15-750
2/11/19

# Searching or Exploring a Graph

There are at least 3 fundamental Diff methods

1) DFS ( depth-first search )
2) BFS ( breadth-first )
3) Random Walk.

Each generate a spanning tree
Namely: First-visit tree.
ie The set of edges used to first visit a vertex.

These trees are very different.

---

Next 2 lectures will be DFS

We start with Basic Depth-first search

<u>Input</u>: $G=(V,E)$ (directed graph)
$s \in V$ (start vertex)

# Basic Depth First Search

Alg: DFS(G)

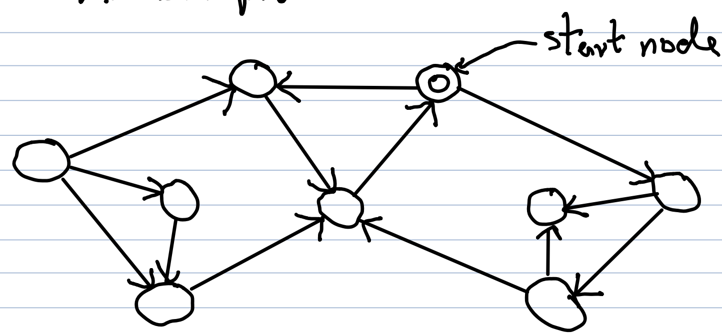
1) $\forall u \in V$ color(u) $\leftarrow$ white ; time $\leftarrow 0$
2) $\forall u \in V$ <u>if</u> color(u) $\equiv$ white <u>then</u> DFS-Visit (u)
   (what order?)

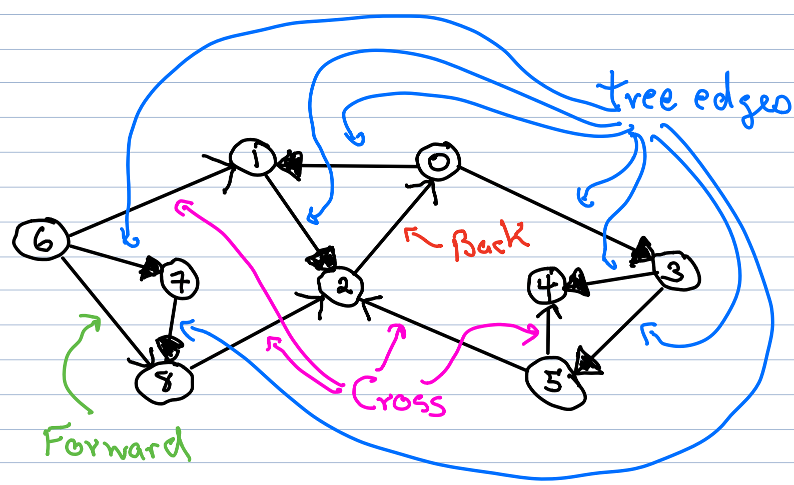
Alg: DFS-Visit (u)

1) color(u) $\leftarrow$ gray ; push-time (u) $\leftarrow$ time++
2) $\forall v \in$ adj(u)
   <u>if</u> color(v) $\equiv$ white <u>then</u> DFS-Visit (v)
3) color(u) $\leftarrow$ black ; pop-time (u) $\leftarrow$ time++

Note: dfs (u) $\equiv$ push-time (u)

# An Example




4-types

Tree
Back-edges
Cross
Forward

# Testing Edge Types

Conside time that edge $(\overset{e}{\overset{\shortparallel}{u}}, v)$ is first used.

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 edge

"   " > "   "   cross edge

# 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 | $POP(v) < pop(u)$ |

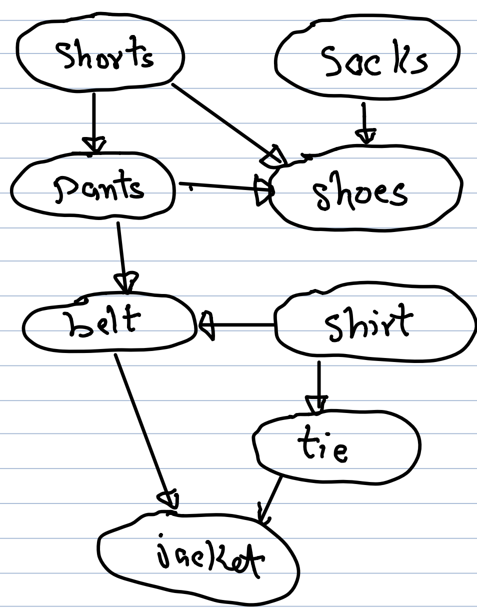
**Thm** If G is a DAG & $(u,v) \in E$ then
$pop(v) < pop(u)$

