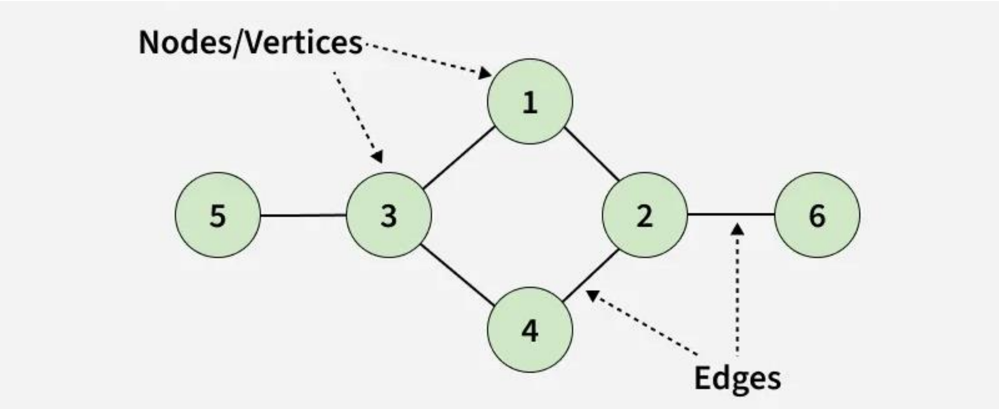
Lecture 8

Graphs IV –Digraphs and Topo Sort



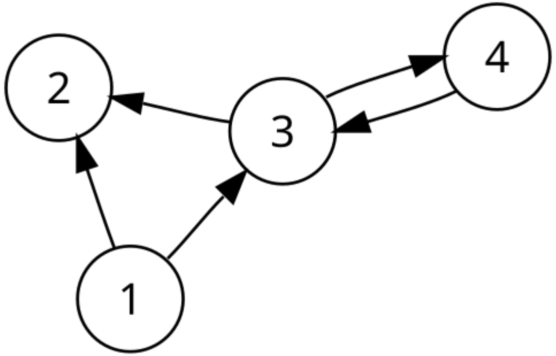
Announcements

- Homework 4 due Sunday night
- Group Meetings continue
- Quiz 1 due next Friday
  - Released earlier next week
  - A few questions from each topic

Directed Graphs

$G = (V, E)$

- $(u, v) \in E$ is a directed edge
- We say that u point to v
- $e = (u, v)$ leaves u, enters v, is an outgoing edge from u, is an incoming edge to v



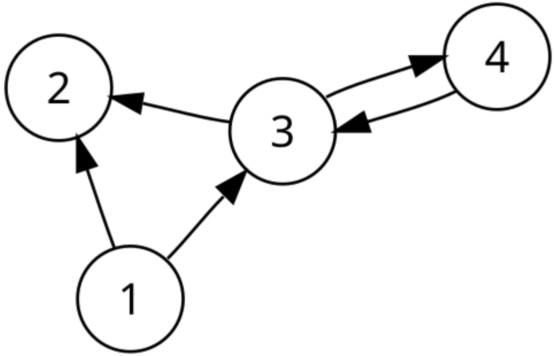
Directed Graphs

$G = (V, E)$

- $(u, v) \in E$ is a directed edge
- We say that u point to v
- $e = (u, v)$ leaves u, enters v, is an outgoing edge from u, is an incoming edge to v

Examples:

- Facebook, LinkedIn: undirected
- Twitter, Instagram: directed
- Web links: directed
- Road network: directed



Digraph Definitions

Most definitions extend naturally from undirected to directed graphs by mapping the word "edge" to "directed edge"

- **Directed path**: a sequence $P = v_1, v_2, \ldots, v_{k-1}, v_k$ such that each consecutive pair $v_i, v_{i+1}$is joined by a directed edge in G. We call this a $v_1 \to v_k$ path.
- **Directed cycle**: a directed path with $v_1 = v_k$
- When referring to a directed graph, the words "path" and "cycle" mean "directed path" and "directed cycle"
- **Connected? Connected component?** More subtle, because now there can be a path from s to t but vice versa.

