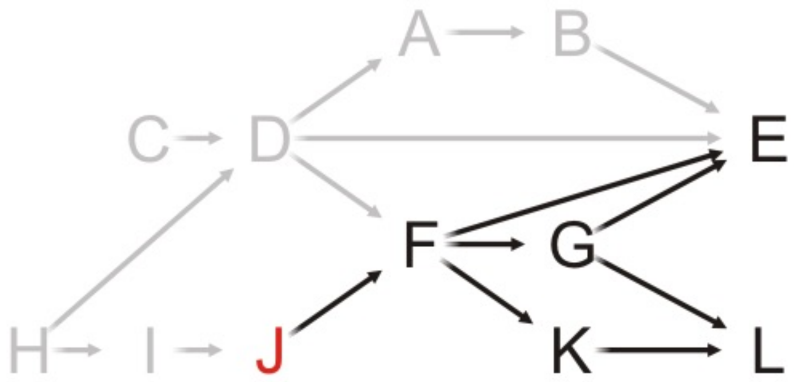
- □ Having completed Task I, what now?



C, H, D, A, B, I

Idea: Example

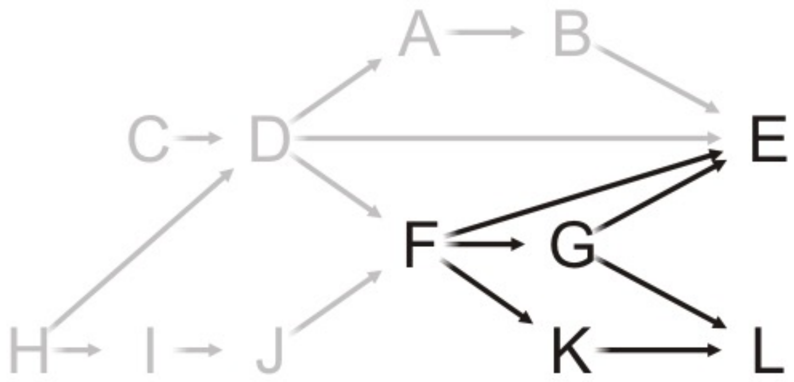
- Only Task J has in-degree zero: choose it



C, H, D, A, B, I

Idea: Example

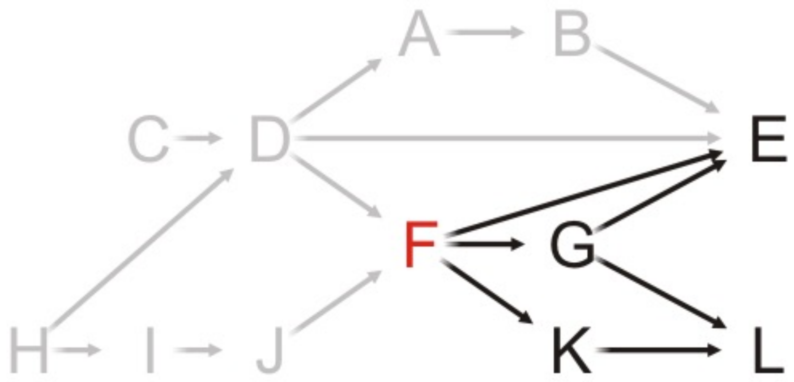
- □ Having completed Task J, what now?



C, H, D, A, B, I, J

Idea: Example

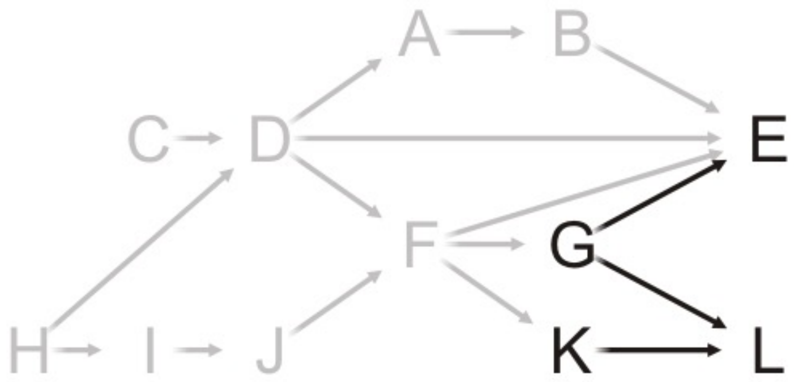
- Only Task F can be completed, so choose it



C, H, D, A, B, I, J

Idea: Example

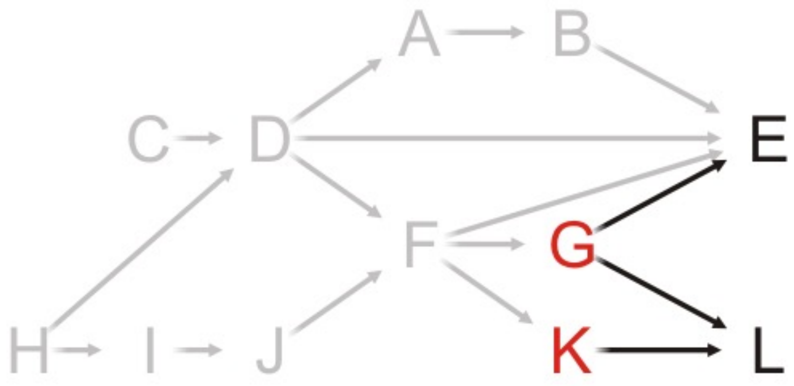
- □ What choices do we have now?



C, H, D, A, B, I, J, F

Idea: Example

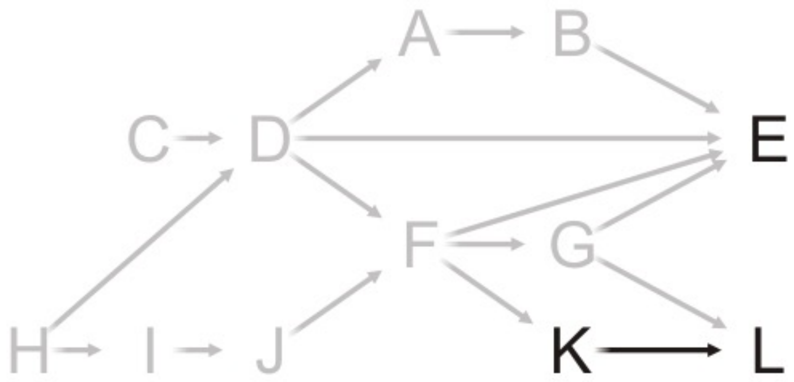
- We can perform Tasks G or K
  - Choose Task G



C, H, D, A, B, I, J, F

Idea: Example

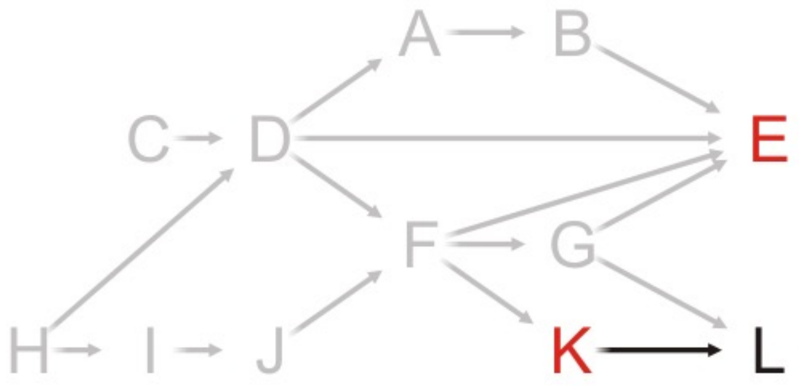
- □ Having removed Task G from the graph, what next?



C, H, D, A, B, I, J, F, G

Idea: Example

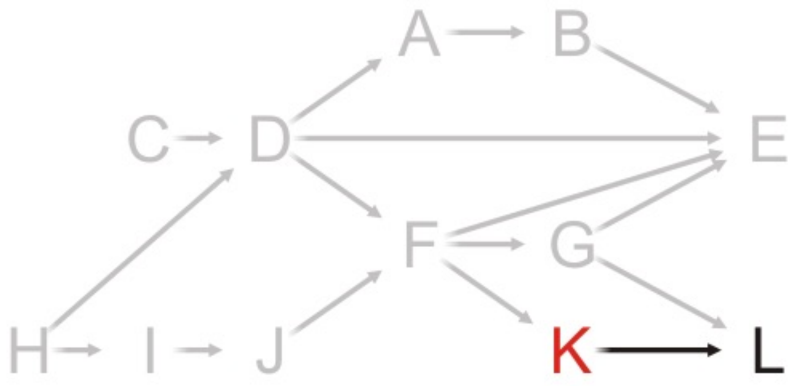
- □ Choosing between Tasks E and K, choose Task E



C, H, D, A, B, I, J, F, G

Idea: Example

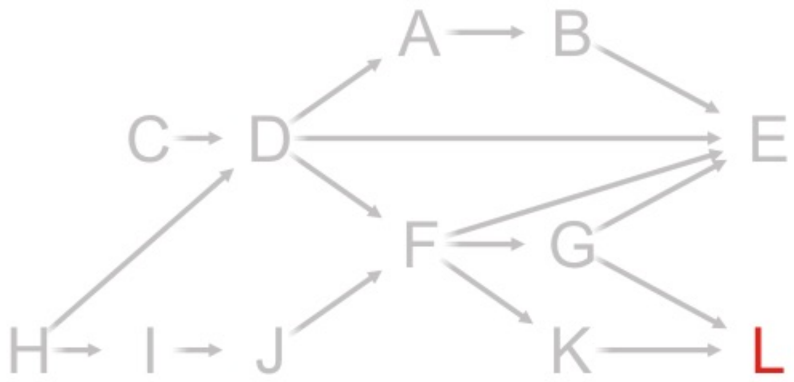
- At this point, Task K is the only one that can be run



