Basic Depth-First-Search (DFS)

Input: $G = (V, E)$ (directed graph)
      $v \in V$ (start vertex)
Algorithm: DFS (G)

1) \( \forall u \in V \), \( \text{color}(u) \leftarrow \text{white} \); \( \text{time} \leftarrow 0 \)

2) \( \forall u \in V \) if \( \text{color}(u) = \text{white} \) then DFS-visit (u)

What order?

DFS-visit (u)

1) \( \text{color}(u) \leftarrow \text{gray} \); \( \text{push-time}(u) \leftarrow \text{time} \leftarrow \text{time} + 1 \)

2) \( \forall v \in \text{Adj}_G(u) \)

if \( \text{color}(v) = \text{white} \) then DFS-Visit (v)

3) \( \text{color}(u) \leftarrow \text{black} \); \( \text{pop-time}(u) \leftarrow \text{time} \leftarrow \text{time} + 1 \)

Note: \( \text{dfs}(u) = \text{push-time}(u) \)
An Example

Tree

Forward

Cross

Back

Cross

Tree

Back/Edge

Forward
Testing Edge Types

Consider that edge \( e \) is first used.

Tree \( e \) if \( \text{color}(v) = \text{white} \)

BackEdge \( e \) if \( \text{color}(v) = \text{gray} \)

\( \text{color}(v) = \text{black} \) if \( \text{Cross}(e) \) or \( \text{Forward}(e) \)

\( \text{color}(v) = \text{black} \) and \( \text{dfs}(u) < \text{dfs}(v) \) forward

\( n > n' \) cross
Thm. The intervals \([\text{push}(u), \text{pop}(u)]\) are well nested in

\[\text{push}(u) < \text{push}(v) < \text{pop}(v) < \text{pop}(u)\]

or
\[\text{push}(u) < \text{pop}(u) < \text{push}(v) < \text{pop}(v)\]

<table>
<thead>
<tr>
<th>Type edge</th>
<th>pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>pop(v) &gt; pop(u)</td>
</tr>
<tr>
<td>back</td>
<td>pop(v) &gt; pop(u)</td>
</tr>
<tr>
<td>cross &amp; forward</td>
<td>pop(v) &lt; pop(u)</td>
</tr>
</tbody>
</table>

Thm. If \(G\) is a DAG & \((u,v) \in E\) then

\[\text{pop}(v) < \text{pop}(u)\]

\(\text{DAG} \equiv \text{Directed Acyclic Graph}\)
**Thm.** The following are equivalent:

a) $G$ has a cycle

b) Every DFS generates a back edge.

c) Some DFS generates a back edge

**Proof:**

$b) \implies c) \implies a)$ Easy

$a) \implies b)$

Suppose $C$ is a cycle in $G$, DFS

Assume that $X_i$ is first vertex searched

$$C = X_1 \rightarrow X_2 \rightarrow \ldots \rightarrow X_k$$

Claim: $(X_k, X_i)$ is a back edge

push($X_i$) < push($X_k$) < pop($X_k$) ≤ pop($X_i$)
Topological Sort

Def: If $G = (V, E)$ is a DAG then an ordering $x_1, \ldots, x_n$ is a topological sort if

$(V_i, V_j) \in E \Rightarrow i < j$
In a DAG

Thm: reverse pop times is a topological sort.

if \( n \rightarrow b \) \( \Rightarrow \) \( \text{pop}(a) > \text{pop}(b) \)

Thm: Top-Sort is \( O(n+m) \) time
Biconnected Components

G is undirected

G is connected if \( \forall v, w \in V \ \exists \text{ path from } v \text{ to } w. \)

V is an articulation point if \( \exists \text{ distinct } x, y \text{ s.t.} \)

all paths from x to y visit V.

Def. G is biconnected if \( \nexists \) an articulation point.

A graph consisting of a single edge is called a trivial biconnected graph.

Def. A biconnected component is a maximal subgraph which is biconnected.
Using DFS for Biconnectivity

Thin In undirected case all edges are
tree or backedges

Definition
\[ \text{low}(v) = \min \{ \text{dfs}(w) \mid \exists u \text{ u descendent of } v \land \text{u} \rightarrow \text{w} \text{ back edge } \} \cup \{\text{dfs}(v)\} \]
The Articulation Points of a DFS

1) Leaves are not Arts
2) The root is an Art iff \( \# \text{children} \geq 2 \)
3) \( u \) is not leaf & not a root then
   \( u \) is an Art iff \( \exists \text{ child } v \) at \( \text{low}(v) \geq \text{dfs}(u) \)

\[ \text{Def} \]
1) If \( v \) is a leaf then \( T-\{v\} \) is connected
2) If root has 1 child then \( T-\{\text{root}\} \) is connected.
   \( \geq 2 \) children

Any path from one child to other uses root

\[ \text{Diagram} \]
3) \( \Rightarrow \) Suppose paths from \( X \) to \( Y \) use \( U \)

3a) \( \forall y \in \text{Subtree}(U) \land \text{low}(x), \text{low}(y) < \text{dfs}(U) \)
then \( \exists \) path from \( x \) to \( y \) not using \( U \). Contr.

3b) \( x \notin \text{subtree}(U) \) then
\( \text{low}(y) \geq \text{dfs}(U) \)

\[ \leq \]
(\( \leq \)) \( v = \text{child}(U) \) \( \text{low}(v) \geq \text{dfs}(U) \)

\( U \) separates \( v \) from \( r \).

All backedges in \( T_1 \)
can reach at most \( U \).
Computing \( \text{low}(u) \)

1) For \( \text{low}(u) \leftarrow \text{dfs}(u) \)

   if \((u,v)\) is back edge
   \[ \text{low}(u) \leftarrow \min \{ \text{low}(u), \text{dfs}(v) \} \]

   if \((u,v)\) tree edge
   \[ \text{low}(u) \leftarrow \min \{ \text{low}(u), \text{low}(v) \} \]