15-750
2/24/10

Basic Depth-First-Search
DFS

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

Alg: DFS(G)

1) $\forall u \in V$ color(u) ← white ; time ← 0

2) $\forall u \in V$ <u>if</u> color(u) ≡ white <u>then</u> DFS-visit(u)

what order?

DFS-visit (u)

1) color(u) ← gray ; push-time(u) ← time++

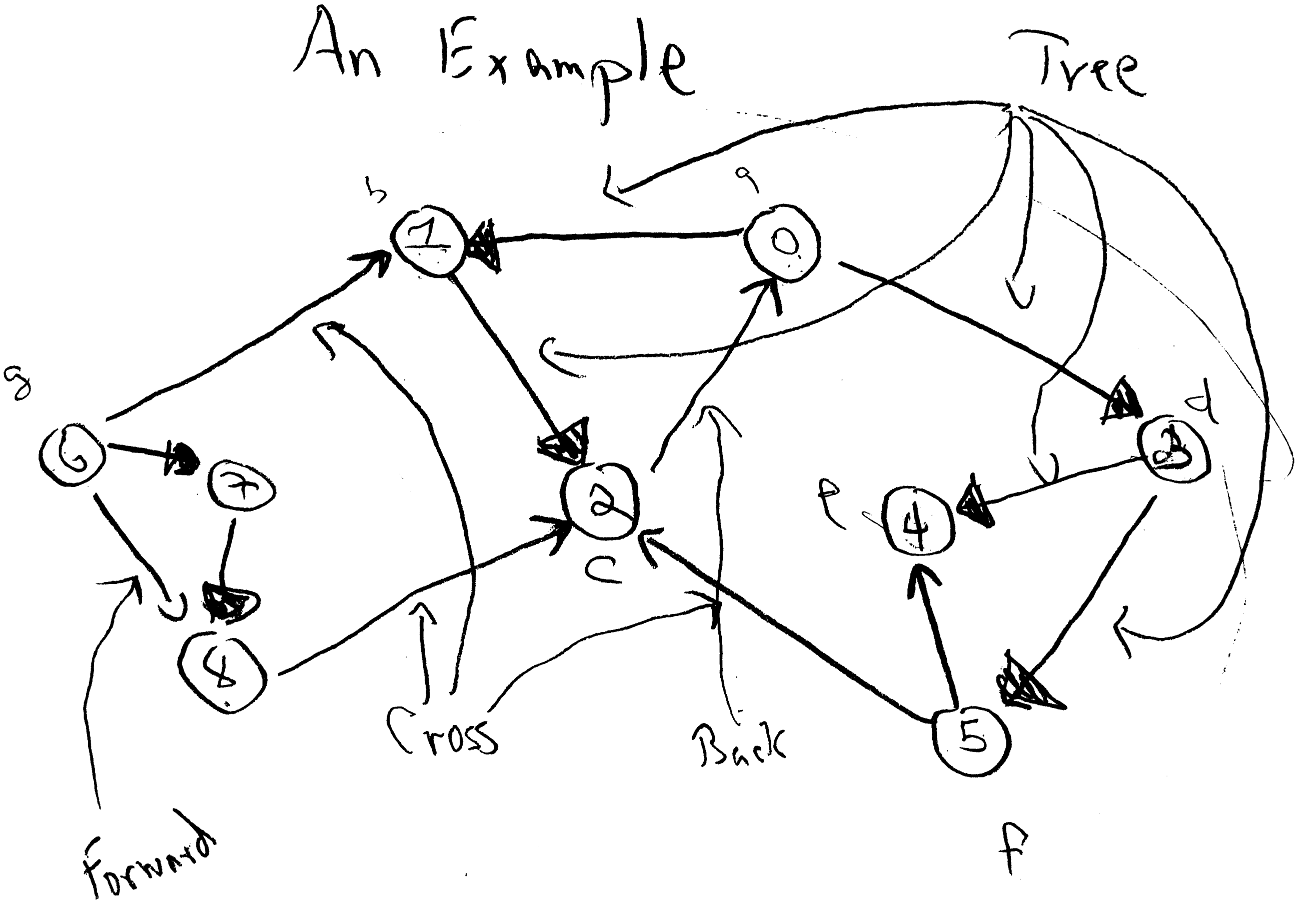
– 2) $\forall v \in$ adj(u)

