CSE 373

Graphs 4: Topological Sort
reading: Weiss Ch. 9

slides created by Marty Stepp
http://www.cs.washington.edu/373/

© University of Washington, all rights reserved.
Ordering a graph

• Suppose we have a directed acyclic graph (DAG) of courses, and we want to find an order in which the courses can be taken.
  ▪ Must take all prereqs before you can take a given course. Example:
    • [142, 143, 140, 154, 341, 374, 331, 403, 311, 332, 344, 312, 351, 333, 352, 373, 414, 410, 417, 413, 415]
  ▪ There might be more than one allowable ordering.
  ▪ How can we find a valid ordering of the vertices?
Topological Sort

- **topological sort**: Given a digraph $G = (V, E)$, a total ordering of $G$'s vertices such that for every edge $(v, w)$ in $E$, vertex $v$ precedes $w$ in the ordering. Examples:
  - determining the order to recalculate updated cells in a spreadsheet
  - finding an order to recompile files that have dependencies
  - (any problem of finding an order to perform tasks with dependencies)
Topo sort example

- How many valid topological sort orderings can you find for the vertices in the graph below?
  - [A, B, C, D, E, F], [A, B, C, D, F, E],
  - [A, B, D, C, E, F], [A, B, D, C, F, E],
  - [B, A, C, D, E, F], [B, A, C, D, F, E],
  - [B, A, D, C, E, F], [B, A, D, C, F, E],
  - [B, C, A, D, E, F], [B, C, A, D, F, E],
  - ... 

- What if there were a new vertex G unconnected to the others?
Topo sort: Algorithm 1

- function topologicalSort():
  - ordering := { }.
  - Repeat until graph is empty:
    - Find a vertex \( v \) with in-degree of 0 (no incoming edges).
      - (If there is no such vertex, the graph cannot be sorted; stop.)
    - Delete \( v \) and all of its outgoing edges from the graph.
    - ordering += \( v \).

![Graph Diagram]
Topo sort example

- function topologicalSort():
  - ordering := \{ \}\.
  - Repeat until graph is empty:
    - Find a vertex \(v\) with in-degree of 0 (no incoming edges).
      - (If there is no such vertex, the graph cannot be sorted; stop.)
    - Delete \(v\) and all of its outgoing edges from the graph.
    - ordering += \(v\).
  - ordering = \{ B \}
Topo sort example

- function topologicalSort():
  - ordering := \{ \}
  - Repeat until graph is empty:
    - Find a vertex \( v \) with in-degree of 0 (no incoming edges).
      - (If there is no such vertex, the graph cannot be sorted; stop.)
    - Delete \( v \) and all of its outgoing edges from the graph.
    - ordering += \( v \).
  - ordering = \{ B, C \}
Topo sort example

• function topologicalSort():
  ▪ ordering := { }.
  ▪ Repeat until graph is empty:
    ▪ Find a vertex v with in-degree of 0 (no incoming edges).
      ▪ (If there is no such vertex, the graph cannot be sorted; stop.)
    ▪ Delete v and all of its outgoing edges from the graph.
    ▪ ordering += v .
  ▪ ordering = { B, C, A }
Topo sort example

- function topologicalSort():
  - \textit{ordering} := \{ \}. \\
  - Repeat until graph is empty:
    - Find a vertex \( v \) with in-degree of 0 (no incoming edges).
      - (If there is no such vertex, the graph cannot be sorted; stop.)
    - Delete \( v \) and all of its outgoing edges from the graph.
    - \textit{ordering} += \( v \).
  - \textit{ordering} = \{ B, C, A, D \}
Topo sort example

- function topologicalSort():
  - ordering := { }.
  - Repeat until graph is empty:
    - Find a vertex \( v \) with in-degree of 0 (no incoming edges).
      - (If there is no such vertex, the graph cannot be sorted; stop.)
    - Delete \( v \) and all of its outgoing edges from the graph.
    - ordering += \( v \).
  - ordering = { B, C, A, D, F }
Topo sort example

- function topologicalSort():
  - ordering := \{ \}.
  - Repeat until graph is empty:
    - Find a vertex v with in-degree of 0 (no incoming edges).
      - (If there is no such vertex, the graph cannot be sorted; stop.)
    - Delete v and all of its outgoing edges from the graph.
    - ordering += v.
  - ordering = \{ B, C, A, D, F, E \}
Revised algorithm

- We don't want to literally delete vertices and edges from the graph while trying to topological sort it; so let's revise the algorithm:

  - $map := \{\text{each vertex } \rightarrow \text{ its in-degree}\}$.
  - $queue := \{\text{all vertices with in-degree} = 0\}$.
  - $ordering := \{\}$.  
  - Repeat until queue is empty:
    - Dequeue the first vertex $v$ from the queue.
    - $ordering += v$.
    - Decrease the in-degree of all $v$'s neighbors by 1 in the $map$.
    - $queue += \{\text{any neighbors whose in-degree is now 0}\}$.
  - If all vertices are processed, success. Otherwise, there is a cycle.
Topo sort example 2

- function topologicalSort():
  - \textit{map} := \{ each vertex $\rightarrow$ its in-degree \}.
  - \textit{queue} := \{ all vertices with in-degree = 0 \}.
  - \textit{ordering} := \{ \}.
  - Repeat until queue is empty:
    - Dequeue the first vertex \( v \) from the queue.
    - \textit{ordering} += \( v \).
    - Decrease the in-degree of all \( v \)'s neighbors by 1 in the \textit{map}.
    - \textit{queue} += \{ any neighbors whose in-degree is now 0 \}.

