Analysis of Algorithms



LECTURES 20-21

Greedy Algorithms

- Graphs
- Minimum spanning trees
- Optimal substructure
- Greedy choice
- Prim's greedy MST algorithm

February 19, 2009

nary 19, 2009



Definition. A *directed graph* (*digraph*) G = (V, E) is an ordered pair consisting of

- a set V of vertices (singular: vertex),
- a set $E \subseteq V \times V$ of *edges*.

In an *undirected graph* G = (V, E), the edge set E consists of *unordered* pairs of vertices.

In either case, we have $|E| = O(V^2)$. Moreover, if *G* is connected, then $|E| \ge |V| - 1$, which implies that $\lg |E| = \Theta(\lg V)$.

(Review CLRS, Appendix B.)

February 19, 2009

Adjacency-matrix representation

The *adjacency matrix* of a graph G = (V, E), where $V = \{1, 2, ..., n\}$, is the matrix A[1 ... n, 1 ... n] given by

 $A[i,j] = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{if } (i,j) \notin E. \end{cases}$

February 19, 2009

ALGORITHMS

Adjacency-matrix representation

The *adjacency matrix* of a graph G = (V, E), where $V = \{1, 2, ..., n\}$, is the matrix A[1 ... n, 1 ... n] given by

$$A[i,j] = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{if } (i,j) \notin E. \end{cases}$$



	4	2	2	l 1	4
				1	
$\Theta(V^2)$ storage	0	1	1	0 0 0 0	1
\Rightarrow dense	0	1	0	0	2
representation.	0	0	0	0	3
	0	1	0	0	4

February 19, 2009

ALGORITHM

Adjacency-list representation

An *adjacency list* of a vertex $v \in V$ is the list Adj[v] of vertices adjacent to v.



 $Adj[1] = \{2, 3\}$ $Adj[2] = \{3\}$ $Adj[3] = \{\}$ $Adj[4] = \{3\}$

February 19, 2009

ALGORITHMS

Adjacency-list representation

An *adjacency list* of a vertex $v \in V$ is the list Adj[v] of vertices adjacent to v.



 $Adj[1] = \{2, 3\}$ $Adj[2] = \{3\}$ $Adj[3] = \{\}$ $Adj[4] = \{3\}$

For undirected graphs, |Adj[v]| = degree(v). For digraphs, |Adj[v]| = out-degree(v).

February 19, 2009



Adjacency-list representation

An *adjacency list* of a vertex $v \in V$ is the list Adj[v] of vertices adjacent to v.



 $Adj[1] = \{2, 3\}$ $Adj[2] = \{3\}$

 $Adj[3] = \{\}$

 $Adi[4] = \{3\}$

For undirected graphs, |Adj[v]| = degree(v). For digraphs, |Adj[v]| = out-degree(v).

Handshaking Lemma: $\sum_{v \in V} degree(v) = 2 |E|$ for undirected graphs \Rightarrow adjacency lists use $\Theta(V + E)$ storage — a *sparse* representation.

February 19, 2009

ALGORITHMS

February 19, 2009

Minimum spanning trees

Input: A connected, undirected graph G = (V, E) with weight function $w : E \to \mathbb{R}$.

• For simplicity, assume that all edge weights are distinct. (CLRS covers the general case.)



February 19, 2009

Minimum spanning trees

Input: A connected, undirected graph G = (V, E) with weight function $w : E \to \mathbb{R}$.

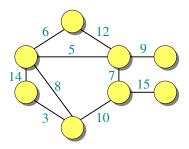
• For simplicity, assume that all edge weights are distinct. (CLRS covers the general case.)

Output: A *spanning tree T* — a tree that connects all vertices — of minimum weight:

$$w(T) = \sum_{(u,v)\in T} w(u,v).$$

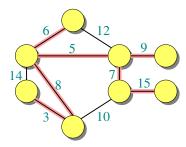
8



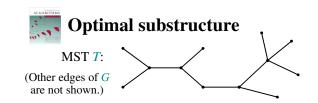


February 19, 2009 10

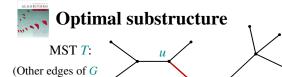




February 19, 2009

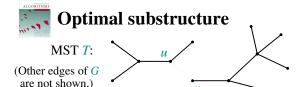


February 19, 2009

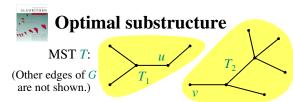


Remove any edge $(u, v) \in T$.

are not shown.)



