CS180 – Directed Acyclic Graphs

Note Title 12/5/2012

Announcements

- Final HW not graded
  one question on final
- Sample final is printed
- Lab tomorrow
- Office hours next Friday: 9-11 am

Directed Graphs

Directed graphs are encountered in many applications.

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

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

(And out-degree is the # of edges leaving the vertex.)

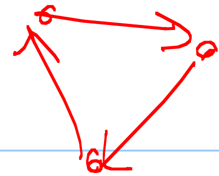
Traversals in directed graphs

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

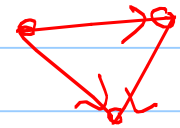
Idea: Modify BFS/DFS to only add outgoing edges to stack/queue.

Directed Acyclic Graphs

directed cycle



Not:



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

While specialized, still useful:

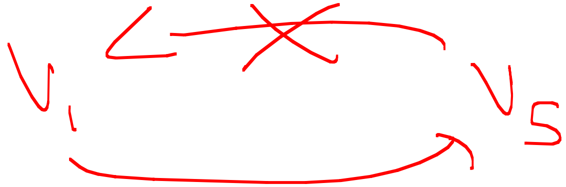
Ex: – prereqs in a degree program

cs150 → cs180 → cs443 →

math 135 → cs180 → cs224 →

Ex: Inheritance in C++
(Compilation)

Ex: Completing a large project by breaking into smaller ones

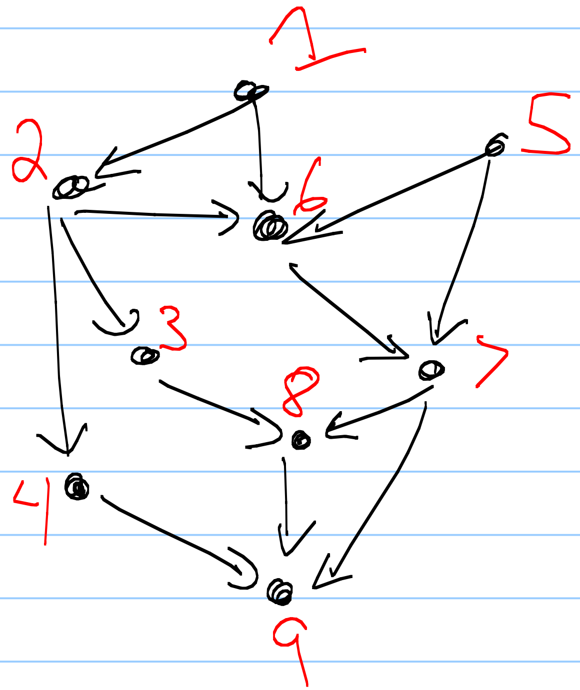


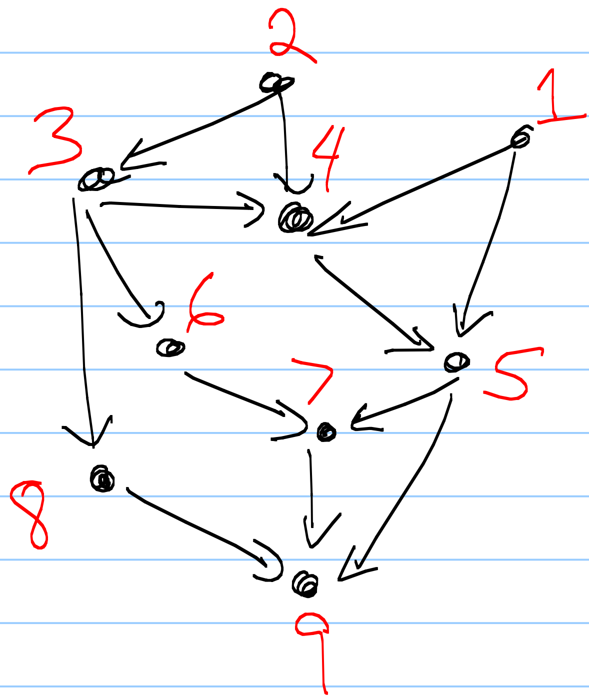
Let G be a directed graph with n vertices.

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:



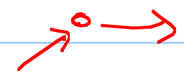


Prop: G has a topological ordering if & only if it is acyclic.

pf: $\Rightarrow$: Spps top ordering $v_1 \dots v_n$

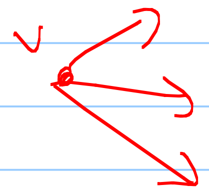
then no cycle

$\Leftarrow$: Spps acyclic.



$\Rightarrow$ some vertex has indegree 0

& can go first



remove v & repeat

Algorithm:

Implement previous proof.

Find v of indegree 0

Put it next

Remove v & its edges

Pseudocode:

```
S = initially empty stack
For all u ∈ V
    Let I[u] = in-degree of u        ← all u
    if I[u] == 0                       in(d(v))
        S.push(u)                         ⇓
                                        O(m)
i = 1
while !S.empty()
O(1) → u = S.pop()
O(1) → Let u be vertex i, & i = i+1
       for all (u,v) ∈ E
O(m)       I[v] = I[v] - 1
           if I[v] == 0
               S.push(v)
```

<u>Claim:</u> Yields a topological ordering

Key insight:

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

Runtime: $O(m+n)$