Decompositions of Graphs

Hengfeng Wei

hfwei@nju.edu.cn

June 12, 2018

John Hopcroft

Robert Tarjan

"For fundamental achievements in the design and analysis of algorithms and data structures."

— *Turing Award, 1986*

Depth-first search

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.

Reference

- "Depth-First Search And Linear Graph Algorithms" by Robert Tarjan.

Depth-first search

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.

"We have seen how the depth-first search method may be used in the construction of very efficient graph algorithms. . . .
Depth-first search is a powerful technique with many applications."

Reference

- "Depth-First Search And Linear Graph Algorithms" by Robert Tarjan.

The POWER of DFS

Graph decomposition vs. Graph traversal

Structures!

The POWER of DFS

Graph decomposition vs. Graph traversal

Structures!

1. states of vertices
2. types of edges
3. lifetime of vertices (DFS)
   - v : d$[v]$, f$[v]$
   - f$[v]$: DAG, SCC
   - d$[v]$: biconnectivity

Types of edges

Definition (Classifying edges)

Given a DFS/BFS traversal $\Rightarrow$ DFS/BFS tree:

Tree edge: $\rightarrow$ child

Back edge: $\rightarrow$ ancestor

Forward edge: $\rightarrow$ *nonchild* descendant

Cross edge: $\rightarrow$ neither ancestor nor descendant

Types of edges

Definition (Classifying edges)

Given a DFS/BFS traversal $\Rightarrow$ DFS/BFS tree:

Tree edge: $\rightarrow$ child

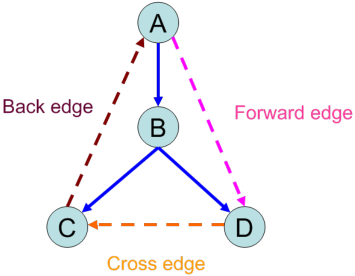
Back edge: $\rightarrow$ ancestor

Forward edge: $\rightarrow$ *nonchild* descendant

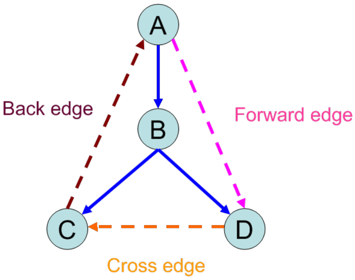
Cross edge: $\rightarrow$ neither ancestor nor descendant

Remarks

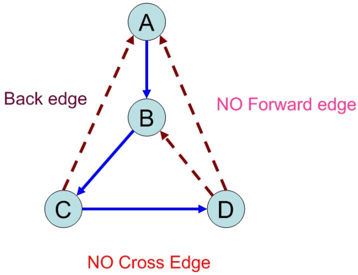
- applicable to both DFS and BFS
- w.r.t. DFS/BFS trees



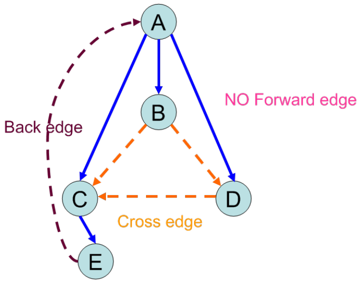
DFS on directed graph



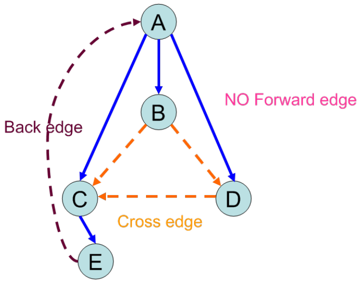
DFS on directed graph



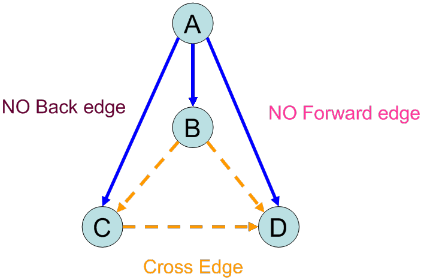
DFS on undirected graph



BFS on directed graph



BFS on directed graph



BFS on undirected graph

DFS tree and BFS tree coincide (Problem 5.7)

Undirected connected graph $G = (V, E), v \in V$

DFS tree T from $v \equiv$ BFS tree T' from v

DFS tree and BFS tree coincide (Problem 5.7)

Undirected connected graph $G = (V, E), v \in V$

DFS tree T from $v \equiv$ BFS tree T' from v

$$G \equiv T$$

DFS tree and BFS tree coincide (Problem 5.7)

Undirected connected graph $G = (V, E), v \in V$

DFS tree T from $v \equiv$ BFS tree T' from v

$$G \equiv T$$

Proof.

