CSE 101, Winter 2018

Design and Analysis of Algorithms

Lecture 3: Connected Components

Class URL: http://vlsicad.ucsd.edu/courses/cse101-w18/
Last Time: DFS in Directed Graphs

- Same algorithm as in undirected graphs!

- Back edge reveals existence of directed cycle

- Cross edges lead to neither descendant nor ancestor → must lead to a node that has already been completely explored

- Cannot have $\text{pre}(u) < \text{post}(u) < \text{pre}(v) < \text{post}(v)$ in a $(u,v)$ cross edge

### Edge Types by Pre/Post Numbers

<table>
<thead>
<tr>
<th>Edge Type</th>
<th>Pre/Post Properties of $(u,v)$</th>
<th>In Search Tree?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>$\text{pre}(u) &lt; \text{pre}(v) &lt; \text{post}(v) &lt; \text{post}(u)$</td>
<td>Yes</td>
</tr>
<tr>
<td>Forward</td>
<td>$\text{pre}(u) &lt; \text{pre}(v) &lt; \text{post}(v) &lt; \text{post}(u)$</td>
<td>No</td>
</tr>
<tr>
<td>Back</td>
<td>$\text{pre}(v) &lt; \text{pre}(u) &lt; \text{post}(u) &lt; \text{post}(v)$</td>
<td>No</td>
</tr>
<tr>
<td>Cross</td>
<td>$\text{pre}(v) &lt; \text{post}(v) &lt; \text{pre}(u) &lt; \text{post}(u)$</td>
<td>No</td>
</tr>
</tbody>
</table>

Every edge must fall into one of these groups
**DAGs**

Acyclic = no directed cycles

- vertices = tasks, (directed) events
- edges = causality, dependency,

Source: no incoming edges / no predecessors
Sink: no outgoing / no successors

* DAG must have at least one source (P,P argument) (and, at least one sink)
Directed Acyclic Graph
- E.g., Vertices = tasks; Directed Edges = dependencies
- Acyclic: no solution if there’s a cycle of dependencies

Terminology: “sources” and “sinks”
- The vertex with largest post number has no incoming edges (we call this vertex a “source”)
- vertex with smallest post number has no outgoing edges (we call this vertex a “sink”)

Fact: In a DAG, every edge leads to a vertex with lower post number
- From last time, edge (u,v) with post(v) > post(u) is a back edge. But a DAG has no cycles ⇔ no back edges
Topological Ordering of a DAG

• **TOP_ORDER PROBLEM:** Given a DAG $G = (V,E)$ with $|V|=n$, assign labels $1,\ldots,n$ to $v_i \in V$ such that every edge is from a lower label to a higher label.

  - “Inductive” thinking
    - Any DAG always has a vertex with indegree $= 0$ (“source”)
    - Give this vertex the next label, delete vertex and its edges,… keep doing this to get a topological ordering

• **DFS-based algorithm:** Run DFS, then perform tasks (label vertices) in decreasing order of *post* numbers
  - *Because in a DAG, every edge leads to a vertex with lower post number*
Connectivity and Connected Components

Undirected Case:
Connected components found by DFS
What is Connectivity in a Directed Graph?

- Is this *digraph* (= another term for “directed graph”) connected? (?!)
  - E.g., can’t reach A from L
  “Only one sensible definition”
- Definition: u is connected to w iff there is a path from u to w and a path from w to u “mutually reachable from” relation
- With this definition, we partition V into strongly connected components

\( \text{equiv relation} \)

\( \sqrt{B \rightarrow B} \) (reflexivity)
\( \sqrt{B \rightarrow E, E \rightarrow B} \) (symmetry)
\( \sqrt{H \rightarrow K, K \rightarrow L} \Rightarrow H \rightarrow L \) (transitivity)

SCC’s are “equivalence classes” under “mutually reachable from” relation in digraphs
Strongly Connected Components

- Is this digraph (= another term for "directed graph") connected?
  - E.g., can’t reach A from L
- “Only one sensible definition”
- Definition: $u$ is connected to $w$ iff there is a path from $u$ to $w$ and a path from $w$ to $u$
- With this definition, we partition $V$ into strongly connected components
  - Example shown (Figure 3.9(a)):
    5 SCC’s

Is $\{G, I, J\}$ an SCC? (no; not maximal)
What Is Connectivity Between SCC’s?