C, H, D, A, B, I, J, F, G, E

Idea: Example

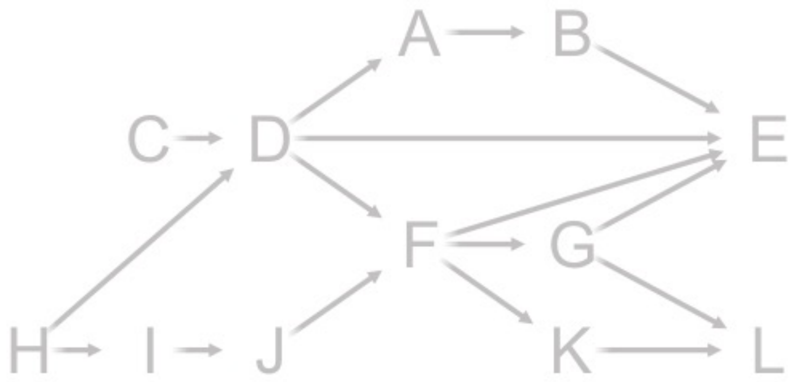
- □ And now that both Tasks G and K are complete, we can complete Task L



C, H, D, A, B, I, J, F, G, E, K

Idea: Example

- □ There are no more vertices left

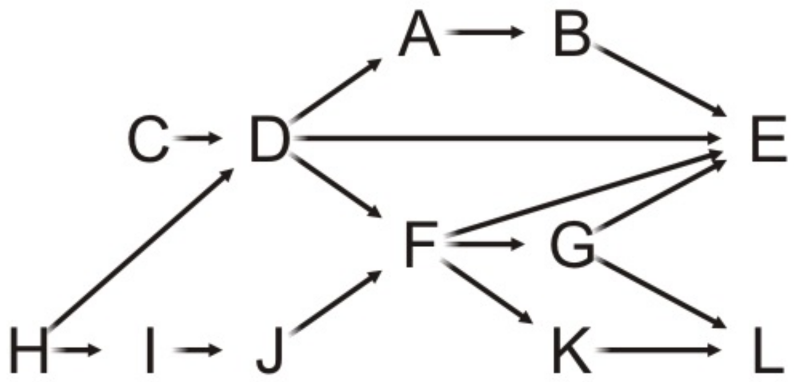


C, H, D, A, B, I, J, F, G, E, K, L

Idea: Example

- Thus, one possible topological sort would be:

  C, H, D, A, B, I, J, F, G, E, K, L

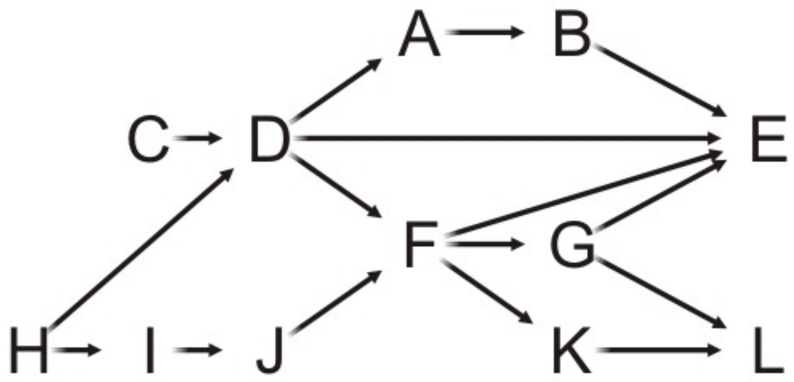


Idea: Example

- □ Note that topological sorts need not be unique:

C, H, D, A, B, I, J, F, G, E, K, L

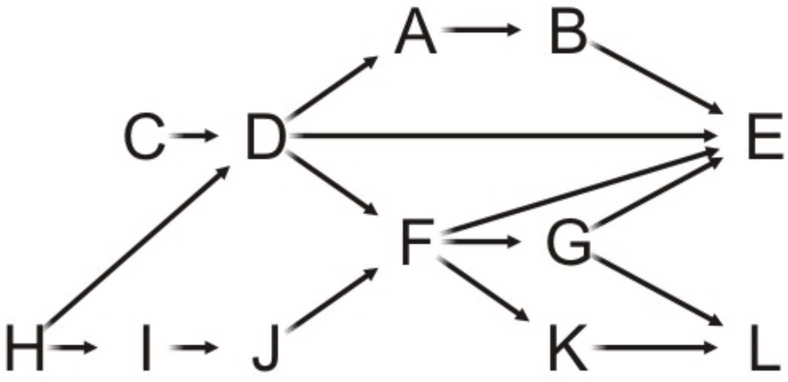
H, I, J, C, D, F, G, K, L, A, B, E



Kahn’s Algorithm

- □ Kahn’s algorithm solves the topological sort problem

- □ **Step #1: Preparing In-degree array**
  - ▪ Construct an array, maintaining the in-degrees of each vertex
  - ▪ Requires Θ(|*V* |) memory



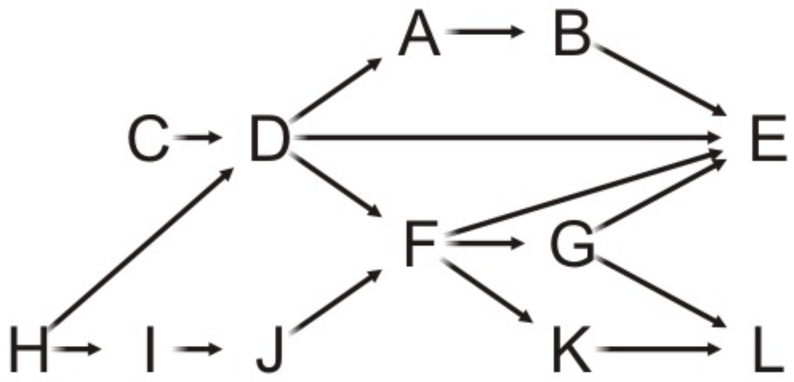
A	1
B	1
C	0
D	2
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

Kahn’s Algorithm

- □ **Step #2: Enumerating in-degree array**
  - #2-A. Prepare an empty queue
  - #2-B. Enqueue all the vertices with the in-degree of zero
  - #2-C. While the queue is not empty
    - Dequeue a vertex
    - Add this vertex to the sequence of topological sort
    - Decrement the in-degree of all its neighboring vertices
    - Enqueue the neighboring vertices with the in-degree of zero

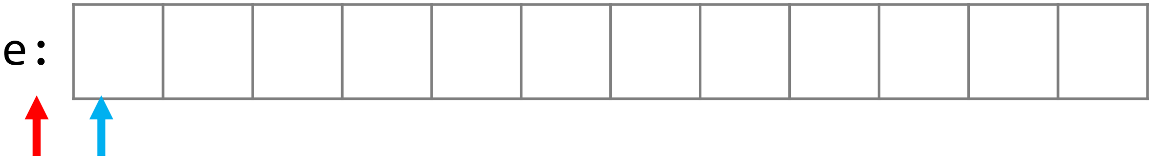
Example: Kahn’s Algorithm

- □ With the previous example, we initialize:
  - ▪ The array of in-degrees
  - ▪ The queue



A	1
B	1
C	0
D	2
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

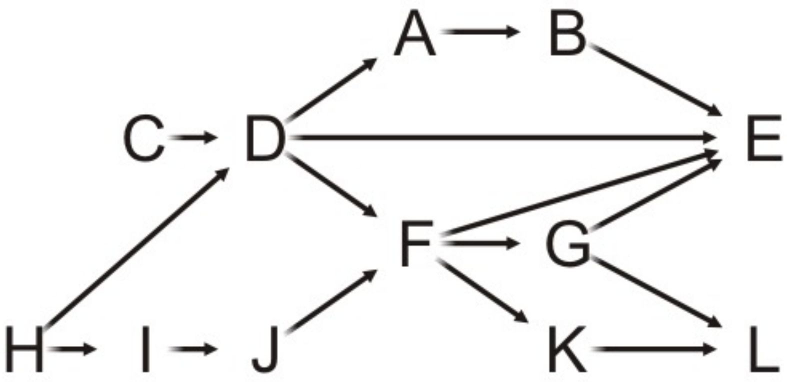
Queue:



The queue is empty

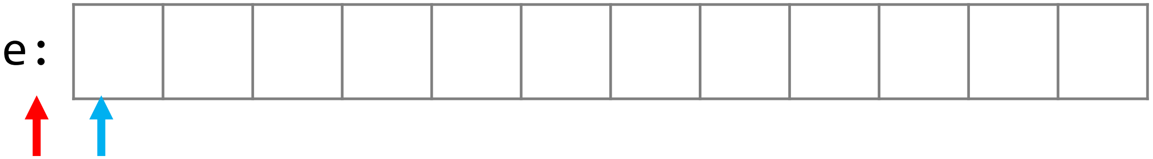
Example: Kahn’s Algorithm

□ Stepping through the table, push all source vertices into the queue



A	1
B	1
C	0
D	2
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

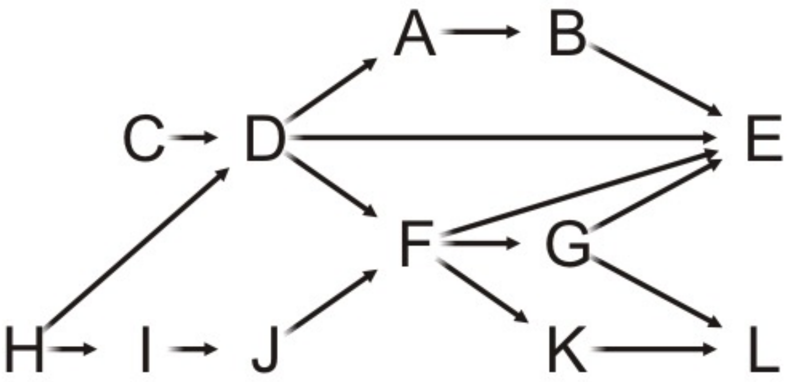
Queue:



The queue is empty

Example: Kahn’s Algorithm

□ Stepping through the table, push all source vertices into the queue



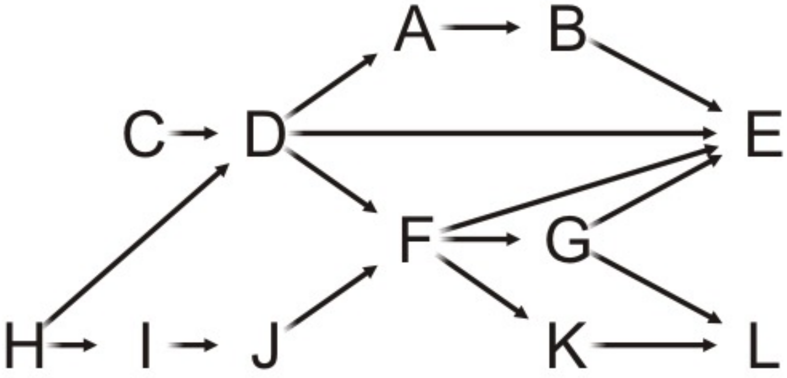
A	1
B	1
C	**0**
D	2
E	4
F	2
G	1
H	**0**
I	1
J	1
K	1
L	2

Queue:

C	**H**										

Example: Kahn’s Algorithm

□ Pop the front of the queue



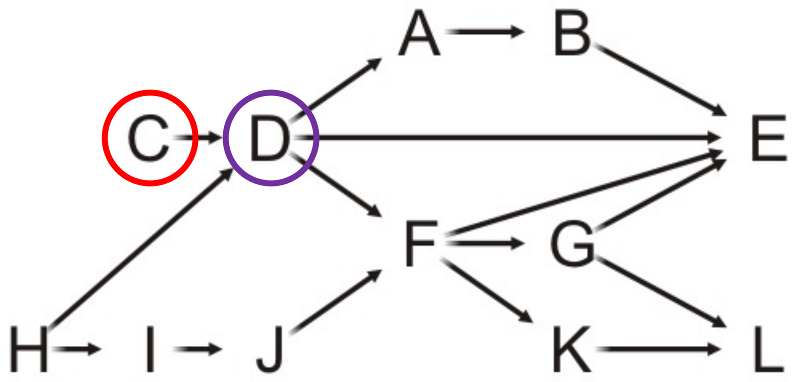
A	1
B	1
C	0
D	2
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

Queue:

C	H										
↑	↑										

Example: Kahn’s Algorithm

- Pop the front of the queue
  - C has one neighbor: D



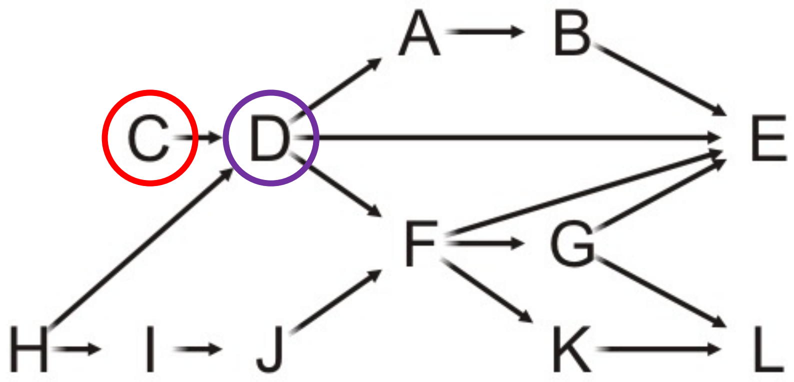
A	1
B	1
C	**0**
D	**2**
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

Queue:

C	H										

Example: Kahn’s Algorithm

- Pop the front of the queue
  - C has one neighbor: D
  - Decrement its in-degree



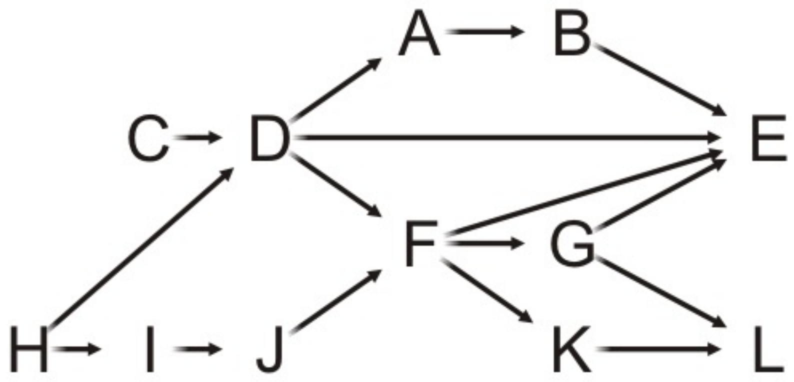
A	1
B	1
C	**0**
D	**1**
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

Queue:

C	H										

Example: Kahn’s Algorithm

□ Pop the front of the queue



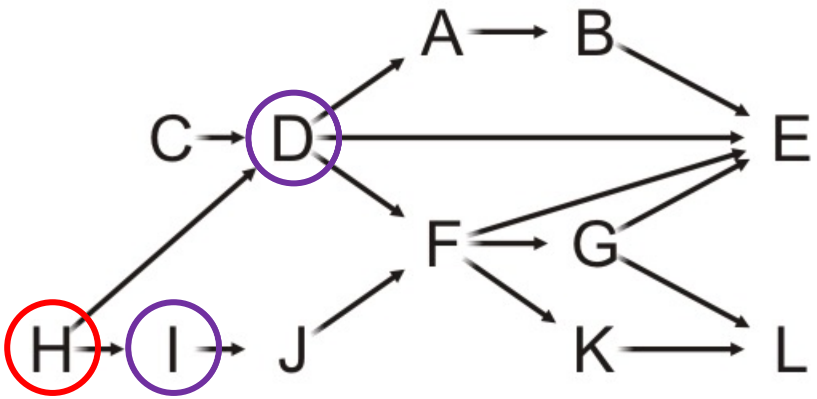
Queue:

C	H										

A	1
B	1
C	0
D	1
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

Example: Kahn’s Algorithm

- Pop the front of the queue
  - H has two neighbors: D and I

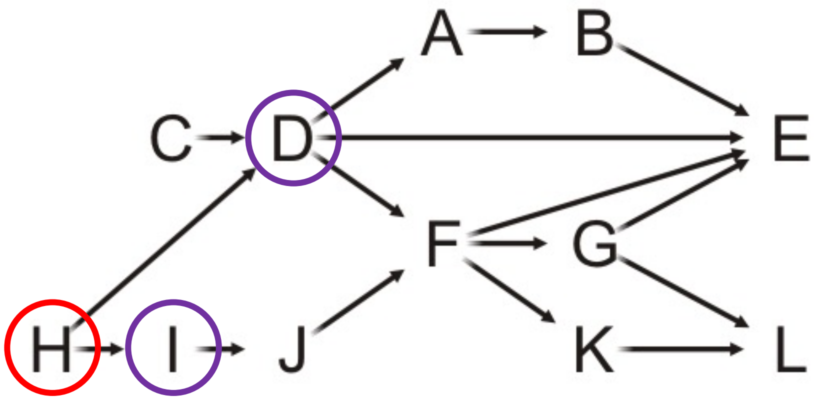


Queue:	C	H										

A	1
B	1
C	0
D	**1**
E	4
F	2
G	1
H	**0**
I	**1**
J	1
K	1
L	2

Example: Kahn’s Algorithm

- □ Pop the front of the queue
  - H has two neighbors: D and I
  - Decrement their in-degrees



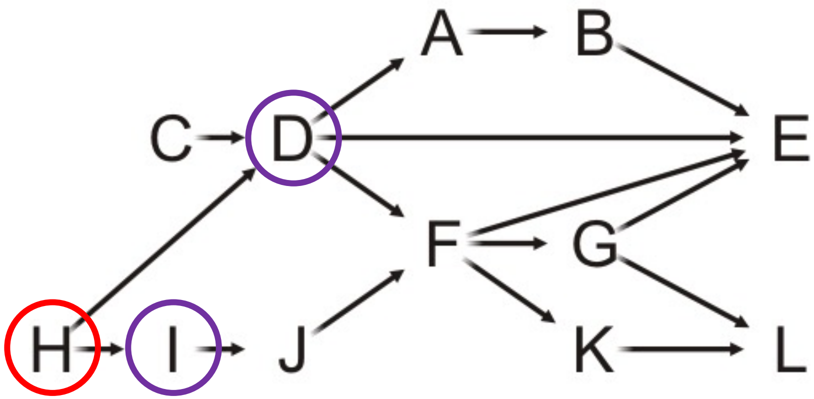
Queue:

C	H										

A	1
B	1
C	0
D	**0**
E	4
F	2
G	1
H	**0**
I	**0**
J	1
K	1
L	2

Example: Kahn's Algorithm

- Pop the front of the queue
  - H has two neighbors: D and I
  - Decrement their in-degrees
    - Both are decremented to zero, so push them onto the queue

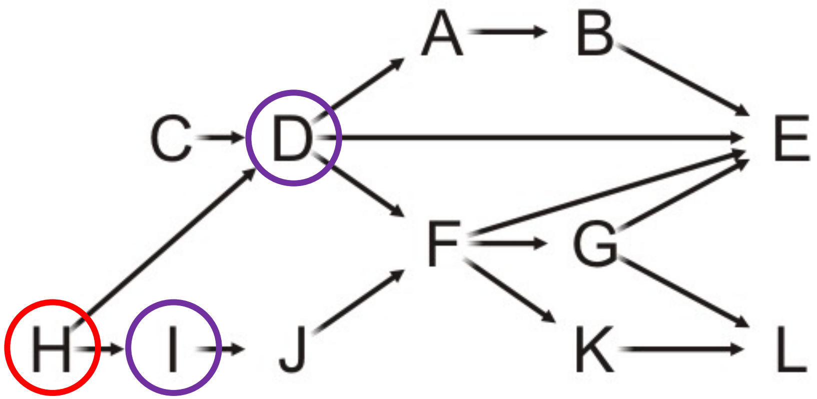


Queue:	C	H										

A	1
B	1
C	0
D	**0**
E	4
F	2
G	1
H	**0**
I	**0**
J	1
K	1
L	2

Example: Kahn's Algorithm

- □ Pop the front of the queue
  - ▪ H has two neighbors: D and I
  - ▪ Decrement their in-degrees
    - Both are decremented to zero, so push them onto the queue



