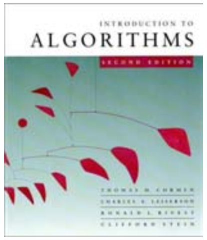
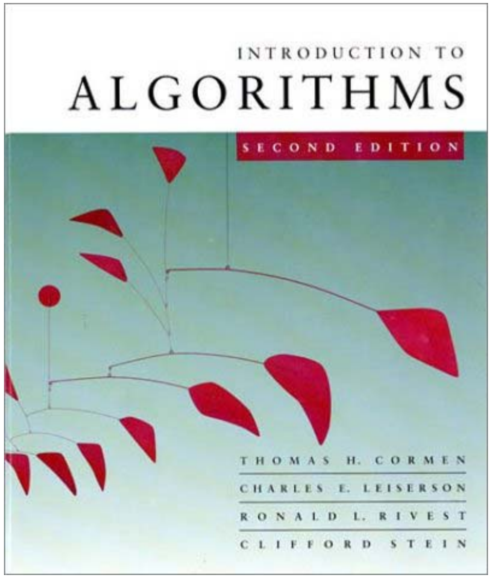


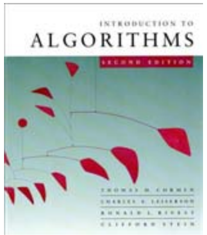
CS 5633 -- Spring 2004



Minimum Spanning Trees

Carola Wenk

Slides courtesy of Charles Leiserson with small changes by Carola Wenk



Graphs (review)

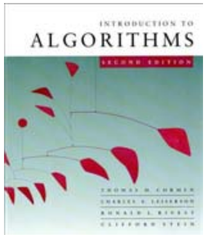
Definition. A ***directed graph (digraph)*** $G = (V, E)$ is an ordered pair consisting of

- 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 $|E| \geq |V| - 1$.

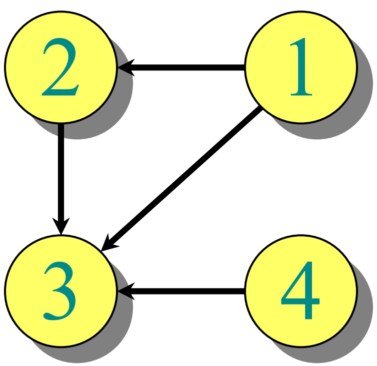
(Review CLRS, Appendix B.4 and B.5.)



Adjacency-matrix representation

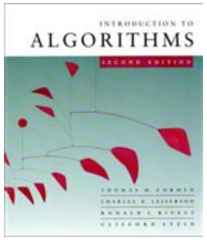
The ***adjacency matrix*** of a graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$, is the matrix $A[1 \,.\,.\, n, 1 \,.\,.\, n]$ given by

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



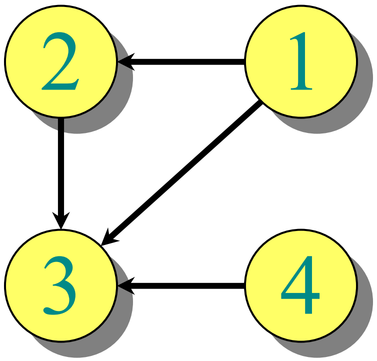
A	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	0	0	0	0
4	0	0	1	0

$\Theta(|V|^2)$ storage $\Rightarrow$ ***dense*** representation.



Adjacency-list representation

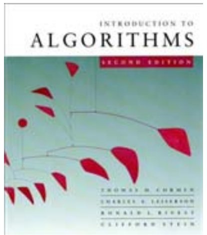
An ***adjacency list*** of a vertex $v \in V$ is the list $Adj[v]$ of vertices adjacent to v.



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

For undirected graphs, $|Adj[v]| = degree(v)$.

For digraphs, $| Adj[v] | = out\text{-}degree(v)$.



Adjacency-list representation

Handshaking Lemma:

- For undirected graphs:

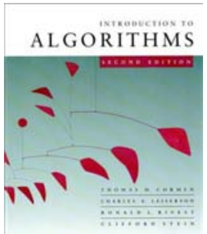
  $\sum_{v \in V} \mathit{degree}(v) = 2|E|$

- For digraphs:

  $\sum_{v \in V} \mathit{in\text{-}degree}(v) + \sum_{v \in V} \mathit{out\text{-}degree}(v) = 2 \,|\, E \,|$

$\Rightarrow$ adjacency lists use $\Theta(|V| + |E|)$ storage

$\Rightarrow$ a ***sparse*** representation



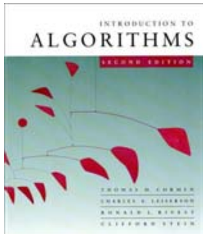
Minimum spanning trees

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

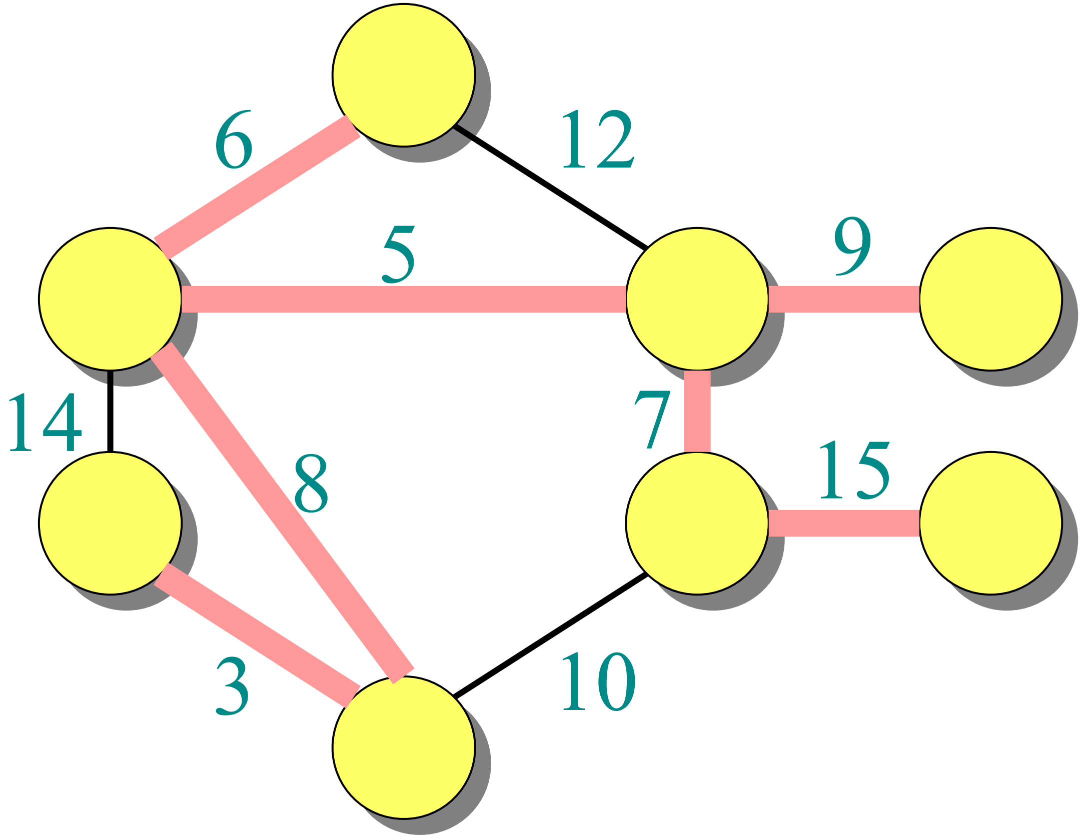
- For simplicity, assume that all edge weights are distinct. (CLRS covers the general case.)

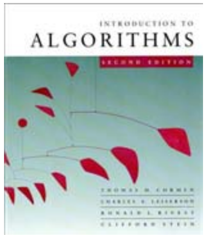
Output: A ***spanning tree*** T — a tree that connects all vertices — of minimum weight:

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



Example of MST



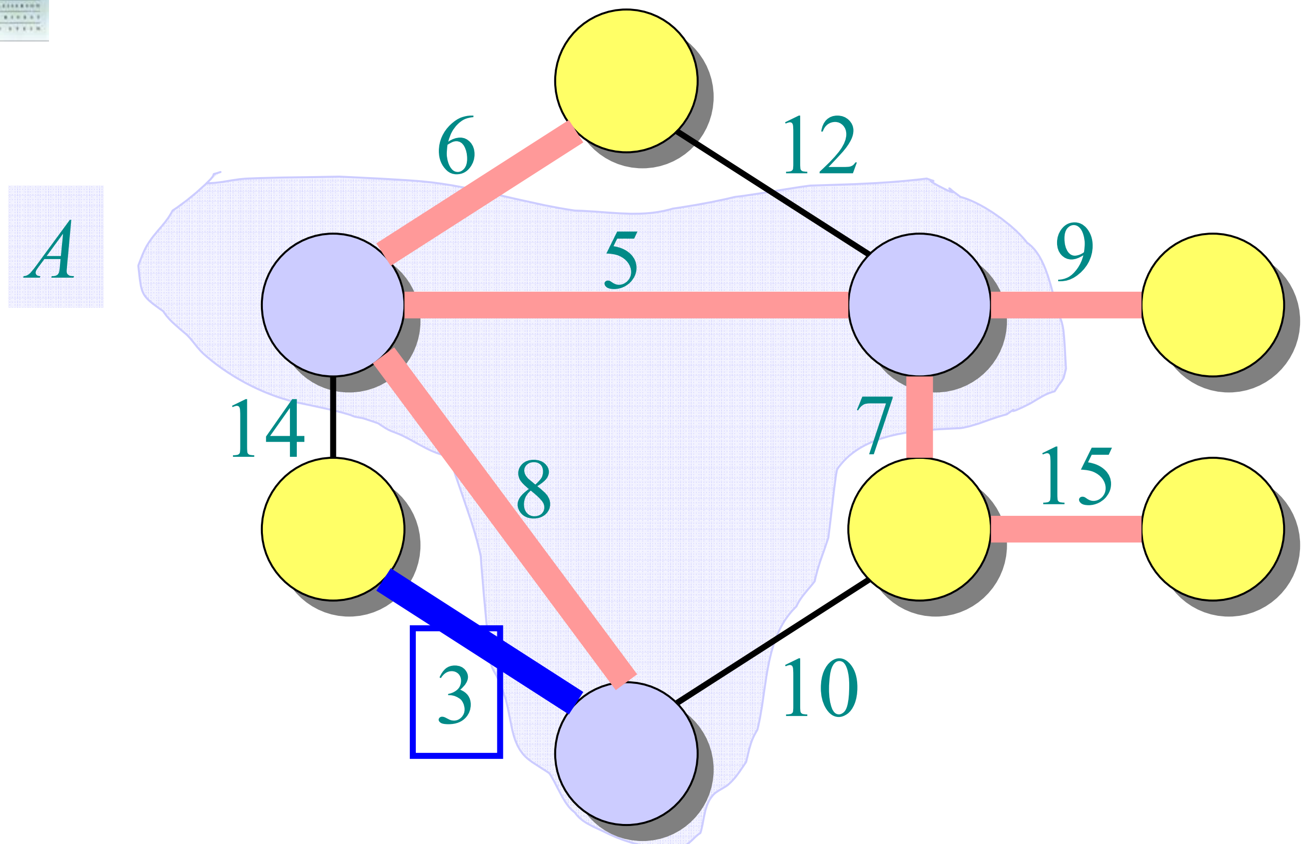


Hallmark for "greedy" algorithms

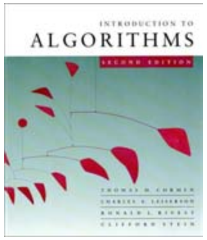
Greedy-choice property
A locally optimal choice is globally optimal.

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 $V \setminus A$. Then, $(u, v) \in T$.

Example of MST



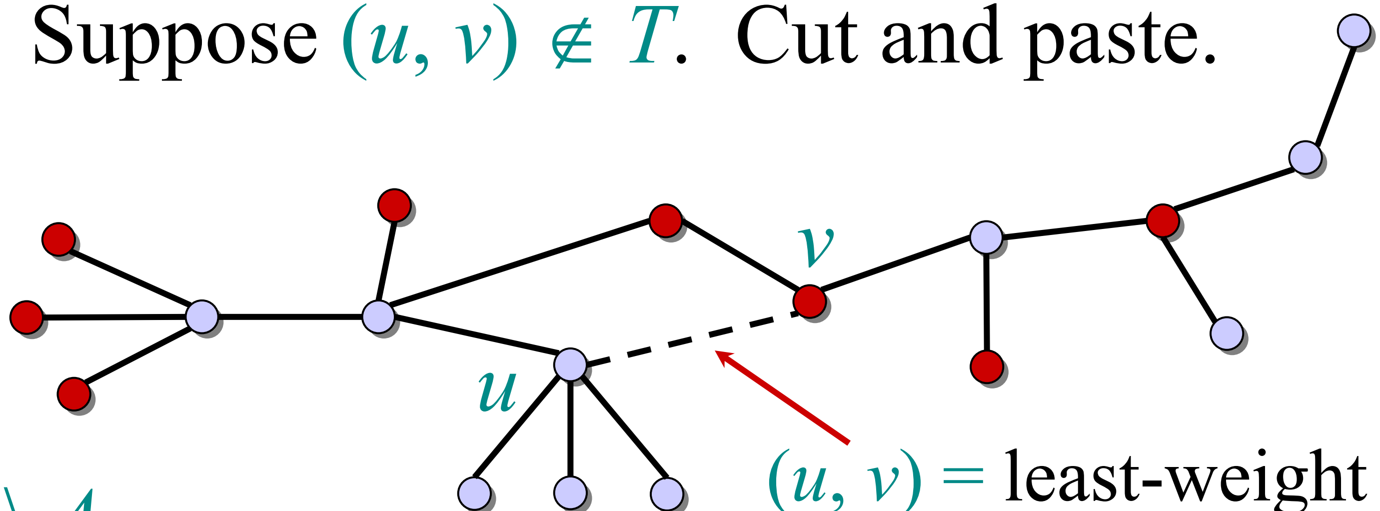
Theorem. Let T be the MST of $G = (V, E)$, and let $A \subseteq V$. Suppose that $(u, v) \in E$ is the least-weight edge connecting A to $V \setminus A$. Then, $(u, v) \in T$.



Proof of theorem

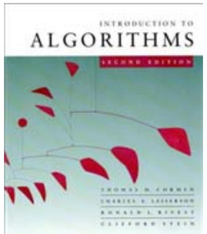
Proof. Suppose $(u, v) \notin T$. Cut and paste.

T:



(u, v) = least-weight edge connecting A to $V \setminus A$

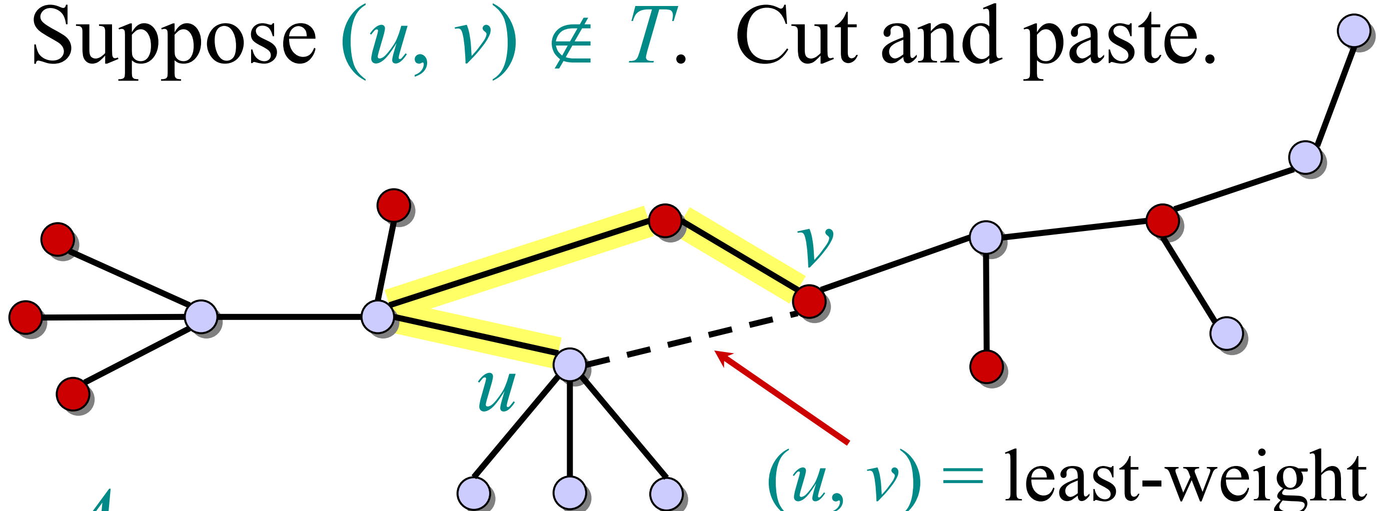
- $\in A$
- $\in V \setminus A$



Proof of theorem

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

T:



● $\in A$

● $\in V - A$

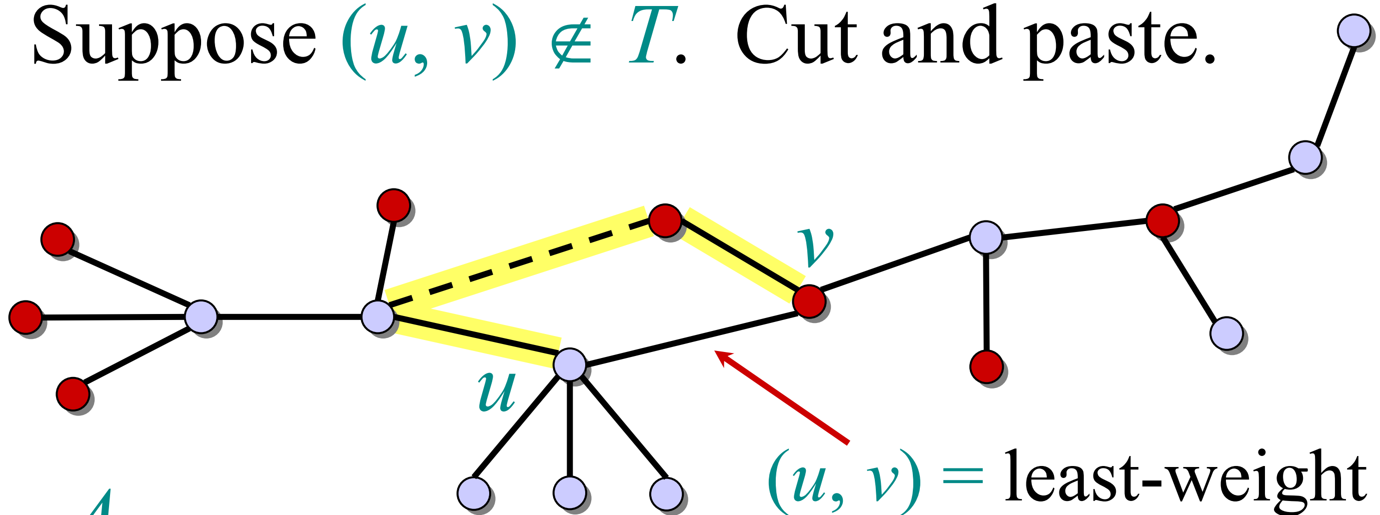
(u, v) = least-weight edge connecting A to $V - A$

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

Proof of theorem

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

T:



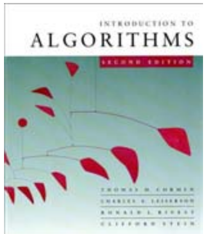
$\in A$

$\in V - A$

(u, v) = least-weight edge connecting A to $V - A$

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

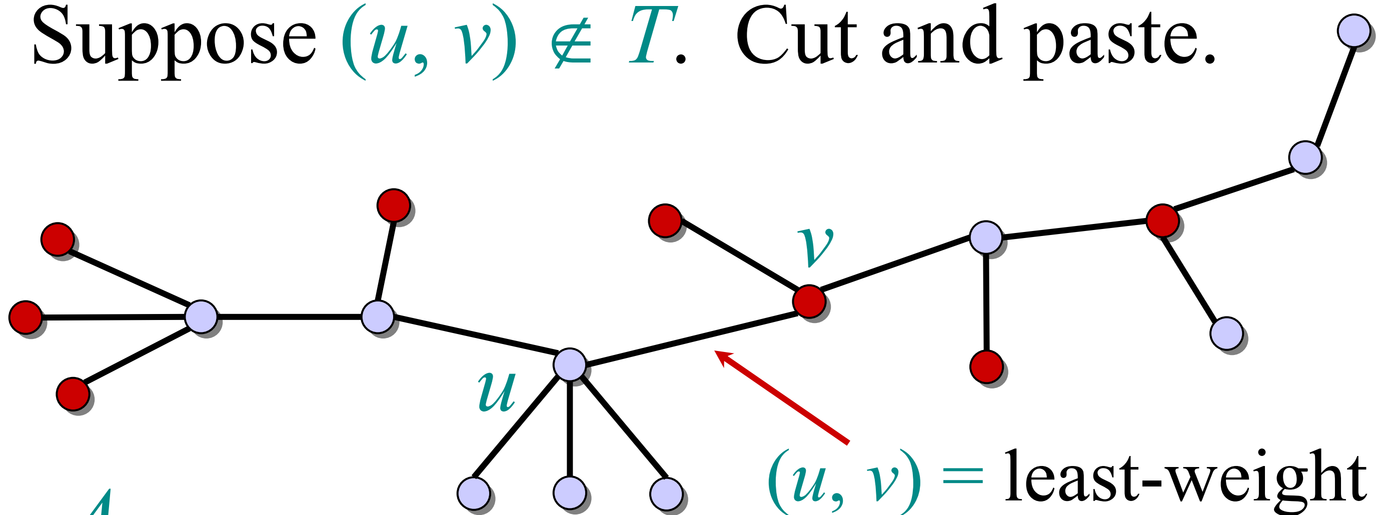
Swap (u, v) with the first edge on this path that connects a vertex in A to a vertex in $V \setminus A$.



Proof of theorem

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

T':

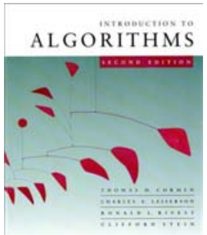


(u, v) = least-weight edge connecting A to $V - A$

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. □

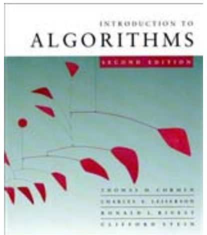


Prim's algorithm

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

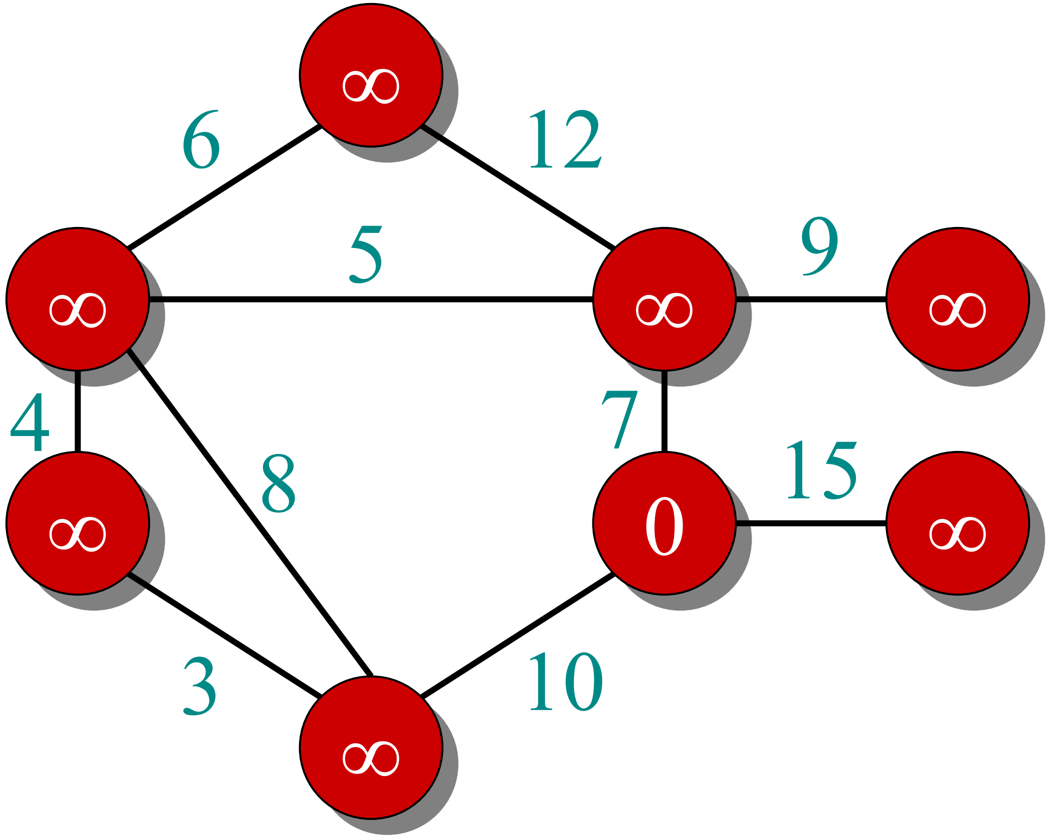
$Q \leftarrow V$
$key[v] \leftarrow \infty$ for all $v \in V$
$key[s] \leftarrow 0$ for some arbitrary $s \in V$
while $Q \neq \emptyset$
    do $u \leftarrow$ EXTRACT-MIN(Q)
        for each $v \in Adj[u]$
            do if $v \in Q$ and $w(u, v) < key[v]$
                then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
                    $\pi[v] \leftarrow u$

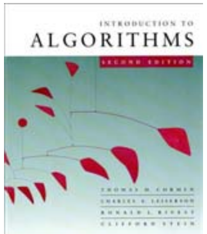
At the end, $\{(v, \pi[v])\}$ forms the MST.



