Basic Depth-First-Search
DFS

Input: \( G = (V, E) \) (directed graph)
\( V \subseteq V \) (start vertex)
Alg: 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 \( \text{DFS-visit}(u) \)

\( \text{What order?} \)

\( \text{DFS-visit}(u) \)

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

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

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

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

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

Tree

Forward

Cross

Backedge

Forward

Cross
Testing Edge Types

Consider the time that edge \((u,v)\) is first used.

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

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

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

\(\text{color}(v) = \text{black} \& \text{dfs}(u) < \text{dfs}(v)\) \hspace{1cm} \text{forward}

\(\text{dfs}(u) > \text{dfs}(v)\) \hspace{1cm} \text{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

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

\[
\begin{array}{c|c}
\text{Type edge} & \text{pop} \\
\hline
\text{Tree} & \text{pop}(v) < \text{pop}(u) \\
\text{Back} & \text{pop}(v) > \text{pop}(u) \\
\text{Cross & Forward} & \text{pop}(v) < \text{pop}(u)
\end{array}
\]

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

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

\(\text{DAG} = \text{Directed Acyclic Graph}\)
Then 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.

$\text{push}(x_i) \leq \text{push}(x_k) < \text{pop}(x_k) < \text{pop}(x_i)$
Topological Sort

Def: If $G = (V, E)$ is a DAG then an ordering $x_1, \ldots, x_n$ is a topological sort if $(x_i, x_j) \in E \Rightarrow i < j$
In a DAG

Thm reverse pop times is a topological sort.

If \( a \rightarrow b \) then \( \text{pop}(a) > \text{pop}(b) \)

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

$G$ is undirected.

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

A vertex $v$ is an articulation point if $\exists \text{ distinct } x, y \text{ s.t. all paths from } x \text{ to } y \text{ visit } v$.

**Def**: $G$ is biconnected if $\exists$ no 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

Thm. In undirected case, all edges are tree or backedges.

Def.
\[ \text{low}(v) = \min \{ \text{dfs}(w) \mid \exists u \text{ descendant of } v \land u \to w \text{ back edge} \} \cup \{\text{dfs}(v)\} \]

Diagram:
- Tree
  - One back edge
The Articulation Points after DFS

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

Proof:

1) If \( v \) is a leaf then \( T - \{v\} \) is connected
2) If root has 1 child then \( T - \{\text{root}\} \) is connected.
   \( > \) 2 children \( \Rightarrow \) not connected

Any path from one child to other uses root
3) \( (\Rightarrow) \) suppose paths from \( X \) to \( Y \) use \( U \)

3a) \( X, Y \in \text{Subtree}(U) \land \text{low}(X), \text{low}(Y) < \text{dfs}(U) \) then \( \exists \) path from \( X \) to \( Y \) not using \( U \) contra.

3b) \( X \notin \text{subtree}(U) \) then \( \text{low}(Y) \geq \text{dfs}(U) \)

\[
(\Leftarrow) \quad \forall v = \text{child}(U) \quad \text{low}(v) \geq \text{dfs}(U)
\]

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

All backedges in \( T_1 \) can reach at most \( U \).
Example

Arts 2, 4
Computing $\text{low}(u)$

1) $\forall u \; \text{low}(u) \leftarrow \text{dfs}(u)$

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

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