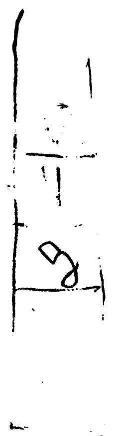
<u>if</u> color(v) ≡ white <u>then</u> DFS-Visit(v)

3) color(u) ← black ; pop-time(u) ← time++

note: dfs(u) ≡ push-time(u)

An Example





Tree

Backedge

Forward

Cross

Testing Edge Types

Conside time that edge (u,v) is first used.
$e = (u,v)$

Tree (e) iff color(v) = white

Back Edge (e) iff color(v) = gray

color(v) = black iff Cross(e) or Forward(e)

---

color(v) = black & dfs(u) < dfs(v) forward

" " > " " cross

Pop Times

Thm The intervals [push(u), pop(u)] are well nested ie

push(u) < push(v) < pop(v) < pop(u)

or push(u) < pop(u) < push(v) < pop(v)

(u,v) type edge	pop
Tree	pop(v) < pop(u)
back	pop(v) > pop(u)
Cross & Forward	POP(v) < pop(u)

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

pop(v) < pop(u)

DAG ≡ Directed Acyclic Graph

<u>Thm</u> The following are $\equiv$

a) G has a cycle

b) Every DFS generates a back edge.

c) Some DFS generates a backedge

<u>pf</u> b) $\Rightarrow$ c) $\Rightarrow$ a) Easy

a) $\Rightarrow$ b)

