Russell Impagliazzo and Miles Jones
Thanks to Janine Tiefenbruck

http://cseweb.ucsd.edu/classes/sp16/cse21-bd/
April 27, 2016
Today's plan

1. Definition of DAG.
2. Ordering algorithm on a DAG.
3. Graph search and reachability.

*In the textbook: Sections 10.4 and 10.5*
Prerequisites for CSE classes

2-year plan:

Take classes in some order.
If course A is prerequisite for course B, must take course A before we take course B.
Which of the following orderings are ok?

A. 30, 145, 151, 100.
B. 110, 105, 21, 101.
C. 21, 105, 130.
D. More than one of the above.
E. None of the above.
Prerequisites for CSE classes

What if we want to include all vertices (i.e. courses)？

Is this possible for any graph?
What if we want to include all vertices (i.e. courses)?

Is this possible for any graph?

1. Classify graphs for which it is.

2. For those, find a good ordering.
Which of the following graphs have good orderings?

A. 

B. 

C. 

D. 

E. None of the above.
Barriers to ordering

A can't be first (because B is before it).
B can't be first (because C is before it).
C can't be first (because D is before it).
D can't be first (because A is before it).

Whenever there is a cycle, can't find a "good" ordering.
Directed graphs with no cycles are called **directed acyclic graphs** (DAGs).
Directed graphs with no cycles are called **directed acyclic graphs** (DAGs).

A **topological ordering** of a graph is an (ordered) list of all its vertices such that, for each edge \((v,w)\) in the graph, \(v\) comes before \(w\) in the list.
Topological ordering

Two algorithmic questions:

1. Given an (ordered) list of all vertices in the graph, is it a topological ordering?
2. Given a graph, produce a topological ordering.
1. Given an (ordered) list of all vertices in the graph, is it a topological ordering?

How would you do it?
2. Given a graph, produce a topological ordering.

At what vertex should we start?

A. Any vertex is okay.
B. We must start at A.
C. Choose any vertex with at least one outgoing edge.
D. Choose any vertex with no incoming edges.
E. None of the above.
In a DAG, vertices with no incoming edges are called **sources**.

Which of these vertices are sources?

A. Only A and G.
B. Only A.
C. Only I.
D. Only I and F.
E. None of the above.
Sources of a DAG

Lemma 1: Every DAG has a (at least one) source

How would you prove this?
Lemma 1: Every DAG has a (at least one) source

*How would you prove this?*

Not a source … look at incoming edges
Sources of a DAG

Lemma 1: Every DAG has a (at least one) source

How would you prove this?

Not a source … look at incoming edges
Lemma 1: Every DAG has a (at least one) source

How would you prove this?
Sources of a DAG

Lemma 1: Every DAG has a (at least one) source

Let G be a DAG. We want to show that G has a source vertex.

In a proof by contradiction (aka indirect proof), what should we assume?

A. G has a source vertex.
B. All the vertices in G are sources.
C. No vertex in G is a source.
D. G has at least one source vertex and at least one vertex that's not a source.
E. None of the above.
Lemma 1: Every DAG has a (at least one) source

**Proof of Lemma 1:** Let $G$ be a DAG with $n$ ($n>1$) vertices. We want to show that $G$ has a source vertex.

**Assume towards a contradiction that no vertex in $G$ is a source.**

Let $v_0$ be a vertex in $G$. Since $v$ is not a source (by assumption), it has an incoming edge. Let $v_1$ be a vertex in $G$ so that $(v_1, v_0)$ is an edge in $G$. Since $v_1$ is also not a source, let $v_2$ be a vertex in $G$ so that $(v_2, v_1)$ is an edge in $G$. Keep going to find $v_0, v_1, v_2, \ldots, v_n$ vertices. There must be a **repeated** vertex in this list (Pigeonhole Principle). **Contradiction** with $G$ being acyclic.
Sources of a DAG

Notation: $G-v$ is the graph that results when remove $v$ and all of its incident edges from $G$.

Lemma 2: If $v$ is a source vertex of $G$, then $G$ is a DAG if and only if $G-v$ is a DAG.
Sources of a DAG

Notation: $G-v$ is the graph that results when remove $v$ and all of its incident edges from $G$.

Lemma 2: If $v$ is a source vertex of $G$, then $G$ is a DAG if and only if $G-v$ is a DAG.

Proof of Lemma 2: Let $G$ be a DAG and assume $v$ is a vertex in $G$.

Assume $G$ is a DAG. WTS $G-v$ is a DAG.

Assume $G-v$ is a DAG. WTS $G$ is a DAG.
Sources of a DAG

Notation: \( G-v \) is the graph that results when remove \( v \) and all of its incident edges from \( G \).

Lemma 2: If \( v \) is a source vertex of \( G \), then \( G \) is a DAG if and only if \( G-v \) is a DAG.

Proof of Lemma 2: Let \( G \) be a DAG and assume \( v \) is a vertex in \( G \).

Assume \( G \) is a DAG. WTS \( G-v \) is a DAG. Can't introduce any cycles by removing edges.

Assume \( G-v \) is a DAG. WTS \( G \) is a DAG.
Sources of a DAG

Notation: $G-v$ is the graph that results when remove $v$ and all of its incident edges from $G$.

Lemma 2: If $v$ is a source vertex of $G$, then $G$ is a DAG if and only if $G-v$ is a DAG.

Proof of Lemma 2: Let $G$ be a DAG and assume $v$ is a vertex in $G$.

Assume $G$ is a DAG. WTS $G-v$ is a DAG. Can't introduce any cycles by removing edges.