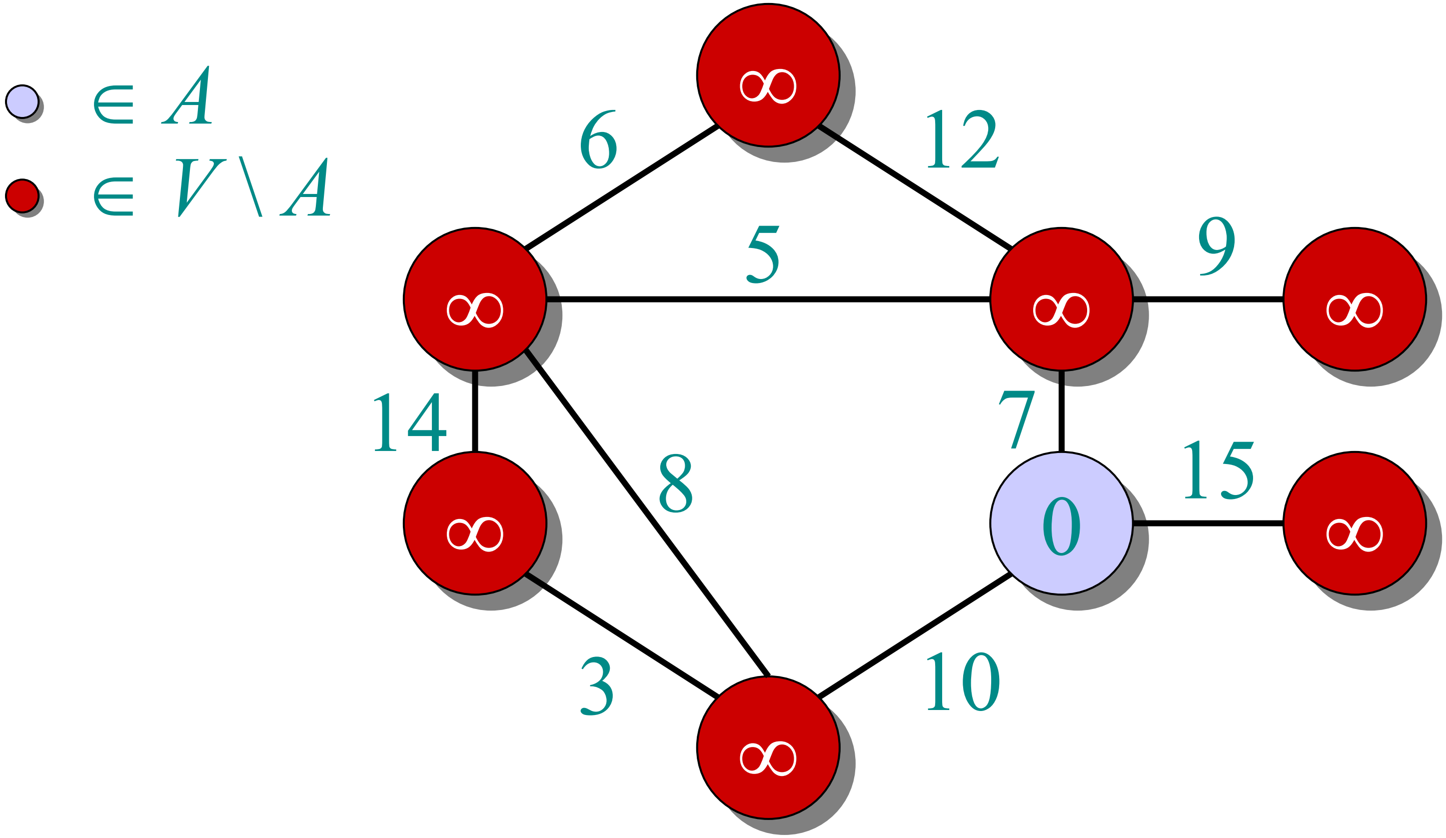
Example of Prim's algorithm

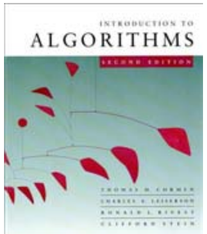
- $\in A$
- $\in V \setminus A$





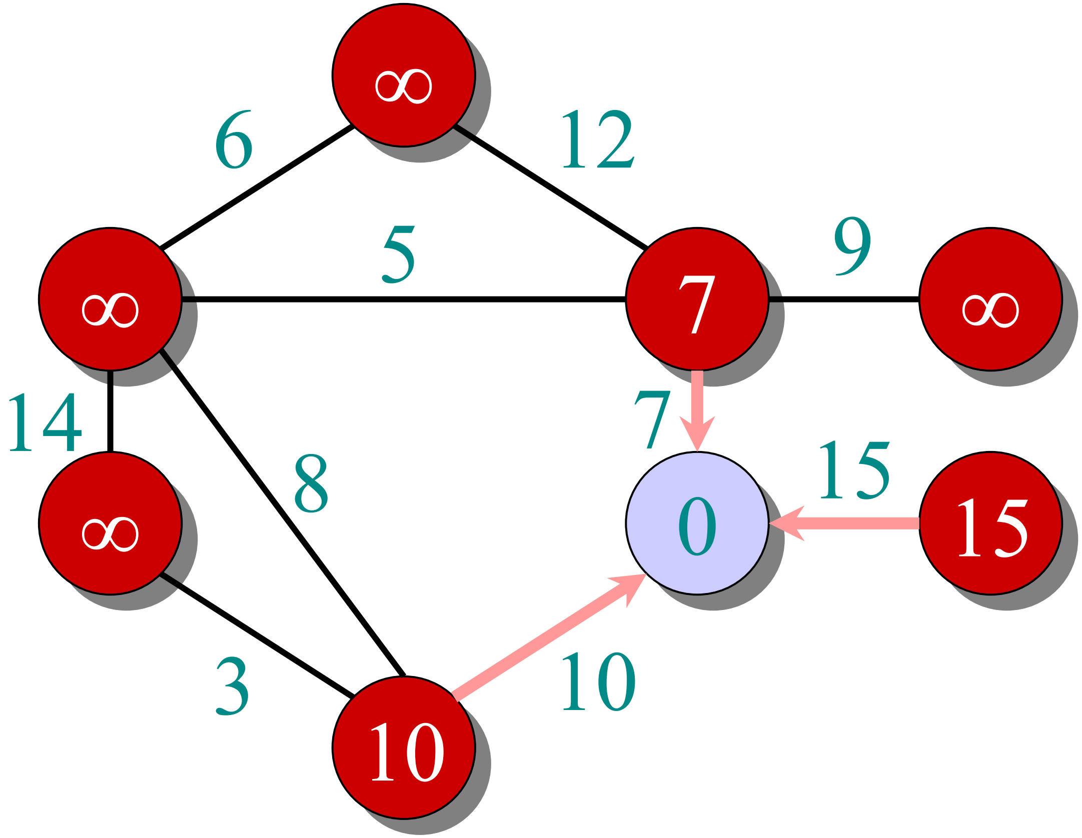
Example of Prim's algorithm

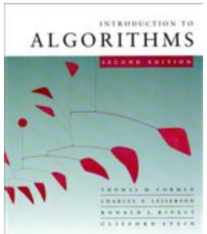




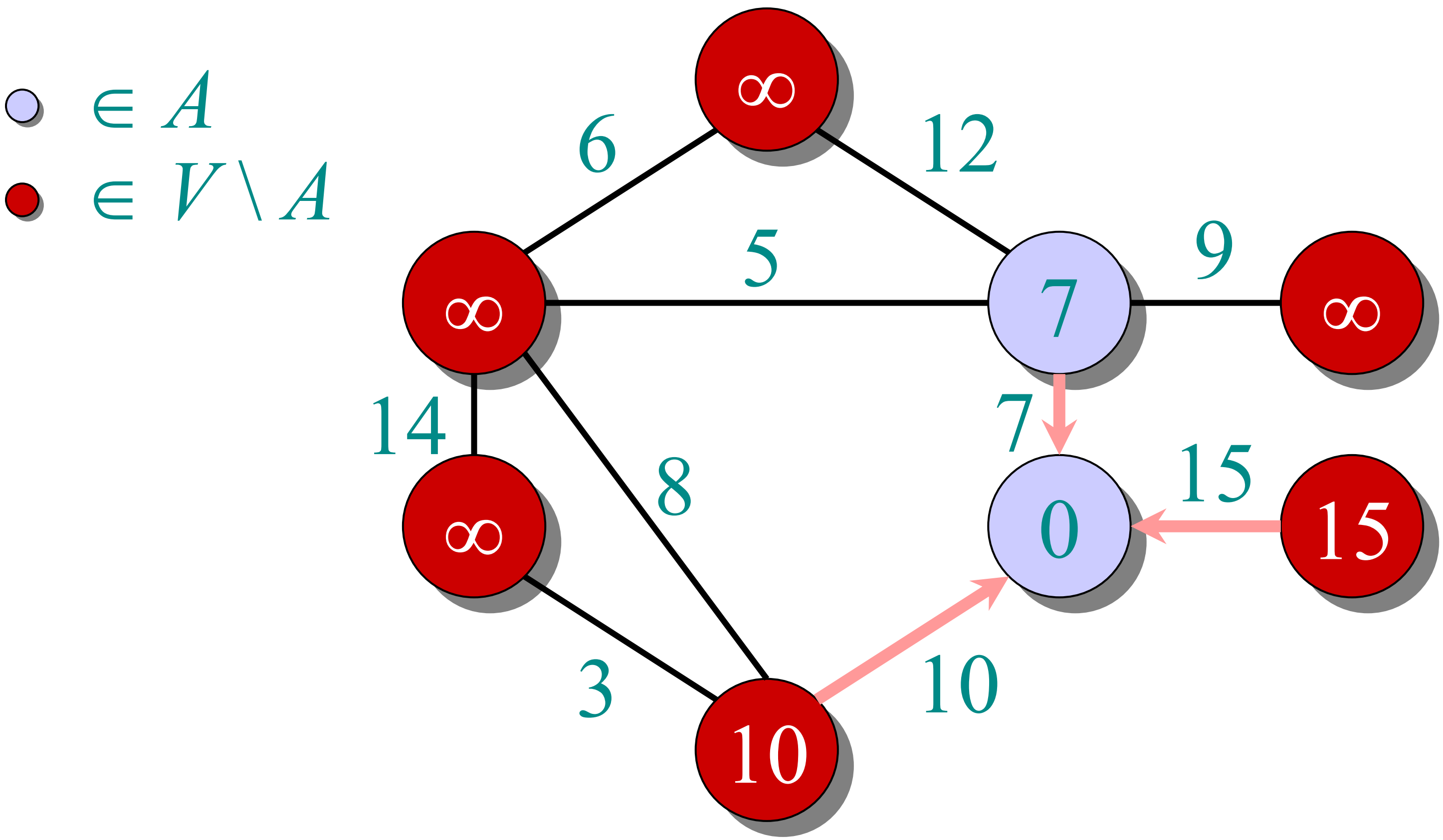