Suppose C is a cycle in G, DFS

Assume that X_1 is first vertex searched

$$C = \; X_1 \to X_2 \to \cdots \to X_K \to X_1$$

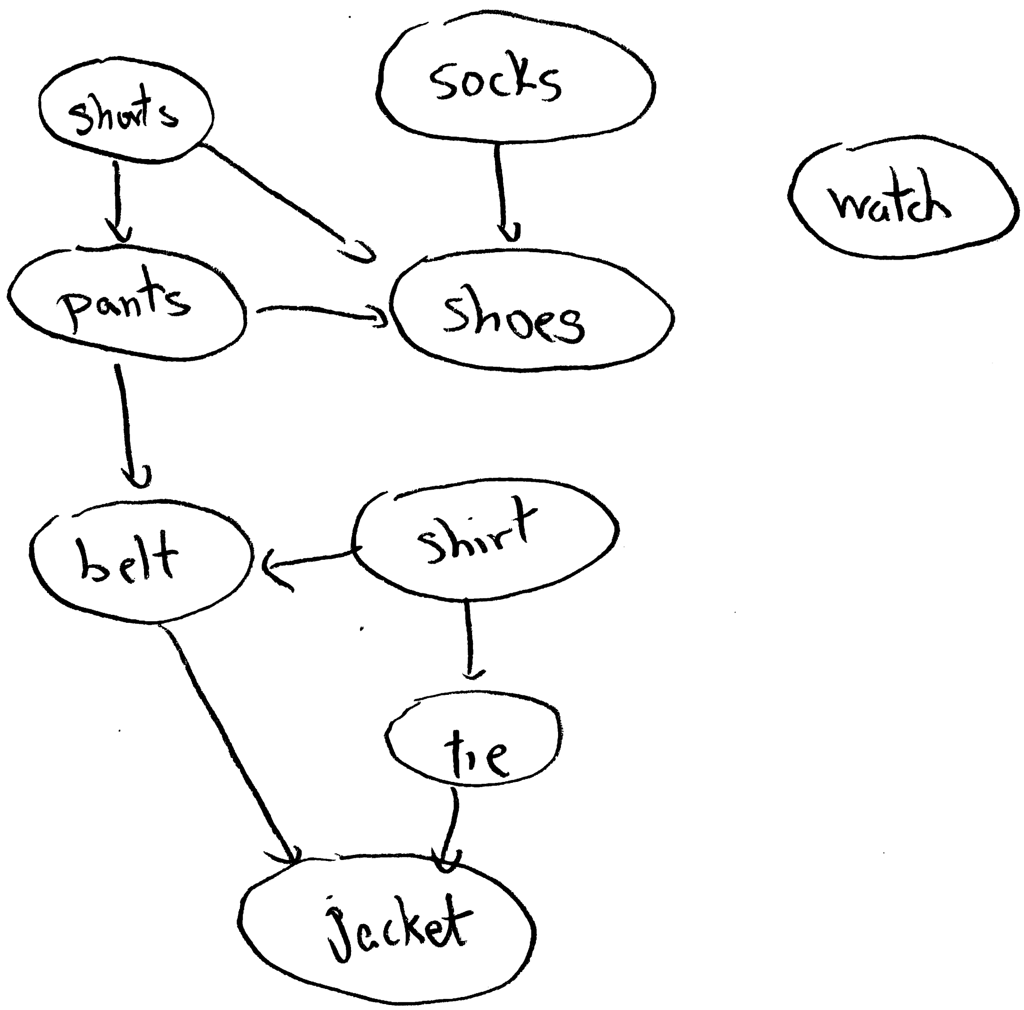
<u>Claim</u> (X_K, X_1) in a backedge

$$push(X_1) < push(X_K) < pop(X_K) < pop(X_1)$$

Topological Sort

<u>Def</u> If $G=(V,E)$ is a DAG then an ordering $x_1 \cdots x_n$ is a <u>topological Sort</u> if

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



<u>Thm</u> In A DAG reverse pop times is a topological Sort.

ie $a \to b \Rightarrow pop(a) > pop(b)$

<u>Thm</u> Top-Sort is $O(n+m)$ time

Biconnected Components

G is undirected

G is **connected** if $\forall v, w \in V$ $\exists$ path from v to w.

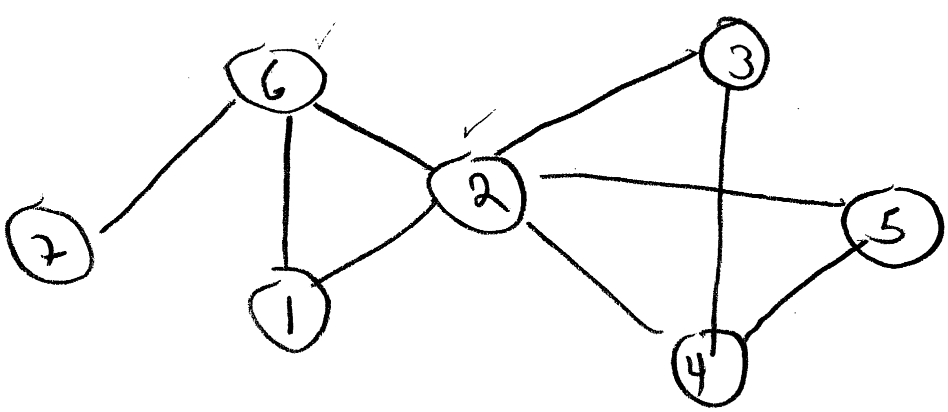
V is an **articulation point** if $\exists$ distinct x, y s.t. all paths from x to y visit v.