Directed Graph Traversal

Problem: Directed Reachability

- Find all nodes reachable from some node s
- What is the length of the shortest directed path from s to t?

Directed Graph Traversal

Problem: Directed Reachability

- Find all nodes reachable from some node s
- What is the length of the shortest directed path from s to t?
- BFS/DFS naturally extend to directed graphs

Directed Graph Traversal

Dir-BFS(s):

mark s as "discovered"

$L[0] \leftarrow \{s\}; i \leftarrow 0$

while $L[i]$ is not empty **do**

$L[i+1] \leftarrow$ empty list

for all vertices v in $L[i]$ **do**

for all outgoing neighbors w of v **do**

if w is not marked "discovered" **then**

mark w as "discovered"

$L[i+1].append(w)$

$i \leftarrow i+1$

Exercise I

Q: Suppose G is a directed path on n vertices and Dir-BFS is called repeatedly starting from any unexplored vertex until all vertices are explored. What is the maximum number of times Dir-BFS may be called?

a) $n-1$

b) n

c) 1

d) m

Exercise I

Q: Suppose G is a directed path on n vertices and Dir-BFS is called repeatedly starting from any unexplored vertex until all vertices are explored. What is the maximum number of times Dir-BFS may be called?

a) $n - 1$

b) n

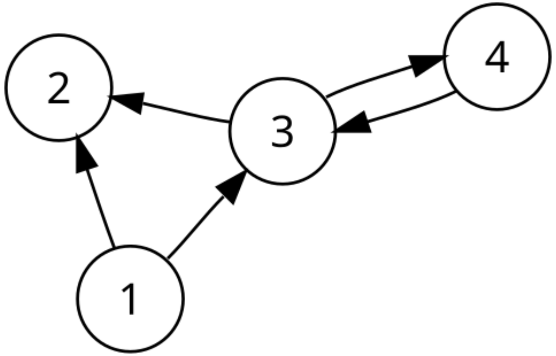
c) 1

d) m

Directed Graph Traversal

How can we find all vertices v with a $v \to t$ path

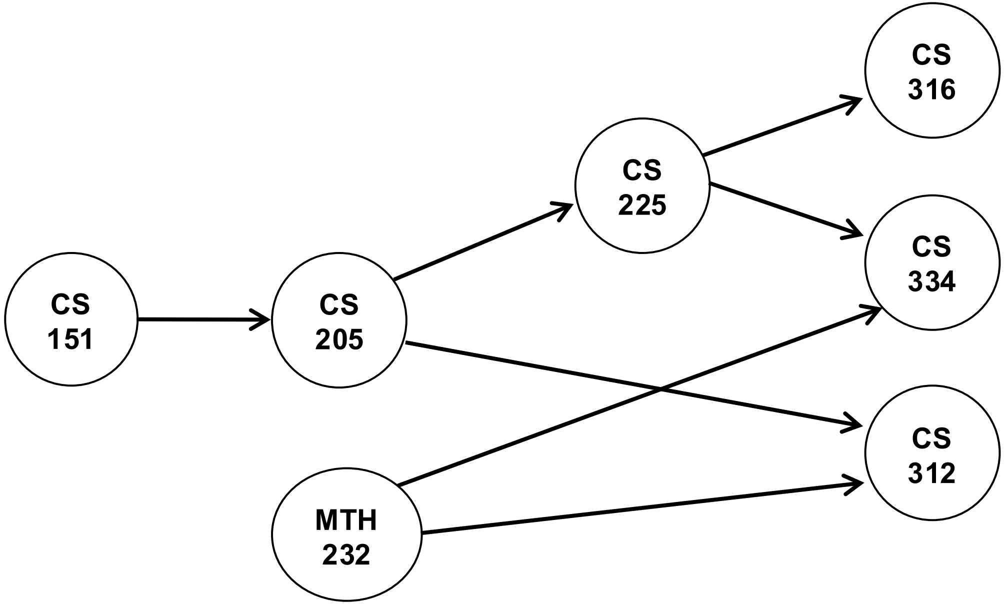
- BFS following edges in reverse direction
- Change "outgoing" to "incoming" in the Dir-BFS algo



