3-6 Decompositions of Graphs

(Part II: DFS, SCC, Bicomponent)

Hengfeng Wei

hfwei@nju.edu.cn

November 05, 2018

The Power of the Hammer of DFS

Graph Traversal $\Longrightarrow$ Graph Decomposition

Structure! Structure! Structure!

SIAM J. COMPUT.
Vol. 1, No. 2, June 1972

DEPTH-FIRST SEARCH AND LINEAR GRAPH ALGORITHMS*

ROBERT TARJAN†

Abstract. The value of depth-first search or "backtracking" as a technique for solving problems is illustrated by two examples. An improved version of an algorithm for finding the strongly connected components of a directed graph and an algorithm for finding the biconnected components of an undirect graph are presented. The space and time requirements of both algorithms are bounded by $k_1V + k_2E + k_3$ for some constants k_1, k_2, and k_3, where V is the number of vertices and E is the number of edges of the graph being examined.

Key words. Algorithm, backtracking, biconnectivity, connectivity, depth-first, graph, search, spanning tree, strong-connectivity.

"Depth-First Search And Linear Graph Algorithms", Robert Tarjan

SIAM J. COMPUT.
Vol. 1, No. 2, June 1972

DEPTH-FIRST SEARCH AND LINEAR GRAPH ALGORITHMS*

ROBERT TARJAN†

Abstract. The value of depth-first search or "backtracking" as a technique for solving problems is illustrated by two examples. An improved version of an algorithm for finding the strongly connected components of a directed graph and an algorithm for finding the biconnected components of an undirect graph are presented. The space and time requirements of both algorithms are bounded by $k_1V + k_2E + k_3$ for some constants k_1, k_2, and k_3, where V is the number of vertices and E is the number of edges of the graph being examined.

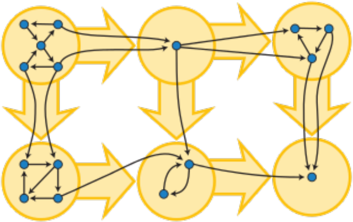
Key words. Algorithm, backtracking, biconnectivity, connectivity, depth-first, graph, search, spanning tree, strong-connectivity.

"Depth-First Search And Linear Graph Algorithms", Robert Tarjan

Tarjan's SCC   Bicomponent

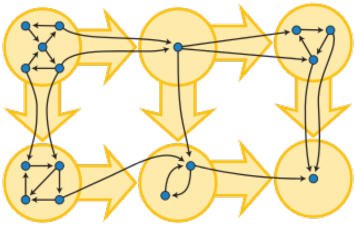
Theorem (Digraph as DAG)

Every digraph is a dag of its SCCs.



Theorem (Digraph as DAG)

Every digraph is a dag of its SCCs.



Two tiered structure of digraphs:

digraph $\equiv$ a dag of SCCs

SCC: equivalence class over reachability

Semiconnected Digraph (Problem 4.14)

$$\forall u, v \in V : u \rightsquigarrow v \vee v \rightsquigarrow u$$

Semiconnected Digraph (Problem 4.14)

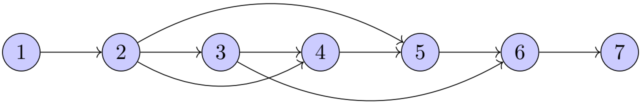
$$\forall u, v \in V : u \rightsquigarrow v \lor v \rightsquigarrow u$$

$$\boxed{\text{digraph} \equiv \text{a dag of SCCs}}$$

Semiconnected Digraph (Problem 4.14)

$$\forall u, v \in V : u \leadsto v \vee v \leadsto u$$

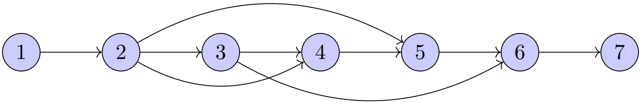
$\boxed{\text{digraph} \equiv \text{a dag of SCCs}}$