G is connected

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

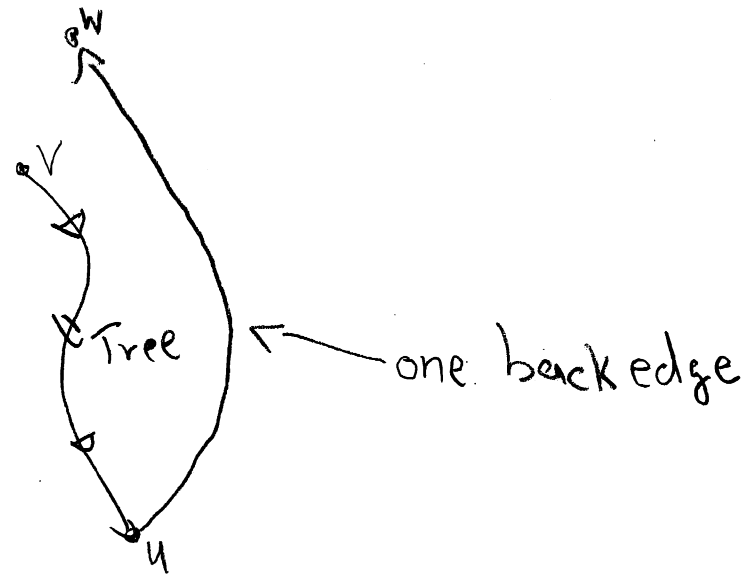


Using DFS for Biconnectivity

Thm In undirected case all edges are tree or backedges

Def

$$low(v) = \min\left\{ \{ DFS(w) \mid \exists u \ u \text{ descendent of } v \wedge u \to w \text{ backedge} \} \cup \{dfs(v)\} \right\}$$



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 st $low(v) \geq dfs(u)$

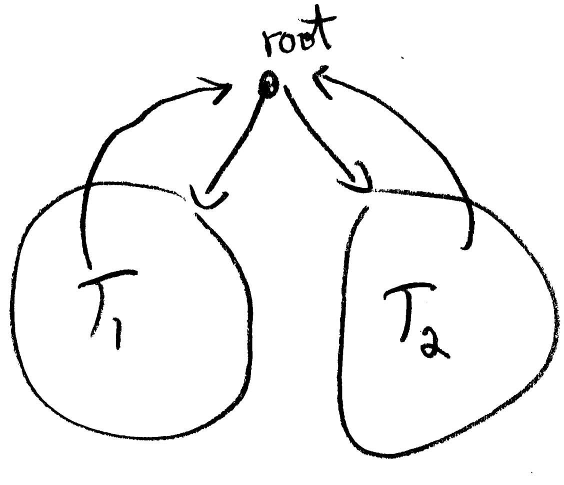
Pf

1) If v is a leaf then $T - \{v\}$ is connected

2) If root has 1 child then $T - \{root\}$ is connected.

≥ 2 children

Any path from one child to other uses root



pf

3) $(\Rightarrow)$ suppose paths from x to y use u

3a) $x, y \in \text{Subtree}(u) \wedge$ suppose $\text{low}(x), \text{low}(y) < \text{dfs}(u)$

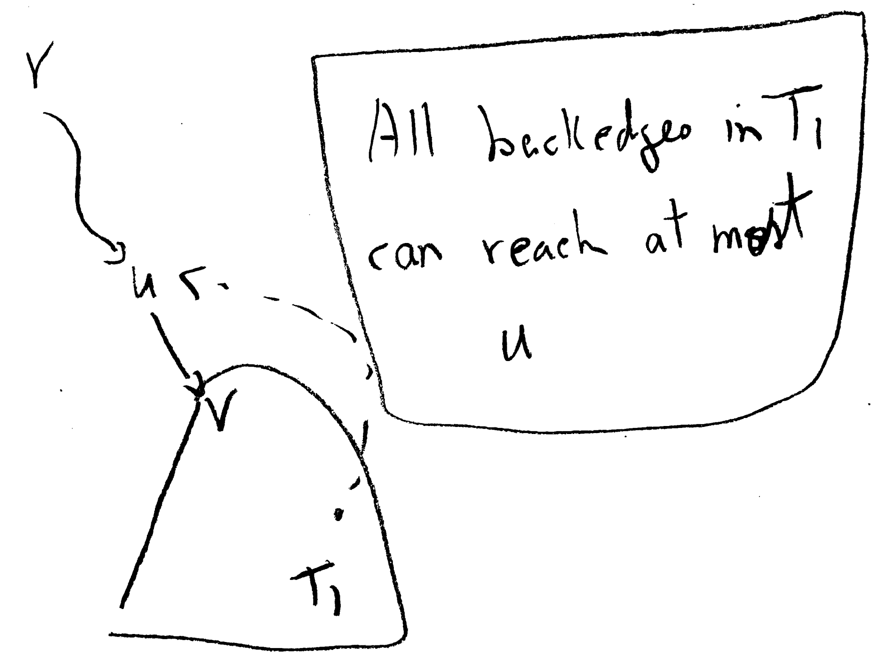
then $\exists$ path from x to y not using u. contra!