Directed Acyclic Graphs

Def: a **directed acyclic graph** (DAG) is a directed graph with no cycles

❖ Models dependences, e.g., course prerequisites



Topological Sorting

Def: a **topological sorting** of a directed graph is an ordering of the vertices such that all edges go "forward" in the ordering

- e.g., a way to order the classes so that all prerequisites are satisfied
- Label vertices $v_1, \dots, v_n$ such that every edge (v_i, v_j) we have $i < j$

Topological Sorting

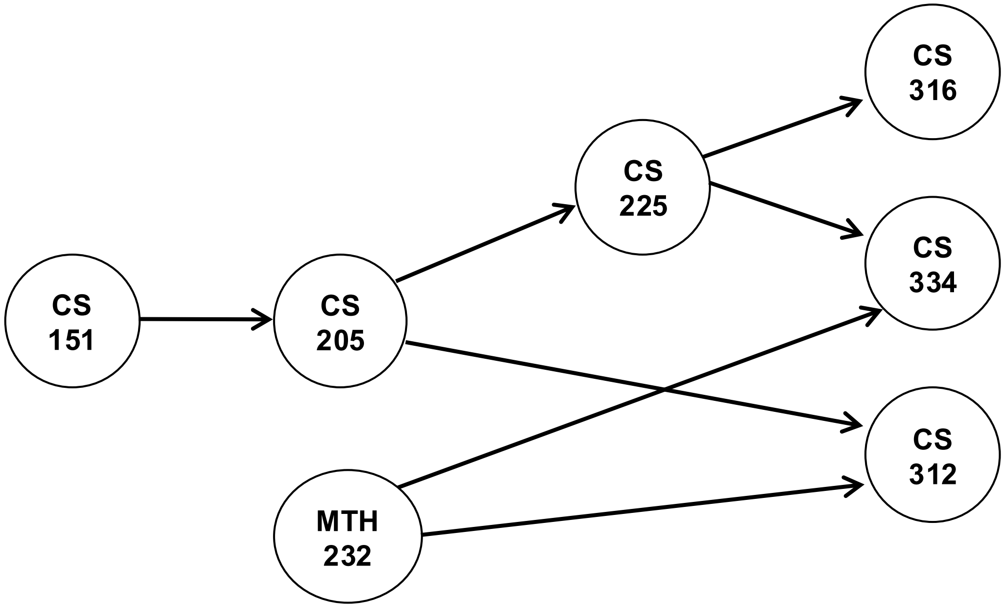
Def: a **topological sorting** of a directed graph is an ordering of the vertices such that all edges go "forward" in the ordering

- ❖ e.g., a way to order the classes so that all prerequisites are satisfied
- ❖ Label vertices $v_1, \ldots, v_n$ such that every edge (v_i, v_j) we have $i < j$

Q: Is a topological ordering possible for any directed graph?