DAG ≡ Directed Acyclic Graph

# Topological Sort

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

$$(x_i, x_j) \in E \Rightarrow i < j$$




---

**Thm** For a DAG reverse pop times is a topological Sort.

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

**Thm** Topological Sort is $O(n+m)$ time assume $G$ is a DAG.

**Thm** The following are $\equiv$

a) $G$ has a cycle

b) Every DFS generates a back edge.

c) Some DFS generates a backedge

**pf** 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

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

**Claim** $(X_K, X_1)$ is a backedge

$$\text{push}(X_1) < \text{push}(X_K) < \text{pop}(X_K) < \text{pop}(X_1)$$

# Biconnected Components

G is undirected

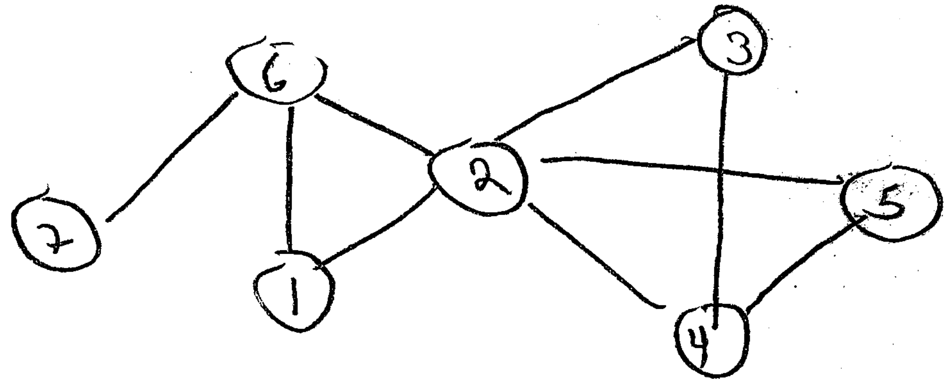
G is <u>connected</u> if $\forall v, w \in V \; \exists$ path from v to w.

v is an <u>articulation</u> <u>point</u> if $\exists$ distinct x, y s.t. all paths from x to y visit v.

**Def** G is <u>biconnected</u> if $\nexists$ an articulation point
G is connected

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

**Def** A <u>biconnected</u> <u>component</u> is a maximal subgraph which is biconnected

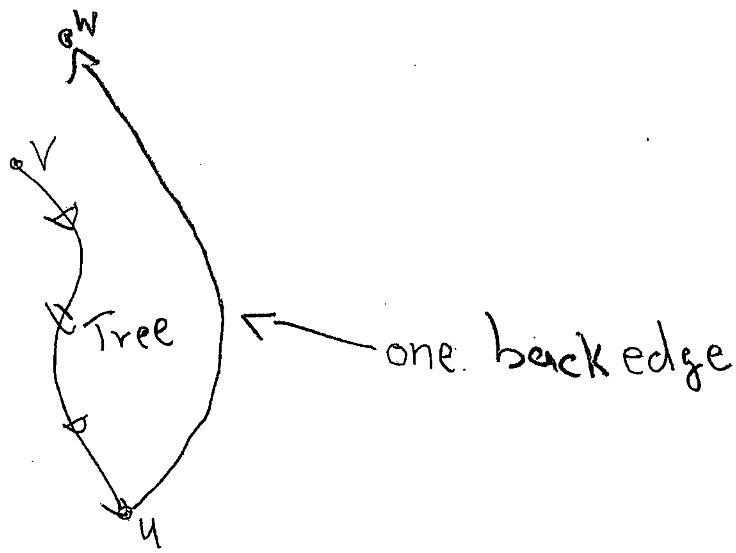


# Using DFS for Biconnectivity

Thm In undirected case all edges are tree or backedges

Def

$$low(v) = \min\left\{ \{ Dfs(w) \mid \exists u \;\; u \text{ descendent of } v \wedge u \to w \text{ back edge} \} \cup \{dfs(v)\} \right\}$$



# The Articulation Points after DFS

**Thm** Suppose $G$ is connected & we have run DFS.

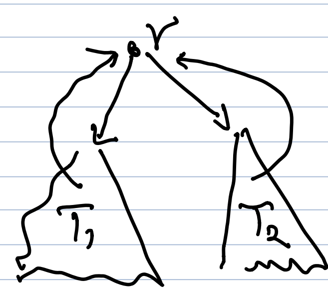
1) Leaves are not Arts.
2) The root is an Art iff #children $\geq 2$
3) If $u$ is not leaf and not the root then
   $u$ is an Art iff $\exists$ child $v$ of $u$ st $low(v) \geq dfs(u)$.

