CS2100

directed graphs

Recap:

See last set of slides.

I am here next Tuesday.

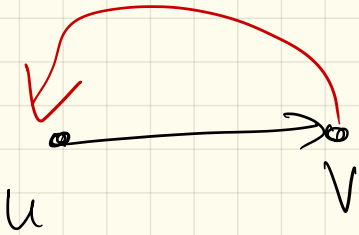
↳ may be in 117 Riffer

Today: Directed graph searching

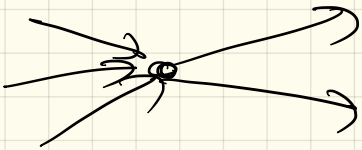
Recall:

$(u,v) \in E$:

not $\{u,v\}$



$(v,u) \in E$



$\text{indegree}(u) \neq \text{outdegree}(u)$

H-S Lemma:

$$\sum_v \text{indeg}(v) + \sum_v \text{outdeg} = 2|E|$$

First: traversal

We can modify BFS/DFS code easily:

queue stack

```
TRAVERSE(s):
    put s into the bag
    while the bag is not empty
        take v from the bag
        if v is unmarked
            mark v
            for each edge vw
                put w into the bag
```

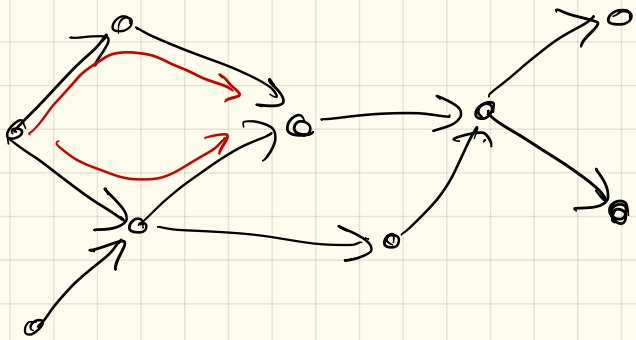
only add outgoing edges

Runtime: $O(m+n)$

Special interesting case:

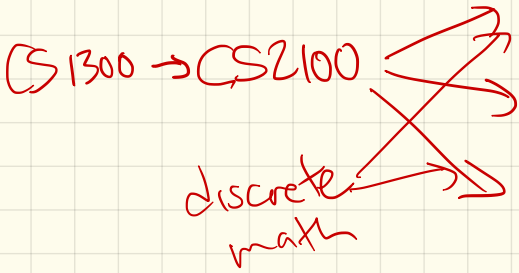
Directed acyclic graphs:

graphs with no directed cycles



These are specialized, but still useful:

Ex: preregs/classes in a major



Ex (cont):

Inheritance or makefiles in C++

error

BinaryTree → BST → AVL

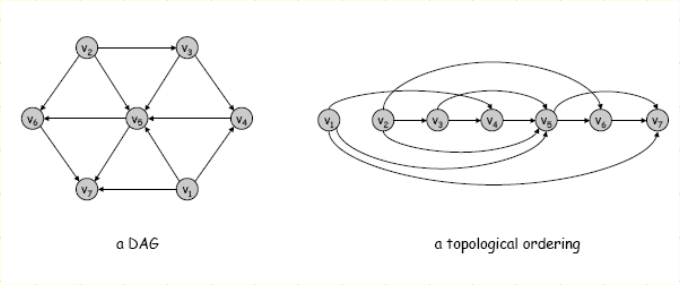
I/O

Ex: In software engineering, dependency charts

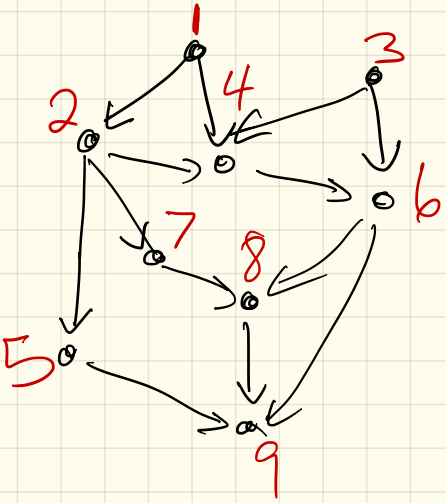
divide task up into pieces

Definition: Given G = a directed graph with n vertices, a **topological ordering** of G is a vertex list: $v_1, v_2, \ldots, v_n$ such that for every edge $(v_i, v_j) \in G$, $i < j$.