Example of Prim's algorithm

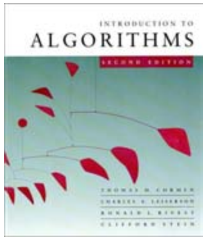
- $\in A$
- $\in V \setminus A$



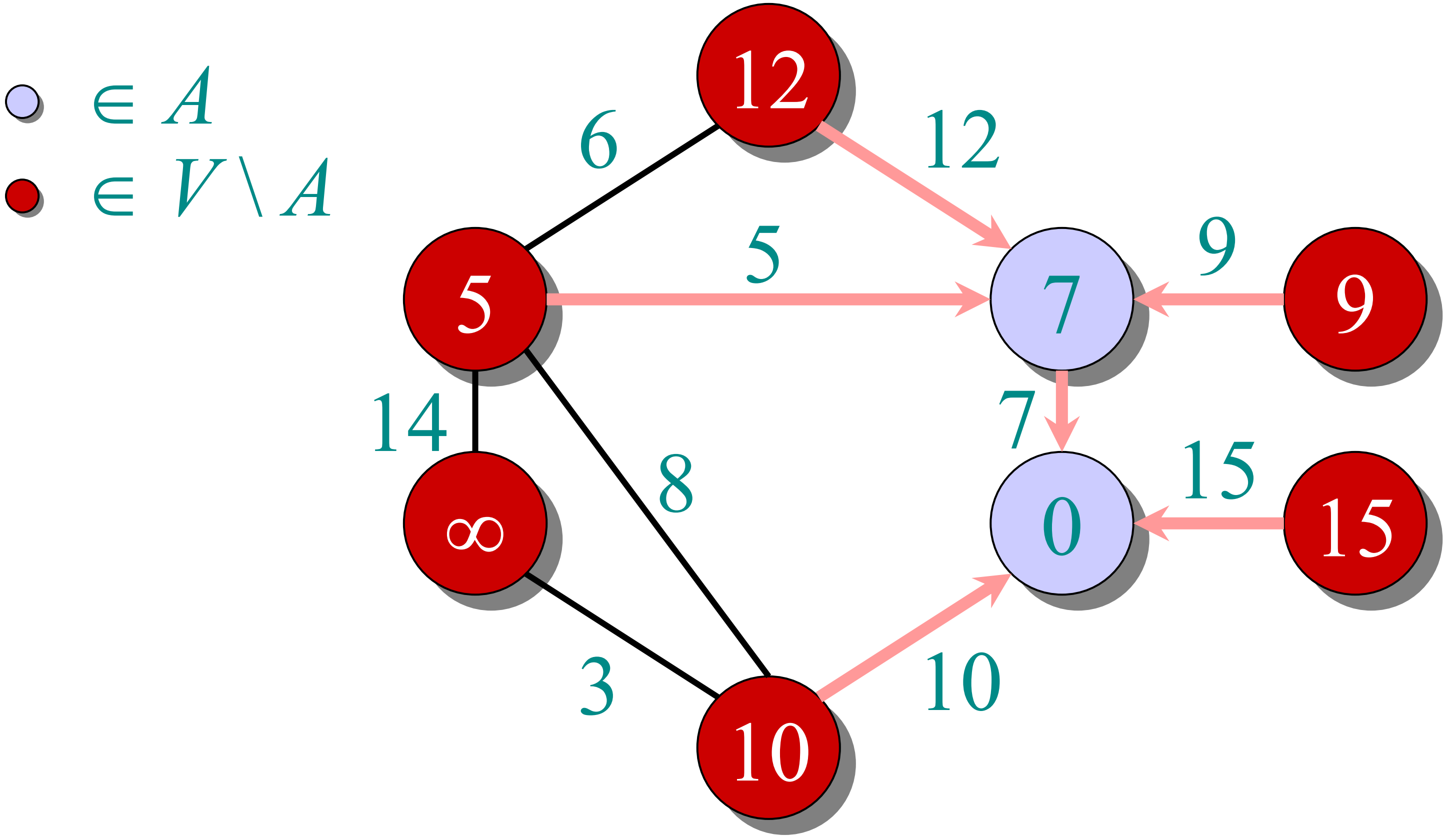


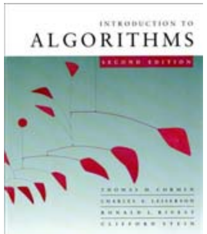
Example of Prim's algorithm





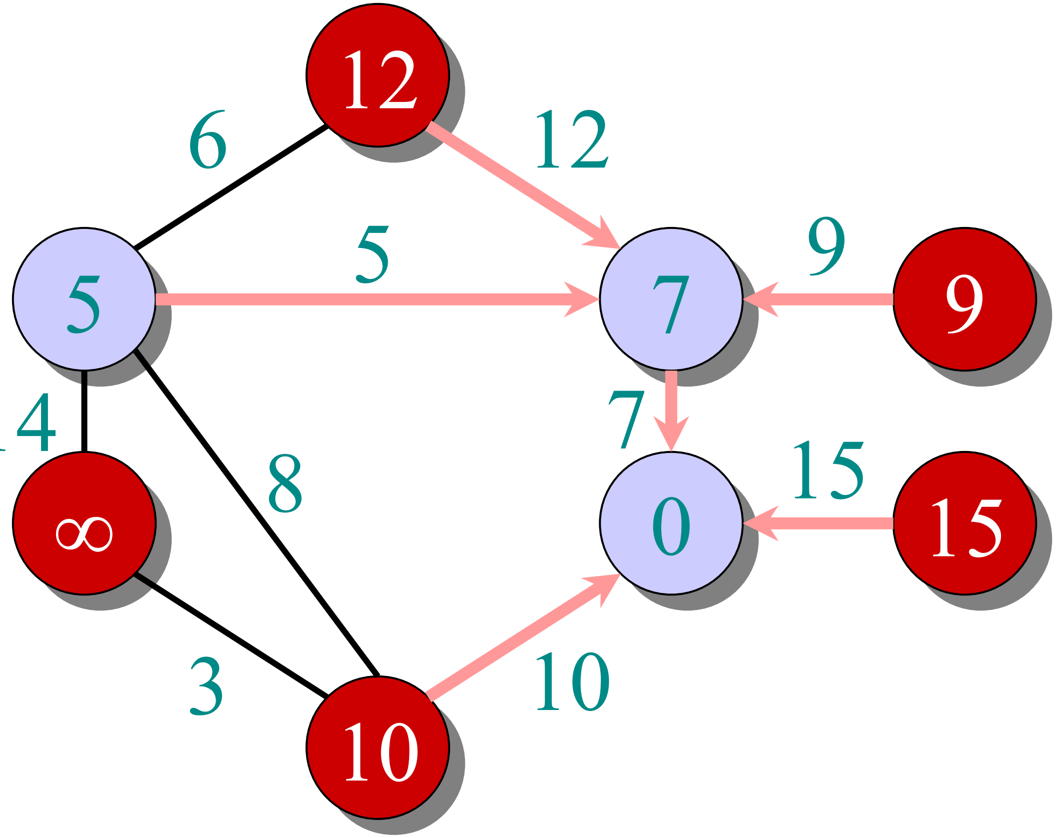