Remove any edge $(u, v) \in T$.



Remove any edge $(u, v) \in T$. Then, T is partitioned into two subtrees T_1 and T_2 .

February 19, 2009

February 19, 2009 14

February 19, 2009

Optimal substructure MST T: (Other edges of G are not shown.)

Remove any edge $(u, v) \in T$. Then, T is partitioned into two subtrees T_1 and T_2 .

Theorem. The subtree T_1 is an MST of $G_1 = (V_1, E_1)$, the subgraph of G induced by the vertices of T_1 :

$$V_1 = \text{vertices of } T_1,$$

 $E_1 = \{ (x, y) \in E : x, y \in V_1 \}.$

Similarly for T_2 .



Proof of optimal substructure

Proof. Cut and paste:

$$w(T) = w(u, v) + w(T_1) + w(T_2).$$

If T_1' were a lower-weight spanning tree than T_1 for G_1 , then $T' = \{(u, v)\} \cup T_1' \cup T_2$ would be a lower-weight spanning tree than T for G.



Proof of optimal substructure

Proof. Cut and paste:

$$w(T) = w(u, v) + w(T_1) + w(T_2).$$

If T_1' were a lower-weight spanning tree than T_1 for G_1 , then $T' = \{(u, v)\} \cup T_1' \cup T_2$ would be a lower-weight spanning tree than T for G.

Do we also have overlapping subproblems? • Yes.



Proof of optimal substructure

Proof. Cut and paste:

$$w(T) = w(u, v) + w(T_1) + w(T_2).$$

If T_1 were a lower-weight spanning tree than T_1 for G_1 , then $T' = \{(u, v)\} \cup T_1' \cup T_2$ would be a lower-weight spanning tree than T for G.

Do we also have overlapping subproblems?

Yes.

Great, then dynamic programming may work!

• Yes, but MST exhibits another powerful property which leads to an even more efficient algorithm.

February 19, 2009

Hallmark for "greedy" algorithms

Greedy-choice property A locally optimal choice is globally optimal.

February 19, 2009

Hallmark for "greedy" algorithms

Greedy-choice property A locally optimal choice is globally optimal.

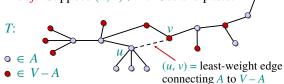
Theorem. Let T be the MST of G = (V, E). and let $A \subseteq V$. Suppose that $(u, v) \in E$ is the least-weight edge connecting A to V-A. Then, $(u, v) \in T$.

February 19, 2009 21



Proof of theorem

Proof. Suppose $(u, v) \notin T$. Cut and paste.

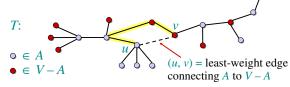


February 19, 2009

22

Proof of theorem

Proof. Suppose $(u, v) \notin T$. Cut and paste.

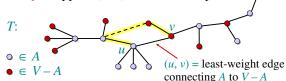


Consider the unique simple path from u to v in T.



Proof of theorem

Proof. Suppose $(u, v) \notin T$. Cut and paste.



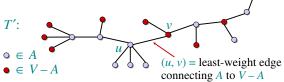
Consider the unique simple path from u to v in T. Swap (u, v) with the first edge on this path that connects a vertex in A to a vertex in V-A.

February 19, 2009



Proof of theorem

Proof. Suppose $(u, v) \notin T$. Cut and paste.



Consider the unique simple path from u to v in T.

Swap (u, v) with the first edge on this path that connects a vertex in A to a vertex in V-A.

A lighter-weight spanning tree than *T* results.

February 19, 2009

February 19, 2009

February 19, 2009

Prim's algorithm

IDEA: Maintain V - A as a priority queue Q. Key each vertex in Q with the weight of the leastweight edge connecting it to a vertex in A.

$$Q \leftarrow V$$

$$key[v] \leftarrow \infty \text{ for all } v \in V$$

$$key[s] \leftarrow 0 \text{ for some arbitrary } s \in V$$

$$\mathbf{while } Q \neq \emptyset$$

$$\mathbf{do } u \leftarrow \text{EXTRACT-MIN}(Q)$$

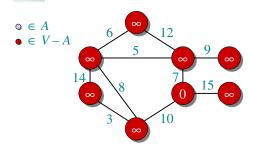
$$\mathbf{for } \text{ each } v \in Adj[u]$$

$$\mathbf{do } \mathbf{if } v \in Q \text{ and } w(u,v) < key[v]$$

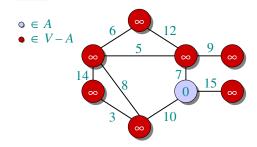
$$\mathbf{then } key[v] \leftarrow w(u,v) \qquad \triangleright \text{ DECREASE-KEY}$$

$$\pi[v] \leftarrow u$$
At the end, $\{(v,\pi[v])\}$ forms the MST.