Assume $G-v$ is a DAG. WTS $G$ is a DAG. ?? Contrapositive …
Sources of a DAG

Notation: \( G-v \) is the graph that results when remove v and all of its incident edges from G.

Lemma 2: If v is a source vertex of G, then G is a DAG if and only if G-v is a DAG

Proof of Lemma 2: Let G be a DAG and assume v is a vertex in G.

Assume G is a DAG. WTS G-v is a DAG. Can't introduce any cycles by removing edges.

Assume G is not a DAG. WTS G-v is not a DAG. A cycle in G can't include a source (because no incoming edges). So this cycle will also be in G-v.
Find Topological Ordering (if possible)

While G has at least one vertex
   If G has some source,
       Choose one source and output it.
       Delete the source and all its outgoing edges from G.
   Else
       Return that G is not a DAG.
Find Topological Ordering (if possible)

While G has at least one vertex
  If G has some source,
    Choose one source and output it.
    Delete the source and all its outgoing edges from G.
  Else
    Return that G is not a DAG.

Implementation details:
Choose first x in S.
For each y adjacent to x,
  Decrement InDegree[y]
If InDegree[y]=0, add y to S.
Maintain integer array, InDegree[], of length n
Maintain collection of sources, S, as list, stack, or queue.
Example

InDegree[]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Collection of sources: \( S = A, G \)

Output:

Choose first \( x \) in \( S \).
For each \( y \) adjacent to \( x \),
  - Decrement \( \text{InDegree}[y] \)
  - If \( \text{InDegree}[y] = 0 \), add \( y \) to \( S \).
InDegree[]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Collection of sources: $S = G, B, C$

Output: A

Choose first $x$ in $S$.
For each $y$ adjacent to $x$,
  - Decrement InDegree[$y$]
  - If InDegree[$y$] = 0, add $y$ to $S$. 
InDegree[]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Collection of sources: S = B, C, E

Output: A, G

Choose first x in S.
For each y adjacent to x,
  Decrement InDegree[y]
  If InDegree[y]=0, add y to S.
InDegree[]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Collection of sources: \( S = C, E \)

Output: A, G, B

Choose first \( x \) in \( S \).
For each \( y \) adjacent to \( x \),
Decrement InDegree[\( y \)]
If InDegree[\( y \)] = 0, add \( y \) to \( S \).
Example

InDegree[]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Collection of sources: $S = E, D, H$

Output: A, G, B, C

Choose first $x$ in $S$.
For each $y$ adjacent to $x$,
Decrement InDegree[$y$]
If InDegree[$y$] = 0, add $y$ to $S$. 
InDegree[]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Collection of sources: $S = D, H$


Choose first $x$ in $S$.
For each $y$ adjacent to $x$,
  Decrement InDegree[$y$]
  If InDegree[$y$]=$0$, add $y$ to $S$. 
Example

InDegree[]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Collection of sources: \( S = H, F \)

Output: A, G, B, C, E, D

Choose first \( x \) in \( S \).
For each \( y \) adjacent to \( x \),
  \( \text{Decrement } \text{InDegree}[y] \)
  \( \text{If } \text{InDegree}[y]=0, \text{add } y \text{ to } S. \)
Example

InDegree[]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Collection of sources: $S = \{F, I\}$

Output: A, G, B, C, E, D, H

Choose first $x$ in $S$.
For each $y$ adjacent to $x$,
  Decrement $\text{InDegree}[y]$
  If $\text{InDegree}[y]=0$, add $y$ to $S$. 
Example

**InDegree[]**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Collection of sources: $S = \{I\}$

**Output:** A, G, B, C, E, D, H, F

Choose first $x$ in $S$.
For each $y$ adjacent to $x$,
  *Decrement InDegree[$y$]*
  *If InDegree[$y$] = 0, add $y$ to $S$.*
### Example

**InDegree[]**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Collection of sources: $S =$

**Output:** A, G, B, C, E, D, H, F, I

Choose first $x$ in $S$.
For each $y$ adjacent to $x$,
Decrement InDegree[$y$]
If InDegree[$y$] = 0, add $y$ to $S.$
Find Topological Ordering (if possible)

Make an array of indegree[v] for each v in |V|  
Initialize a queue of sources S

While G has at least one vertex
  If G has some source,  
    Choose one source and output it.  
    v := head(S), output v, eject(S,v)  
    Delete the source and all its outgoing edges from G.  
    for each (v,u) in E  
      decrement indegree(u)  
      if indegree(u)=0 then insert (S,u)
  Else
    Return that G is not a DAG.
Find Topological Ordering (if possible)

Make an array of indegree[v] for each v in |V|  \( O(|V|) \)
Initialize a queue of sources S  \( O(|V|) \)

While G has at least one vertex

If G has some source,  \( O(1) \)
   Choose one source and output it.  \( v:= \text{head}(S), \text{output } v, \text{eject}(S,v) \)  \( O(1) \)
   Delete the source and all its outgoing edges from G.
      for each \((v,u) \in E\)  \( O(\text{degree}(v)) \)
         decrement indegree(u)
         if indegree(u)=0 then insert (S,u)

Else

Return that G is not a DAG.

Since each v is ejected once, total time is

\[
\sum_{v \in V} \text{degree}(v) = |E|
\]
Prerequisites for (some) CSE classes

2-year plan:
Take classes in some order.
If course A is prerequisite for course B, must take course A before we take course B.

How many quarters?
Layers of a DAG

First layer  all nodes that are sources

Next layer  all nodes that are now sources
            (once we remove previous layer and its outgoing edges)

Repeat…
Prerequisites for CSE classes

How many quarters (layers) before take all classes?

A. 1.
B. 2.
C. 3.
D. 4.
E. More than four.