G_{DFS}: tree + back *vs.* G_{BFS}: tree + cross

$\square$

DFS tree and BFS tree coincide (Problem 5.7)

Undirected connected graph $G = (V, E), v \in V$

DFS tree T from $v \equiv$ BFS tree T' from v

$$G \equiv T$$

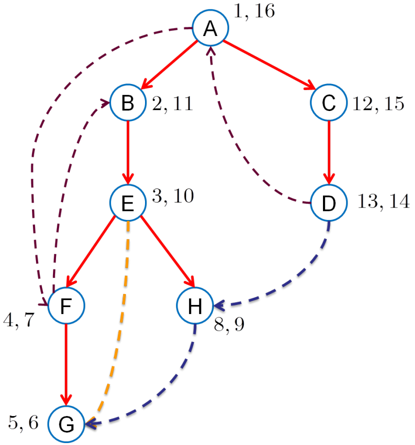
Proof.

G_{DFS}: tree + back *vs.* G_{BFS}: tree + cross

□

Q : What if G is a digraph?

Lift time of vertices in DFS



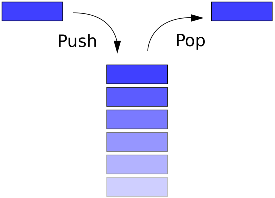
Theorem (Disjoint or Contained (Problem 4.2: (1) & (2)))

$$\forall u, v : [_u\ ]_u \cap [_v\ ]_v = \emptyset \bigvee \left([_u\ ]_u \subsetneqq [_v\ ]_v \vee [_v\ ]_v \subsetneqq [_u\ ]_u \right)$$

Theorem (Disjoint or Contained (Problem 4.2: (1) & (2)))

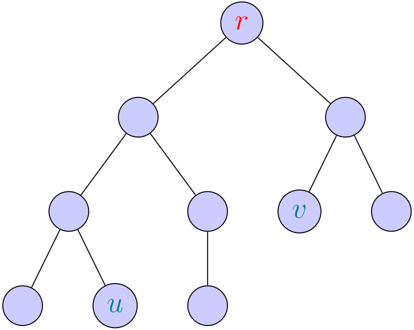
$$\forall u, v : [_u\ ]_u \cap [_v\ ]_v = \emptyset \bigvee \left([_u\ ]_u \subsetneq [_v\ ]_v \vee [_v\ ]_v \subsetneq [_u\ ]_u \right)$$

Proof.



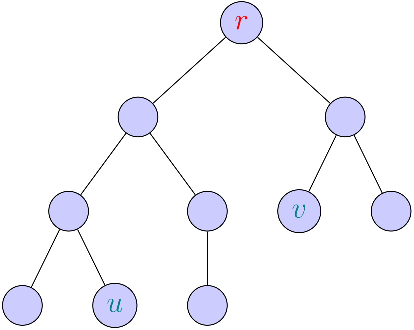
$\square$

Preprocessing for ancestor/descendant relation (Problem 5.23)



Q : Is u an ancestor of v? $O(1)$

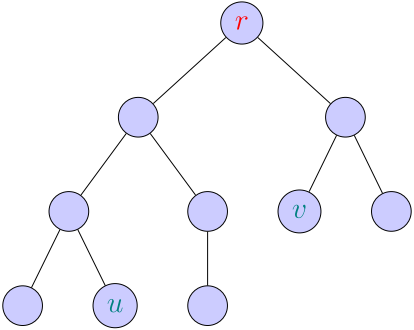
Preprocessing for ancestor/descendant relation (Problem 5.23)



Q : Is u an ancestor of v? $O(1)$

$$v : \mathsf{d}[v], \mathsf{f}[v]$$

Preprocessing for ancestor/descendant relation (Problem 5.23)



Q : Is u an ancestor of v? $O(1)$

$$v : \mathsf{d}[v], \mathsf{f}[v]$$

Q : # of descendants of any v?

