CS 2100 - Directed Acyclic Graphs

Note Title 12/5/2012

Friday

- HW due in class
- Review lecture

Monday

- Review Session

Next Wed @ 8 am: Final

Practice final is up front

Directed Graphs

Directed graphs are encountered in many applications.

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

We say the number of edges going into u is the in-degree.
(And 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.

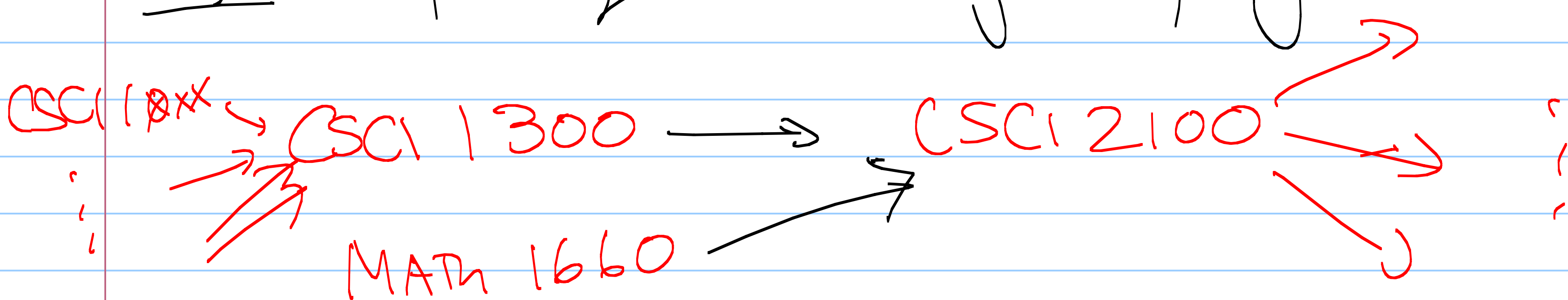
Idea: Modify BFS/DFS to only add outgoing edges to stack/queue.

Directed Acyclic Graphs

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

While specialized, still useful:

Ex: - prereqs in a degree program



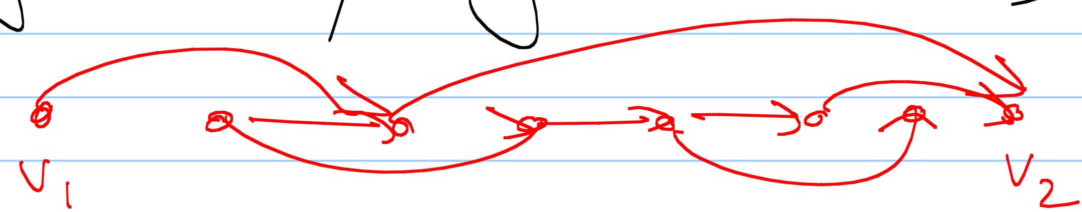
Ex: Inheritance in C++

Ex: Completing a large project by breaking into smaller ones

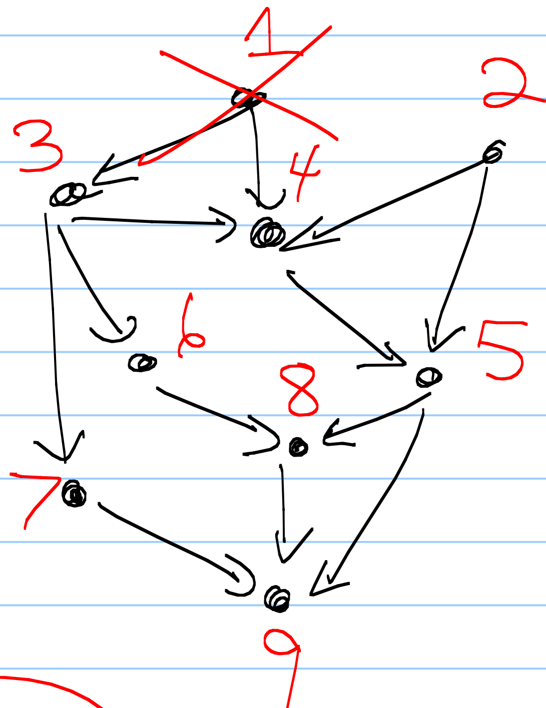
Let G be a directed graph with n vertices.

