COMP251: DATA STRUCTURES & ALGORITHMS

Instructor: Maryam Siahbani

Computer Information System
University of Fraser Valley

* Some slides from “Algorithms and Data Structures” by Douglas Wilhelm Harder

Topological Sort

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)

Definition of topological sorting

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

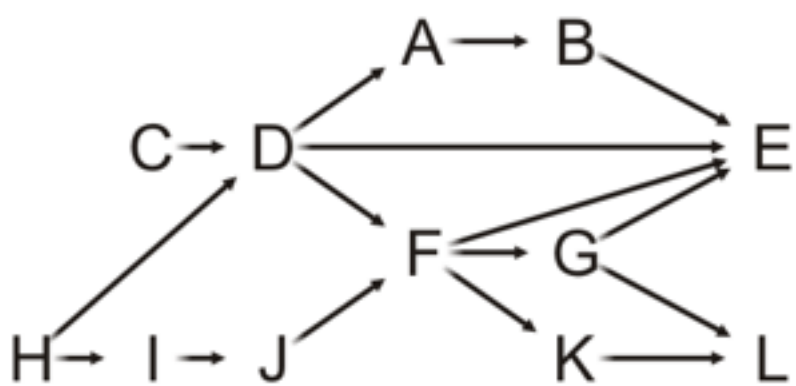
$$v_1, v_2, v_3, \ldots, v_{|V|}$$

such that v_j appears 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

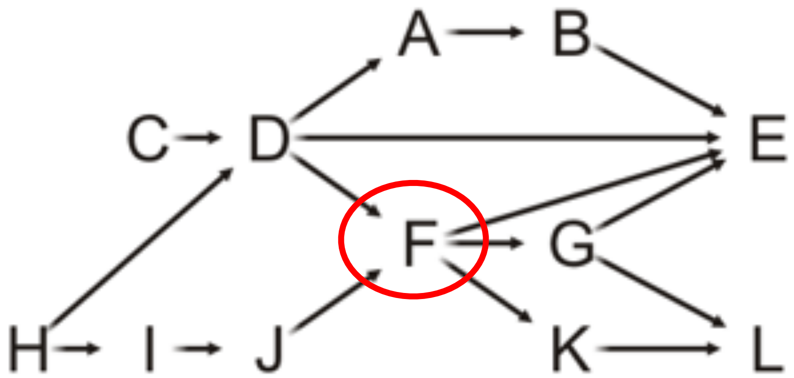
H, C, I, D, J, A, F, B, G, K, E, L



Example

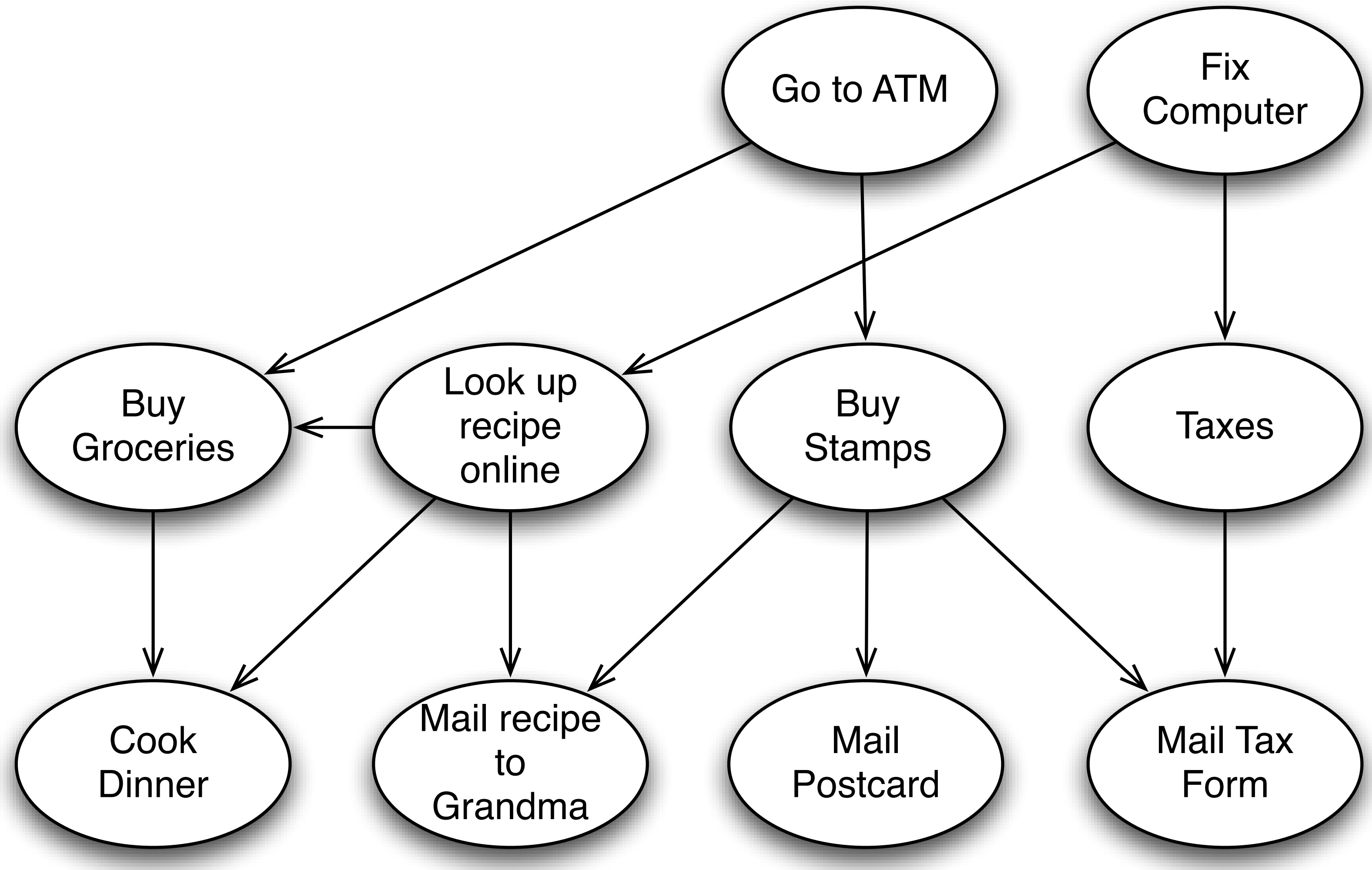
For example, there are paths from H, C, I, D and J to F, so all these must come before F in a topological sort

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



Clearly, this sorting need not be unique

Applications



Topological Sort

A naïve algorithm:

–Given a DAG G iterate:

1. Find the in-degrees of all nodes
2. Find a node v with in-degree zero
3. Print v as the next vertex in the topological sort
4. Remove the node and continue iterating (go to step 1).

Topological Sort

A better algorithm:

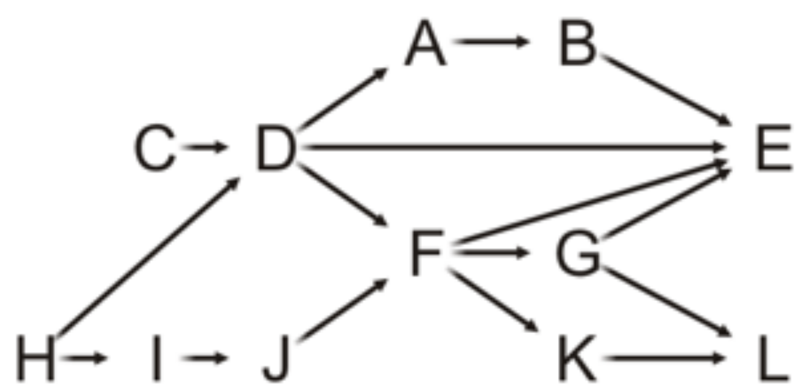
–Given a DAG G

1. Compute all in-degrees
2. Put all in-degree 0 nodes in a Collection
3. Print and remove a node from Collection
4. Decrement in-degrees of the neighbours (of the removed node)
5. If any neighbours has in-degree 0 add it to the Collection go to step 3

Example

Let's step through this algorithm with this example

- First find the in-degrees and store them
- Need a Collection to keep track of nodes with in-degree 0



Implementation

Thus, to implement a topological sort:

- Give a number to each nod and use an array to store in-degrees
- Create a queue and initialize it with all vertices that have in-degree zero

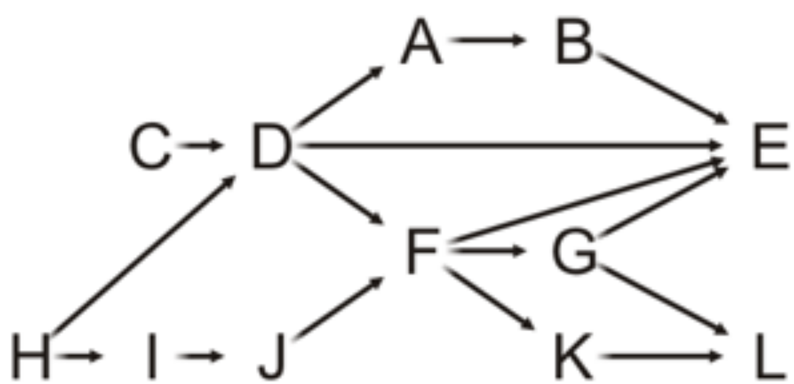
While the queue is not empty:

- Pop a vertex from the queue
- Decrement the in-degree of its neighbours
- Those neighbours whose in-degree was decremented to zero are pushed onto the queue

