Oct 31 Problem:

Input: A directed graph $G = (V, E)$

Output: A topological ordering of G (if one $\exists$)

Lemma 1: If G has a topological ordering $\Rightarrow$ G is a DAG

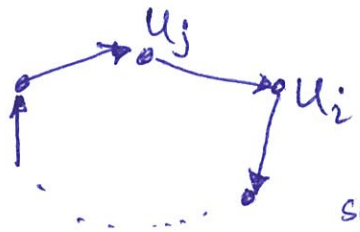
THEOREM: If G is a DAG $\Rightarrow$ G has a topological ordering

---

Pf (idea) of Lemma 1: By contradiction [Assume $\neg(A \Rightarrow B) \equiv A \wedge \neg B$]

Assume $u_1, \ldots, u_n$ is a topological ordering of G

BUT G has a directed cycle C.



Let i be the smallest index ($i \in [n]$) s.t. $u_i \in C$

$\Rightarrow$ $\exists\, j$ s.t. $(u_j, u_i) \in C$

since C is a cycle

$\Rightarrow (u_j, u_i) \in E$

BUT $j > i$ (by definition of i) $\Rightarrow$ a backward edge (u_j, u_i),

$\Rightarrow$ contradiction ∎

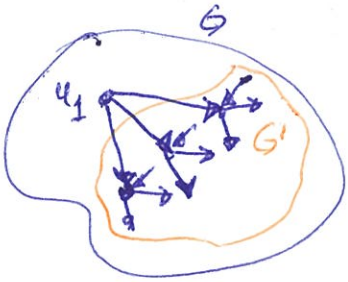
Pf (idea) Thm: Algorithmically prove that any DAG has a topological ordering

---

"Reverse Engineer" the algo: Assume $u_1, \ldots, u_n$ is a topological ordering

Q: How many incoming edges does u_1 have

A: 0 as all edges (u_1, u_i) should be forward edges.



$G' \stackrel{\text{def}}{=} G \setminus \{u\} = (V \setminus \{u\}, E \setminus \{(u, w) \mid w \in V\})$.

Lemma 2: G' is a DAG (EX)

Idea: Deleting edges/vertices cannot create a cycle.

Lemma 3: If G is a DAG, then $\exists\ w \in V$ s.t. w has no incoming edges in (V,E).

Algo idea: Recurse! TopOrd $(G=(V,E))$

$\exists$ by Lemma 3

$O(1)$ → 1. If $V=\{u\}$, output u

2. Let w be a vertex with no incoming edges } $O(n)$

3. $G' = G \setminus \{w\}$ ← (ASSUME: $O(n)$)

4. Output w, TopOrd(G') $O(1)$

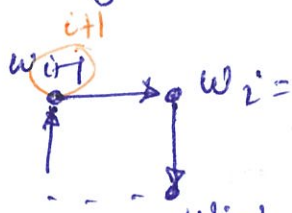
Pf idea of Lemma 3: By contradiction

Assume G is a DAG BUT all vertices have ≥ 1 incoming edge.

$n+1$ labels $\in V$ $(w_1, \ldots, w_{n+1})$

w_{n+1} w_n ... w_2 w_1 but n nodes

Pigeon hole principle ⟹ $\exists\ i < j$ s.t. $w_i = w_j$

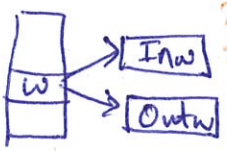


$\Rightarrow \exists$ a cycle in G

$\Rightarrow$ a contradiction ∎

Ex: TopOrd is correct.

Adjacency list. Q: Given $u \in V$, how quickly can you figure out if w has no incoming edge?



A: $O(1)$

$\Rightarrow$ Step 2 can be done in $O(n)$ time (just look at all vertices $u \in V$)

$\Rightarrow$ Any recursive call takes $O(n)$ time

But $\leq n$ recursive calls (since in each step algo throws away one vertex)

$\Rightarrow$ OVERALL: $\leq n \cdot O(n) = O(n^2)$.

Can get $O(m+n)$