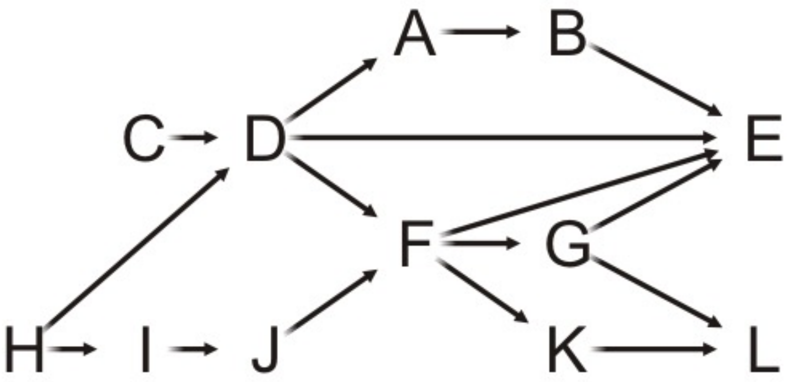
Queue:

C	H	D	I								
		↑	↑								

A	1
B	1
C	0
D	**0**
E	4
F	2
G	1
H	**0**
I	**0**
J	1
K	1
L	2

Example: Kahn's Algorithm

□ Pop the front of the queue



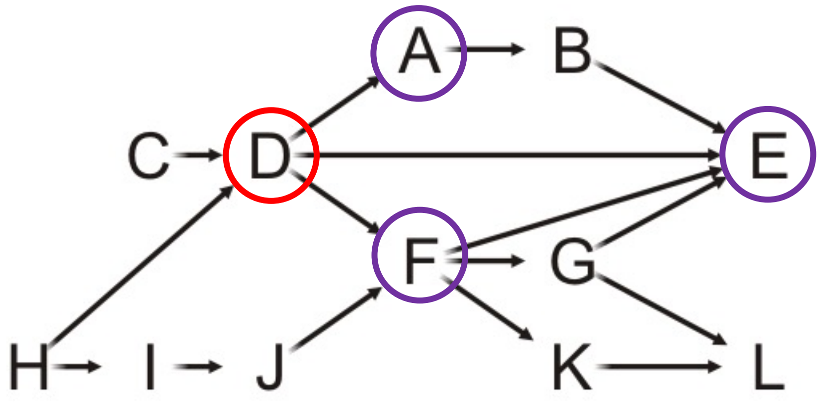
A	1
B	1
C	0
D	0
E	4
F	2
G	1
H	0
I	0
J	1
K	1
L	2

Queue:

C	H	D	I								

Example: Kahn’s Algorithm

- Pop the front of the queue
  - D has three neighbors: A, E and F



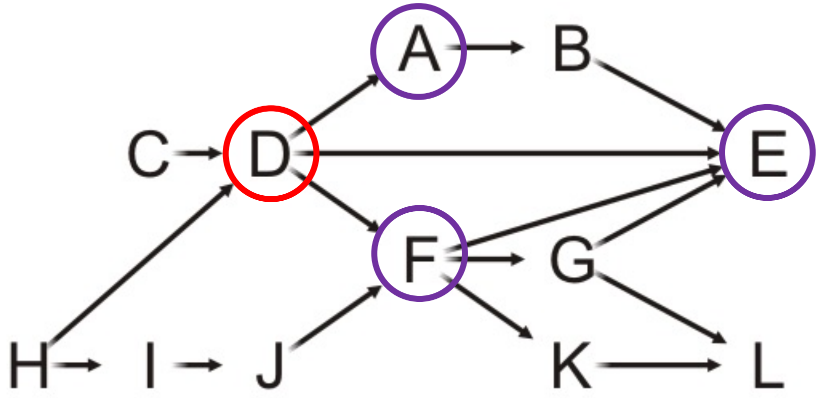
A	1
B	1
C	0
D	0
E	4
F	2
G	1
H	0
I	0
J	1
K	1
L	2

Queue:

C	H	D	I								

Example: Kahn's Algorithm

- Pop the front of the queue
  - D has three neighbors: A, E and F
  - Decrement their in-degrees



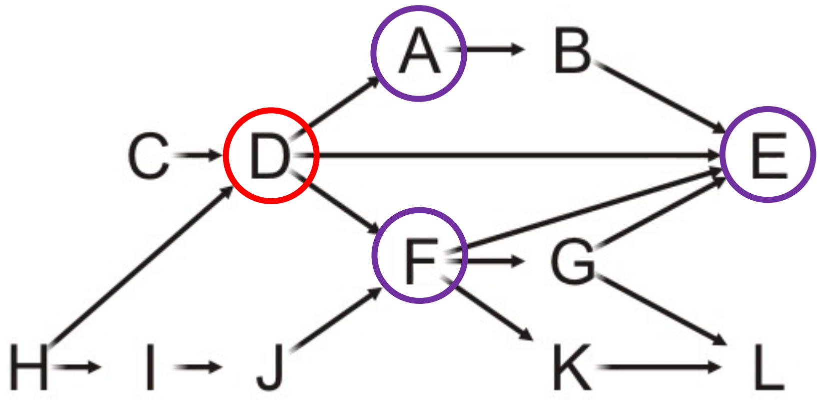
A	0
B	1
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	1
K	1
L	2

Queue:

C	H	D	I								

Example: Kahn’s Algorithm

- Pop the front of the queue
  - D has three neighbors: A, E and F
  - Decrement their in-degrees
    - A is decremented to zero, so push it onto the queue



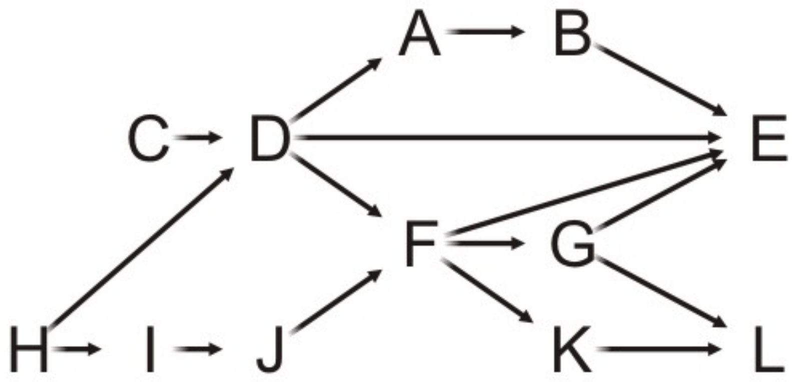
A	0
B	1
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	1
K	1
L	2

Queue:

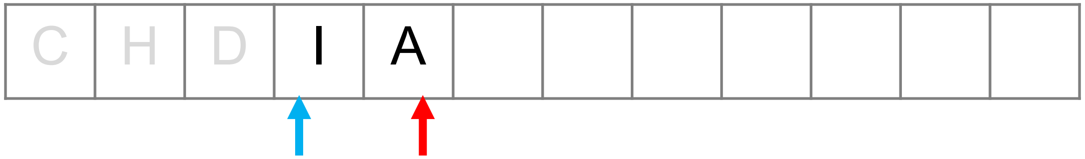
C	H	D	I	A							

Example: Kahn’s Algorithm

- □ Pop the front of the queue



Queue:

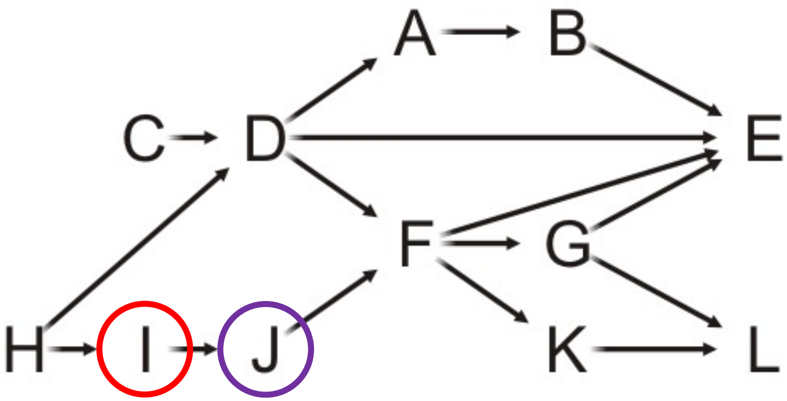


C	H	D	I	A							

A	0
B	1
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	1
K	1
L	2

Example: Kahn's Algorithm

- Pop the front of the queue
  - I has one neighbor: J



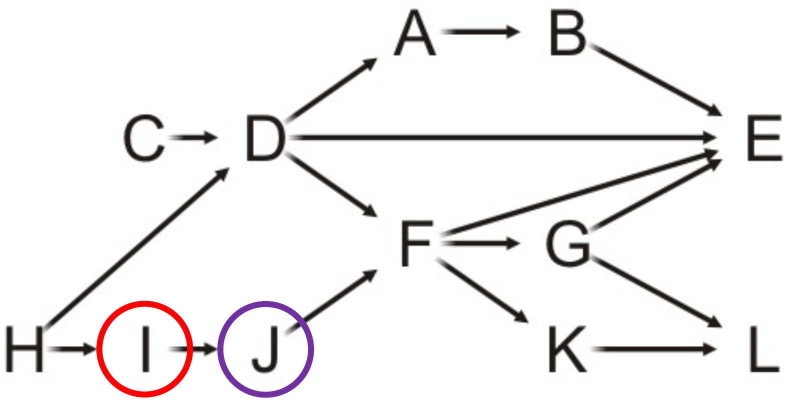
Queue:

C	H	D	I	A							

A	0
B	1
C	0
D	0
E	3
F	1
G	1
H	0
I	**0**
J	**1**
K	1
L	2

Example: Kahn’s Algorithm

- Pop the front of the queue
  - I has one neighbor: J
  - Decrement its in-degree