Example

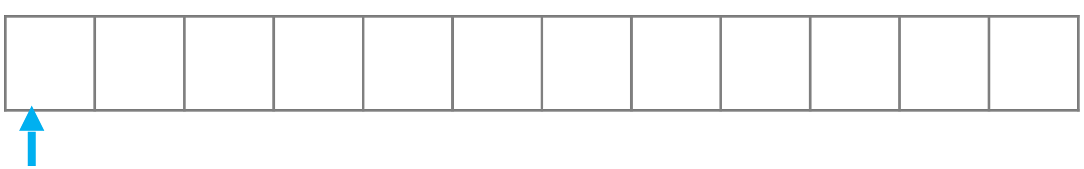
With the previous example, we initialize:

- 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:`

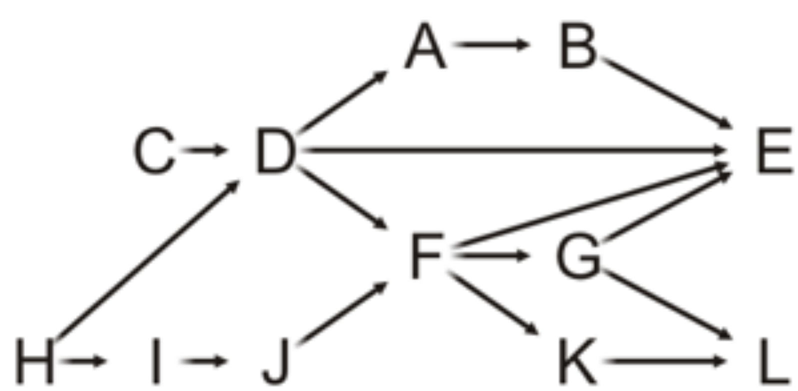


The queue is empty

Example

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

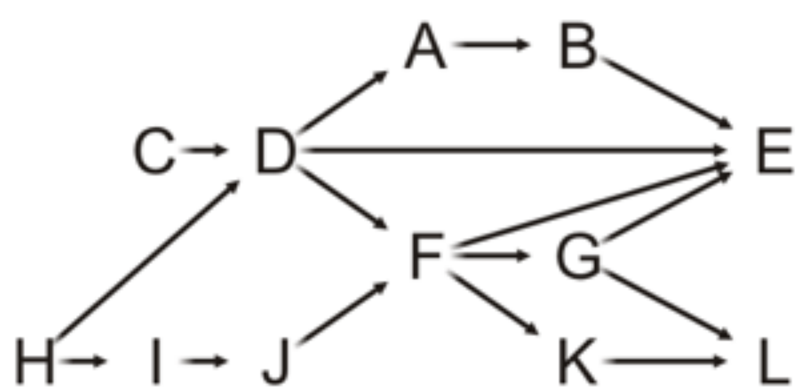


Queue:

The queue is empty

Example

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

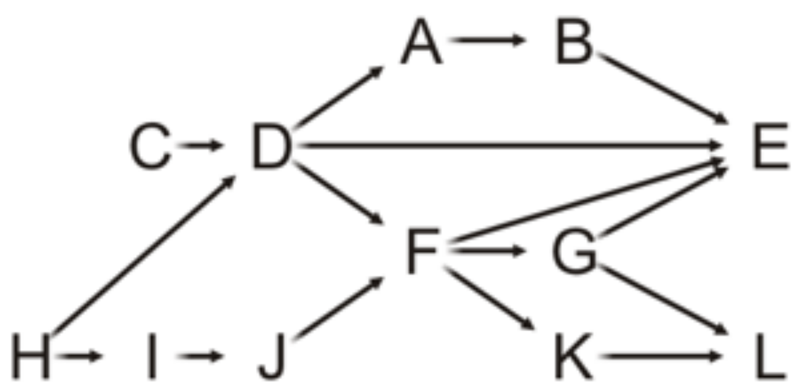
Queue:

C	**H**										

The queue is empty

Example

Pop the front of the queue



Queue:

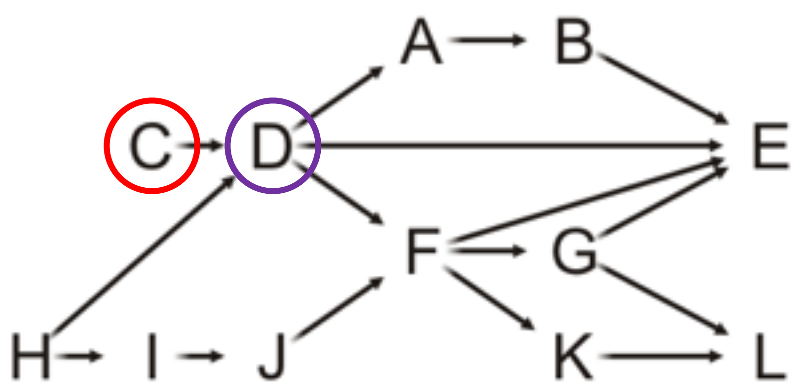
C	H										

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

Example

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

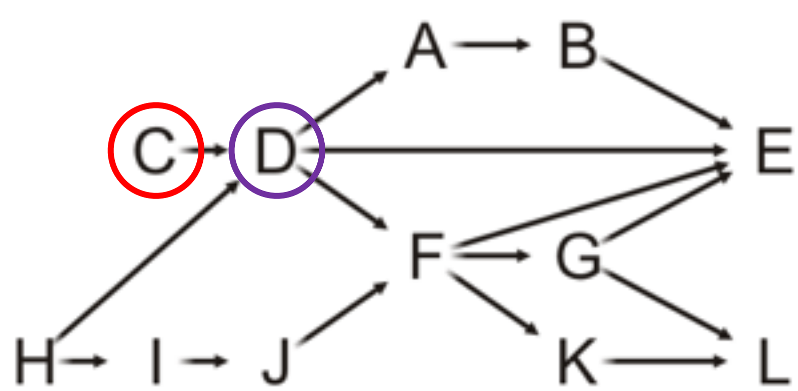
Queue:

C	H										

Example

Pop the front of the queue

- C has one neighbor: D
- Decrement its in-degree



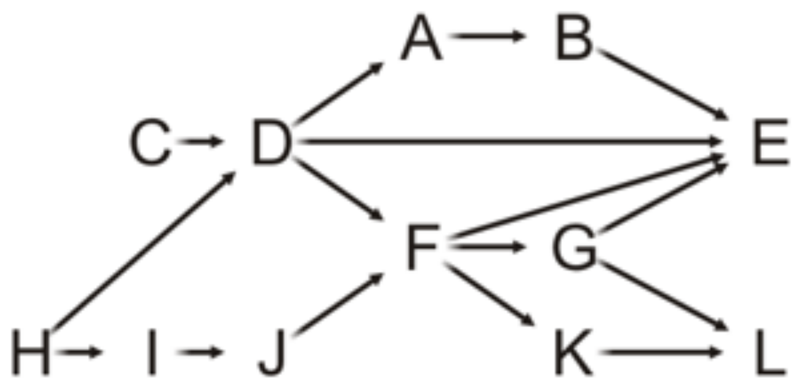
Queue:

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

Pop the front of the queue



Queue:

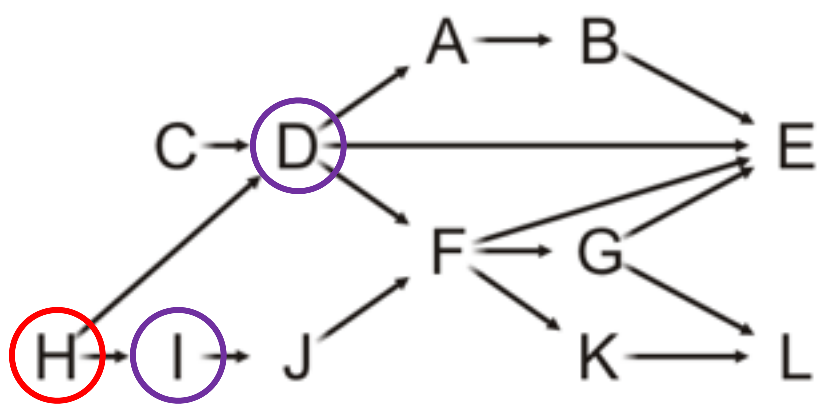
C	H										

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

Pop the front of the queue

- H has two neighbors: D and I



Queue:

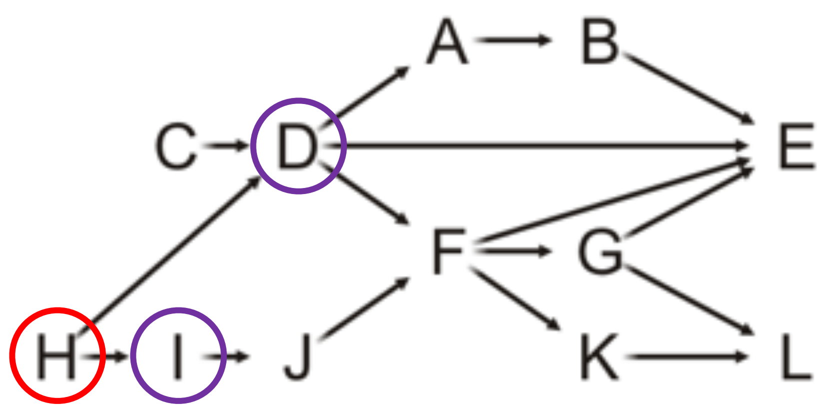
C	H										

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

Pop the front of the queue

- H has two neighbors: D and I
- Decrement their in-degrees



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

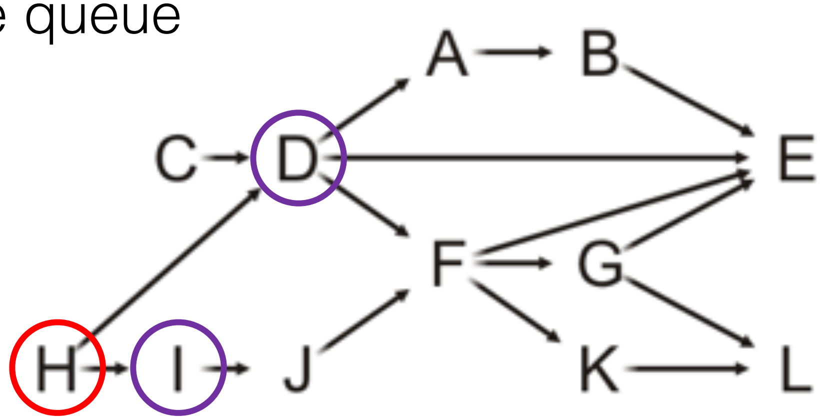
Queue:

C	H										

Example

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



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

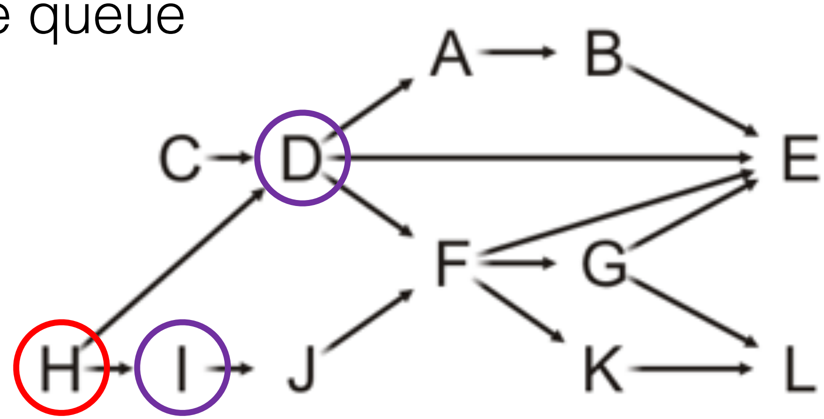
Queue:

C	H										

Example

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



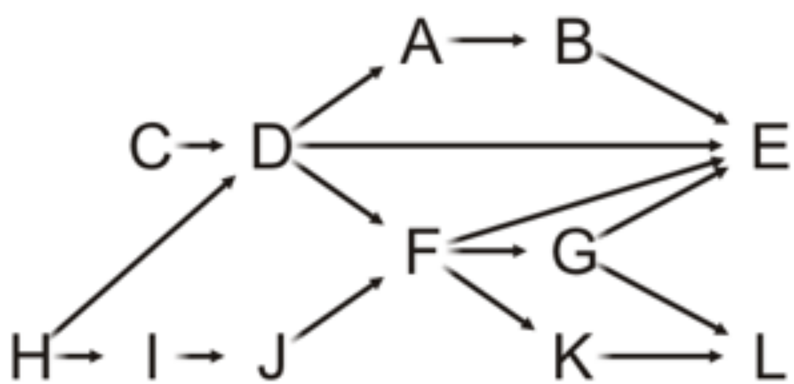
Queue:

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

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

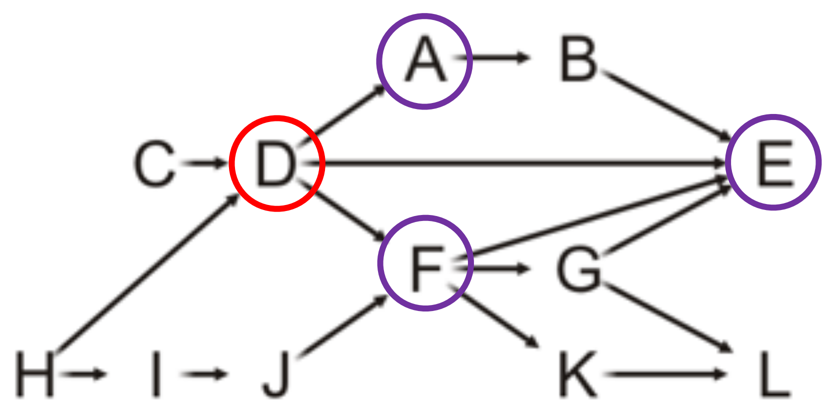
`Queue:`

C	H	D	I								

Example

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

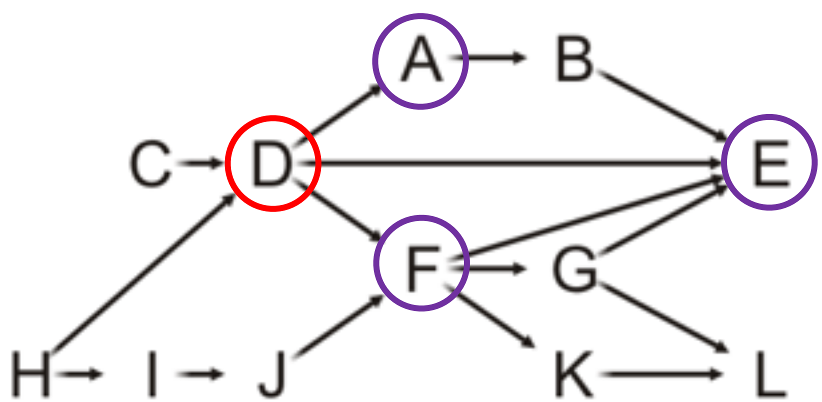
Queue:

C	H	D	I								

Example

Pop the front of the queue

- D has three neighbors: A, E and F
- Decrement their in-degrees



Queue:

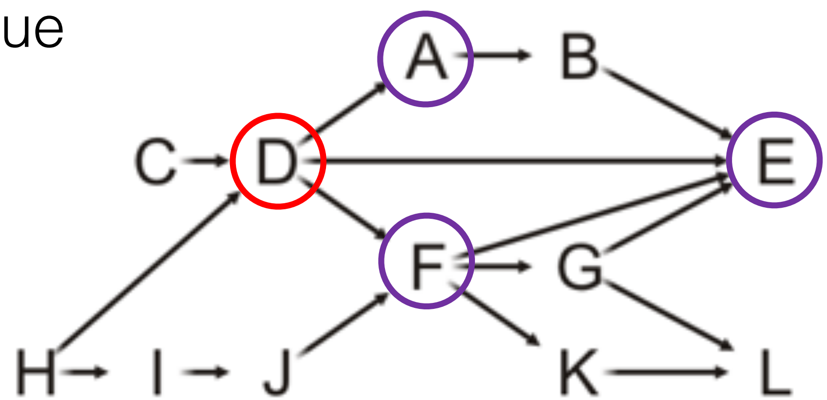
C	H	D	I								

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

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



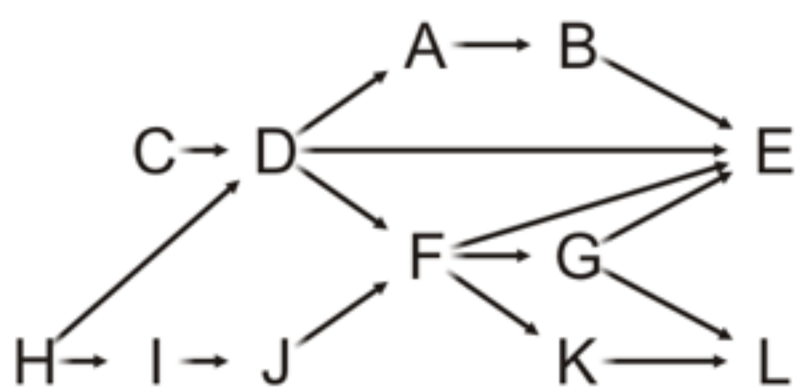
Queue:

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

Pop the front of the queue



Queue:

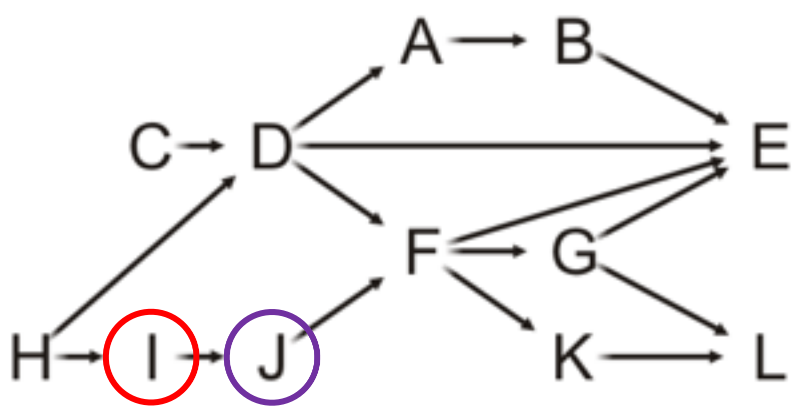
C	H	D	I	A							

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

Pop the front of the queue

- I has one neighbor: J



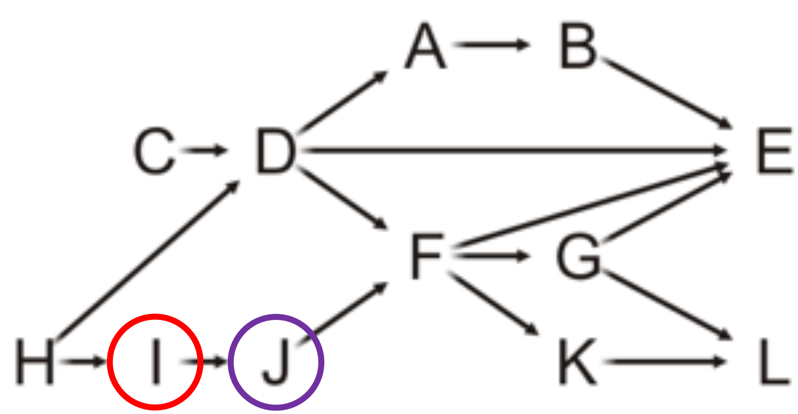
Queue:												
	C	H	D	I	A							

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

Pop the front of the queue

- I has one neighbor: J
- Decrement its in-degree



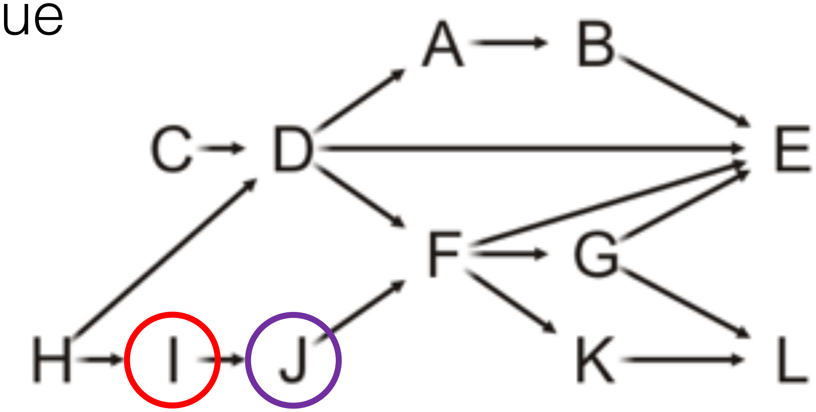
Queue:												
	C	H	D	I	A							

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

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



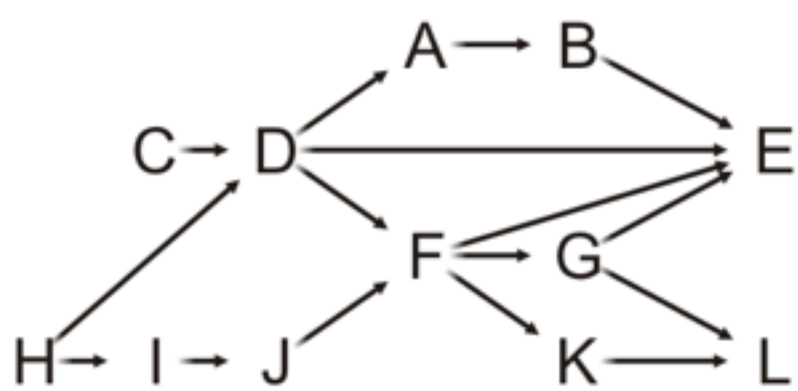
Queue:

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

Pop the front of the queue



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

Queue:

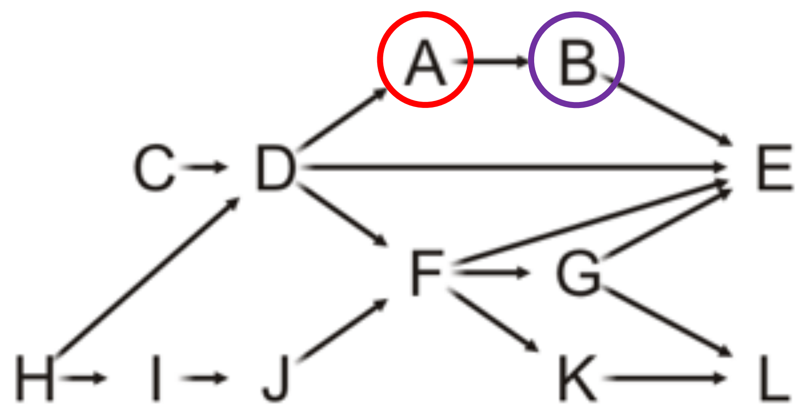
C	H	D	I	A	J						

Example

Pop the front of the queue

– A has one neighbor: B

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

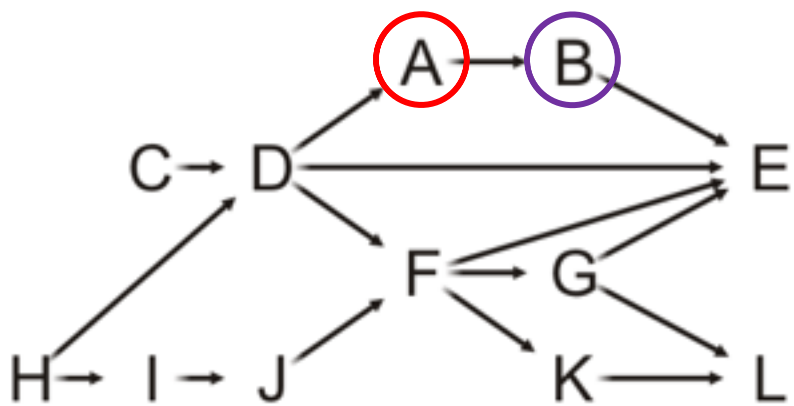


`Queue:`	C	H	D	I	A	J						

Example

Pop the front of the queue

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



Queue:

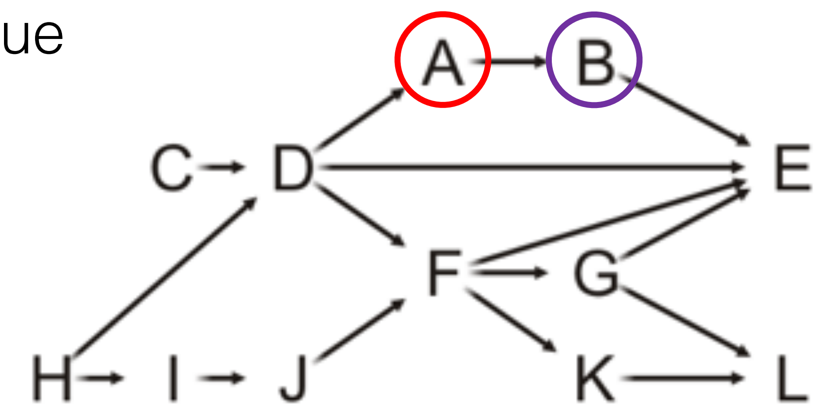
C	H	D	I	A	J						

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

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



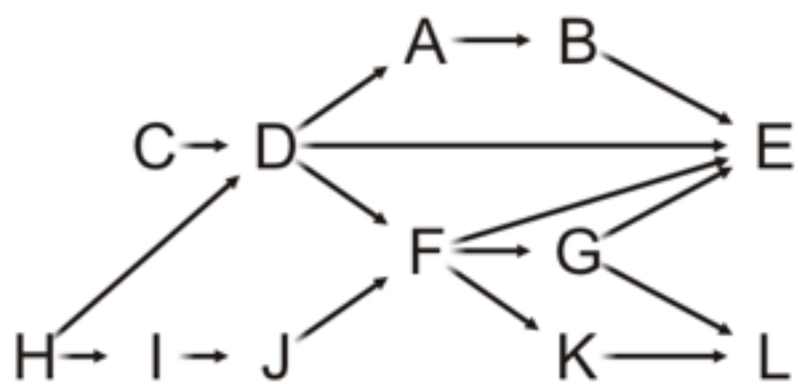
Queue:

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

Pop the front of the queue



Queue:

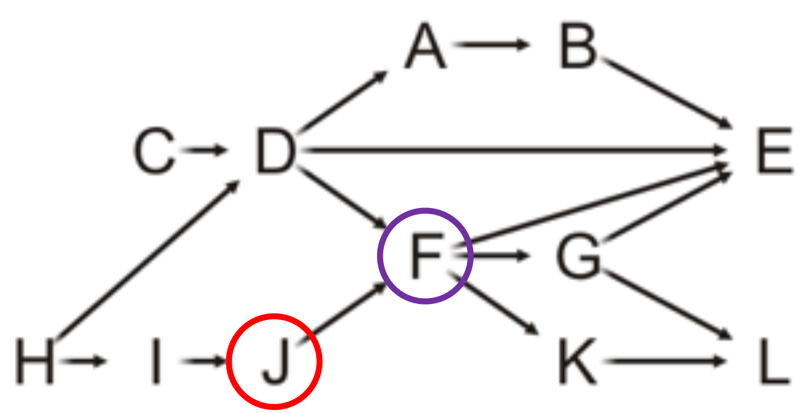
C	H	D	I	A	J	B					

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

Pop the front of the queue

- J has one neighbor: F



Queue:

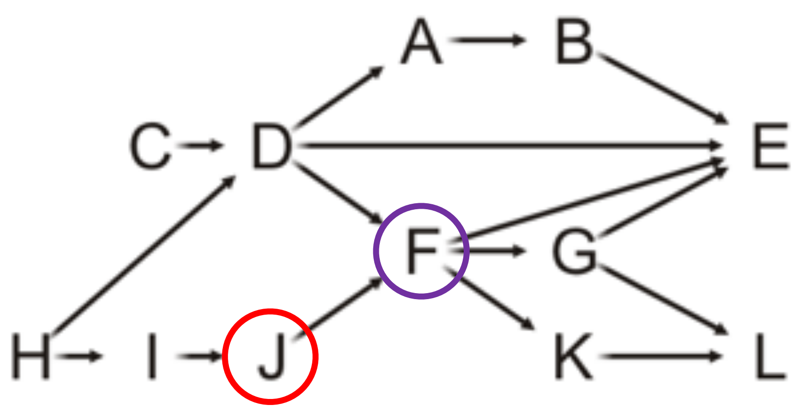
C	H	D	I	A	J	B					

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

Pop the front of the queue

- J has one neighbor: F
- Decrement its in-degree



Queue:

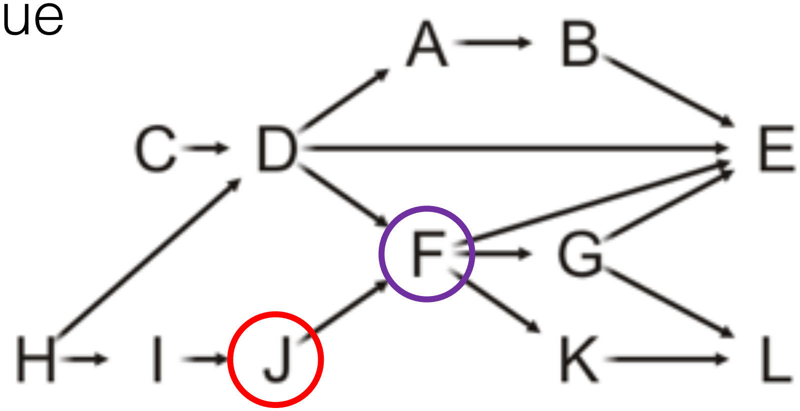
C	H	D	I	A	J	B					

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

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



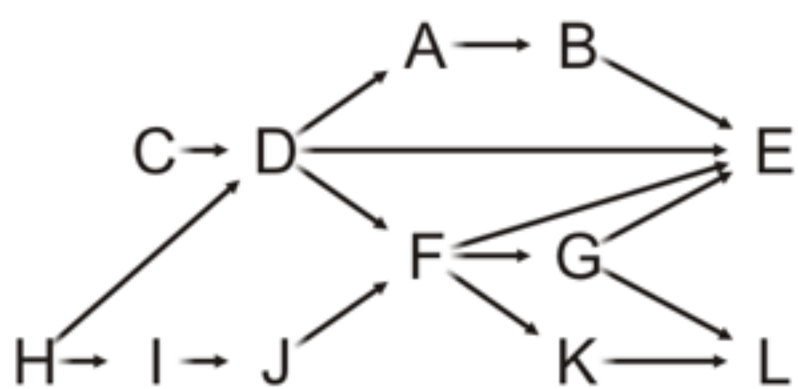
Queue:

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

Pop the front of 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

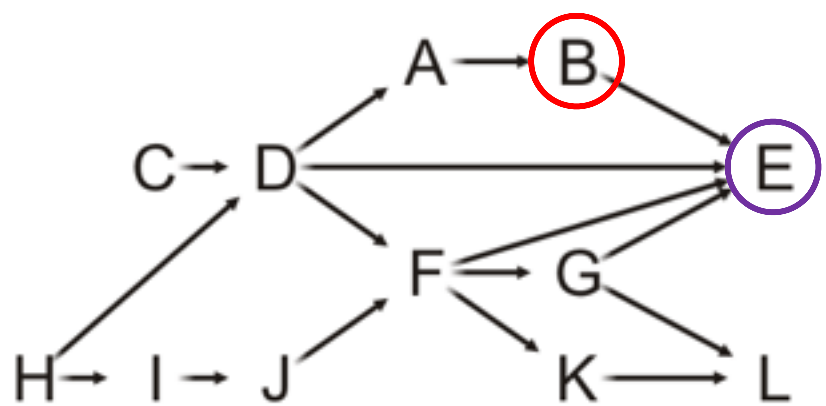
Queue:

C	H	D	I	A	J	B	F				

Example

Pop the front of the queue

– B has one neighbor: E



Queue:

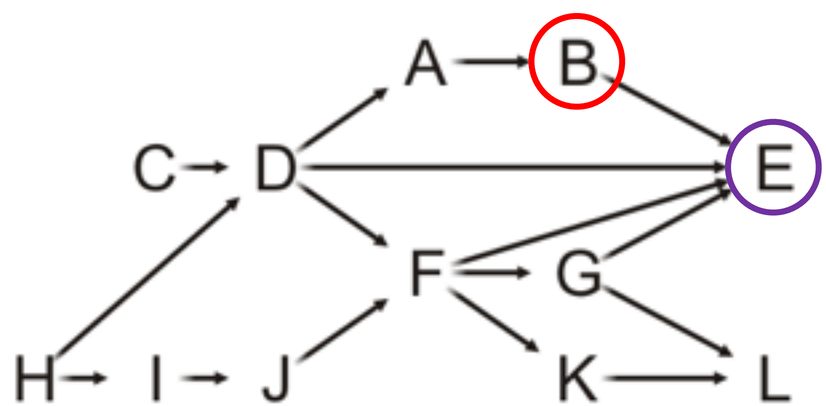
C	H	D	I	A	J	B	F				

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

Pop the front of the queue

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



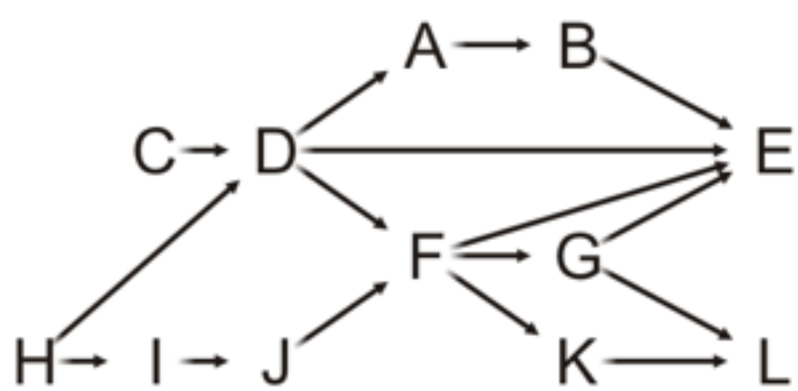
A	0
B	**0**
C	0
D	0
E	**2**
F	0
G	1
H	0
I	0
J	0
K	1
L	2

`Queue:`

C	H	D	I	A	J	B	F				

Example

Pop the front of the queue



Queue:

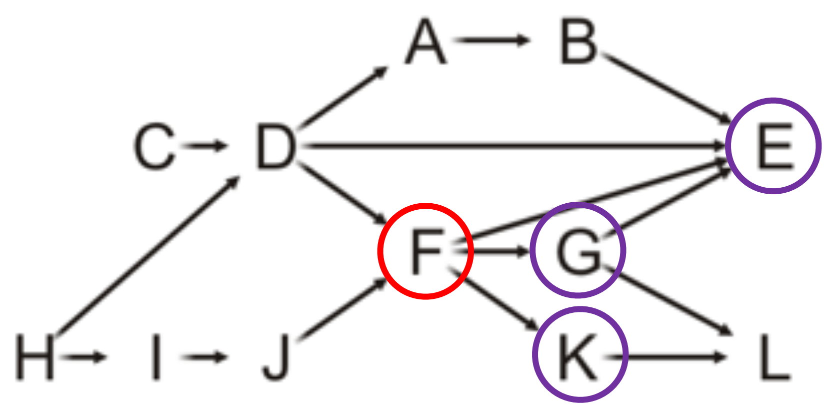
C	H	D	I	A	J	B	F				

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

Pop the front of the queue

– F has three neighbors: E, G and K



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

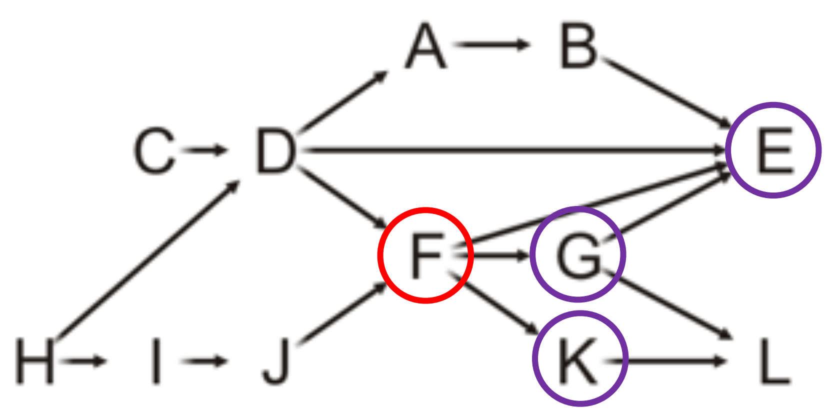
Queue:

C	H	D	I	A	J	B	F				

Example

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

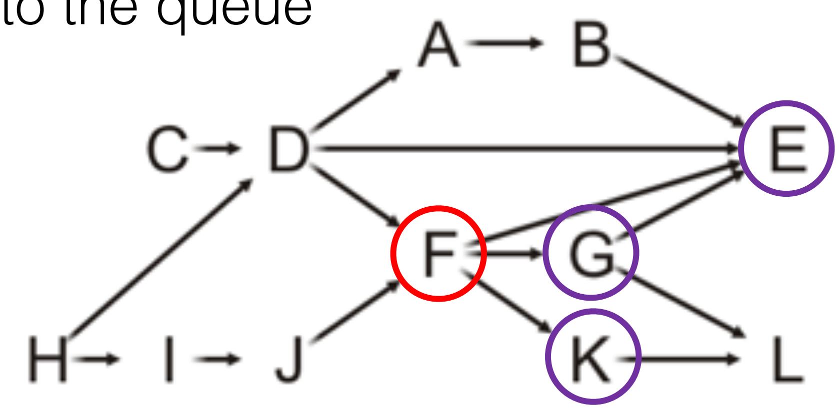
Queue:

C	H	D	I	A	J	B	F				

Example

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:

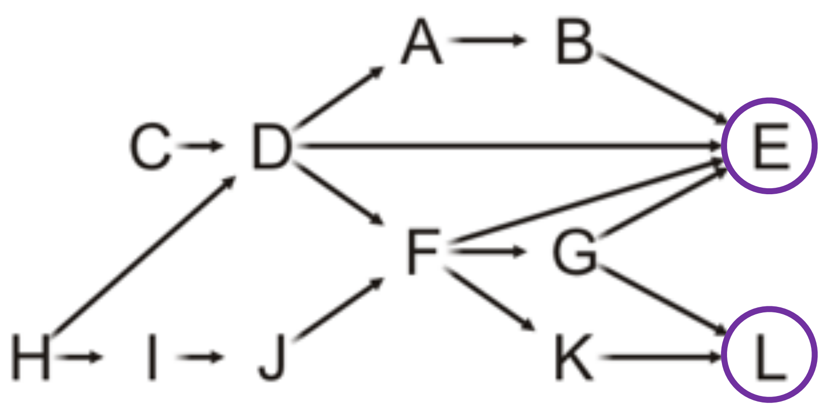
C	H	D	I	A	J	B	F	G	K		

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

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



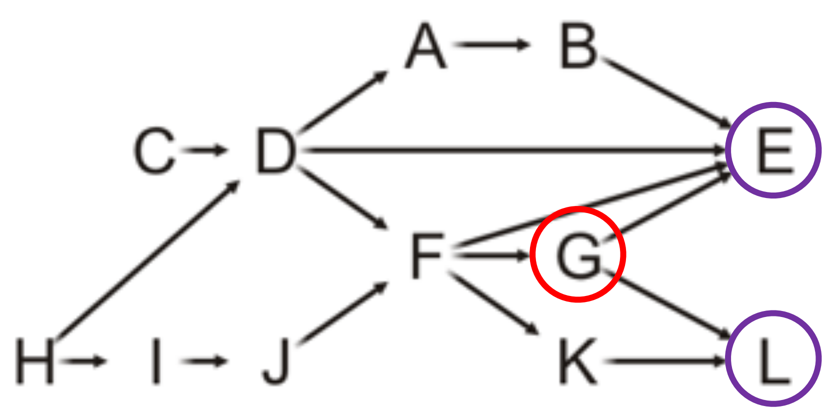
Queue:

C	H	D	I	A	J	B	F	G	K		

Example

Pop the front of the queue

- G has two neighbors: E and L



Queue:

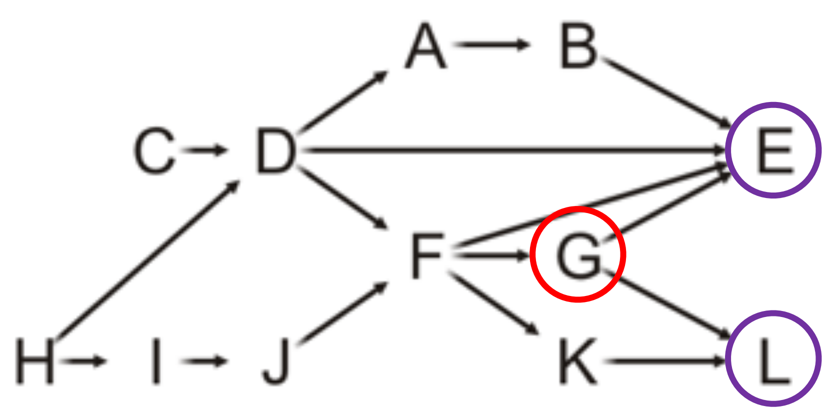
C	H	D	I	A	J	B	F	G	K		

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

Pop the front of the queue

- G has two neighbors: E and L
- Decrement their in-degrees



Queue:

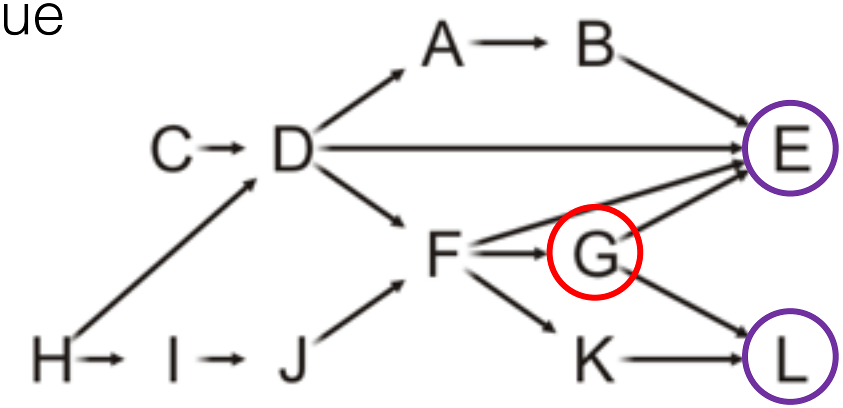
C	H	D	I	A	J	B	F	G	K		

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

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

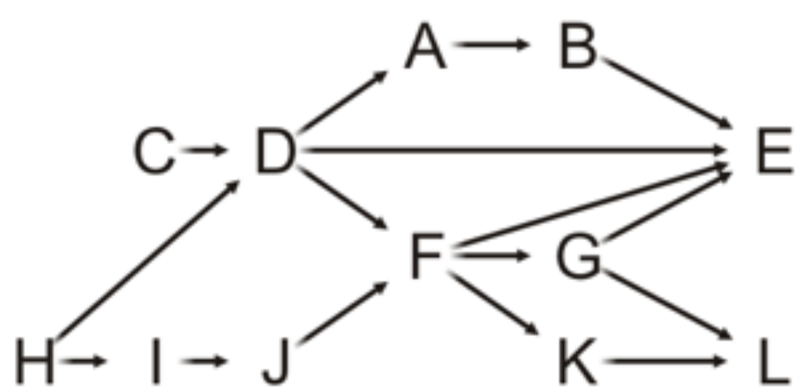


Queue:	C	H	D	I	A	J	B	F	G	K	E	

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

Pop the front of 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	1

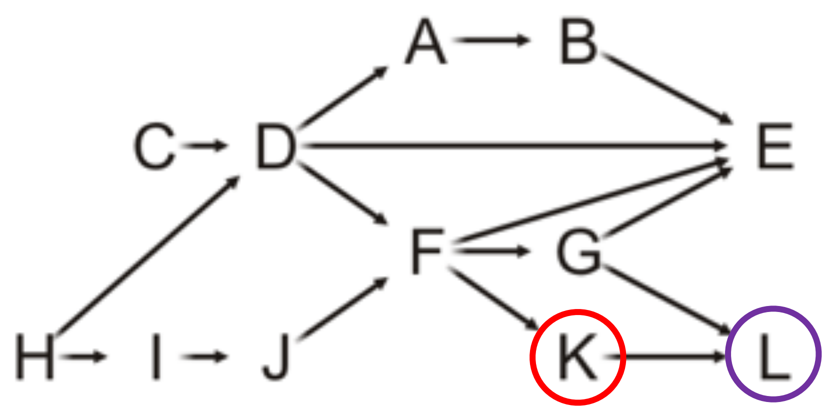
Queue:

C	H	D	I	A	J	B	F	G	K	E	

Example

Pop the front of the queue

– K has one neighbors: L



Queue:

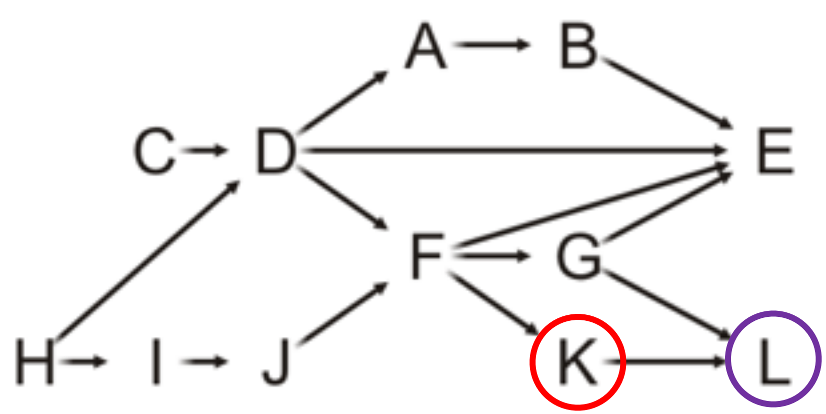
C	H	D	I	A	J	B	F	G	K	E	

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

Pop the front of the queue

- K has one neighbors: L
- Decrement its in-degree



Queue:

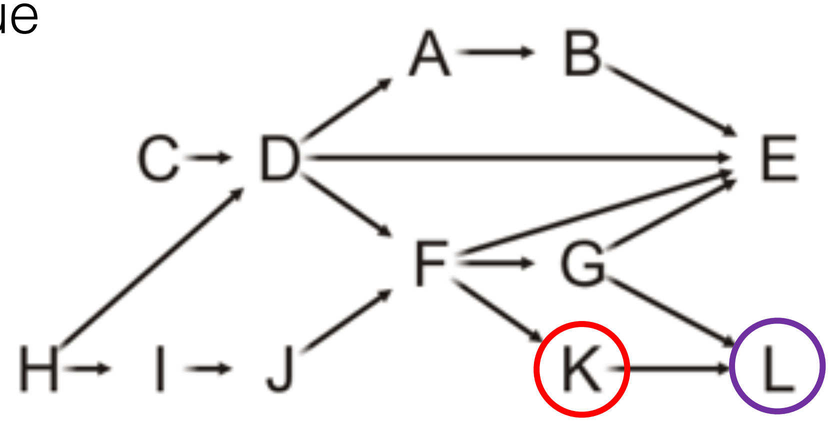
C	H	D	I	A	J	B	F	G	K	E	

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

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



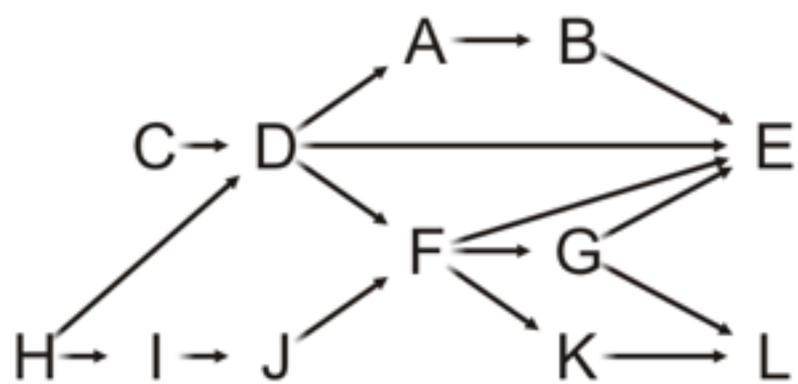
Queue:

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

Pop the front of the queue



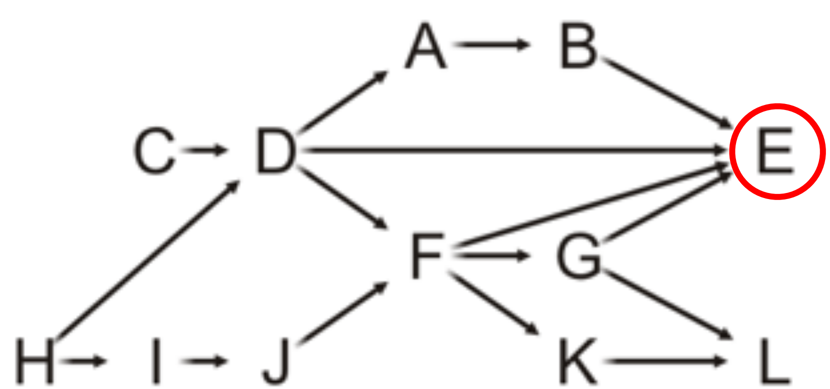
Queue:	C	H	D	I	A	J	B	F	G	K	E	L

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

Pop the front of the queue

– E has no neighbours



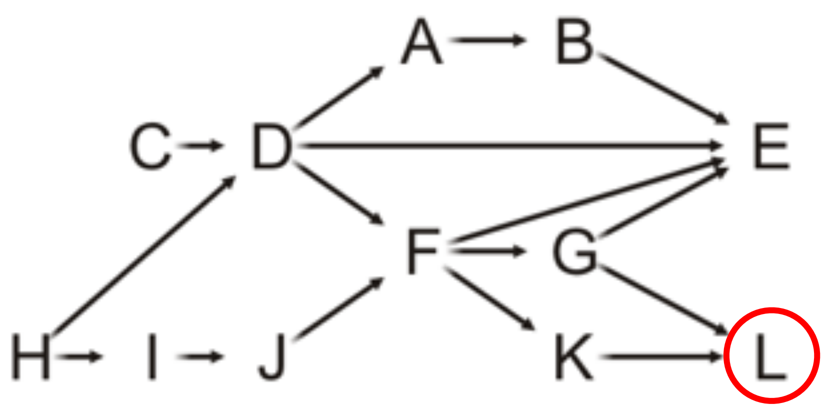
A	0
B	0
C	0
D	0
E	**0**
F	0
G	0
H	0
I	0
J	0
K	0
L	0

Queue:

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

Example

Pop the front of 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

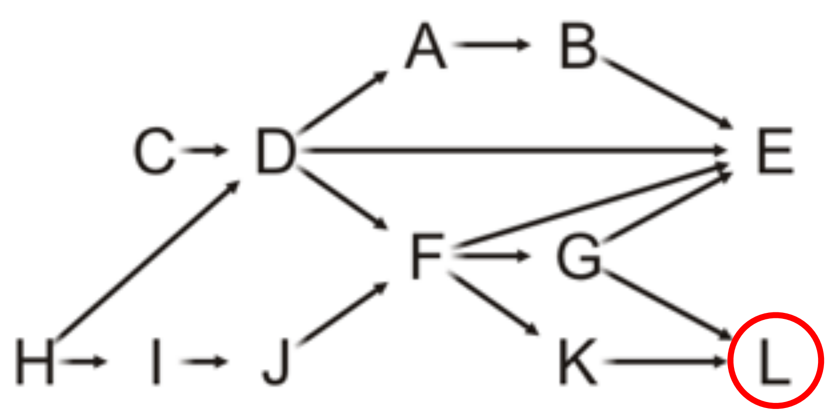
`Queue:`

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

Example

Pop the front of the queue

- L has no neighbours



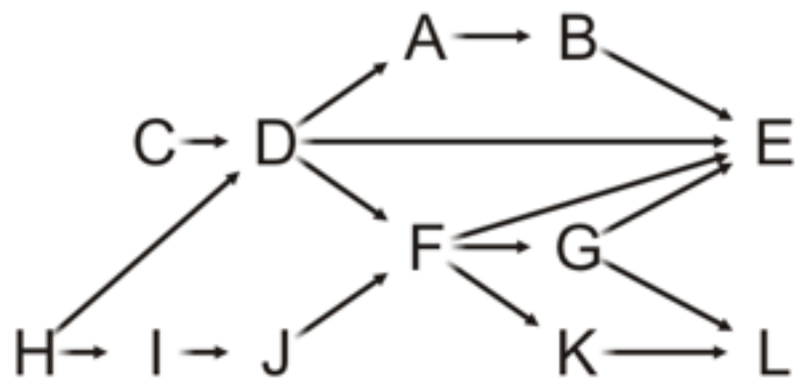
Queue:

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

The queue is empty, so we are done



Queue:

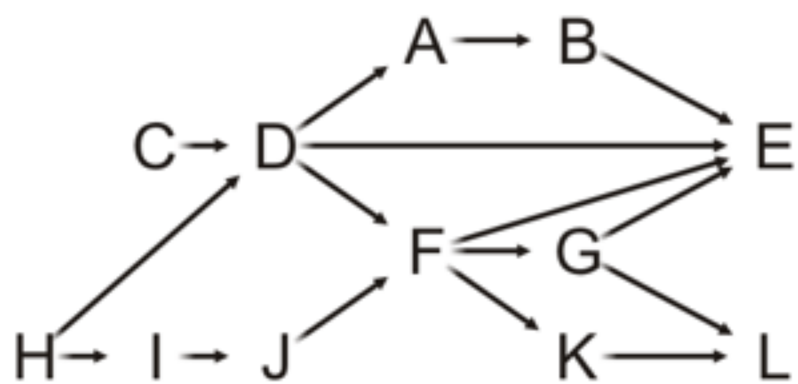
C	H	D	I	A	J	B	F	G	K	E	L

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

We deallocate the memory for the temporary in-degree array

The array stores the topological sorting



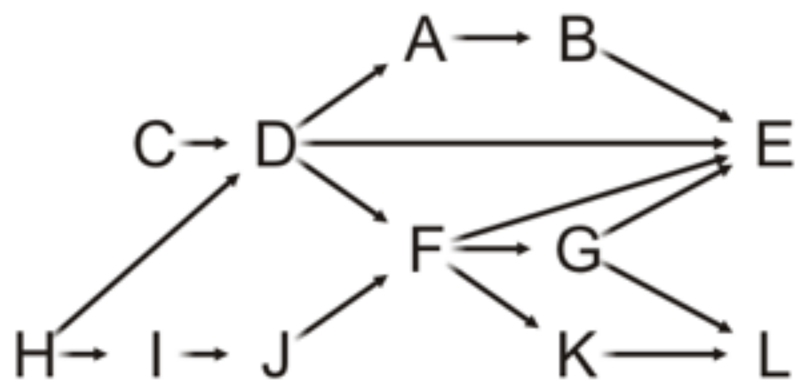
C	H	D	I	A	J	B	F	G	K	E	L

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

Example

Thus, one possible topological sort would be:

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

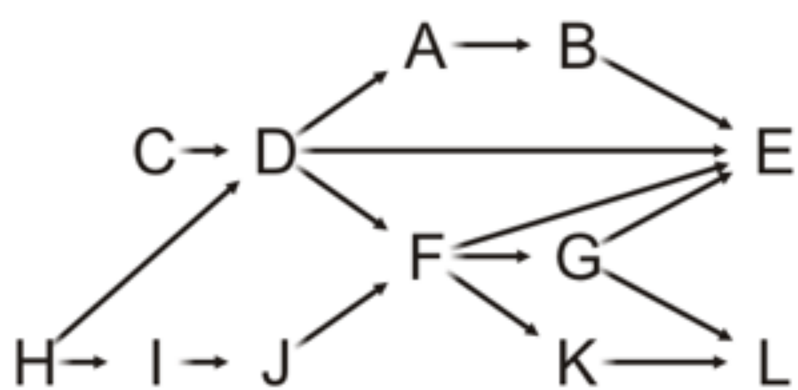


Example

Note that topological sorts need not be unique:

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

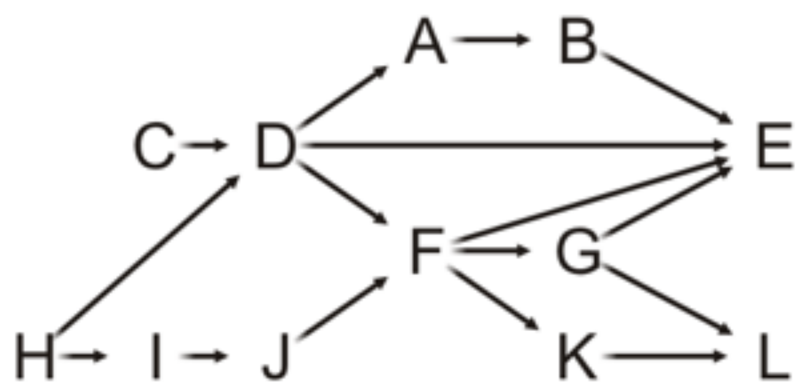
H, I, J, C, D, F, G, K, L, A, B, E



Analysis

What are the tools necessary for a topological sort?