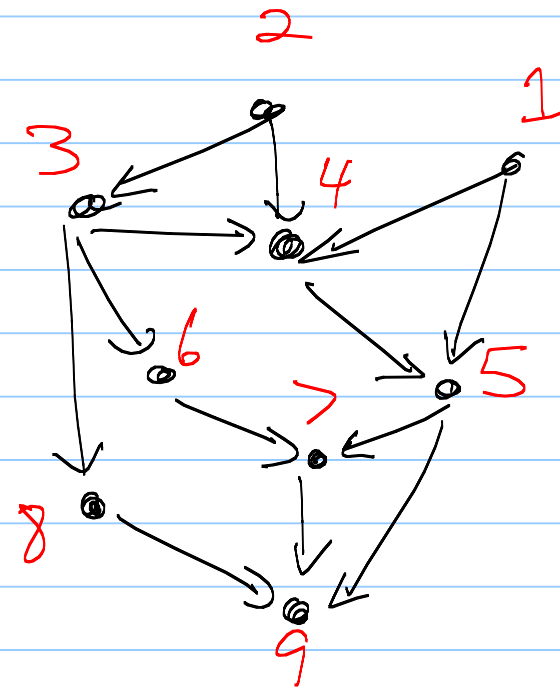
A topological ordering of G is a list: $v_1, v_2, \ldots, v_n$ such that for every edge $(v_i, v_j) \in E$, $i < j$.

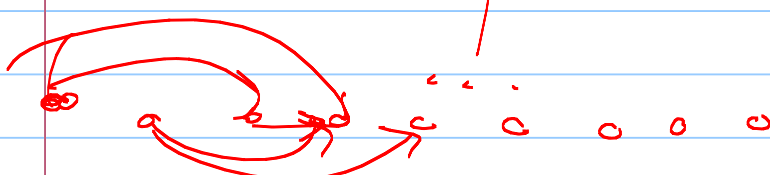
(So we order vertices so that edges only go forward.)



Not unique:

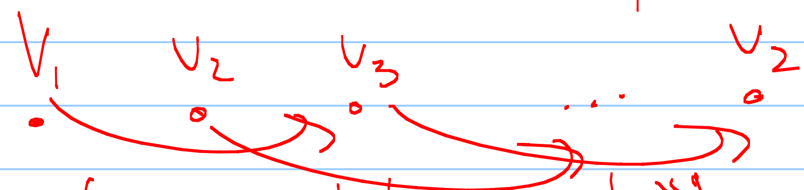






Prop: G has a topological ordering if & only if it is acyclic.

pf: $\Rightarrow$: Assume G has top ordering



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

$\Leftarrow$: Spps G is acyclic:

Find a vertex with 0 indegree.

↳ Can do this since G is acyclic.

Put that vertex first, delete it & repeat.

Algorithm:

Track indegrees

Repeat until no vertices.

Find one that is 0,

put vertex first

delete v from G

Pseudocode:

```
S = initially empty stack
For all u ∈ V
    Let I[u] = 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) ∈ E
        I[v] = I[v] - 1
        if I[v] == 0
            S.push(v)
```

$O(m+n)$

← repeats for every vertex

← $O(1)$

← $O(1)$

← outdegree of u

$O(1)$

Claim: Yields a topological ordering

Key insight:

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

Runtime:

Total: $\underbrace{O(m+n)}_{\text{setting up } I} + \underbrace{\sum_{v \in V} (1 + d^+(v))}_{\text{stack loop}}$

$\sum_{v \in V} (1 + d^+(v)) = O(m+n)$

$= \boxed{O(m+n)}$