Plan for today:

- topological ordering algorithm
- quiz + break

- greedy algorithms

Last time:

directed acyclic graph

If G is a DAG, then it has a topological order.

Back to topo sort... proof by induction

Let G be an arbitrary DAG.

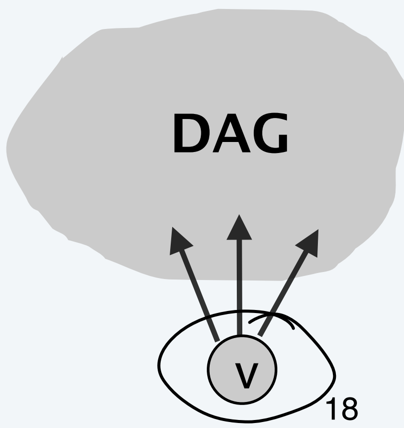
Assume that all DAGs with fewer nodes than G have topological orderings. (IH)

Case 1: G has one node. G has a topological ordering. ✓

Case 2. G has n > 1.

There exists a node with no entering edges. Remove this node to form G'. Notice that G' is a DAG with fewer nodes than G, so by our inductive hypothesis, G' has a topological ordering $v_1, v_2, \ldots, v_n$. Since the node we removed has no incoming edges, we can add it to this topological ordering in the first position.

So G has a topological ordering. Because G was an arbitrary DAG, all DAGs have topological orderings.



Back to topo sort...

Let G be an arbitrary DAG.

Assume that all DAGs with fewer nodes than G have topological orderings. (IH)

Case 1: G has one node. G has a topological ordering.

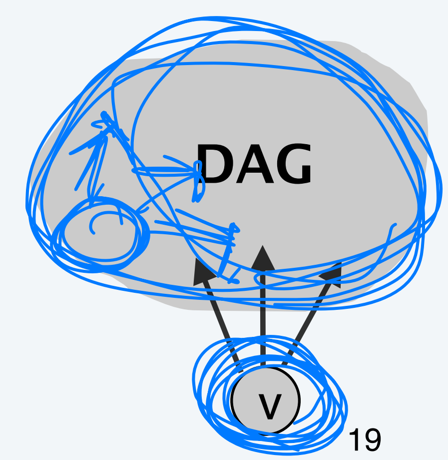
Case 2. G has n > 1.

There exists a node with no entering edges. Remove this node to form G'. Notice that G' is a DAG with fewer nodes than G, so by our inductive hypothesis, G' has a topological ordering $v_1, v_2, \ldots, v_n$. Since the node we removed has no incoming edges, we can add it to this topological ordering in the first position.

So G has a topological ordering. Because G was an arbitrary DAG, all DAGs have topological orderings.

G G' G'' . . . graph w/ single node

What's the algorithm?



Back to topo sort...

order = 1 2 5

Let G be an arbitrary DAG.

Assume that all DAGs with fewer nodes than G have topological orderings. (IH)

$3 \to 4$

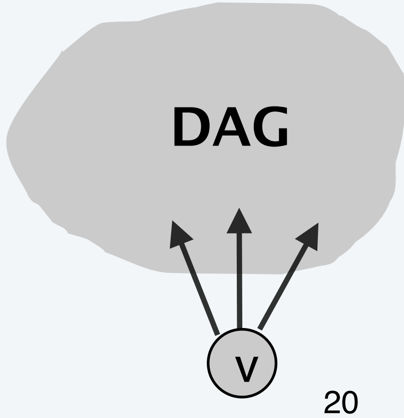
Case 1: G has one node. G has a topological ordering.

Case 2. G has n > 1.

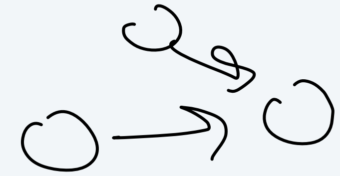
There exists a node with no entering edges. Remove this node to form G'. Notice that G' is a DAG with fewer nodes than G, so by our inductive hypothesis, G' has a topological ordering $v_1, v_2, \ldots, v_n$. Since the node we removed has no incoming edges, we can add it to this topological ordering in the first position.

So G has a topological ordering. Because G was an arbitrary DAG, all DAGs have topological orderings.

```
To compute a topological ordering of G:
Find a node v with no incoming edges and order it first
Delete v from G
Recursively compute a topological ordering of G−{v}
  and append this order after v
```



What's the runtime?



```
To compute a topological ordering of G:
Find a node v with no incoming edges and order it first
Delete v from G
Recursively compute a topological ordering of G−{v}
  and append this order after v
```

$n \cdot (n-1)(n-2) \cdots = n!$

$n + n-1 + n-2 + \cdots$

Naive upper bound:

- $n-1$ recursive calls (one for every node except the first one we remove)
- at each recursive call, we need to find v w/no incoming edges – runtime is upper bounded by n.

$O(n^2)$ and in fact $\Theta(n^2)$.

Topological sorting algorithm: running time

Theorem. Algorithm finds a topological order in $O(m + n)$ time.

Pf.

- Maintain the following information:
  - $count(w)$ = remaining number of incoming edges
  - S = set of remaining nodes with no incoming edges
- Initialization: $O(m + n)$ via single scan through graph.
- Update: to delete v
  - remove v from S
  - decrement $count(w)$ for all edges from v to w; and add w to S if $count(w)$ hits 0
  - this is $O(1)$ per edge ▪

Homework Quiz

back from break @

10:20