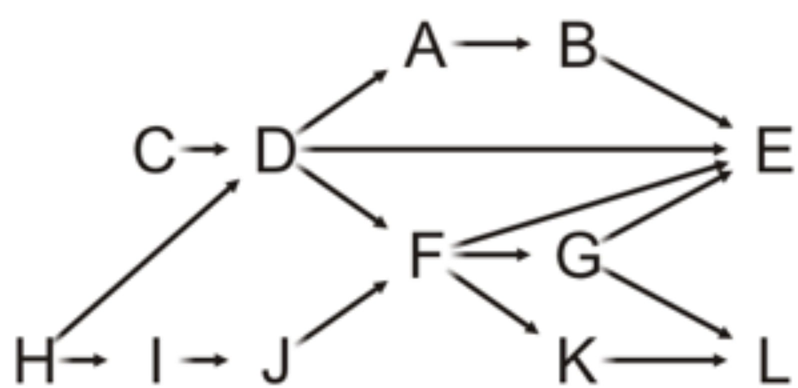
- This requires $\Theta(|V|)$ memory to store in-degrees
- Also requires $\Theta(|V|)$ memory for the queue



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

Analysis

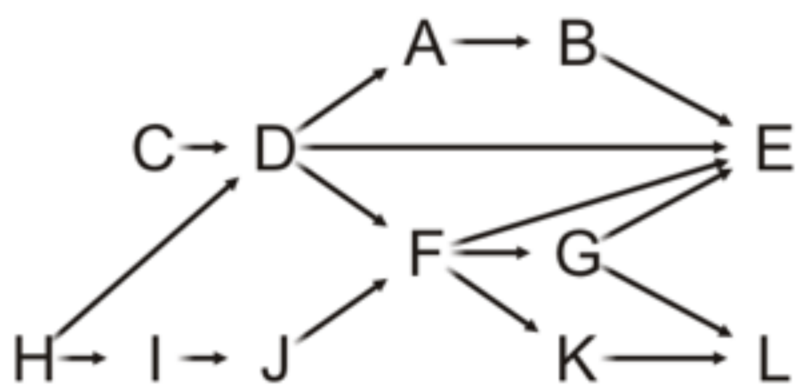
We must iterate $|V|$ times



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

Analysis

1. Each time we need to find vertices with in-degree zero
   - We could loop through the array with each iteration: run time would be $O(|V|^2)$
   - *Better approach*: each time the in-degree of a vertex is decremented to zero, push it onto the queue. It needs $O(1)$!, in total: $O(|V|)$

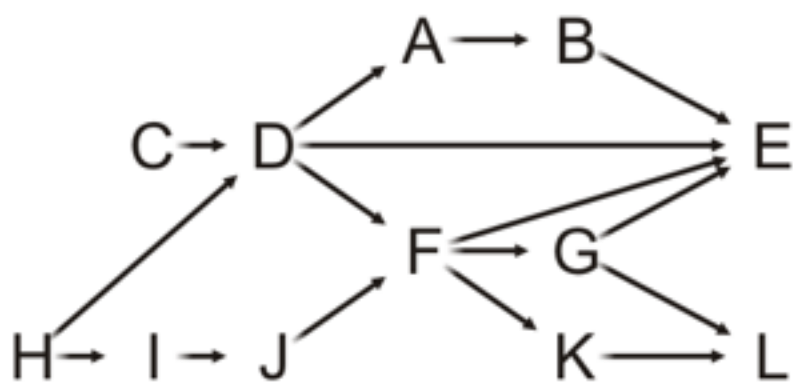


A	1
B	1
C	0
D	2
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

Analysis

2. What are the run times associated with the queue?

- Initially, we must scan through each of the vertices: $\Theta(|V|)$
- For each vertex, we will have to push onto and pop off the queue once ($O(1)$), in total: $\Theta(|V|)$

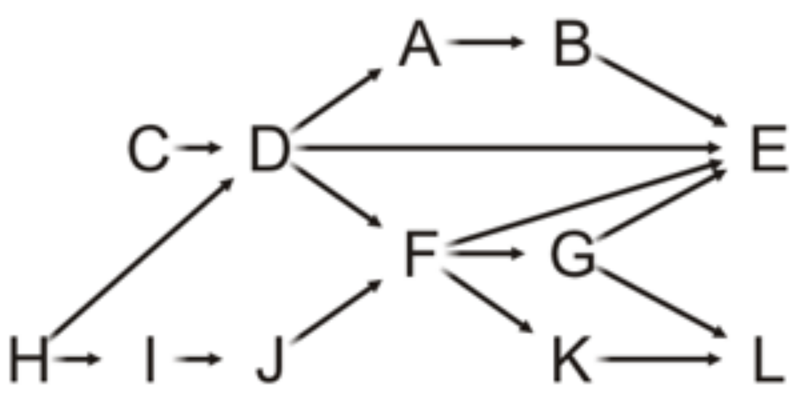


A	1
B	1
C	0
D	2
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

Analysis

3. Finally, each value in the in-degree table is associated with an edge

- Here, $|E| = 16$
- Each of the in-degrees must be decremented to zero
- Each edge is used, but never repeated: $\Theta(|E|)$



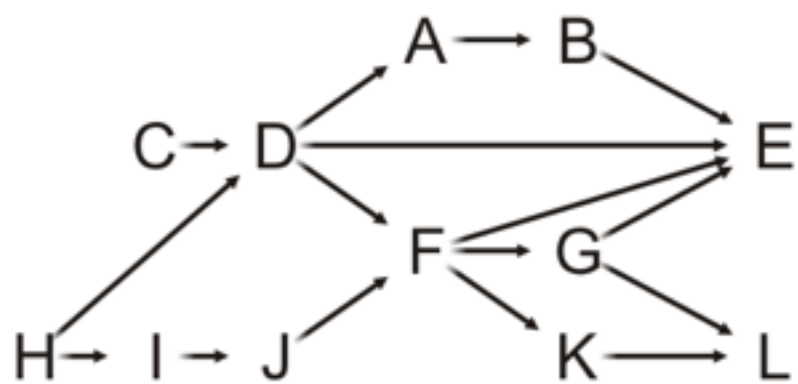
A	1
B	1
C	0
D	2
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2
+	16

Analysis

Therefore, the run time of a topological sort is: $\Theta(|V| + |E|)$

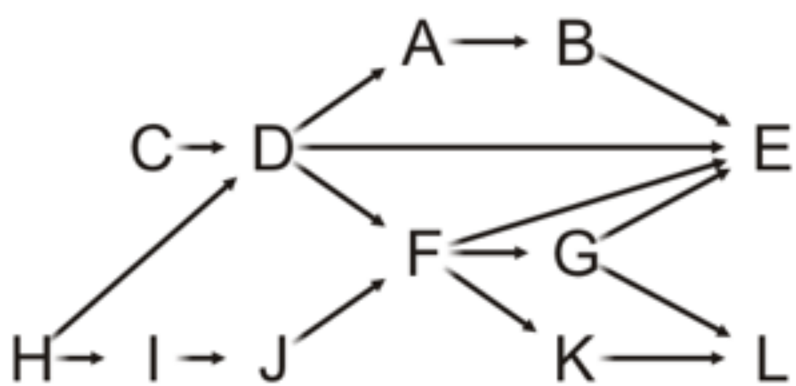
And the memory requirements is $\Theta(|V|)$

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



Analysis

What happens if at some step, all remaining vertices have an in-degree greater than zero?



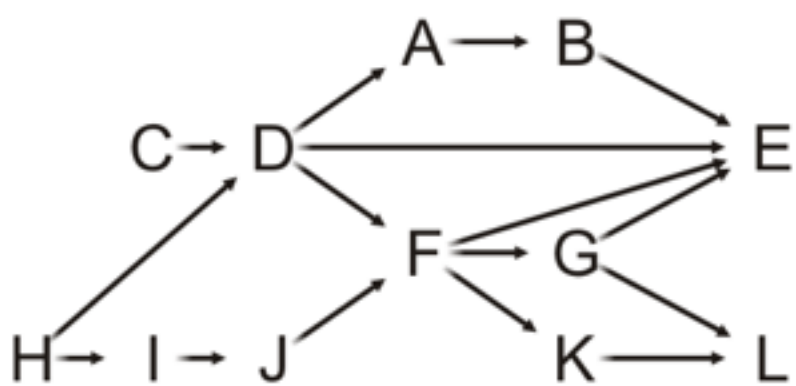
A	1
B	1
C	0
D	2
E	4
F	2
G	1
H	0
I	1
J	1
K	1
L	2

Analysis

What happens if at some step, all remaining vertices have an in-degree greater than zero?

- There must be at least one cycle within that sub-set of vertices

Consequence: we now have an $\Theta(|V| + |E|)$ algorithm for determining if a graph has a cycle



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

References

Wikipedia, http://en.wikipedia.org/wiki/Topological_sorting

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

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