- map := \{ A=0, B=0, C=1, D=2, E=2, F=2 \}
- queue := \{ B, A \}
- ordering := \{ \}
Topo sort example 2

- function topologicalSort():
  - map := {each vertex → its in-degree}.
  - queue := {all vertices with in-degree = 0}.
  - ordering := { }.
  - Repeat until queue is empty:
    - Dequeue the first vertex v from the queue. // B
    - ordering += v.
    - Decrease the in-degree of all v's // C, D
      neighbors by 1 in the map.
    - queue += {any neighbors whose in-degree is now 0}.

- map := { A=0, B=0, C=0, D=1, E=2, F=2 }
- queue := { A, C }
- ordering := { B }
Topo sort example 2

- function topologicalSort():
  - map := {each vertex → its in-degree}.
  - queue := {all vertices with in-degree = 0}.
  - ordering := { }.
  - Repeat until queue is empty:
    - Dequeue the first vertex \( v \) from the queue.  // A
    - ordering += \( v \).
    - Decrease the in-degree of all \( v \)'s  // D
      neighbors by 1 in the map.
    - queue += {any neighbors whose in-degree is now 0}.

- map := \{ A=0, B=0, C=0, D=0, E=2, F=2 \}
- queue := \{ C, D \}
- ordering := \{ B, A \}
Topo sort example 2

- function topologicalSort():
  - map := {each vertex \(\rightarrow\) its in-degree}.
  - queue := {all vertices with in-degree = 0}.
  - ordering := { }.
  - Repeat until queue is empty:
    - Dequeue the first vertex \(v\) from the queue. // C
    - ordering += \(v\).
    - Decrease the in-degree of all \(v\)'s // E, F neighbors by 1 in the map.
    - queue += {any neighbors whose in-degree is now 0}.

- map := \{ A=0, B=0, C=0, D=0, E=1, F=1 \}
- queue := \{ D \}
- ordering := \{ B, A, C \}
Topo sort example 2

- function topologicalSort():
  - map := {each vertex → its in-degree}.
  - queue := {all vertices with in-degree = 0}.
  - ordering := { }.
- Repeat until queue is empty:
  - Dequeue the first vertex v from the queue.  // D
  - ordering += v.
  - Decrease the in-degree of all v's  // F, E neighbors by 1 in the map.
  - queue += {any neighbors whose in-degree is now 0}.

- map := { A=0, B=0, C=0, D=0, E=0, F=0 }
- queue := { F, E }
- ordering := { B, A, C, D }
Topo sort example 2

- function topologicalSort():
  - map := {each vertex → its in-degree}.
  - queue := {all vertices with in-degree = 0}.
  - ordering := { }.
  - Repeat until queue is empty:
    - Dequeue the first vertex $v$ from the queue.  // F
    - ordering += $v$.
    - Decrease the in-degree of all $v$'s // none neighbors by 1 in the map.
    - queue += {any neighbors whose in-degree is now 0}.

- map := { A=0, B=0, C=0, D=0, E=0, F=0 }
- queue := { E }
- ordering := { B, A, C, D, F }
Topo sort example 2

- function topologicalSort():
  - $map := \{\text{each vertex } \rightarrow \text{ its in-degree}\}$. 
  - $queue := \{\text{all vertices with in-degree } = 0\}$. 
  - $ordering := \{\}$.
  - Repeat until queue is empty:
    - Dequeue the first vertex $v$ from the queue.  // $E$
    - $ordering += v$.
    - Decrease the in-degree of all $v$'s  // none neighbors by 1 in the $map$.
    - $queue += \{\text{any neighbors whose in-degree is now } 0\}$.

- $map := \{A=0, B=0, C=0, D=0, E=0, F=0\}$
- $queue := \{\}$
- $ordering := \{B, A, C, D, F, E\}$
Topo sort runtime

- What is the runtime of our topological sort algorithm?
  - (with an "adjacency map" graph internal representation)

  - function topologicalSort():
    - map := {each vertex → its in-degree}. \(\text{\# O(V)}\)
    - queue := {all vertices with in-degree = 0}.
    - ordering := { }.
    - Repeat until queue is empty: \(\text{\# O(V)}\)
      - Dequeue the first vertex \(v\) from the queue. \(\text{\# O(1)}\)
      - ordering += v. \(\text{\# O(1)}\)
      - Decrease the in-degree of all \(v\)'s neighbors by 1 in the \text{map}. \(\text{\# O(E) for all passes}\)
      - queue += {any neighbors whose in-degree is now 0}.

- Overall: \(\text{O}(V + E)\); essentially \(O(V)\) time on a sparse graph (fast!)