Example of Prim's algorithm

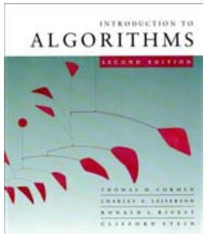




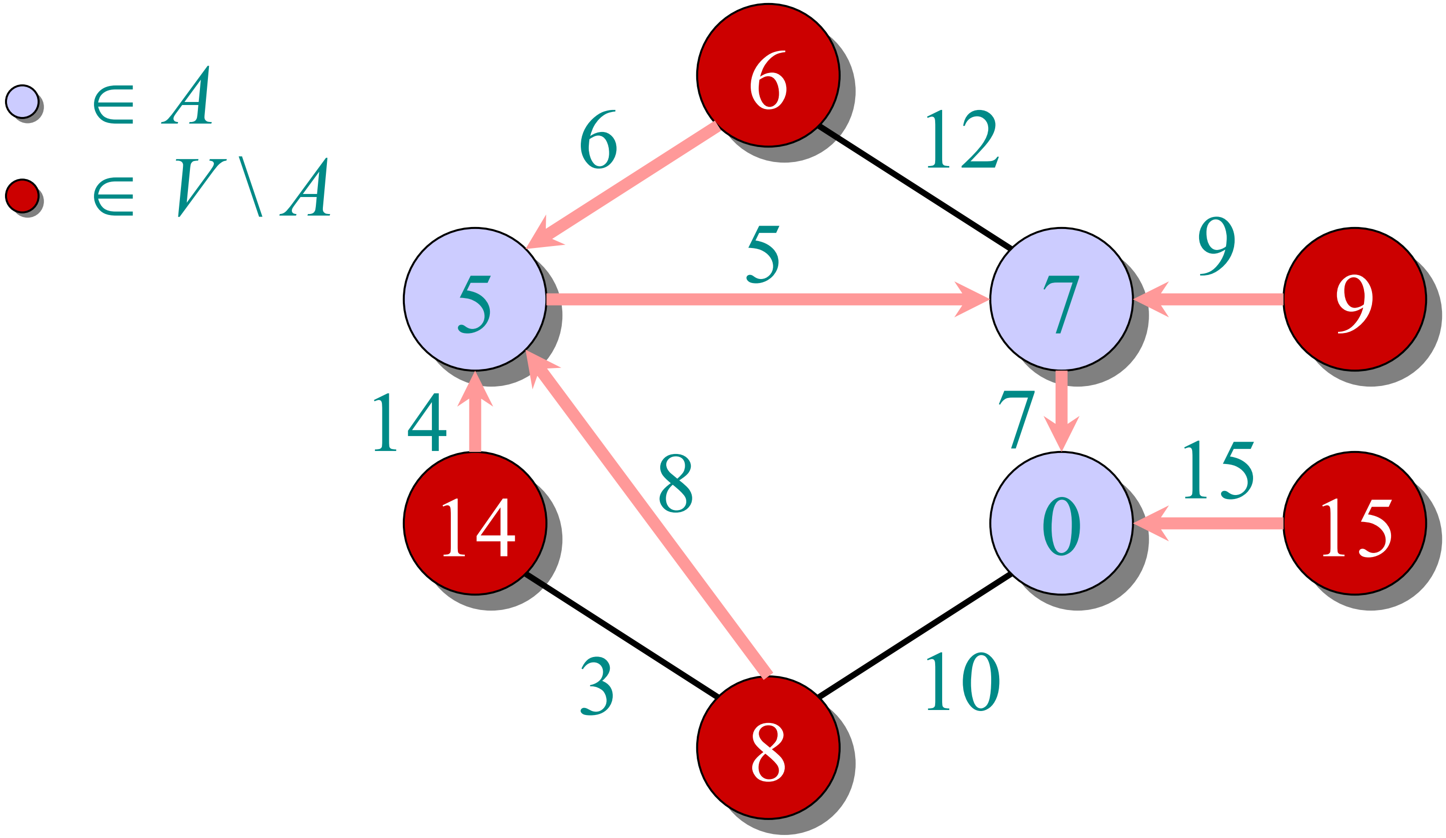
Example of Prim's algorithm

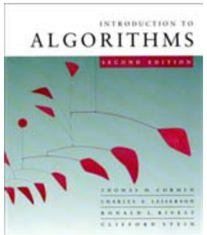
- $\in A$
- $\in V \setminus A$



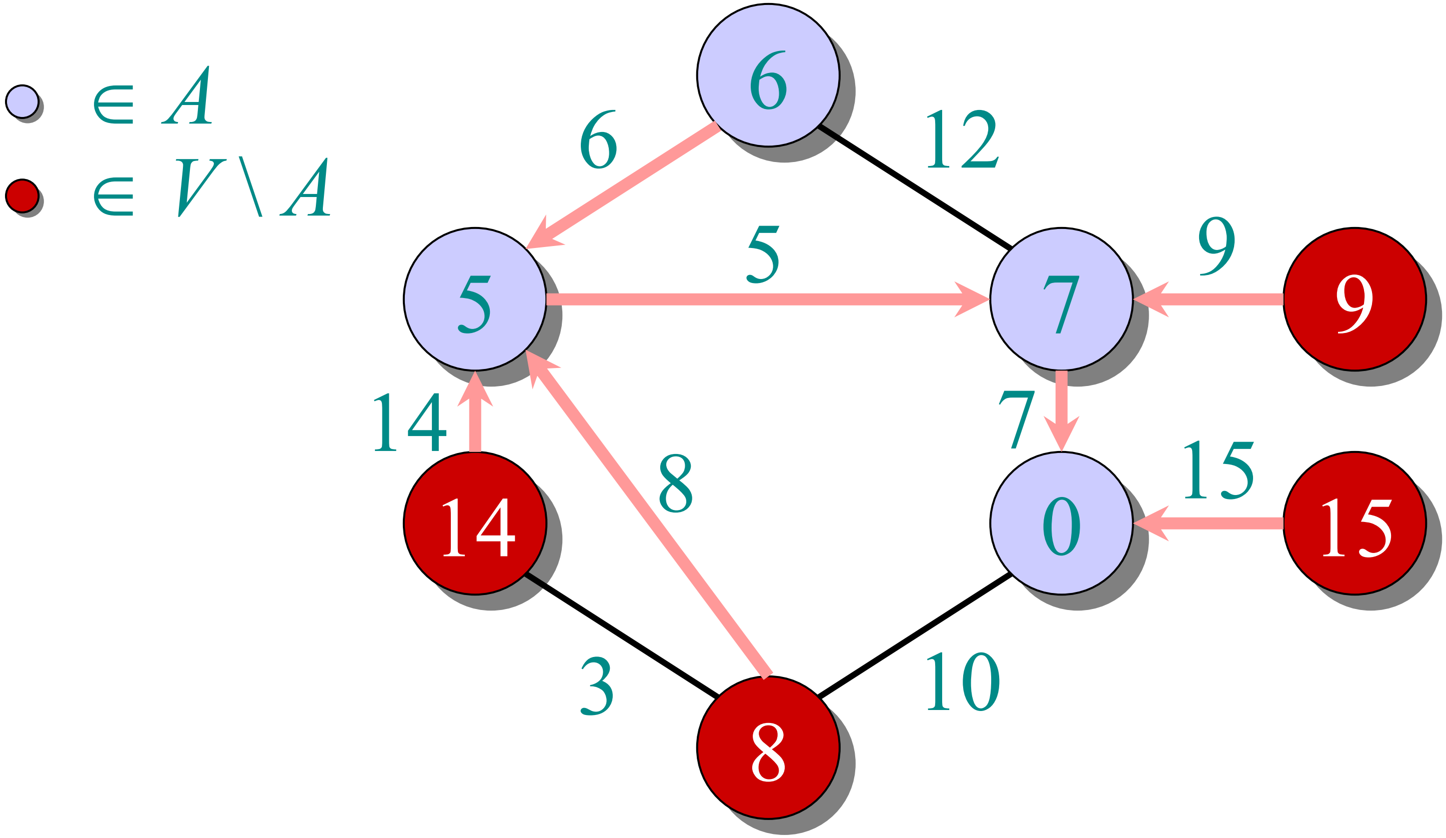


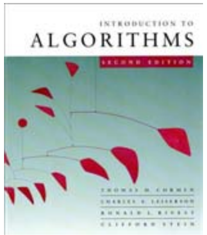