• “Meta-graph”
• Shrink each SCC to a “meta-node”
• Put a directed edge from one meta-node to another if there is an edge in that direction between any of their respective vertices

The Meta-Graph
A Digraph Has a DAG Over Its SCC’s

- “Meta-graph”
- Shrink each SCC to a “meta-node”
- Put a directed edge from one meta-node to another if there is an edge in that direction between any of their respective vertices

The meta-graph is a DAG – WHY?
- Every directed graph is the DAG of its strongly connected components

We have a two-tiered connectivity structure of directed graph:
- **Coarse** = Top-level DAG
- **Fine** = Inside a meta-node / SCC
Finding SCC’s in Directed Graphs

- **Goal**: An algorithm for finding SCC’s in directed graphs.
- **Fact 1**: If the explore subroutine is started at node u, it will terminate when all nodes reachable from u have been visited.
- **Consequence**: If we start in a sink SCC, then we will identify precisely that SCC.

**IDEA**:
- (1) Find a node that is guaranteed to be in a sink SCC
- (2) After identifying a sink SCC, somehow continue (e.g., with explore(u))
What Node is Guaranteed to Be in a Sink SCC?

- **Fact 2:** Run DFS on $G$. The node with highest post number lies in a source SCC.
- **Why?**
- **Fact 3:** If $C$, $C'$ are SCC’s and there is an edge from $C$ to $C'$, then the highest post number in $C$ is larger than the highest post number in $C'$.

- **Fact 3 implies Fact 2**
What Node is Guaranteed to Be in a Sink SCC?

- **Fact 2:** Run DFS on $G$. The node with highest post number lies in a source SCC.

- **Why?**

- **Fact 3:** If $C$, $C'$ are SCC’s and there is an edge from $C$ to $C'$, then the highest post number in $C$ is larger than the highest post number in $C'$.

- **Case 1:** DFS sees $C$ before $C'$. $\rightarrow$ The first node visited in $C$ will have higher post number than any node in $C'$. 

![Diagram showing DFS traversal and SCCs](image-url)
What Node is Guaranteed to Be in a Sink SCC?

- **Fact 2:** Run DFS on G. The node with highest post number lies in a source SCC.

- **Why?**

- **Fact 3:** If C, C’ are SCC’s and there is an edge from C to C’, then the highest post number in C is larger than the highest post number in C’.

- **Case 2:** DFS sees C’ before C. → DFS will get stuck after seeing all of C’ but before seeing any node of C.
Topological Ordering of SCC’s

• Fact 2: Run DFS on G. The node with highest post number lies in a source SCC.

• Why?

• Fact 3: If C, C’ are SCC’s and there is an edge from C to C’, then the highest post number in C is larger than the highest post number in C’.

→ SCC’s can be topologically sorted by arranging them in decreasing order of their highest post numbers.

= Generalization of earlier algorithm for topological ordering. (In DAG, each SCC is a singleton node.)
Back To Finding SCC’s

- (Recall) IDEA:
  - (1) Find a node that is guaranteed to be in a sink SCC
  - (2) After identifying a sink SCC, somehow continue

- Definition: reverse graph \( G^R \) = same as \( G \), with edges reversed

- Source SCC in \( G^R \) = sink SCC in \( G \)
- \( \rightarrow \) Do DFS on \( G^R \), pick node with highest post number
- This node is in a sink SCC of \( G \)
Back To Finding SCC’s

• (Recall) IDEA:
  – (1) Find a node that is guaranteed to be in a sink SCC
  – (2) After identifying a sink SCC, somehow continue

• Identify sink SCC, delete it from graph
• Of remaining nodes, the one with highest post number will be in a sink SCC of whatever is left of G
Algorithm for Finding SCC’s in Digraph

- (Recall) IDEA:
  - (1) Find a node that is guaranteed to be in a sink SCC
  - (2) After identifying a sink SCC, somehow continue

### Algorithm

Run DFS on $G^R$

for $v \in V$ in decreasing order of post numbers

  if not visited[$v$]

    explore($G, v$)

  output nodes seen as a SCC
Meta-graph: 2 source SCC’s, 1 sink SCC
SCC Example

Construct reverse graph $G^R$
Run DFS on $G^R$ (lexicographic tie-breaking)
Run DFS on G in order of decreasing POST numbers from $G^R$
Run DFS on G in order of decreasing POST numbers from $G^R$
Run DFS on G in order of decreasing POST numbers from $G^R$
Run DFS on $G$ in order of decreasing POST numbers from $G^R$

Exercise: What happens when you run this algorithm on a DAG?
Examples 1

- We leveraged “highest post number is in a source SCC”.
  \[ \Rightarrow G^R, \text{ etc.} \]

- Could we have instead leveraged “lowest post number is in a sink SCC”?
Examples 2

• Find SCC’s in this directed graph G.
  (Do this with different vertex labels
   ⇒ different tie-breaking in DFS)
Examples 3

- Practice finding tree, forward, back, cross edges in the DFS forest…
  (Do this with different vertex labels ⇒ different tie-breaking in DFS)