Example of Prim's algorithm

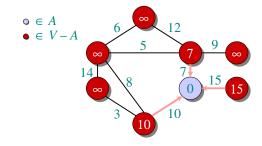


Example of Prim's algorithm



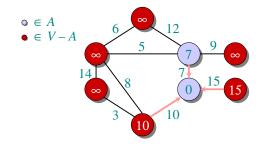
February 19, 2009





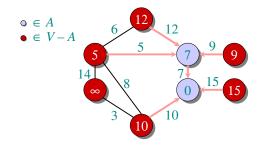
February 19, 2009

Example of Prim's algorithm



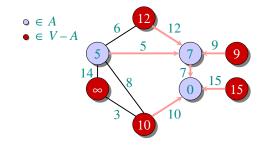
February 19, 2009 3

Example of Prim's algorithm



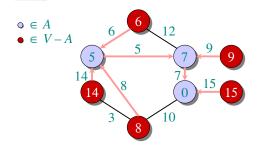
February 19, 2009 3

Example of Prim's algorithm



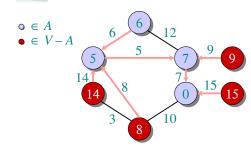
February 19, 2009

Example of Prim's algorithm

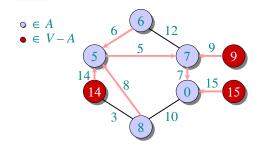


February 19, 2009 3

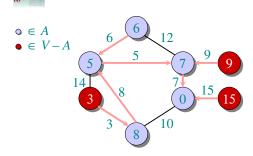
Example of Prim's algorithm



Example of Prim's algorithm



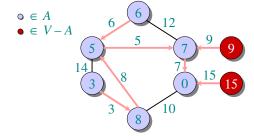
Example of Prim's algorithm



bruary 19, 2009 35 February 19, 2009



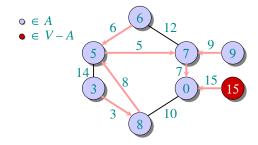
Example of Prim's algorithm



February 19, 2009

37

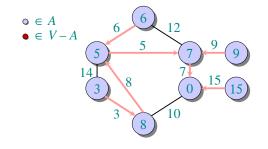
Example of Prim's algorithm



February 19, 2009

ALGORITHMS

Example of Prim's algorithm



February 19, 2009 39



Analysis of Prim

```
\begin{aligned} Q &\leftarrow V \\ key[v] &\leftarrow \infty \text{ for all } v \in V \\ key[s] &\leftarrow 0 \text{ for some arbitrary } s \in V \\ \textbf{while } Q \neq \varnothing \\ \textbf{do } u &\leftarrow \text{EXTRACT-MIN}(Q) \\ \textbf{for each } v \in Adj[u] \\ \textbf{do if } v \in Q \text{ and } w(u,v) < key[v] \\ \textbf{then } key[v] \leftarrow w(u,v) \\ \pi[v] \leftarrow u \end{aligned}
```

February 19, 2009

40

43

ALGORITHMS

Analysis of Prim

```
\Theta(V) \begin{cases} Q \leftarrow V \\ key[v] \leftarrow \infty \text{ for all } v \in V \\ key[s] \leftarrow 0 \text{ for some arbitrary } s \in V \end{cases}
\mathbf{vhile} \ Q \neq \emptyset
\mathbf{do} \ u \leftarrow \text{EXTRACT-MIN}(Q)
\mathbf{for} \ \text{each} \ v \in Adj[u]
\mathbf{do} \ \mathbf{if} \ v \in Q \ \text{and} \ w(u,v) < key[v]
\mathbf{then} \ key[v] \leftarrow w(u,v)
\pi[v] \leftarrow u
```