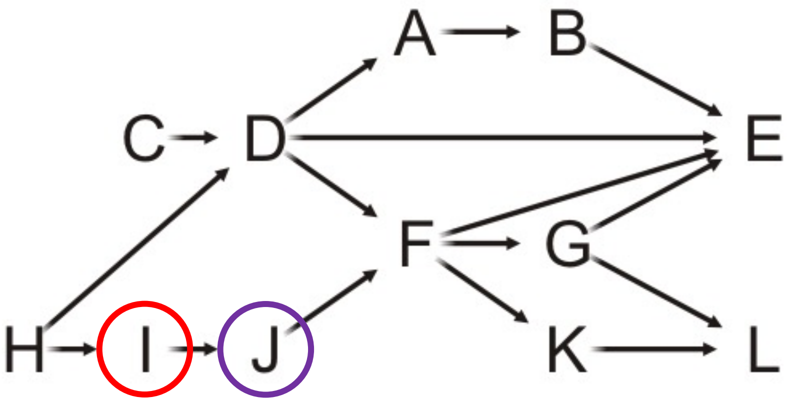
Queue:

C	H	D	I	A							

A	0
B	1
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	0
K	1
L	2

Example: Kahn’s Algorithm

- Pop the front of the queue
  - I has one neighbor: J
  - Decrement its in-degree
    - J is decremented to zero, so push it onto the queue



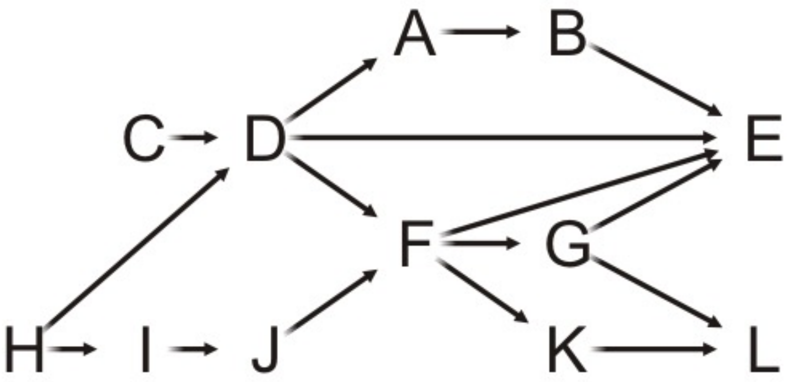
Queue:

C	H	D	I	A	J						

A	0
B	1
C	0
D	0
E	3
F	1
G	1
H	0
I	**0**
J	**0**
K	1
L	2

Example: Kahn’s Algorithm

□ Pop the front of the queue



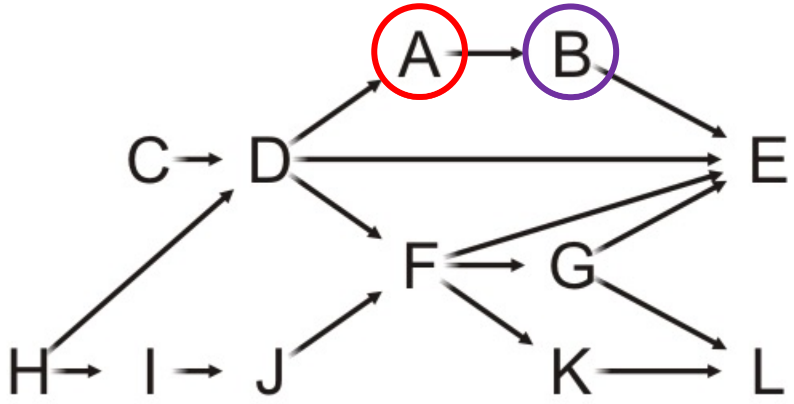
Queue:

C	H	D	I	A	J						

A	0
B	1
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	0
K	1
L	2

Example: Kahn’s Algorithm

- □ Pop the front of the queue
    - ▪ A has one neighbor: B



Queue:

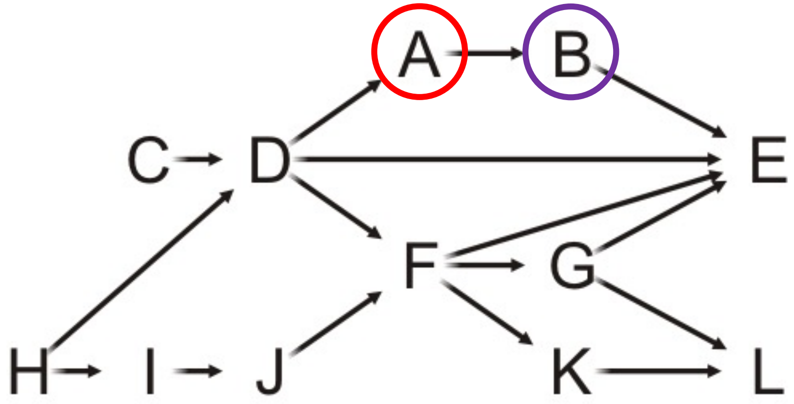
C	H	D	I	A	J						

A	0
B	1
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	0
K	1
L	2

Example: Kahn's Algorithm

□ Pop the front of the queue

- A has one neighbor: B
- Decrement its in-degree



A	0
B	0
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	0
K	1
L	2

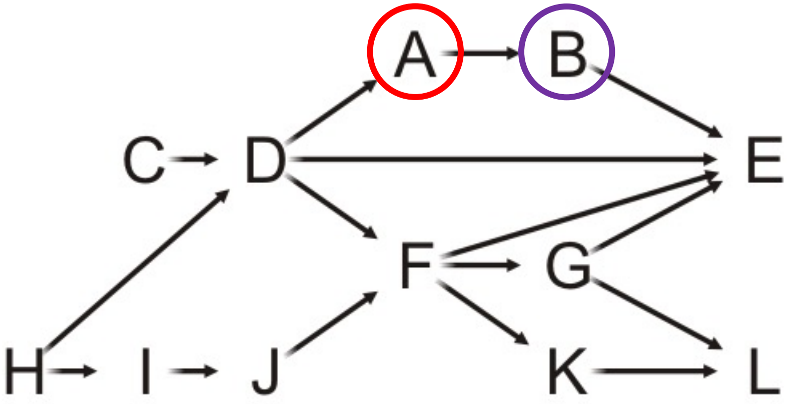
Queue:

C	H	D	I	A	J						

Example: Kahn’s Algorithm

- Pop the front of the queue
  - A has one neighbor: B
  - Decrement its in-degree
    - B is decremented to zero, so push it onto the queue

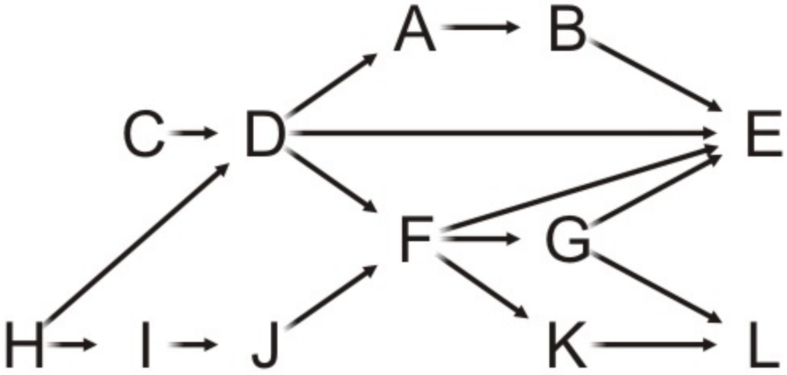
A	0
B	0
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	0
K	1
L	2



Queue:	C	H	D	I	A	J	B					

Example: Kahn’s Algorithm

- Pop the front of the queue



Queue:

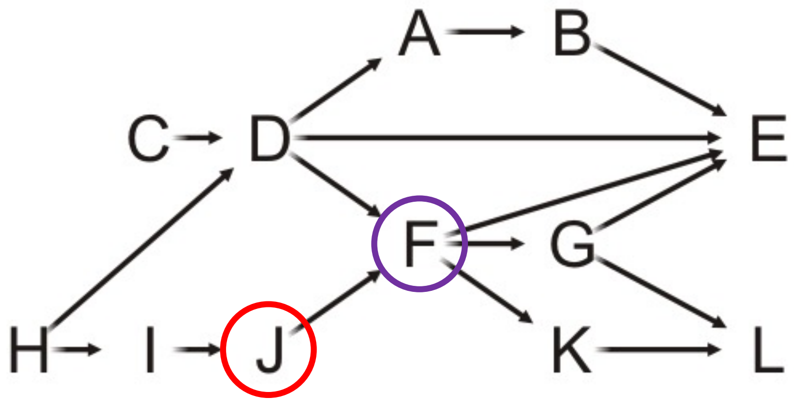
C	H	D	I	A	J	B					

A	0
B	0
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	0
K	1
L	2

Example: Kahn's Algorithm

□ Pop the front of the queue

- J has one neighbor: F



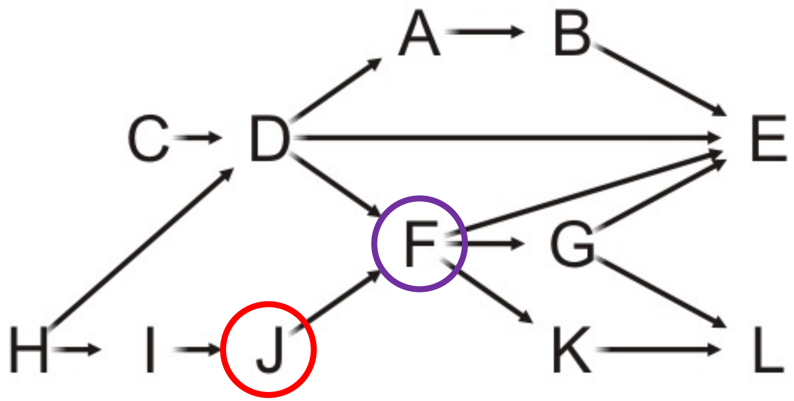
Queue:

C	H	D	I	A	J	B					

A	0
B	0
C	0
D	0
E	3
F	1
G	1
H	0
I	0
J	0
K	1
L	2

Example: Kahn’s Algorithm

- Pop the front of the queue
  - J has one neighbor: F
  - Decrement its in-degree