[In other words, edges only point forward.]

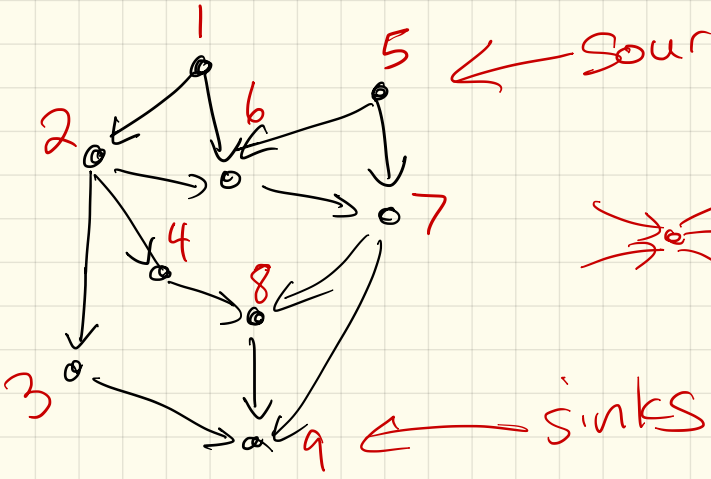


Note: these are not unique.



Pick a source & put it 1st.

delete it & repeat



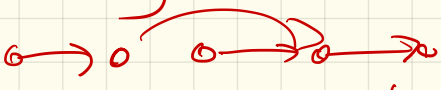
← sources

← sinks

Proposition: G has a topological ordering $\iff$ G is acyclic.

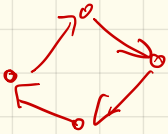
find a source or sink

$\Rightarrow$: If have a top ordering



can't have a cycle.

$\Leftarrow$: If have a dir. cycle.



can't lay these out.

Leads to algorithm:

Sort by indegree. source

Someone must have indegree 0.

Put him next, delete & update indegrees.

(Repeat)

Or – nicer:

might be on final

O(n)-ish

```
TopologicalSort(G):
  n ← |V|
  for i ← 1 to n
      v ← any source in G
      S[i] ← v
      delete v and all edges leaving v
  return S[1..n]
```

out-d(v)

```
TopologicalSort(G):
  n ← |V|
  for i ← n down to 1
      v ← any sink in G
      S[i] ← v
      delete v and all edges entering v
  return S[1..n]
```

i

in-deg

Runtime:

$O(n^2)$ or $O(m+n)$

Weighted Graphs

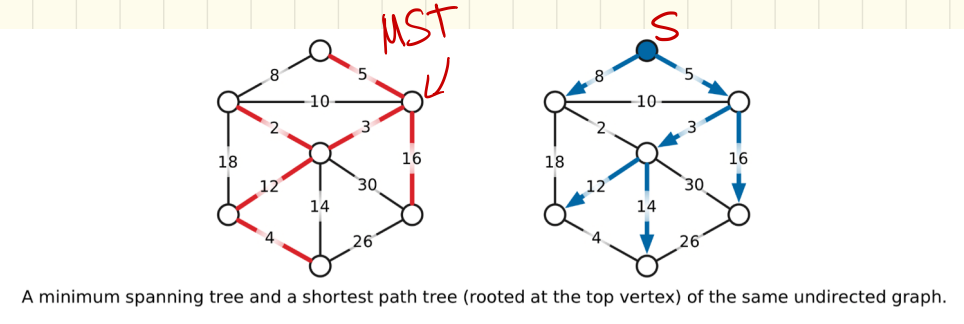
Just a note or two:

- If the graph is weighted, each edge e has a value $w(e)$.

Then, can ask for

- shortest paths
- minimal spanning trees

These are not the same:



A minimum spanning tree and a shortest path tree (rooted at the top vertex) of the same undirected graph.

(Not on final)