Homework 12

Bryce Tolman

2024-03-10

Contents

Graphs Continued

- Forward edges: go down the tree to descendant (not child)
- Back edges: back up the tree to ancesotr (not parent)
- Cross edges: to other node not ancestor or descendant

pre/post ordering of (u,v)	Edge Type
u[v[ ]v]u	Tree/Forward
v[u[ ]u]v	Back
u[ ]u v[ ]v	Cross

- If you have a back edge, there is a cycle in the graph

DAGs

- There are things called DAGs, these are directed acyclic graphs.
- We can linearize these. We just say that any node that has things going into it has to have visited those incoming nodes first
- There can be multiple linearizations of the same DAG
- How do we linearize it?
  - Get post order numbers
  - The one with the highest post order number (not mattering who you choose to look at in what order) is a source
    - Though you can have multiple sources

  - We linearize as we do the DFS to get the post numbers
    * You just add the node to the end of the list as you pop it off the stack of the DFS
  - You will get one of the possible linearizations no matter what (it will vary as you choose the start node and order)
  - This is complexity E becase we have to visit every edge and node (but E is usually bigger so we call it O(E))
- Sinks are nodes that have no outgoing edges
- Sources are nodes that have no incoming edges
- Strongly Connected Graphs are directed graphs in which any node can get to any other node
- Weakly Connected Graphs are directed graphs that would be strongly connected if they were undirected (there are no lone nodes)
- DAGs cannot be strongly connected (there are no cycles)
- Strongly Connected Components are sub parts of a graph in which there are strongly connected graphs, but the whole graph is not strongly connected
- There is a link between strongly connected components (usually)
- How would we make the whole thing strongly connected?
  - Connect each sink to a source (a whole component can be a sink or source)
- How do we find the strongly connected components?
  - Well, every directed graph is a DAG of its strongly connected components
  - Pretty much, you want to grab the sink over and over again and get the connected component each time and getting rid of the component from the graph each time as you discover it all
  - How do we detect if the node is a sink?
    * The sink node has the lowest post, for DAGs, not for just any directed graph...
    * For any graph, the source has to have the highest post order number
    * So, we reverse the graph, and find the source of that, which is the sink of the actual graph!

Breadth First Search (BFS)

- Helps us find the shortest path between two nodes

```
graph TD;
    A-->B;
    A-->C;
    B-->D;
    C-->D;
```