---

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

2) If $r$ is root with 1 child then $T-\{r\}$ is connected.
   $\geq 2$ children " " not connected.

Since any path from one child to the other uses root.



## Pf of case 3.

($\Rightarrow$) Suppose $u$ not leaf or root and $\exists\, x \neq y \neq u \neq x$ all paths from $x$ to $y$ use $u$.

3 subcases

a) $x, y \notin$ subtree($u$) (false, empty)

b) $x, y \in$ subtree($u$)

c) $x \in$ subtree & $y \notin$ subtree

3b) Suppose $low(x)$ & $low(y) < dfs(u)$ (contra!)

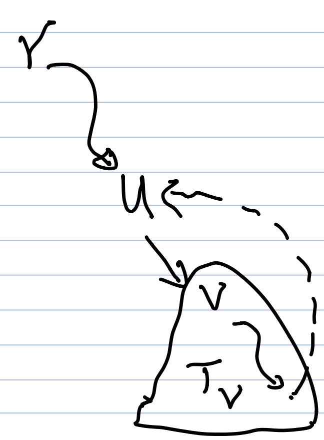
WLOG $low(x) \geq dfs(u)$

3c) $low(x) < dfs(u)$ contra!

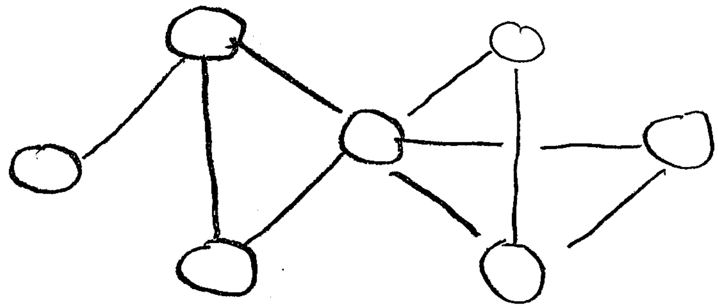
---

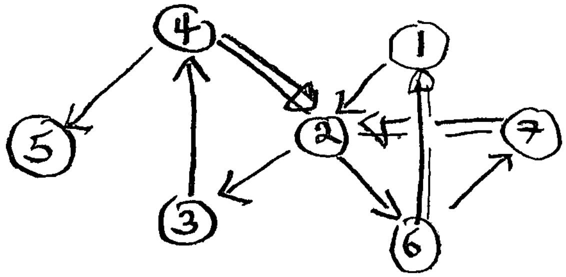
($\Leftarrow$) $v = child(u)$ & $low(v) \geq dfs(u)$

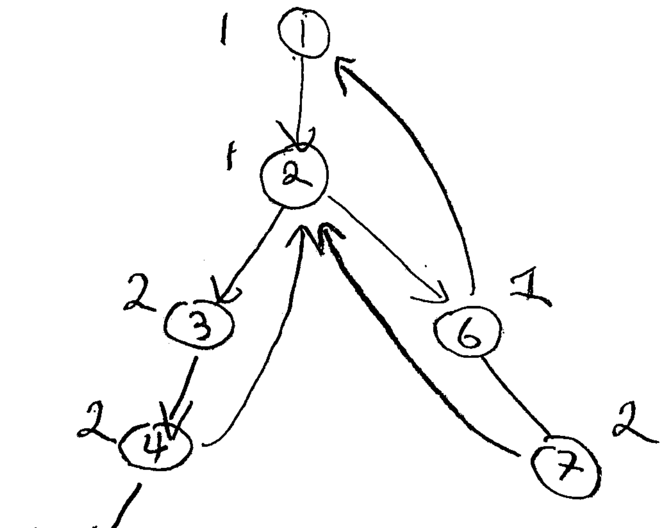
then $u$ separates $v$ from root



# Example






Arts 2, 4

# Depth First Search to compute low(u)

Alg: DFS(G)

1) $\forall u \in V$ color(u) $\leftarrow$ white ; time $\leftarrow 0$
2) $\forall u \in V$ if color(u) = white then DFS-Visit(u)
   (what order?)

Alg: DFS-Visit(u)

1) color(u) $\leftarrow$ gray ; push-time(u) $\leftarrow$ time++ ;
   1a) low(u) $\leftarrow$ dfs(u)
2) $\forall v \in$ adj(u)
   if color(v) = white then DFS-Visit(v)
3) color(u) $\leftarrow$ black ; pop-time(u) $\leftarrow$ time++
   3a) If (u,v) is backedge then
       low(u) $\leftarrow$ min{low(u), dfs(v)}
   3b) If (u,v) is tree edge then
       low(u) $\leftarrow$ min{low(u), low(v)}