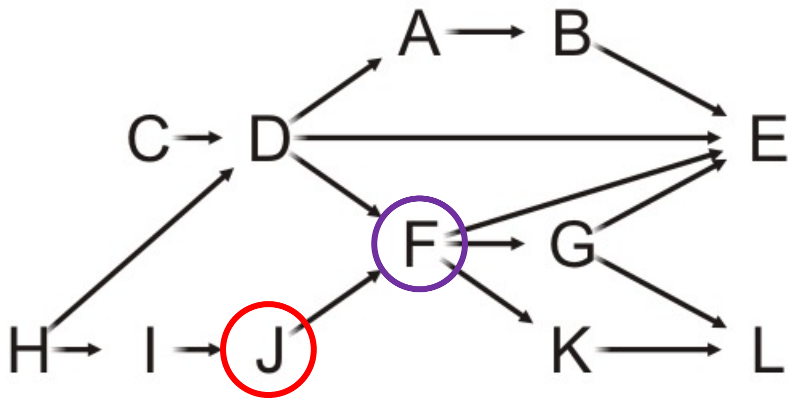
Queue:

C	H	D	I	A	J	B					

A	0
B	0
C	0
D	0
E	3
F	0
G	1
H	0
I	0
J	0
K	1
L	2

Example: Kahn’s Algorithm

- □ Pop the front of the queue
  - ▪ J has one neighbor: F
  - ▪ Decrement its in-degree
    - F is decremented to zero, so push it onto the queue



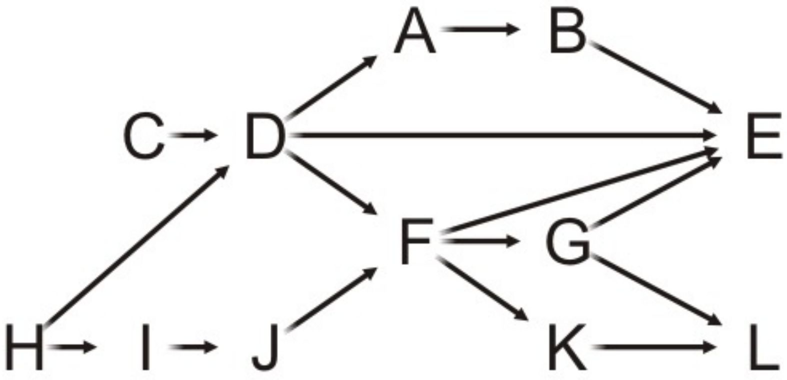
A	0
B	0
C	0
D	0
E	3
F	0
G	1
H	0
I	0
J	0
K	1
L	2

Queue:

C	H	D	I	A	J	B	F				

Example: Kahn’s Algorithm

□ Pop the front of the queue



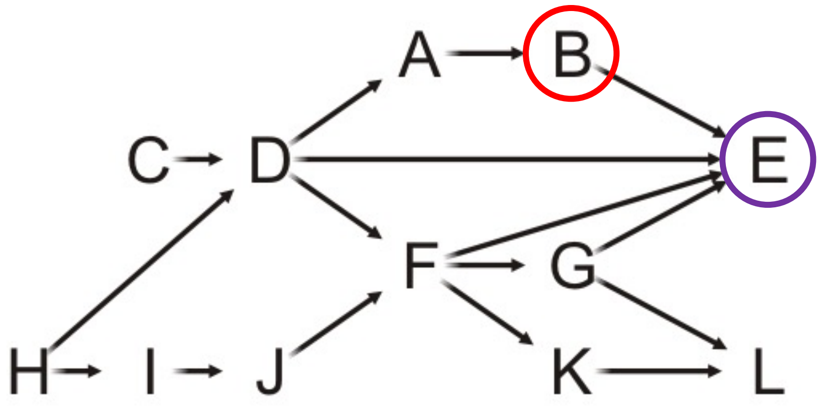
Queue:

C	H	D	I	A	J	B	F				

A	0
B	0
C	0
D	0
E	3
F	0
G	1
H	0
I	0
J	0
K	1
L	2

Example: Kahn's Algorithm

- Pop the front of the queue
  - B has one neighbor: E



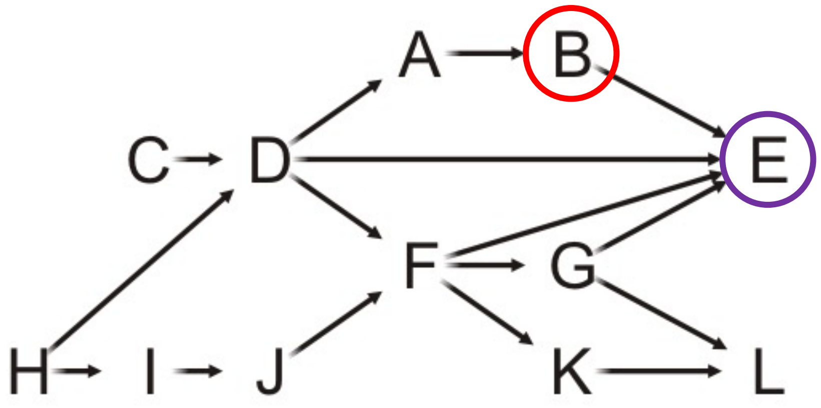
Queue:

C	H	D	I	A	J	B	F				

A	0
B	0
C	0
D	0
E	3
F	0
G	1
H	0
I	0
J	0
K	1
L	2

Example: Kahn’s Algorithm

- Pop the front of the queue
  - B has one neighbor: E
  - Decrement its in-degree



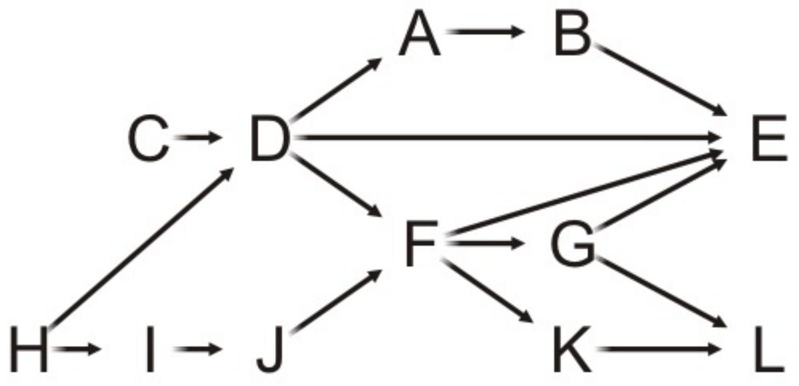
A	0
B	**0**
C	0
D	0
E	**2**
F	0
G	1
H	0
I	0
J	0
K	1
L	2

Queue:

C	H	D	I	A	J	B	F				

Example: Kahn’s Algorithm

- Pop the front of the queue



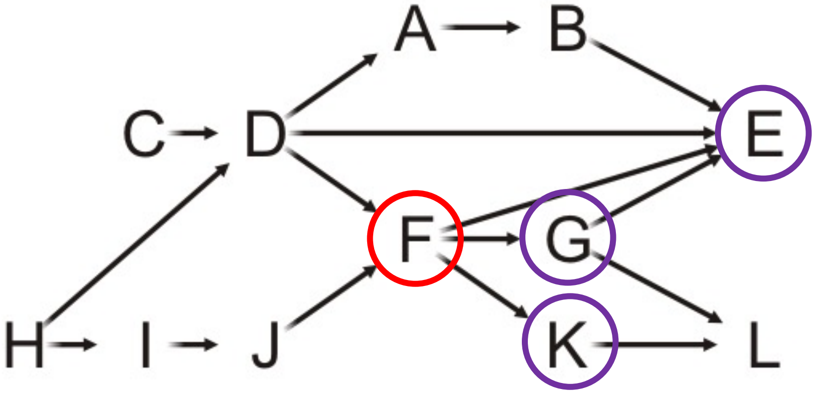
A	0
B	0
C	0
D	0
E	2
F	0
G	1
H	0
I	0
J	0
K	1
L	2

Queue:

C	H	D	I	A	J	B	F				

Example: Kahn’s Algorithm

- Pop the front of the queue
  - F has three neighbors: E, G and K



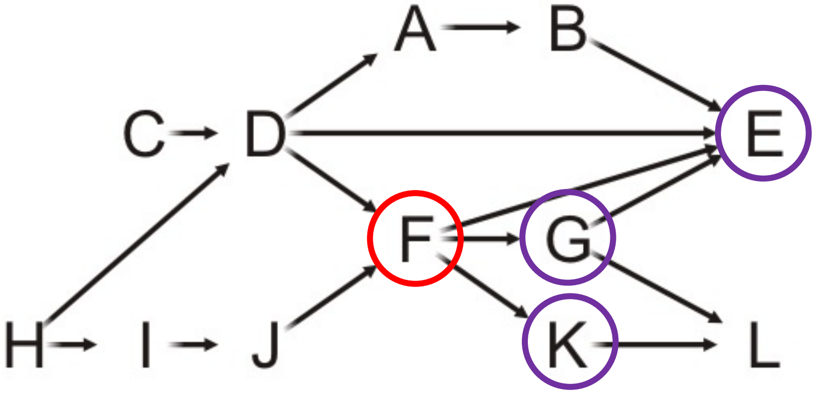
Queue:

C	H	D	I	A	J	B	F				

A	0
B	0
C	0
D	0
E	**2**
F	**0**
G	**1**
H	0
I	0
J	0
K	**1**
L	2

Example: Kahn’s Algorithm

- □ Pop the front of the queue
  - F has three neighbors: E, G and K
  - Decrement their in-degrees



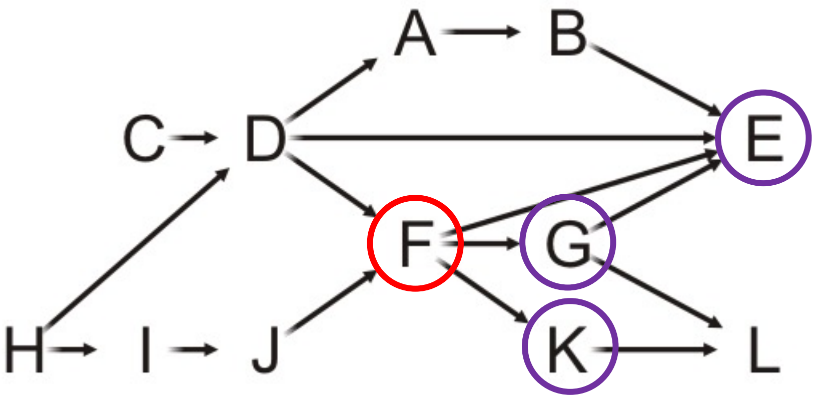
A	0
B	0
C	0
D	0
E	**1**
F	**0**
G	**0**
H	0
I	0
J	0
K	**0**
L	2