Semiconnected Digraph (Problem 4.14)

$$\forall u, v \in V : u \rightsquigarrow v \lor v \rightsquigarrow u$$

$\boxed{\text{digraph} \equiv \text{a dag of SCCs}}$



$\boxed{\text{DAG: Semiconnected} \iff \exists!\ \text{topo. ordering}}$

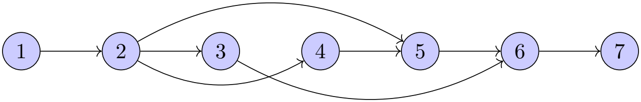
$$\boxed{\text{DAG: Semiconnected} \iff \exists!\ \text{topo. ordering}}$$

$$\boxed{\text{DAG: Semiconnected} \iff \exists! \text{ topo. ordering}}$$

Tarjan's TOPOSORT + Check edges (v_i, v_{i+1})

$$\boxed{\text{DAG: Semiconnected} \iff \exists!\ \text{topo. ordering}}$$

Tarjan's TOPOSORT + Check edges (v_i, v_{i+1})



$\boxed{\text{digraph} \equiv \text{a dag of SCCs}}$

Theorem

Let C and C' be two SCCs in directed graph G.

$$u \in C \to v \in C' \implies f[C] > f[C']$$

$$\boxed{\text{digraph} \equiv \text{a dag of SCCs}}$$

Theorem

Let C and C' be two SCCs in directed graph G.

$$u \in C \to v \in C' \implies f[C] > f[C']$$

$$d[U] = \min_{u \in U} \{d[u]\} \qquad f[U] = \max_{u \in U} \{f[u]\}$$

$$\text{DAG} \implies \boxed{u \to v \iff f[v] < f[u]}$$

Kosaraju's SCC algorithm, 1978

SCCs can be topo-sorted in decreasing order of their $f[\cdot]$.

Kosaraju's SCC algorithm, 1978

SCCs can be topo-sorted in decreasing order of their $f[\cdot]$.

The vertice v with the highest $f[v]$ is in a source SCC.

Kosaraju's SCC algorithm, 1978

SCCs can be topo-sorted in decreasing order of their $f[\cdot]$.

The vertice v with the highest $f[v]$ is in a source SCC.

(I) DFS on G; DFS on G^T ($f[\cdot] \downarrow$)

Kosaraju's SCC algorithm, 1978

SCCs can be topo-sorted in decreasing order of their $f[\cdot]$.

The vertice v with the highest $f[v]$ is in a source SCC.

(I) DFS on G; DFS on G^T $(f[\cdot] \downarrow)$

(II) DFS on G^T; DFS on G

Kosaraju's SCC algorithm, 1978

SCCs can be topo-sorted in decreasing order of their $f[\cdot]$.

The vertice v with the highest $f[v]$ is in a source SCC.

(I) DFS on G; DFS on G^T ($f[\cdot] \downarrow$)

(II) DFS on G^T; DFS on G ($f[\cdot] \downarrow$)

Kosaraju's SCC algorithm, 1978

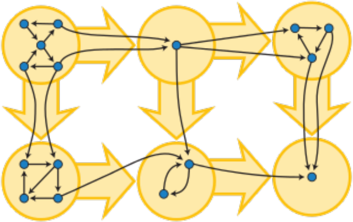
SCCs can be topo-sorted in decreasing order of their $f[\cdot]$.

The vertice v with the highest $f[v]$ is in a source SCC.

(I) DFS on G; DFS/BFS on G^T ($f[\cdot] \downarrow$)

(II) DFS on G^T; DFS/BFS on G ($f[\cdot] \downarrow$)

Component Graph of a Digraph (Problem 22.5-5)



procedure $\textsc{CompGraph}(G, \mathcal{C})$
  for $C, C' \in \mathcal{C}$ **do**
    for $u \in C, v \in C'$ **do**
      if $u \to v$ **then**
        $C \to C'$

procedure $\textsc{CompGraph}(G, \mathcal{C})$
  for $(u, v) \in E$ **do**
    $C[u] \to C[v]$

Definition (Biconnected Graph)

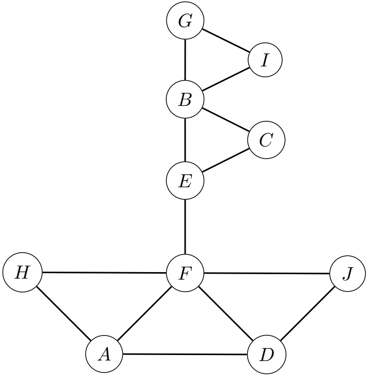
A connected undirected graph is *biconnected* if it contains no "cut-nodes".