Edge types and life time of vertices in DFS (Problem 4.5)

$$\forall u \to v :$$

- tree/forward edge: $[_u \ [_v \ ]_v \ ]_u$
- back edge: $[_v \ [_u \ ]_u \ ]_v$
- cross edge: $[_v \ ]_v \ [_u \ ]_u$

Edge types and life time of vertices in DFS (Problem 4.5)

$$\forall u \to v :$$

- tree/forward edge: $[_u \ [_v \ ]_v \ ]_u$
- back edge: $[_v \ [_u \ ]_u \ ]_v$
- cross edge: $[_v \ ]_v \ [_u \ ]_u$

$$\mathsf{f}[v] < \mathsf{d}[u] \iff \qquad \text{edge}$$

$$\nexists \text{ cycle } \implies$$

Edge types and life time of vertices in DFS (Problem 4.5)

$$\forall u \to v :$$

- tree/forward edge: $[_u \; [_v \;]_v \;]_u$
- back edge: $[_v \; [_u \;]_u \;]_v$
- cross edge: $[_v \;]_v \; [_u \;]_u$

$$\text{f}[v] < \text{d}[u] \iff \text{cross edge}$$

$$\nexists \text{ cycle} \implies$$

Edge types and life time of vertices in DFS (Problem 4.5)

$$\forall u \to v :$$

- tree/forward edge: $[_u\ [_v\ ]_v\ ]_u$
- back edge: $[_v\ [_u\ ]_u\ ]_v$
- cross edge: $[_v\ ]_v\ [_u\ ]_u$

$$\text{f}[v] < \text{d}[u] \iff \text{cross edge}$$

$$\text{f}[u] < \text{f}[v] \iff$$

$$\nexists \text{ cycle} \implies$$

Edge types and life time of vertices in DFS (Problem 4.5)

$$\forall u \to v :$$

- tree/forward edge: $[_u\ [_v\ ]_v\ ]_u$
- back edge: $[_v\ [_u\ ]_u\ ]_v$
- cross edge: $[_v\ ]_v\ [_u\ ]_u$

$$\text{f}[v] < \text{d}[u] \iff \text{cross edge}$$

$$\text{f}[u] < \text{f}[v] \iff \text{back edge}$$

$$\nexists \text{ cycle} \implies$$

Edge types and life time of vertices in DFS (Problem 4.5)

$$\forall u \to v :$$

- tree/forward edge: $[_u \; [_v \;]_v \;]_u$
- back edge: $[_v \; [_u \;]_u \;]_v$
- cross edge: $[_v \;]_v \; [_u \;]_u$

$$\mathsf{f}[v] < \mathsf{d}[u] \iff \text{cross edge}$$

$$\mathsf{f}[u] < \mathsf{f}[v] \iff \text{back edge}$$

$$\nexists \text{ cycle} \implies \boxed{u \to v \iff \mathsf{f}[v] < \mathsf{f}[u]}$$

Height and diameter of tree (Problem 5.4)

Binary tree $T = (V, E)$ with $|V| = n$:

- height $H(T)$ in $O(n)$
- diameter $D(T)$ in $O(n)$

Height and diameter of tree (Problem 5.4)

Binary tree $T = (V, E)$ with $|V| = n$:

- height $H(T)$ in $O(n)$
- diameter $D(T)$ in $O(n)$

$$\begin{cases} H(T) = 0, & T \text{ is a leave} \\ H(T) = \max\Big(H(L_T), H(R_T)\Big) + 1, & \text{o.w.} \end{cases}$$

throught root or not?

Height and diameter of tree (Problem 5.4)

Binary tree $T = (V, E)$ with $|V| = n$:

- height $H(T)$ in $O(n)$
- diameter $D(T)$ in $O(n)$

$$\begin{cases} H(T) = 0, & T \text{ is a leave} \\ H(T) = \max\Big(H(L_T), H(R_T)\Big) + 1, & \text{o.w.} \end{cases}$$

throught root or not?

Question

Diameter of a tree *without* a designated root?

Perfect subtree

Perfect subtree (Problem 5.22)

- binary tree $T = (V, E)$
- root $r \in V$
- goal: find all perfect subtrees

Counting shortest paths

Counting shortest paths (Problem 5.26)

Counting # of shortest paths in (un)directed graphs using BFS.

Counting shortest paths

Counting shortest paths (Problem 5.26)

Counting $\#$ of shortest paths in (un)directed graphs using BFS.

Maybe in the next class. . .


THANK YOU