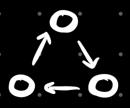
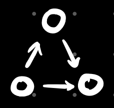
Topological Sorting

- An ordering of nodes in a directed, acyclic graph (DAG) such that for every directed edge (u,v), node u comes before node v in ordering (dependencies satisfied)
  - ↳ acyclic - no cycles in graph



cycle!



non-cycle!

- If a graph contains a cycle, topological sorting can't be implemented because there would be circular dependency, no linear order could satisfy all the dependencies

- TopSort does not need to have a unique output

Traversal Explanation

- Pick an unvisited node

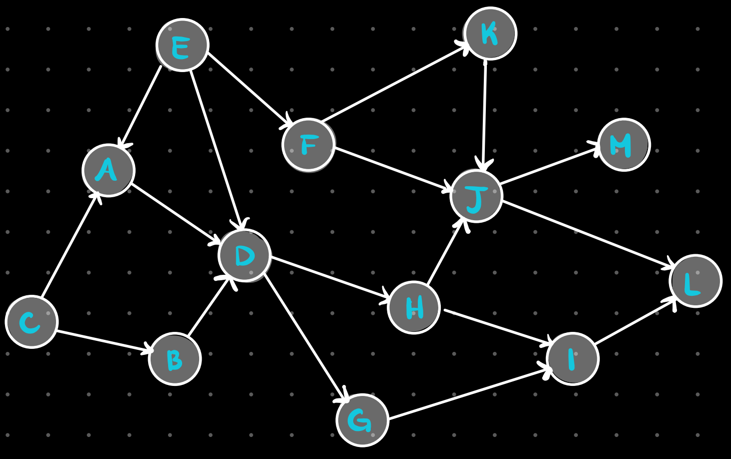
- Beginning with the selected node, do a DFS exploring only unvisited nodes

- On recursive callback of the DFS, add current node to the topological ordering in reverse order (or add to stack then at end, pop from stack)

Example

Assume you start on node "H" (chosen randomly)

DFS Visited	Topological Stack
B	C
C	B
K	E
F	F
G	K
D	A
A	D
E	G
M	H
J	J
L	M
I	I
H	L



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

Kahn's Algorithm

- An algorithm used to find the topological sort of a given graph
- Intuition behind Kahn's algorithm is to repeatedly remove nodes without any dependencies from the graph and add them to topological sorting.
- As nodes without dependencies (and their outgoing edges) are removed from the graph, new nodes without dependencies should become free.
- We repeat removing nodes without dependencies from the graph until all nodes are processed, or a cycle is discovered.

Example

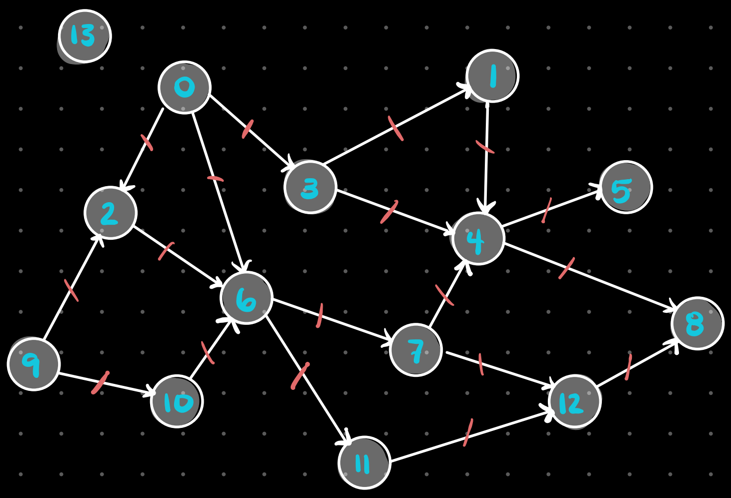
~~8~~
~~5~~
~~12~~
~~4~~
~~11~~
~~7~~
~~6~~
~~1~~
~~10~~
~~2~~
~~3~~
~~13~~
~~9~~
~~0~~

Queue

*Maintain queue of all nodes w/ no incoming edges (indegree = 0)

When a node gets an indegree of 0, add to queue

When you pop a node, remove its outgoing edges



Keep track of node indegrees:

~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~	~~8~~	~~9~~	~~10~~	~~11~~	~~12~~	~~13~~
0	~~1~~	~~2~~	~~1~~	~~3~~	~~1~~	~~3~~	~~1~~	~~2~~	0	~~1~~	~~1~~	~~2~~	0
	0	~~1~~	0	~~2~~	0	~~2~~	0	~~1~~		0	0	~~1~~	
		0		~~1~~		~~1~~		0				0	
				0		0							

Output: 0, 9, 13, 3, 2, 10, 1, 6, 7, 11, 4, 12, 5, 8

1. Add nodes w/ in-degree of 0 to queue
2. When popping from queue, add node to topological sort and remove it and its outgoing edges from graph
3. Add nodes w/ updated in-degree of 0 to queue if needed
4. Repeat until queue is empty