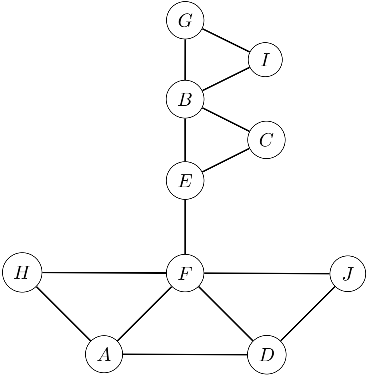
Definition (Biconnected Graph)

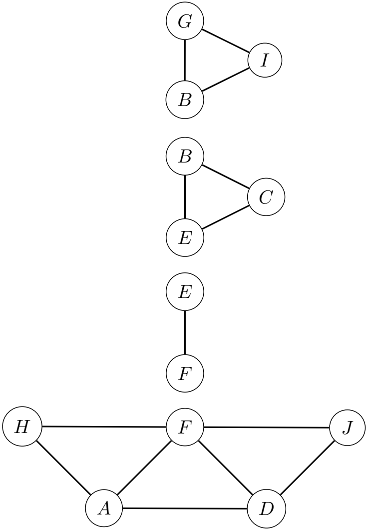
A connected undirected graph is *biconnected* if it contains no "cut-nodes".

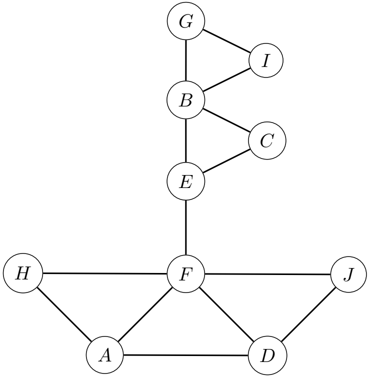
Definition (Biconnected Component (Bicomponent))

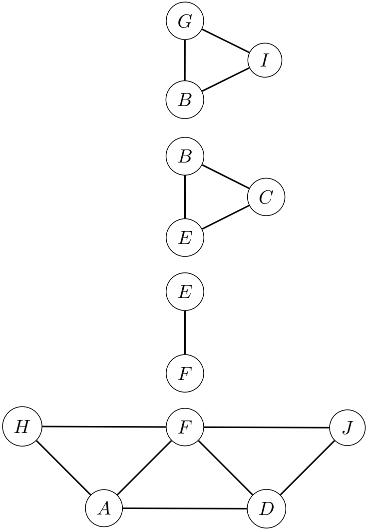
A *bicomponent* of an undirected graph is a maximal biconnected subgraph.


G
I
B
C
E
H
F
J
A
D


G
I
B
C
E
H
F
J
A
D


G
I
B
B
C
E
E
F
H
F
J
A
D





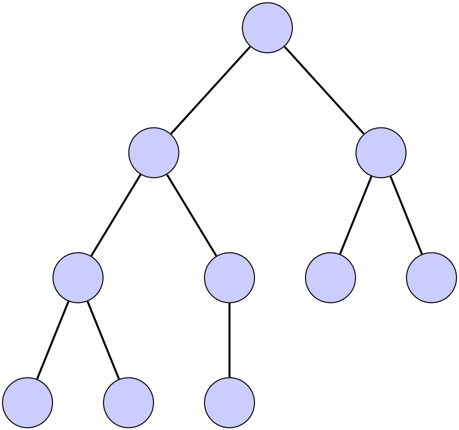
Paritition of edges (not of nodes)

