

CS180 - Directed Acyclic Graphs

Note Title

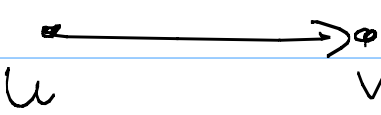
12/5/2012

Announcements

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

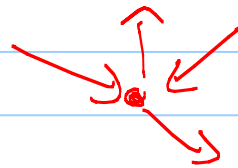
Directed Graphs

Directed graphs are encountered in many applications.

$(u, v) \in E$:  $\xrightarrow{\quad}$
 $\times (v, u)$

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.

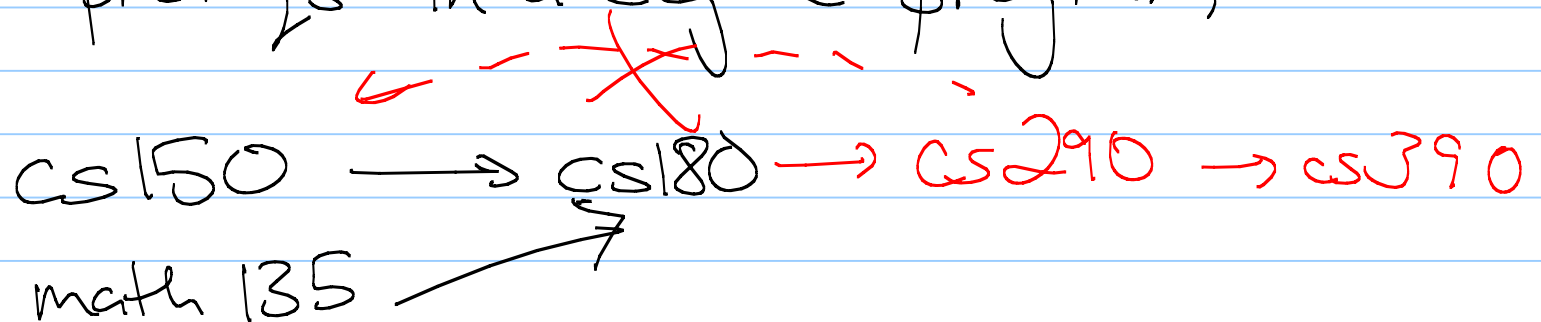
Idea: Modify BFS/DFS to only add outgoing edges to stack/queue.

Directed Acyclic Graphs

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

While specialized, still useful:

Ex: -pre reqs in a degree program



Ex: Inheritance in C++

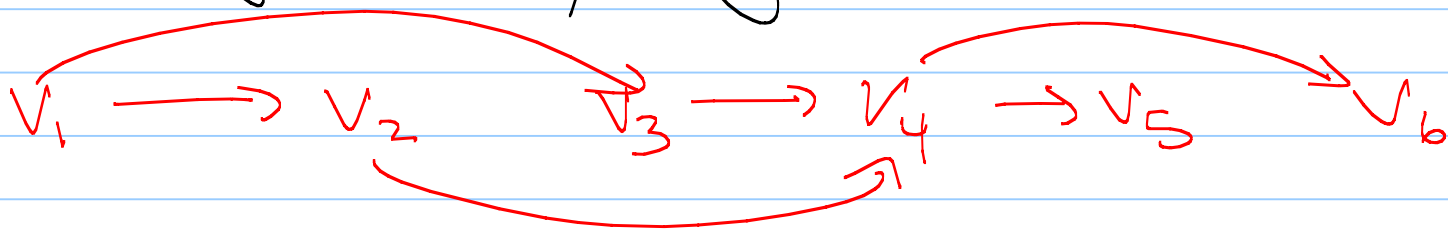
Compile ancestors first
(makefiles)