Exercise II

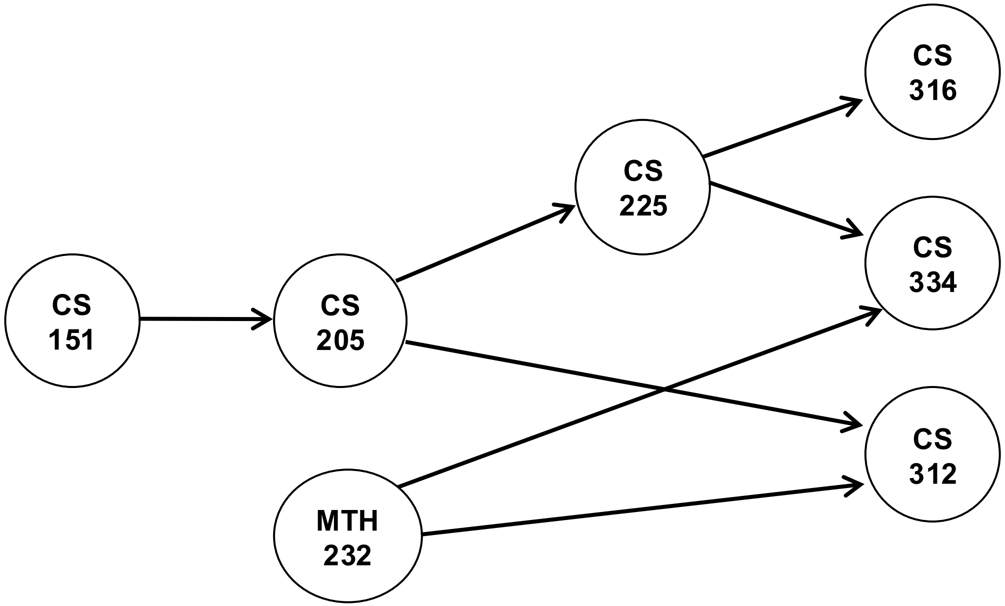
1. Find a topological ordering
2. Think about an algorithm to find a topological ordering in general



Exercise II

1. Find a topological ordering
2. Think about an algorithm to find a topological ordering in general

CS 151, CS 205, MTH 232, CS 225, CS 316, CS 334, CS 312



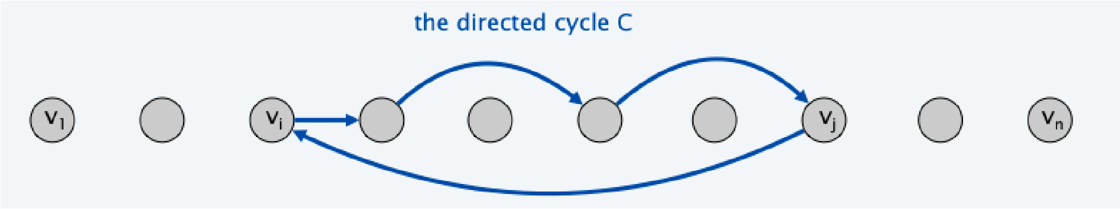
DAGs

Lemma: If G has a topological ordering, then G is a DAG.

DAGs

Lemma: If G has a topological ordering, then G is a DAG.

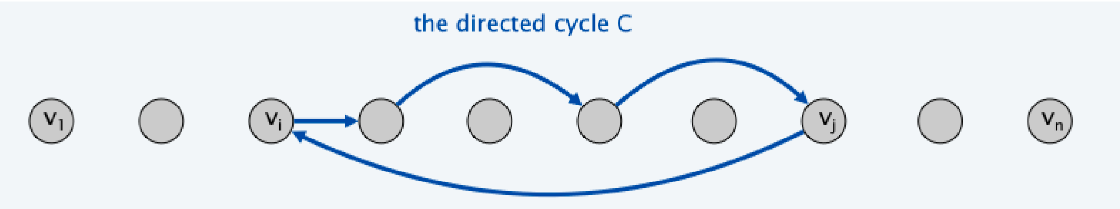
- Suppose G has a topological ordering $v_1, \ldots, v_n$
- For contradiction, suppose G also has a directed cycle C



DAGs

Lemma: If G has a topological ordering, then G is a DAG.

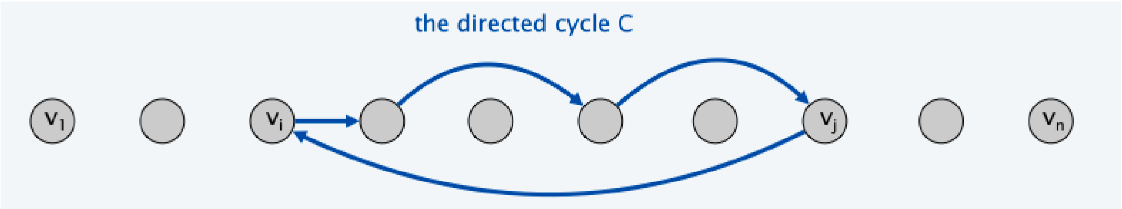
- Suppose G has a topological ordering $v_1, \dots, v_n$
- For contradiction, suppose G also has a directed cycle C
- Let v_i be the lowest-index vertex in C, and let v_j be the vertex just before v_i in C
- Then, (v_i, v_j) is an edge, and $i < j$



DAGs

Lemma: If G has a topological ordering, then G is a DAG.

