Decompositions of Graphs

Hengfeng Wei

hfwei@nju.edu.cn

May 20, 2017 – May 24, 2017

Decompositions of Graphs

Turing Award

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.

Graph decomposition

Graph decomposition vs. Graph traversal

Structures!

Graph decomposition

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

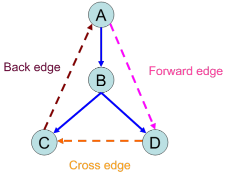
Forward edge: $\rightarrow$ *nonchild* descendant

Cross edge: $\rightarrow$ neither ancestor nor descendant

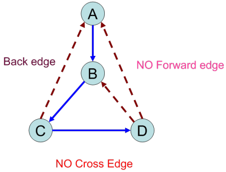
Remarks

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

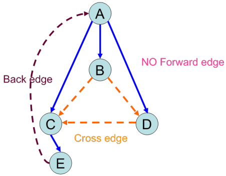
Types of edges (Problem 5.18)



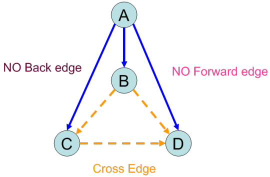
(a) DFS on directed graph.



(b) DFS on undirected graph.



(c) BFS on directed graph.



(d) BFS on undirected graph.

Types of edges

DFS tree and BFS tree coincide (Additional)

$G = (V, E), v \in V$.
DFS tree T = BFS tree T'.

- G is an undirected graph $\implies G = T$
- G is a digraph $\overset{?}{\implies} G = T$

Types of edges

DFS tree and BFS tree coincide (Additional)

$G = (V, E), v \in V$.
DFS tree T = BFS tree T'.

- G is an undirected graph $\implies G = T$
- G is a digraph $\overset{?}{\implies} G = T$

- T: tree + back *vs.* T': tree + cross

Types of edges

DFS tree and BFS tree coincide (Additional)

$G = (V, E), v \in V$.
DFS tree T = BFS tree T'.

- G is an undirected graph $\implies G = T$
- G is a digraph $\stackrel{?}{\implies} G = T$

- T: tree + back *vs.* T': tree + cross
- T: tree + back + forward + cross *vs.* T': tree + back + cross

Lifttime of vertices in DFS

Theorem (Disjoint or contained)

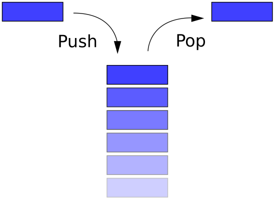
$$\forall u, v :$$
$$[_u\ ]_u \cap [_v\ ]_v = \emptyset$$
$$\bigvee$$
$$([_u\ ]_u \subsetneqq [_v\ ]_v \vee [_v\ ]_v \subsetneqq [_u\ ]_u)$$

Lifttime of vertices in DFS

Theorem (Disjoint or contained)

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

Proof.



Ancestor/descendant relation

Preprocessing for ancestor/descendant relation (Problem 5.23)

- binary tree $T = (V, E)$ (tree)
- $r \in V$

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

Ancestor/descendant relation

Preprocessing for ancestor/descendant relation (Problem 5.23)

- binary tree $T = (V, E)$ (tree)
- $r \in V$

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

Question

$\forall v$: how many descendants?

Ancestor/descendant relation

Preprocessing for ancestor/descendant relation (Problem 5.23)

- binary tree $T = (V, E)$ (tree)
- $r \in V$

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

Question

$\forall v$: how many descendants?

$$(\mathsf{f}[v] - \mathsf{d}[v] - 1)/2$$

Edge types and lifetime of vertices in DFS

Edge types and lifetime of vertices in DFS (Problem 5.2)

$\forall u \to v$:

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

Edge types and lifetime of vertices in DFS

Edge types and lifetime of vertices in DFS (Problem 5.2)

$\forall u \to v$:

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

Remark

- f$[v] <$ d$[u]$: cross edge
- f$[u] <$ f$[v]$: back edge

Edge types and lifetime of vertices in DFS

Edge types and lifetime of vertices in DFS (Problem 5.2)

$\forall u \to v$:

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

Remark

- $\mathsf{f}[v] < \mathsf{d}[u]$: cross edge
- $\mathsf{f}[u] < \mathsf{f}[v]$: back edge

$$u \to v \iff \mathsf{f}[v] < \mathsf{f}[u]$$

Height and diameter of tree

Height and diameter of tree (Problem 5.21)

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

- height ($O(n)$)
- diameter ($O(n)$)

Question

Diameter of a tree *without* 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)

Decompositions of Graphs

Cycle detection

Cycle detection (Problem 5.24)

	Digraph	Undirected graph
DFS		
BFS		

Cycle detection

Cycle detection (Problem 5.24)

	Digraph	Undirected graph
DFS	back edge $\iff$ cycle	
BFS		

Cycle detection

Cycle detection (Problem 5.24)

	Digraph	Undirected graph
DFS	back edge $\iff$ cycle	back edge $\iff$ cycle
BFS		

Cycle detection

Cycle detection (Problem 5.24)

	Digraph	Undirected graph
DFS	back edge $\iff$ cycle	back edge $\iff$ cycle
BFS		cross edge $\iff$ cycle

Cycle detection

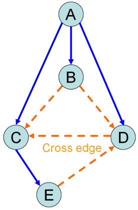
Cycle detection (Problem 5.24)

	Digraph	Undirected graph
DFS	back edge $\iff$ cycle	back edge $\iff$ cycle
BFS	back edge $\implies$ cycle cycle $\not\Longrightarrow$ back edge	cross edge $\iff$ cycle

Cycle detection

Cycle detection (Problem 5.24)

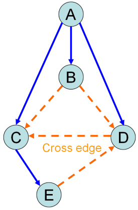
	Digraph	Undirected graph
DFS	back edge $\iff$ cycle	back edge $\iff$ cycle
BFS	back edge $\implies$ cycle cycle $\not\Longrightarrow$ back edge	cross edge $\iff$ cycle



Cycle detection

Cycle detection (Problem 5.24)

	Digraph	Undirected graph
DFS	back edge $\iff$ cycle	back edge $\iff$ cycle
BFS	back edge $\implies$ cycle cycle $\not\implies$ back edge	cross edge $\iff$ cycle



Remark

How to identify back edges?

Edge deletion

Edge deletion (Problem 5.20)

- connected, undirected graph G
- $\exists ? e \in E : G \setminus e$ is connected?
- $O(|V|)$

Edge deletion

Edge deletion (Problem 5.20)

- connected, undirected graph G
- $\exists ? e \in E : G \setminus e$ is connected?
- $O(|V|)$

$$\exists \text{ cycle} \iff \exists \text{ such } e$$

Edge deletion

Edge deletion (Problem 5.20)

- connected, undirected graph G
- $\exists ? e \in E : G \setminus e$ is connected?
- $O(|V|)$

$$\exists \text{ cycle} \iff \exists \text{ such } e$$

$$\text{tree: } |E| = |V| - 1 \implies \text{check } |E| \geq |V|$$

Orientation of undirected graph

Orientation of undirected graph (Problem 5.9)

- undirected (connected) graph G
- edges oriented *s.t.*

$$\forall v, \textsf{in}[v] \geq 1$$

Orientation of undirected graph

Orientation of undirected graph (Problem 5.9)

- undirected (connected) graph G
- edges oriented *s.t.*

$$\forall v, \mathsf{in}[v] \geq 1$$

orientation $\iff \exists$ cycle C

Orientation of undirected graph

Orientation of undirected graph (Problem 5.9)

- undirected (connected) graph G
- edges oriented *s.t.*

$$\forall v, \text{in}[v] \geq 1$$

$$\text{orientation} \iff \exists \text{ cycle } C$$

$$\text{BFS/DFS from } v \in C$$

Shortest cycle of undirected graph

Shortest cycle of undirected graph (Problem 5.8)

Shortest cycle of G:

- DFS on G
- $\forall v$: level$[v]$
- back edge $u \to v$: level$[u] -$ level$[v] + 1$

Shortest cycle of undirected graph

Shortest cycle of undirected graph (Problem 5.8)

Shortest cycle of G:

- DFS on G
- $\forall v$: level$[v]$
- back edge $u \to v$: level$[u]$ $-$ level$[v] + 1$

Question

What about digraphs?

Decompositions of Graphs

Decompositions of Graphs

Decompositions of Graphs


THANK YOU