3b) $x \notin \text{subtree}(u)$ then

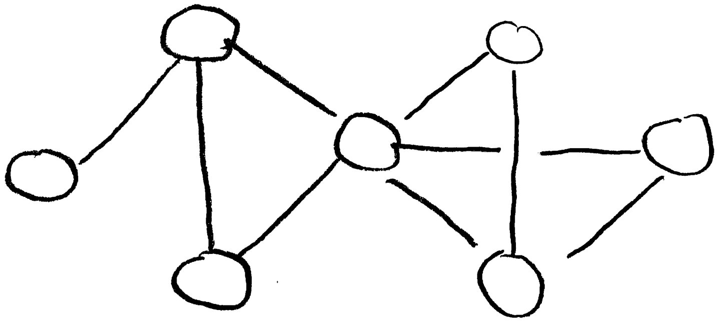
$\text{low}(y) \geq \text{dfs}(u)$

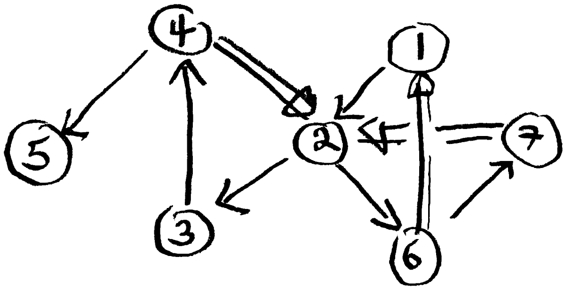
$(\Leftarrow)$ $v = \text{child}(u)$ $\text{low}(v) \geq \text{dfs}(u)$

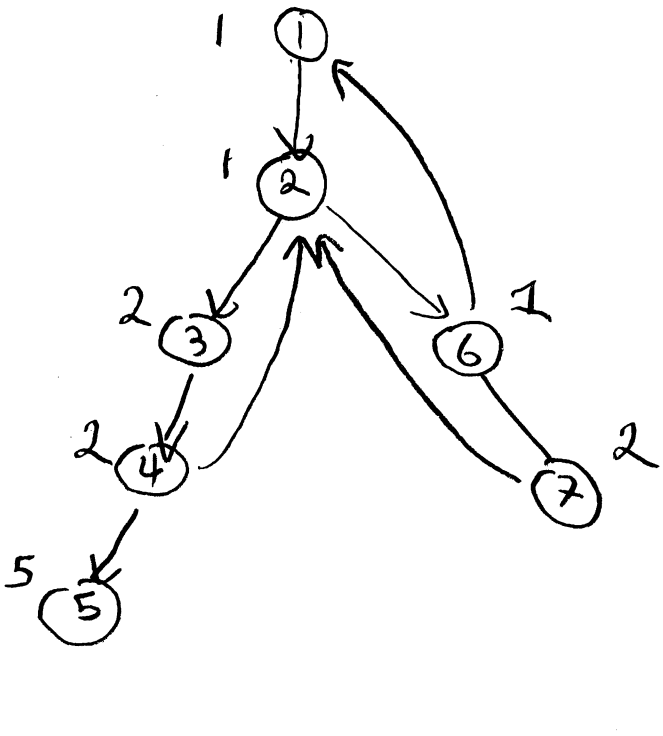
u separates v from r.



Example







Arts 2, 4

Computing low(u)

1) $\forall u$ $low(u) \leftarrow dfs(u)$

if (u,v) is back edge

$low(u) \leftarrow \min\{low(u), dfs(v)\}$

if (u,v) tree edge

$low(u) \leftarrow \min\{low(u), low(v)\}$