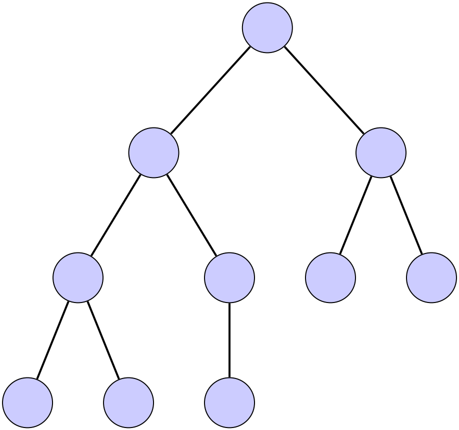
The Power of the Hammer of DFS on Undirected Graphs

The Power of the Hammer of DFS on Undirected Graphs

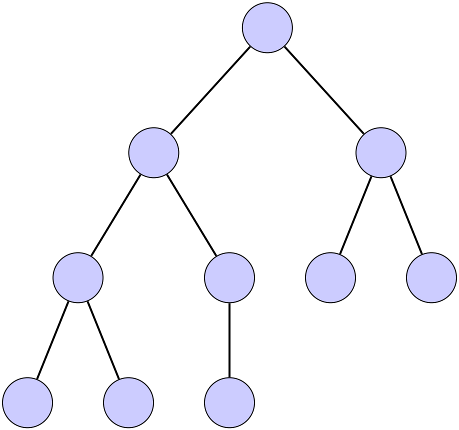
Theorem (Theorem 22.10)

In a depth-first search of an undirected graph G, every edge of G is either a tree edge or a back edge.



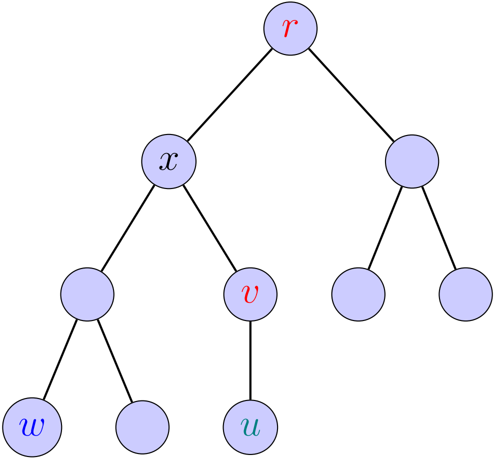


Cut-nodes?

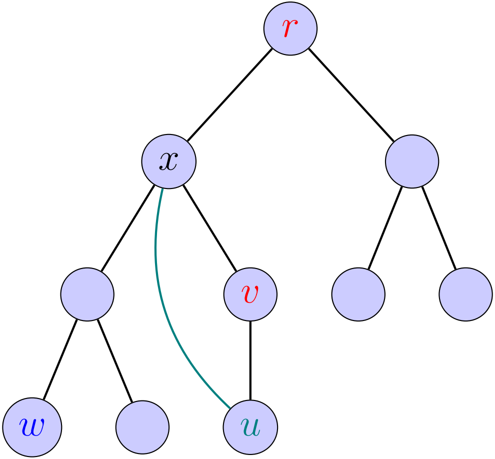


Cut-nodes? Bicomponents?

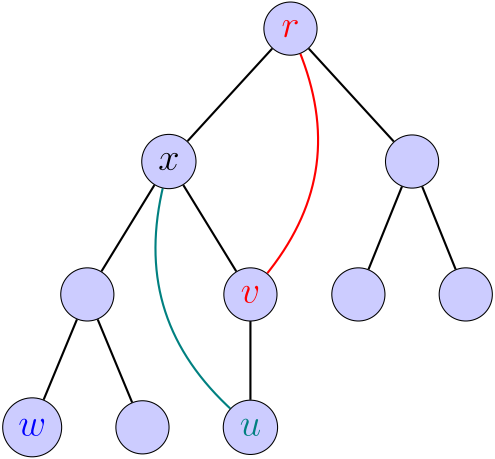
Bicomp: Back!



