Graphs and Connectivity

\[ G = (V, E) \quad w_e = \text{weight of edge } e. \]

undirected, directed
weighted, unweighted

Adjacency Matrix
Adjacency List
def explore(G, u):
    previsit(u)
    visited[u] = True
    for (u, v) in E:
        if not visited[v]:
            explore(G, v)
    postvisit(u)

def dfs(G):
    for u in V:
        visited[u] = False
        pre[u] = -1
        post[u] = -1
        order = 1
        cc = 1
        for u in V:
            if not visited[u]:
                explore(G, u)
                cc += 1

dfs ensures that every vertex will have a chance to be explored.

Runtime of dfs:
The for loops in dfs are $O(|V|)$. The for loop in explore will cause each edge to be examined twice by the time dfs has completed. $\Rightarrow O(2|E|)$

$\Rightarrow$ dfs is $O(|V|+|E|)$. 

explore visits every vertex reachable from u in G, once.
def previsit(u):
    ccnum[u] = cc
    pre[u] = order
    order += 1

def postvisit(u):
    post[u] = order
    order += 1

Accounting to track
pre/post order numbers,
and connected components.
Run dfs on this graph. What are the pre/post numbers? What type are the edges?
Graphs and Connectivity

Draw DFS Tree Here

Directed Graph DFS

<table>
<thead>
<tr>
<th>Vertex</th>
<th>pre</th>
<th>post</th>
<th>ccnum</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Edge</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td></td>
</tr>
<tr>
<td>AF</td>
<td></td>
</tr>
<tr>
<td>BE</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
</tr>
<tr>
<td>DA</td>
<td></td>
</tr>
<tr>
<td>DH</td>
<td></td>
</tr>
<tr>
<td>EF</td>
<td></td>
</tr>
<tr>
<td>EG</td>
<td></td>
</tr>
<tr>
<td>EH</td>
<td></td>
</tr>
<tr>
<td>FG</td>
<td></td>
</tr>
<tr>
<td>HG</td>
<td></td>
</tr>
</tbody>
</table>
Graphs and Connectivity

Directed Graph DFS

<table>
<thead>
<tr>
<th>Vertex</th>
<th>pre</th>
<th>post</th>
<th>ccnum</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>13</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>4</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>8</td>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

Draw DFS Tree Here

<table>
<thead>
<tr>
<th>Edge</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>Tree</td>
</tr>
<tr>
<td>AC</td>
<td>Tree</td>
</tr>
<tr>
<td>AF</td>
<td>Forward</td>
</tr>
<tr>
<td>BE</td>
<td>Tree</td>
</tr>
<tr>
<td>CD</td>
<td>Tree</td>
</tr>
<tr>
<td>DA</td>
<td>Back</td>
</tr>
<tr>
<td>DH</td>
<td>Cross</td>
</tr>
<tr>
<td>EF</td>
<td>Tree</td>
</tr>
<tr>
<td>EG</td>
<td>Forward</td>
</tr>
<tr>
<td>EH</td>
<td>Tree</td>
</tr>
<tr>
<td>FG</td>
<td>Tree</td>
</tr>
<tr>
<td>HG</td>
<td>Cross</td>
</tr>
</tbody>
</table>
Edge \((u, v)\) goes from vertex \(u\) to vertex \(v\). Given: \(\text{pre}[u] < \text{post}[u]\) and \(\text{pre}[v] < \text{post}[v]\).

What possible orderings of \(\text{pre}[u], \text{post}[u], \text{pre}[v], \text{post}[v]\) exist?

\[
\begin{align*}
\text{pre}[u] &< \text{post}[u] < \text{pre}[v] < \text{post}[v] \quad &\text{(1)} \\
\text{pre}[u] &< \text{pre}[v] < \text{post}[u] < \text{post}[v] \quad &\text{(2)} \\
\text{pre}[u] &< \text{pre}[v] < \text{post}[u] < \text{post}[v] \quad &\text{(3)} \\
\text{pre}[v] &< \text{post}[u] < \text{pre}[u] < \text{post}[v] \quad &\text{(4)} \\
\text{pre}[v] &< \text{pre}[u] < \text{post}[u] < \text{post}[v] \quad &\text{(5)} \\
\text{pre}[v] &< \text{pre}[u] < \text{post}[u] < \text{post}[v] \quad &\text{(6)}
\end{align*}
\]

What do each mean about the type of the edge \((u, v)\) in the DFS tree?
Edge \( (u,v) \)

1. \[
\begin{array}{c}
\udots \\
\hspace{1cm} u & u & v & v
\end{array}
\]
   \[\text{if } u < v \text{ and edge } (u,v) \text{ exists, the search would explore } (v) \text{ before postvisit}(u). \text{ This permutation is not possible.}\]

2. \[
\begin{array}{c}
\udots \\
\hspace{1cm} u & v & v & u
\end{array}
\]
   \[\text{vertex } v \text{ is explored after } u, \text{ and is postvisited before } u. \text{ } v \text{ is a descendant of } u. \text{ This is either a tree or a forward edge.}\]

3. \[
\begin{array}{c}
\udots \\
\hspace{1cm} u & v & u & v
\end{array}
\]
   \[\text{It is not possible for postvisit to be called on } u \text{ before } v, \text{ if previsit is called on } u \text{ first. This permutation is not possible.}\]

4. \[
\begin{array}{c}
\udots \\
\hspace{1cm} v & v & u & u
\end{array}
\]
   \[\text{explore}(v) \text{ completed before explore}(u) \text{ is called. } u \text{ is not reachable from } v. \text{ The edge } (u,v) \text{ would cross branches in the tree. Cross edge.}\]

5. \[
\begin{array}{c}
\udots \\
\hspace{1cm} v & u & u & v
\end{array}
\]
   \[u \text{ is a descendant of } v. \text{ (See 2). There is and edge that points back up the tree from } u + v. \text{ Back edge.}\]

6. \[
\begin{array}{c}
\udots \\
\hspace{1cm} v & u & v & u
\end{array}
\]
   \[\text{See 3. Not possible.}\]
A cycle is a path in a graph that returns to the same node: A circular path.

Acyclic graphs have no cycles.

How can you tell if a graph is acyclic?

- If there is a back edge, there is a cycle.
- If there is a cycle, there is a back edge.
- If there is no back edge, there is no cycle!

- A DAG has a cycle iff its DFS tree has a back edge.
- DAG edges lead to a vertex with a lower post-number.
- Every DAG has at least 1 source and at least 1 sink.

Linearization: order DAG vertices so prerequisites will be completed first.

Algorithms?