CS 2100 - Directed Acyclic Graphs

Friday
- HW due in class
- Review lecture

Monday
- Review Session

Next Wed @ 8am: Final
Practice final is up front
Directed Graphs

Directed graphs are encountered in many applications.

$(u,v) \in E$ : $u \rightarrow v$

We say the number of edges going into $u$ is the \underline{in-degree}.

(And the \underline{out-degree} is the \# of edges leaving the vertex.)
Traversals in directed graphs

Detecting if there is a path from $s$ to $t$ in a directed graph can be done in $O(m+n)$ time.

Idea: Modify BFS/DFS to only add outgoing edges to stack/queue.
Directed Acyclic Graphs

If no directed cycles, called a directed acyclic graph, or DAG.

While specialized, still useful:

Ex: -pre-reqs in a degree program

CSCI 1800 → CSCI 1300 → CSCI 2100

MAT1 660
Ex: Inheritance in C++

Ex: Completing a large project by breaking into smaller ones
Let $G$ be a directed graph with $n$ vertices.

A topological ordering of $G$ is a list $v_1, v_2, \ldots, v_n$ such that for every edge $(v_i, v_j) \in E$, $i < j$.

(So we order vertices so that edges only go forward.)
Not unique:
Prop: $G$ has a topological ordering if and only if it is acyclic.

pf: $\Rightarrow$: Assume $G$ has top ordering $V_1, V_2, V_3, \ldots, V_n$.

A cycle would need "backwards" edge which is impossible since it is a top ordering.

$\Leftarrow$: Suppose $G$ is acyclic.

Find a vertex with 0 indegree.

Can do this since $G$ is acyclic.

But that vertex first, delete it and repeat.
Algorithm:

Track in degrees
Repeat until no vertices.
Find one that is 0,
put vertex first,
delete v from G.
Pseudo code:

$S = \text{initially empty stack}$

For all $u \in V$

Let $I[u] = \text{in-degree of } u$

If $I[u] = 0$

$S.push(u)$

$i = 1$

while $!S.empty()$

$u = S.pop()$

Let $u$ be vertex $i$, $i = i + 1$

for all $(u,v) \in E$

$I[v] = I[v] - 1$

If $I[v] = 0$

$S.push(v)$

$\mathcal{O}(m+n)$

repeats for every vertex
Claim: Yields a topological ordering

Key insight:

When \( \text{IN}[v] = 0 \), all vertices with edges going into \( v \) have already been "placed" earlier.
Runtime:  \text{setting up I} \downarrow \text{stack loop} \downarrow \text{Total: } O(m+n) + \sum_{v \in V} \left(1 + d^+(v)\right) \\
= O(m+n)