Topological Sort of a Graph

CSE 373 - Data Structures
May 24, 2002

Readings and References

- Reading
  - Section 9.2, *Data Structures and Algorithm Analysis in C*, Weiss

- Other References

Some slides based on: CSE 326 by S. Wolfman, 2000

Topological Sort

Problem: Find an order in which all these courses can be taken.

Example: 142 → 143 → 378
       → 370 → 321 → 341 → 322
       → 326 → 421 → 401

In order to take a course, you must take all of its prerequisites first.

Given a digraph $G = (V, E)$, find a linear ordering of its vertices such that:

for any edge $(v, w)$ in $E$, $v$ precedes $w$ in the ordering.
Any linear ordering in which all the arrows go to the right is a valid solution.

Note that F can go anywhere in this list because it is not connected.

Any linear ordering in which an arrow goes to the left is not a valid solution.

**Topo sort - good example**

**Topo sort - bad example**

**Topo sort algorithm - 1**

**Step 1:** Identify vertices that have no incoming edges
- The “in-degree” of these vertices is zero

**Topo sort algorithm - 1a**

**Step 1:** Identify vertices that have no incoming edges
- If no such vertices, graph has cycle(s) (cyclic graph)
- Topological sort not possible – Halt.
Topo sort algorithm - 1b

**Step 1**: Identify vertices that have no incoming edges
- Select one such vertex

Select

A → B → C → D → E

Topo sort algorithm - 2

**Step 2**: Delete this vertex of in-degree 0 and all its outgoing edges from the graph. Place it in the output.

Select B. Copy to sorted list. Delete B and its edges.
Select C. Copy to sorted list. Delete C and its edges.

Select D. Copy to sorted list. Delete D and its edges.

Select E. Copy to sorted list. Delete E and its edges. Select F. Copy to sorted list. Delete F and its edges.

E, F

Done

Remove from algorithm and serve.
For input graph $G = (V,E)$, Run Time $= O(?)$

**Break down into total time to:**
- Find a vertex with in-degree 0
- Remove its edges
- Place vertex in output

Assume adjacency list representation

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

**Tracking “in-degree”**

- Calculate and store In-Degree of all vertices in an array
  - Find vertex with in-degree 0: Search the array
  - Remove its edges: Update the array

**Topo Sort run time analysis**

- Find vertices with in-degree 0:
  - $|V|$ vertices, and for each vertex it takes $O(|V|)$ to search the In-Degree array $= O(|V|^2)$
- Remove edges:
  - $|E|$ edges
- Place vertices in output:
  - $|V|$ vertices

- For input graph $G = (V,E)$
  - Run Time $= O(|V|^2 + |E|)$
  - Quadratic in $|V|$

**Key idea:** Initialize and maintain a queue (or stack) of vertices with In-Degree 0
Topo Sort with queue

After each vertex is output, when updating In-Degree array, {
\textit{enqueue any vertex whose In-Degree has become zero}}

**Topological Sort Algorithm #2**

- Store each vertex’s In-Degree in an array
- Initialize queue with all “in-degree=0” vertices
- While there are vertices remaining in the queue:
  - Dequeue and output a vertex
  - Reduce In-Degree of all vertices adjacent to it by 1
  - Enqueue any of these vertices whose In-Degree became zero

Topo Sort run time

- Initialize In-Degree array: \(O(|E|)\)
- Initialize Queue with In-Degree 0 vertices: \(O(|V|)\)
- Dequeue and output vertex:
  - \(|V|\) vertices, each takes only \(O(1)\) to dequeue and output: \(O(|V|)\)
- Reduce In-Degree of all vertices adjacent to a vertex and Enqueue any In-Degree 0 vertices:
  - \(O(|E|)\)
- For input graph \(G=(V,E)\) run time = \(O(|V| + |E|)\)
  - Linear in \(|V|\)