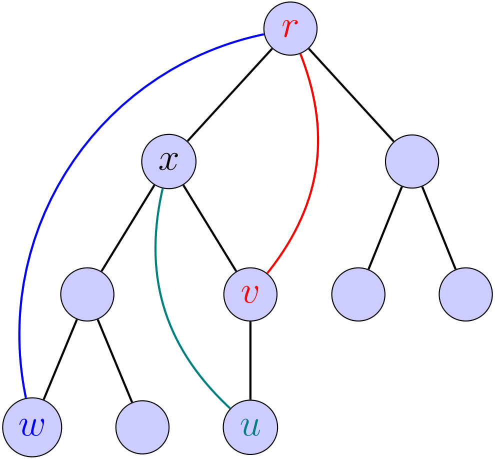
Bicomp: Back!



Bicomp: Back!



Bicomp: Back!




BACK
to the
past

back$[v]$:

The earliest ancestor v can get

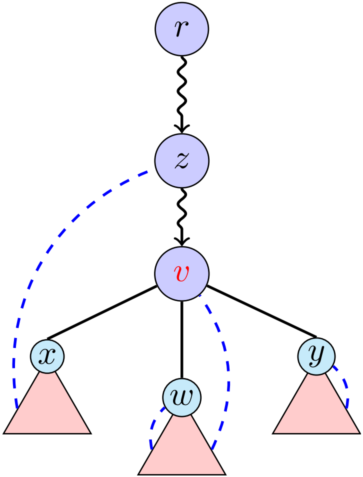
by following tree edges $\mathbb{T}$ and back edges $\mathbb{B}$.

back[v] :

The earliest ancestor v can get

by following tree edges $\mathbb{T}$ and back edges $\mathbb{B}$.

(I) When and how to initialize&update back[v]?



r
z
v
x
w
y

Theorem (Characterization of Cut-nodes)

In a DFS tree, v is a cut-node

$$\Longleftrightarrow$$

(a) *v is the root and $deg(v) \geq 2$*

(b) *v is not the root and some subtree of v has no back edge to a proper ancestor of v*

Theorem (Characterization of Cut-nodes)

In a DFS tree, v is a cut-node

$$\iff$$

(a) v *is the root and* $deg(v) \geq 2$

(b) v *is not the root and some subtree of* v *has no back edge to a proper ancestor of* v

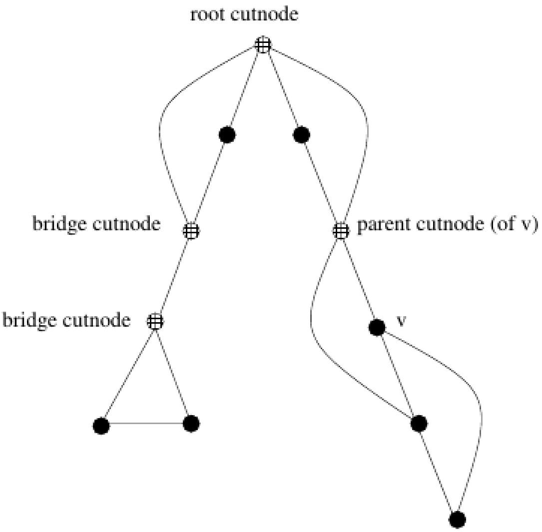
(II) When and how to identify a bicomponent?

back[v] :

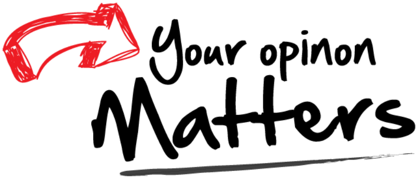
The earliest ancestor v can get

by following tree edges $\mathbb{T}$ and back edges $\mathbb{B}$.

$$
\begin{aligned}
\text{tree edge } (\to v)\text{:}\quad & \text{back}[v] = d[v] \\
\text{back edge } (v \to w)\text{:}\quad & \text{back}[v] = \min\{\text{back}[v], d[w]\} \\
\text{backtracking from } w\text{:}\quad & \text{back}[v] = \min\{\text{back}[v], \text{back}[w] = \text{wBack}\}
\end{aligned}
$$


root cutnode
bridge cutnode
parent cutnode (of v)
bridge cutnode
v


THANK YOU



Office 302

Mailbox: H016

hfwei@nju.edu.cn