**Directed DFS: example**

```
procedure explore(G,v)
    visited[v] = true
    previsit(v)
    for each edge (v,u) in E:
        if not visited[u]:
            explore(G,u)
    postvisit(v)

procedure dfs(G)
    for all v in V:
        visited[v] = false
    for all v in V:
        if not visited[v]:
            explore(G,v)

procedure previsit(v)
    pre[v] = clock++

procedure postvisit(v)
    post[v] = clock++
```

**Directed DFS: terminology**

- **Four types of edges**
  - Tree edge: part of DFS forest
  - Back edge: leads to an ancestor
  - Forward edge: leads to non-child descendant
  - Cross edge: leads to neither descendant nor ancestor

**The pre/post signature of ancestors**

<table>
<thead>
<tr>
<th>Node</th>
<th>pre</th>
<th>post</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>c</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>d</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>e</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>f</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>g</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>h</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

Node u is an ancestor of node v if and only if:

- pre[u] < pre[v] < post[v] < post[u] (for forward edges)
- pre[v] < pre[u] < post[u] < post[v] (for back edges)
- pre[v] < post[v] < pre[u] < post[u] (for cross edges)

**Cycles**

A cycle in a directed graph is a circular path $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow v_1$.

- **Claim:** A directed graph G has a cycle if and only if DFS encounters a back edge.

**Directed acyclic graphs (dags)**

**Claim:** In a dag, every edge leads to a lower post number.

**Topological ordering**

**Input:** a dag

**Goal:** give each node a number so that every edge leads from a lower number to a higher number (i.e. precedence constraints satisfied).

**Solution:**
- Run DFS and perform tasks in order of decreasing POST numbers.

**Graph without cycles: acyclic**

How to tell if a directed graph is acyclic?

Linear time algorithm to check acyclicity!
**DAGs, cont’d**

A source is a node with no in-edges.
A sink is a node with no out-edges.

**Claim**
In a dag, the node with highest post number is a source and lowest post number is a sink.

Another algorithm for topological ordering:
- Find a source, output it
- Delete it from the graph
- Repeat until graph is empty

**Topological sorting, method 2**

- Find a source, output it
- Delete it from the graph
- Repeat until graph is empty

**Topological sorting, method 2**

Justification of correctness  Running time analysis