February 19, 2009

February 19, 2009



Analysis of Prim

```
 \Theta(V) \  \  \, \begin{cases} Q \leftarrow V \\ key[v] \leftarrow \infty \text{ for all } v \in V \\ key[s] \leftarrow 0 \text{ for some arbitrary } s \in V \end{cases}  while Q \neq \emptyset  \text{do } u \leftarrow \text{EXTRACT-MIN}(Q)   \text{for each } v \in Adj[u]   \text{do if } v \in Q \text{ and } w(u,v) < key[v]   \text{then } key[v] \leftarrow w(u,v)   \pi[v] \leftarrow u
```

February 19, 2009 42



Analysis of Prim

```
 \Theta(V) \  \, \text{total} \left\{ \begin{array}{l} Q \leftarrow V \\ key[v] \leftarrow \infty \text{ for all } v \in V \\ key[s] \leftarrow 0 \text{ for some arbitrary } s \in V \end{array} \right. \\ \text{while } Q \neq \varnothing \\ \text{do } u \leftarrow \text{EXTRACT-MIN}(Q) \\ \text{do if } v \in Q \text{ and } w(u,v) < key[v] \\ \text{times} \left\{ \begin{array}{l} \text{do if } v \in Q \text{ and } w(u,v) \\ \text{then } key[v] \leftarrow w(u,v) \\ \pi[v] \leftarrow u \end{array} \right.
```

ALGORITHMS

Analysis of Prim

```
 \begin{array}{l} \Theta(V) \\ \text{total} \end{array} \begin{cases} \begin{array}{l} Q \leftarrow V \\ key[v] \leftarrow \infty \text{ for all } v \in V \\ key[s] \leftarrow 0 \text{ for some arbitrary } s \in V \end{array} \\ \text{while } Q \neq \emptyset \\ \text{do } u \leftarrow \text{EXTRACT-MIN}(Q) \\ \text{do if } v \in Q \text{ and } w(u,v) < key[v] \\ \text{times} \end{array} \end{cases}
```

Handshaking Lemma $\Rightarrow \Theta(E)$ implicit Decrease-Key's.



Analysis of Prim

```
\Theta(V) \begin{tabular}{l} $Q \leftarrow V \\ $key[v] \leftarrow \infty$ for all $v \in V$ \\ $key[s] \leftarrow 0$ for some arbitrary $s \in V$ \\ \hline \begin{tabular}{l} while $Q \neq \varnothing$ \\ $do u \leftarrow \text{EXTRACT-MIN}(Q)$ \\ \hline times \\ \hline \begin{tabular}{l} degree(u) \\ times \\ \hline \end{tabular} \begin{tabular}{l} do \ if $v \in Q$ and $w(u,v) < key[v]$ \\ \hline \end{tabular} \begin{tabular}{l} then $key[v] \leftarrow w(u,v)$ \\ \hline \end{tabular} \begin{tabular}{l} \pi[v] \leftarrow u \\ \hline \end{tabular}
```

Handshaking Lemma $\Rightarrow \Theta(E)$ implicit Decrease-Key's.

$$Time = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$$

44 February 19, 2009 45



Time = $\Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$



Time = $\Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$

 $Q = T_{\text{EXTRACT-MIN}} T_{\text{DECREASE-KEY}}$ Total

Analysis of Prim (continued)

Time = $\Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$

Q	$T_{ m EXTRACT-MIN}$	$T_{ m DECREASE-KEY}$	Total
array	O(V)	<i>O</i> (1)	$O(V^2)$

February 19, 2009 47 February 19, 2009 48



February 19, 2009

Analysis of Prim (continued)

Time = $\Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$

Q	$T_{\rm EXTRACT ext{-}MIN}$	$T_{ m DECREASE-KEY}$	Total
array	O(V)	<i>O</i> (1)	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$

February 19, 2009 49



Solution Analysis of Prim (continued)

Time = $\Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$

Q	T _{EXTRACT-MIN}	T _{DECREASE-KE}	Y Total
array	O(V)	<i>O</i> (1)	$O(V^2)$
oinary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
bonacci heap	$O(\lg V)$ amortized	O(1) amortized	$O(E + V \lg V)$ worst case

February 19, 2009



MST algorithms

Kruskal's algorithm (see CLRS):

- Uses the *disjoint-set data structure* (see CLRS, Ch. 21).
- Running time = $O(E \lg V)$.

February 19, 2009 51



MST algorithms

Kruskal's algorithm (see CLRS):

- Uses the *disjoint-set data structure* (see CLRS, Ch. 21).
- Running time = $O(E \lg V)$.

Best to date:

- Karger, Klein, and Tarjan [1993].
- Randomized algorithm.
- O(V + E) expected time.

February 19, 2009 52