Queue:

C	H	D	I	A	J	B	F				

Example: Kahn’s Algorithm

- Pop the front of the queue
  - F has three neighbors: E, G and K
  - Decrement their in-degrees
    - G and K are decremented to zero,
      so push them onto the queue

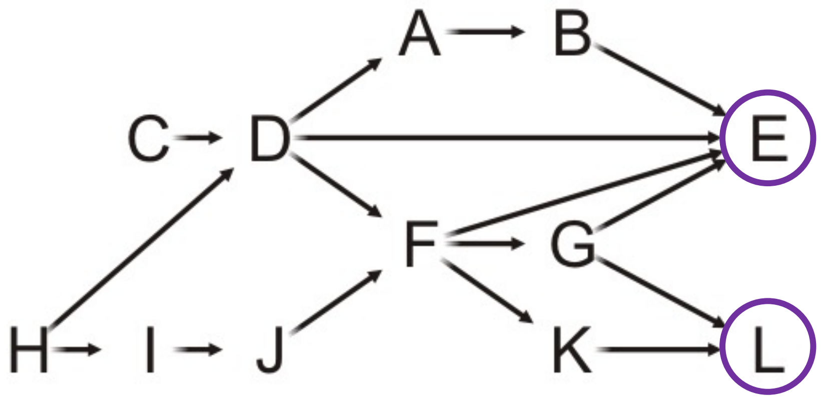


Queue:	C	H	D	I	A	J	B	F	G	K		
									↑	↑		

A	0
B	0
C	0
D	0
E	1
F	0
G	0
H	0
I	0
J	0
K	0
L	2

Example

- □ Pop the front of the queue



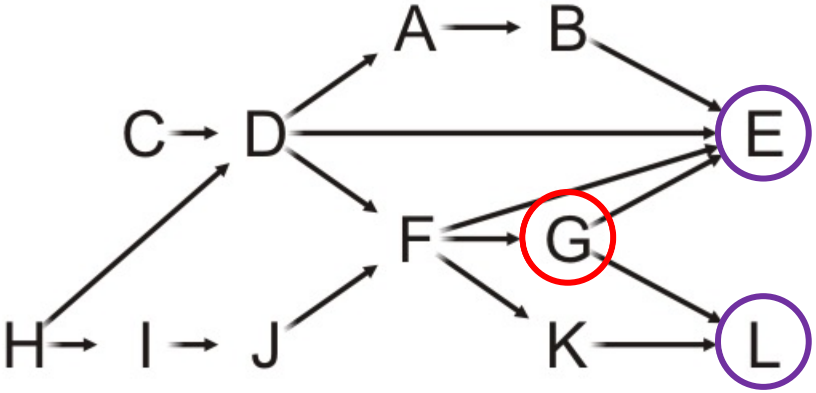
Queue:

C	H	D	I	A	J	B	F	G	K		

A	0
B	0
C	0
D	0
E	1
F	0
G	0
H	0
I	0
J	0
K	0
L	2

Example: Kahn's Algorithm

- □ Pop the front of the queue
  - ▪ G has two neighbors: E and L



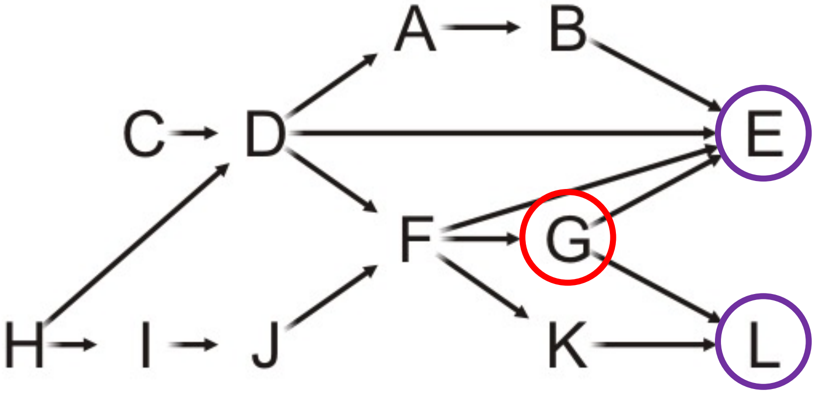
Queue:

C	H	D	I	A	J	B	F	G	K		

A	0
B	0
C	0
D	0
E	1
F	0
G	0
H	0
I	0
J	0
K	0
L	2

Example: Kahn’s Algorithm

- Pop the front of the queue
  - G has two neighbors: E and L
  - Decrement their in-degrees



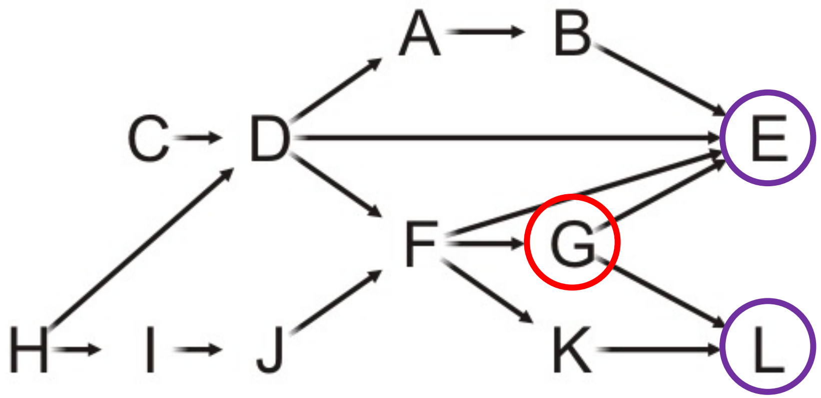
Queue:

C	H	D	I	A	J	B	F	G	K		

A	0
B	0
C	0
D	0
E	**0**
F	0
G	**0**
H	0
I	0
J	0
K	0
L	**1**

Example: Kahn’s Algorithm

- □ Pop the front of the queue
  - ▪ G has two neighbors: E and L
  - ▪ Decrement their in-degrees
    - E is decremented to zero, so push it onto the queue



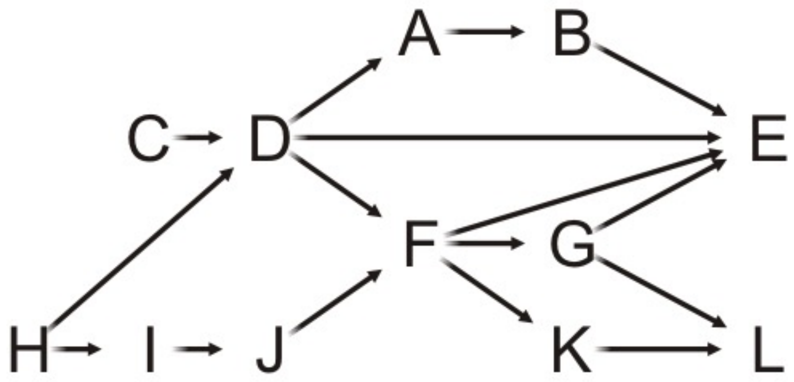
Queue:

C	H	D	I	A	J	B	F	G	K	E	

A	0
B	0
C	0
D	0
E	**0**
F	0
G	**0**
H	0
I	0
J	0
K	0
L	**1**

Example: Kahn’s Algorithm

- □ Pop the front of the queue



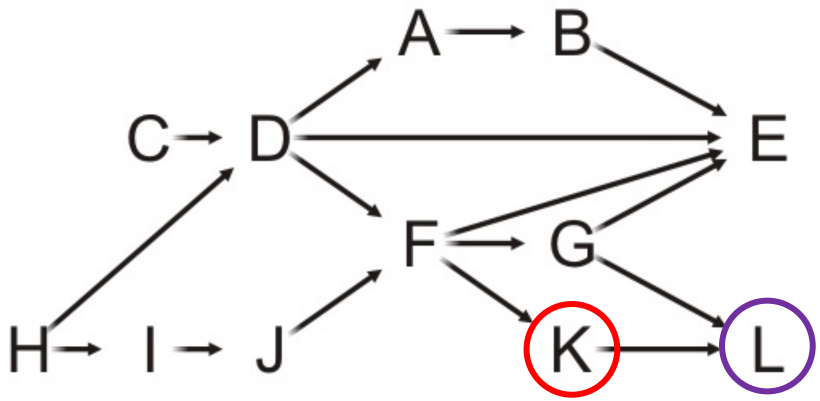
Queue:

C	H	D	I	A	J	B	F	G	K	E	

A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	0
L	1

Example: Kahn’s Algorithm

- Pop the front of the queue
  - K has one neighbors: L



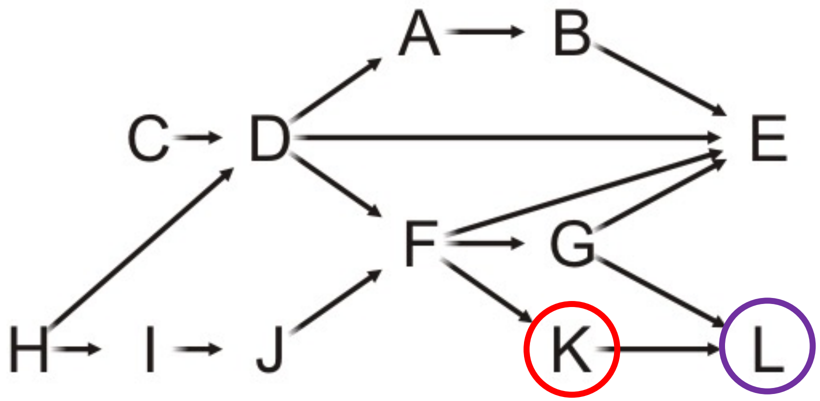
A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	**0**
L	**1**

Queue:

C	H	D	I	A	J	B	F	G	K	E	

Example: Kahn’s Algorithm

- □ Pop the front of the queue
  - K has one neighbors: L
  - Decrement its in-degree



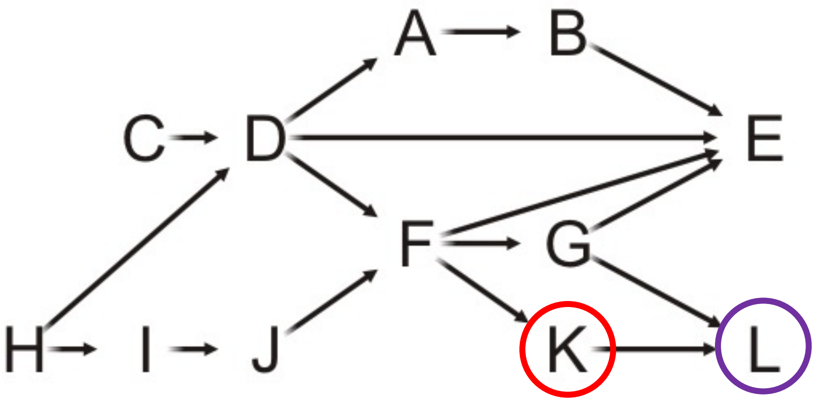
Queue:

C	H	D	I	A	J	B	F	G	K	E	

A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	**0**
L	**0**

Example: Kahn's Algorithm

- Pop the front of the queue
  - K has one neighbors: L
  - Decrement its in-degree
    - L is decremented to zero, so push it onto the queue



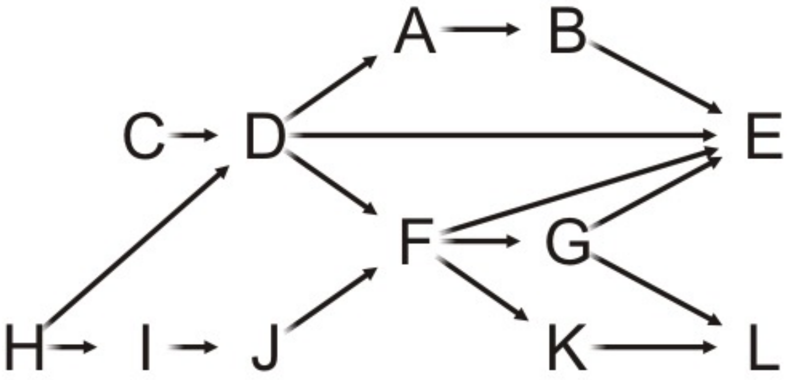
A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	**0**
L	**0**

Queue:

C	H	D	I	A	J	B	F	G	K	E	L

Example: Kahn's Algorithm

□ Pop the front of the queue



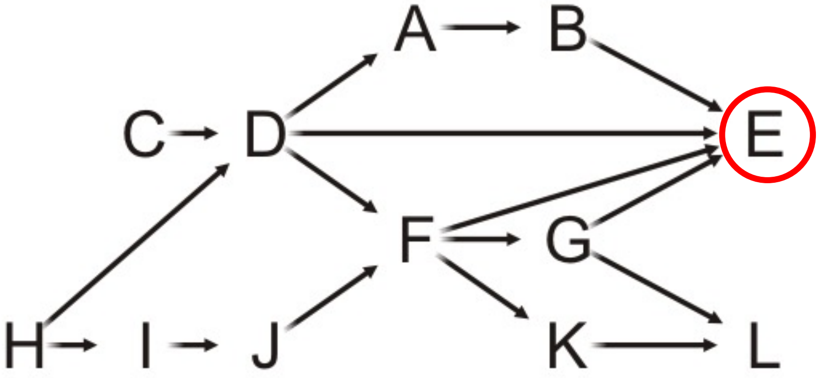
Queue:

C	H	D	I	A	J	B	F	G	K	E	L

A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	0
L	0

Example: Kahn's Algorithm

- Pop the front of the queue
  - E has no neighbors—it is a *sink*



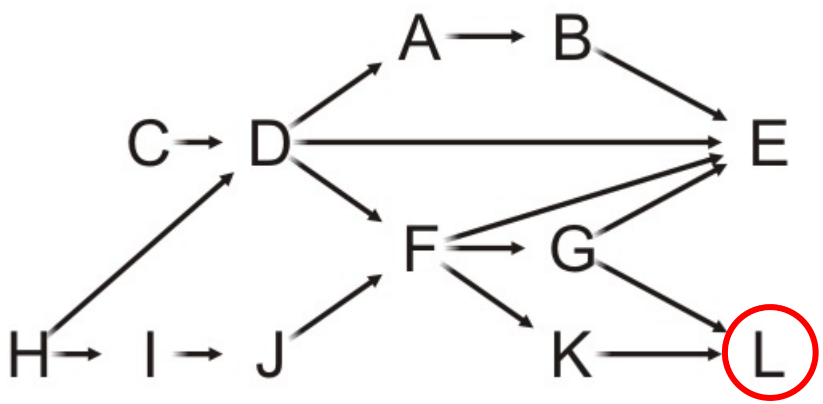
A	0
B	0
C	0
D	0
E	**0**
F	0
G	0
H	0
I	0
J	0
K	0
L	0

Queue:

C	H	D	I	A	J	B	F	G	K	E	L

Example: Kahn’s Algorithm

- □ Pop the front of the queue

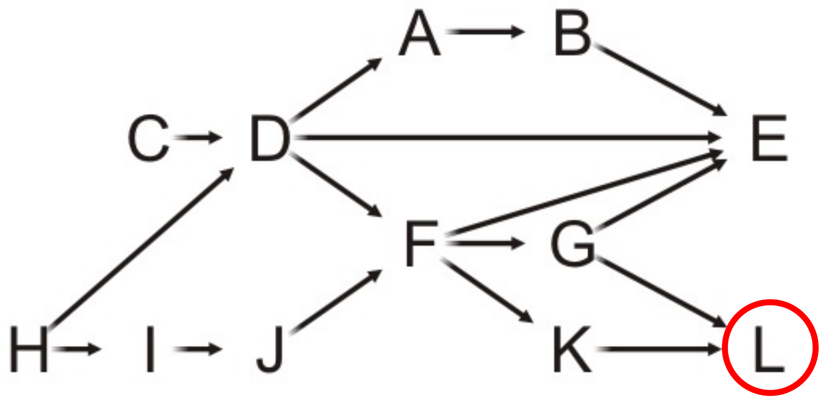


Queue:	C	H	D	I	A	J	B	F	G	K	E	L

A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	0
L	0

Example: Kahn’s Algorithm

- Pop the front of the queue
  - L has no neighbors—it is also a *sink*



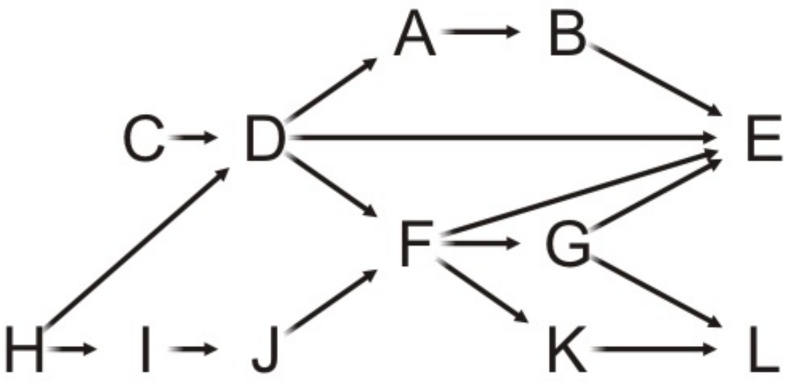
Queue:

C	H	D	I	A	J	B	F	G	K	E	L

A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	0
L	**0**

Example: Kahn's Algorithm

- The queue is empty, so we are done



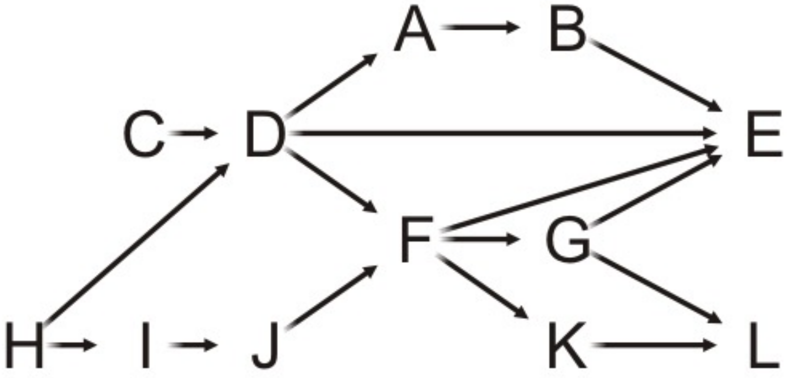
A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	0
L	0

Queue:

C	H	D	I	A	J	B	F	G	K	E	L

Example: Kahn’s Algorithm

□ The array stores the topological sorting



C	H	D	I	A	J	B	F	G	K	E	L

A	0
B	0
C	0
D	0
E	0
F	0
G	0
H	0
I	0
J	0
K	0
L	0

Runtime of Kahn’s Algorithm

- □ Step #1: Preparing the in-degree array
  - ▪ Takes Θ(|V| + |E|) runtime
    - If the DAG was represented as an adjacency list
  - ▪ Takes Θ(|V|^2) runtime
    - If the DAG was represented as an adjacency matrix

- □ Step #2: Keep enumerating the in-degree array
  - ▪ Takes Θ(|V| + |E|) runtime
    - Queued vertices |V| times and decrementing in-degree |E| times

Summary

- □ In this topic, we have discussed topological sorts
  - ▪ Sorting of elements in a DAG
  - ▪ Kahn's Algorithm
    - A table of in-degrees
    - Select that vertex which has current in-degree zero

References

[1] Wikipedia, http://en.wikipedia.org/wiki/Topological_sorting

[2] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, McGraw Hill, 1990, §11.1, p.200.

[3] Weiss, Data Structures and Algorithm Analysis in C++, 3rd Ed., Addison Wesley, §9.2, p.342-5.