Example of Prim's algorithm





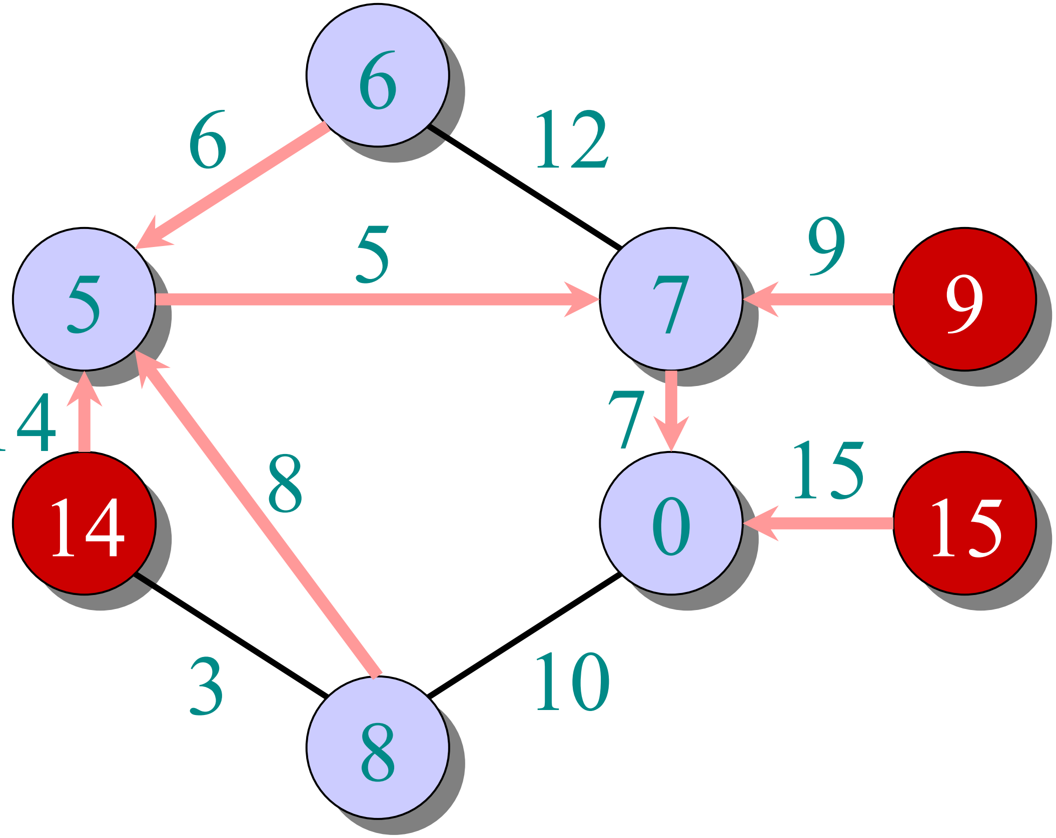
Example of Prim's algorithm

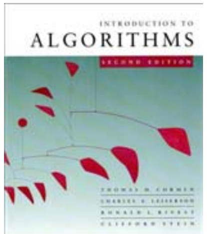




Example of Prim’s algorithm

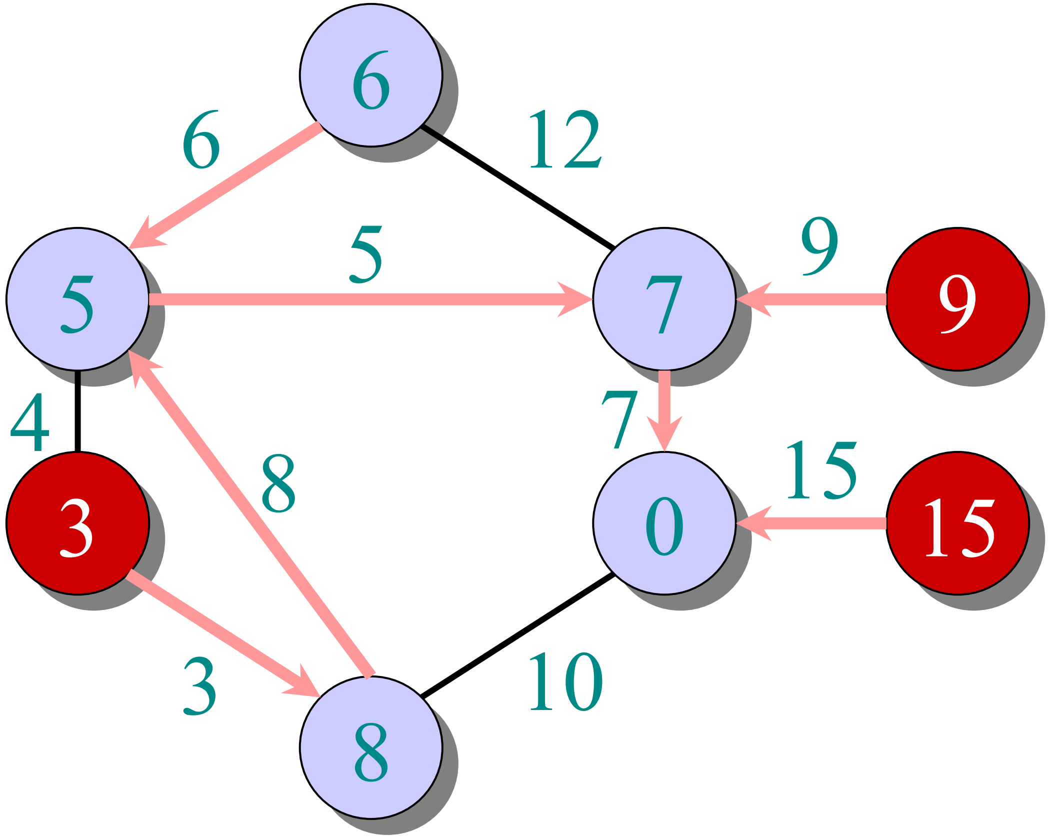
- (light blue circle) $\in A$
- (red circle) $\in V \setminus A$