Ex: Completing a large project
by breaking into smaller ones

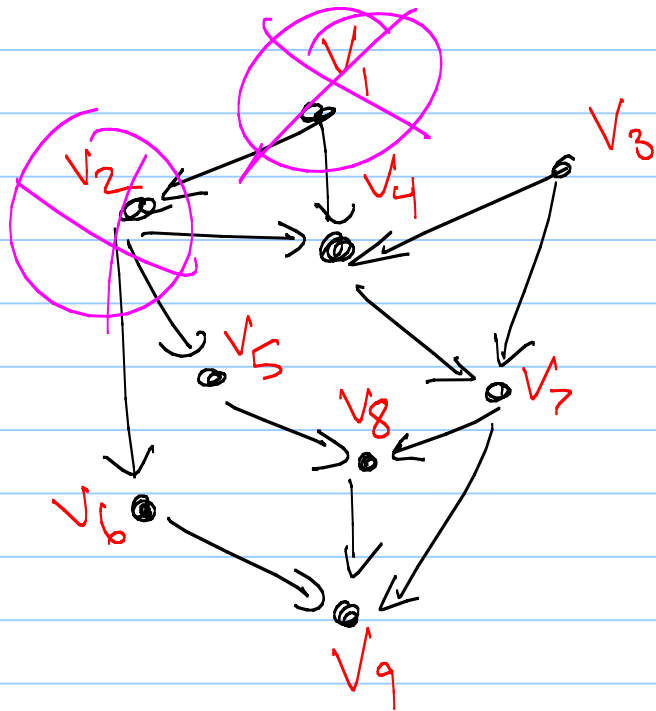
Let G be a directed graph with n vertices.

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

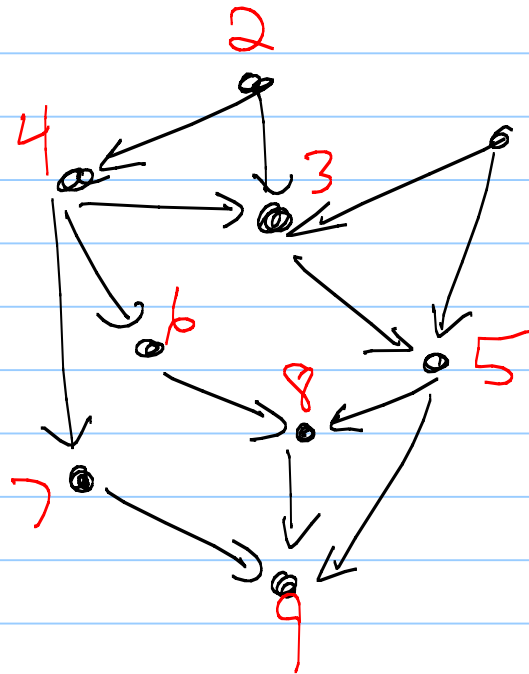
(So we order vertices so that edges only go forward.)

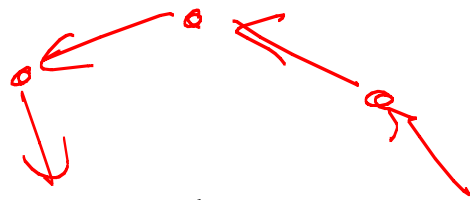


Not unique:



- vertices with
indegree 0 first





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

pf: \Rightarrow : Spps G has top. ordering
 $v_1, v_2, v_3, v_4, \dots, v_n$
 no cycle, since all edges are
 "forward"

\Leftarrow : By Contrapositive: Spps no
 topological ordering.

At some point in any ordering,
 every vertex has indegree > 0 .
 \Rightarrow Can form some cycle \Rightarrow

Algorithm:

~~white~~ vertex is left:

Pick vertex of indegree 0

Put it next in list

Remove it & its edges

Pseudocode:

S = initially empty stack

For all $u \in V$

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

If $I[u] == 0$ } $O(V)$
S.push(u)

} writes
adj list,
 $O(m)$

$i = 1$

while !S.empty()

u = S.pop()

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

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

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

if $I[v] == 0$

S.push(v)

repeats
n times

n
times

Claim: Yields a topological ordering

Key insight:

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

(see proof)

Runtime:

Each time we do $I[v] = I[v] - 1$
in ~~the~~ loop, we've removed
an edge.

$\Rightarrow O(m)$ time repeated total

$O(m+n)$
↑
#edges