CS 180 - Directed Acyclic Graphs

Announcements
- Sample final available
- HW due Monday
- Final a week from Monday
- Review session: 2pm on Friday
Directed Graphs

Directed graphs are encountered in many applications.

\((u,v) \in E\) 

\(\neq (v,u)\)

We say the number of edges going into \(u\) is the \textit{in-degree}.

(And \textit{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 $O(n+m)$ 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

\[ \text{cs150} \rightarrow \text{cs180} \rightarrow \text{cs290} \rightarrow \text{cs390} \]

\[ \text{math 135} \]
Ex: Inheritance in C++

Compile ancestors first (makefiles)

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:

- vertices with indegree 0 first
Prop: \( G \) has a topological ordering if and only if \( G \) is acyclic.

pf:

\( \Rightarrow \): Suppose \( G \) has top. ordering \( v_1, v_2, v_3, v_4, \ldots, v_n \) no cycle, since all edge are "forward".

\( \Leftarrow \): By contrapositive: Suppose no topological ordering.

At some point in any ordering, every vertex has indegree \( \geq 0 \).

\( \Leftarrow \) Can form some cycle.

Algorithm:
while vertex is left:
    Pick vertex of indegree 0
    Put it next in list
    Remove it & its edges
Pseudo code:

\[
S = \text{initially empty stack} \\
\text{repeat} \ n\ \text{times} \\
\quad \text{for all } u \in X \\
\quad \quad \text{let } I[u] = \text{in-degree of } u \\
\quad \quad \text{if } I[u] = 0 \quad \text{write } \quad \text{adj list } \quad O(m) \\
\quad \quad S.push(u) \\
\quad e = 1 \\
\quad \text{while } \! S.\text{empty}() \\
\quad \quad u = S.\text{pop}() \\
\quad \quad \text{let } u \text{ be vertex } i, \quad e = e + 1 \\
\quad \quad \text{for all } (u,v) \in E \\
\quad \quad \quad I[v] = I[v] - 1 \\
\quad \quad \quad \text{if } I[v] = 0 \\
\quad \quad \quad S.push(v) \\
\]
Claim: Yields a topological ordering

Key insight:

When $|I[v]| = 0$, all vertices with edges going into $v$ have already been "placed" earlier.

(see proof)
Runtime:

Each time we do $T[v] = T[v] - 1$ in loop, we’ve removed an edge.

$\Rightarrow O(m)$ time repeated total

$O(m + n)$

#edges