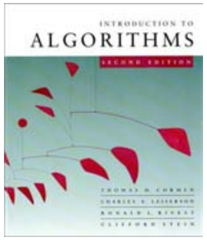




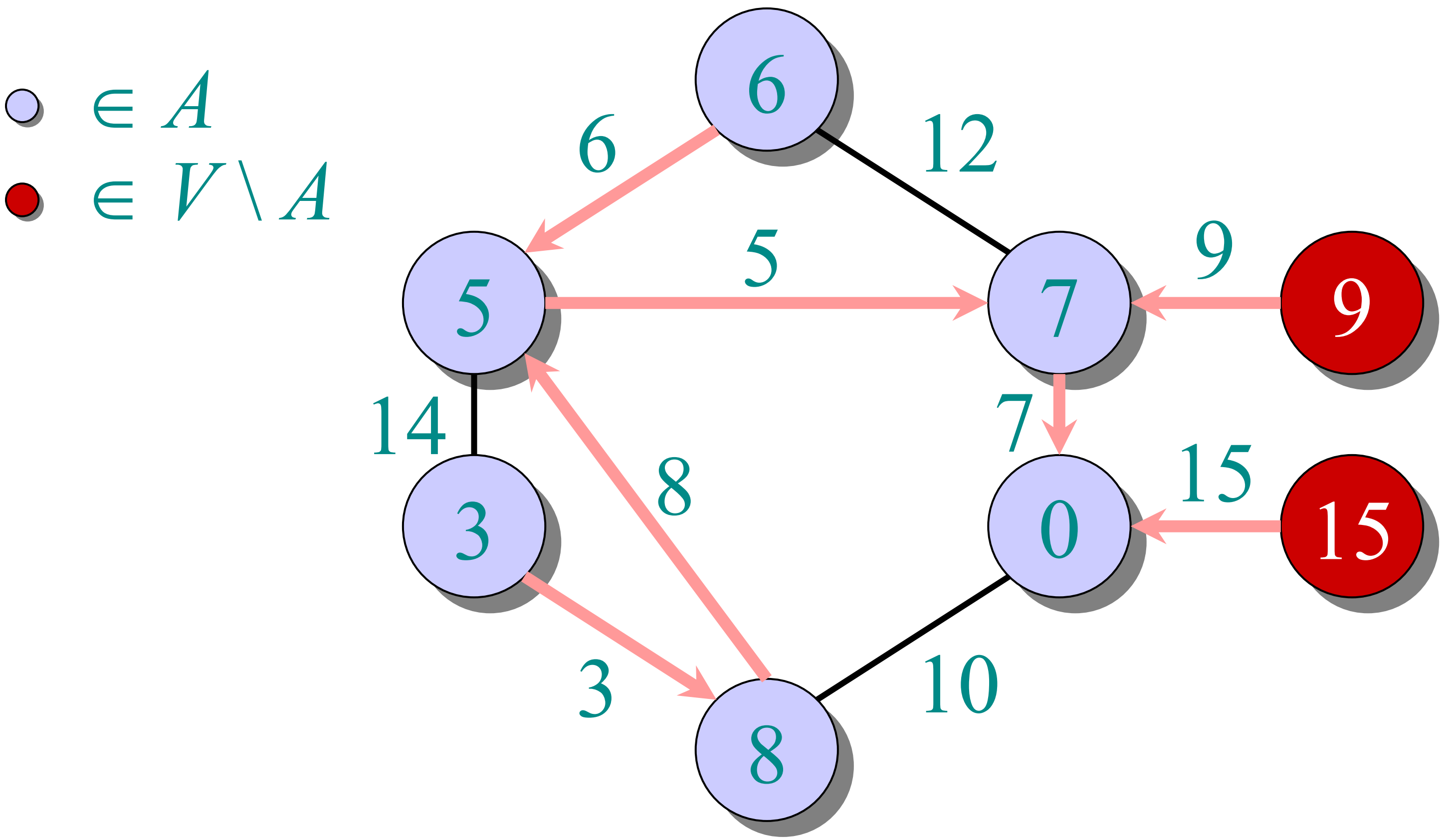
Example of Prim's algorithm

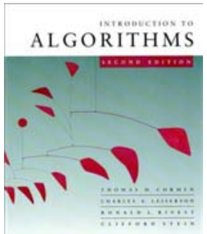
- $\in A$
- $\in V \setminus A$



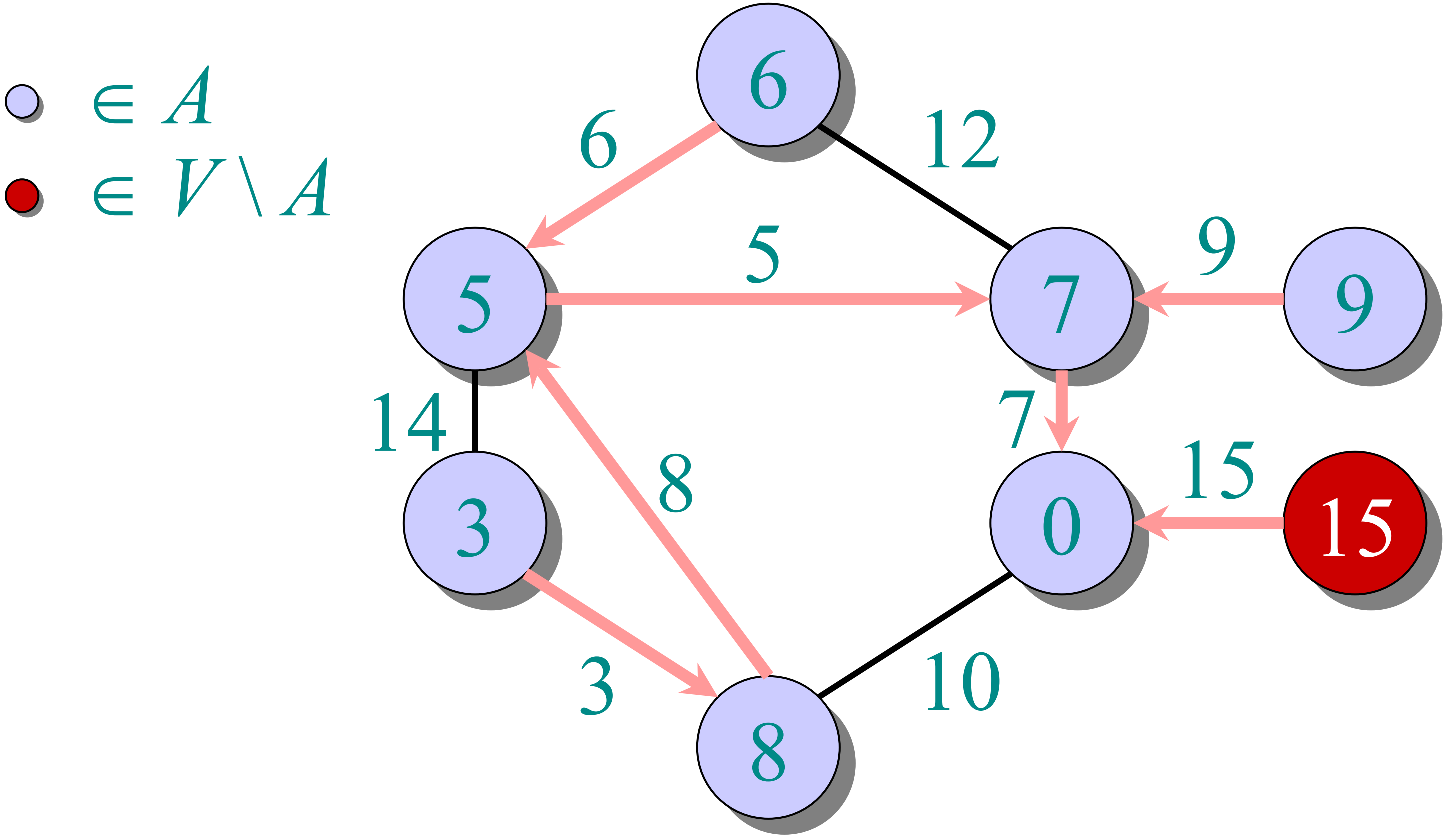


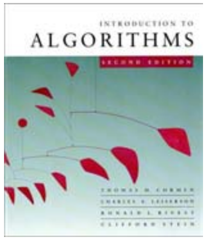
Example of Prim's algorithm



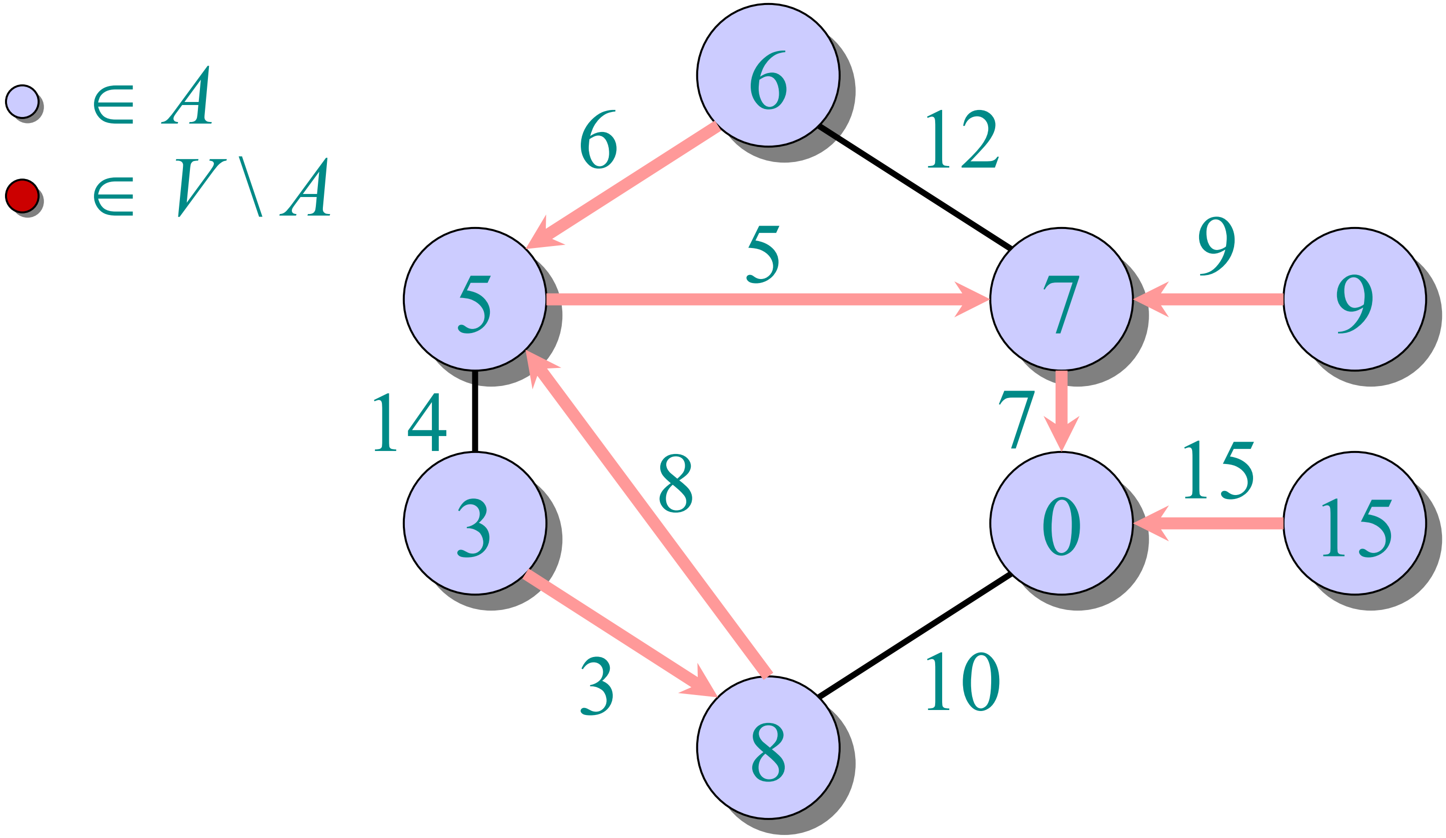


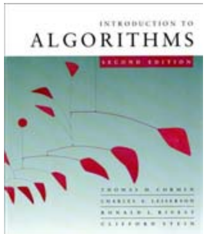
Example of Prim’s algorithm





Example of Prim's algorithm





Analysis of Prim

$\Theta(|V|)$ total:

$Q \leftarrow V$

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

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

$|V|$ times:

while $Q \neq \emptyset$

  do $u \leftarrow$ EXTRACT-MIN(Q)

  $degree(u)$ times:

   for each $v \in Adj[u]$

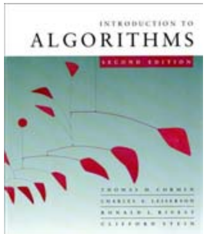
    do if $v \in Q$ and $w(u, v) < key[v]$

     then $key[v] \leftarrow w(u, v)$

      $\pi[v] \leftarrow u$

Handshaking Lemma $\Rightarrow \Theta(|E|)$ implicit DECREASE-KEY's.

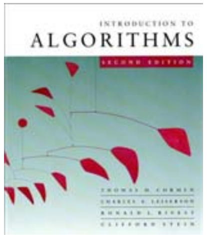
Time $= \Theta(|V|)\cdot T_{\text{EXTRACT-MIN}} + \Theta(|E|)\cdot T_{\text{DECREASE-KEY}}$



Analysis of Prim (continued)

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
array	$O(\|V\|)$	$O(1)$	$O(\|V\|^2)$
binary heap	$O(\log \|V\|)$	$O(\log \|V\|)$	$O(\|E\| \log \|V\|)$
Fibonacci heap	$O(\log \|V\|)$ amortized	$O(1)$ amortized	$O(\|E\| + \|V\| \log \|V\|)$ worst case



MST algorithms

Kruskal's algorithm (see CLRS):

- Uses the ***disjoint-set data structure*** (Lecture 20).
- Running time $= O(|E| \log |V|)$.

Best to date:

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