- Suppose G has a topological ordering $v_1, \dots, v_n$
- For contradiction, suppose G also has a directed cycle C
- Let v_i be the lowest-index vertex in C, and let v_j be the vertex just before v_i in C
- Then, (v_i, v_j) is an edge, and $i < j$
- However, since G has a topological ordering and (v_i, v_j) is an edge, $j < i$ $(\rightarrow\leftarrow)$



Topo Sorting DAGs

Problem: Given DAG G, compute a topological ordering for G

topo-sort(G):
while there are vertices remaining **do**
find a vertex v with no incoming edges
place v next in the order
delete v and all of its outgoing edges from G

Topo Sorting DAGs

Problem: Given DAG G, compute a topological ordering for G

topo-sort(G):
  while there are vertices remaining **do**
    find a vertex v with no incoming edges
    place v next in the order
    delete v and all of its outgoing edges from G

Running time: $O(n + m)$

Topo Sorting DAGs

Theorem: If G is a DAG, topo-sort always finds a topological order

- In a DAG G, there is always a vertex v with no incoming edges (why?)
- Any such vertex v can be first in the topological ordering
- Removing v from G produces a new DAG G'
- Vertex v followed by a topo ordering for G' is a topo ordering for G

DAGs and Topological Orderings

Theorem: G is a DAG if and only if G has a topological ordering

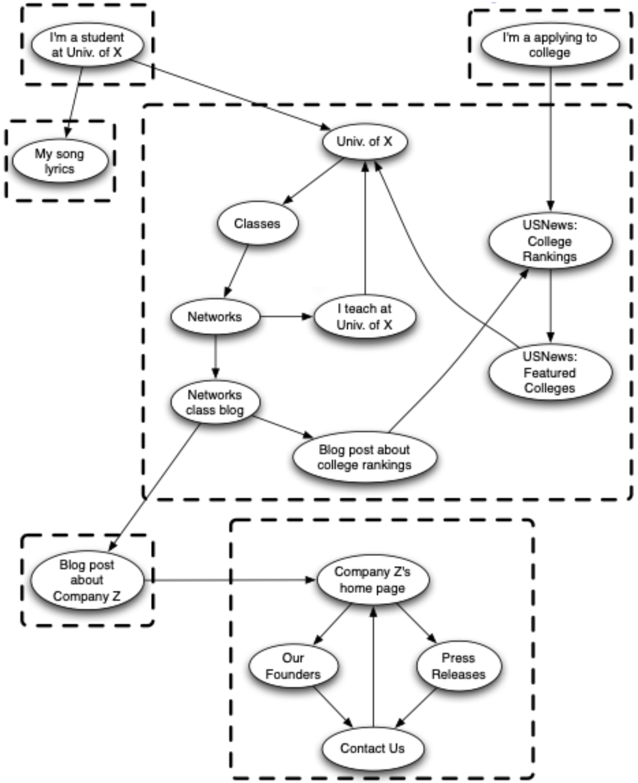
- ❖ We proved → direction by giving an algorithm (topo-sort) that always finds a topological ordering given a DAG
- ❖ We proved ← direction in a Lemma above, show that if a graph has a topological ordering, then it is a DAG

Directed Graph Connectivity

A **strongly connected graph** is a graph with a directed path between any pair of vertices

A **strongly connected component** (SCC) is a maximal subset of vertices with a directed path between any pair

❖ SCCs can be bound in time $O(n+m)$



Next Time

- Start greedy algorithms