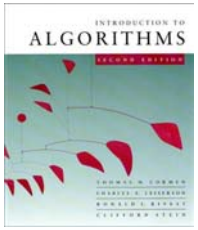
CS 3343 -- Spring 2009



Minimum Spanning Trees

Carola Wenk

Slides courtesy of Charles Leiserson with changes and additions by Carola Wenk

Minimum spanning trees

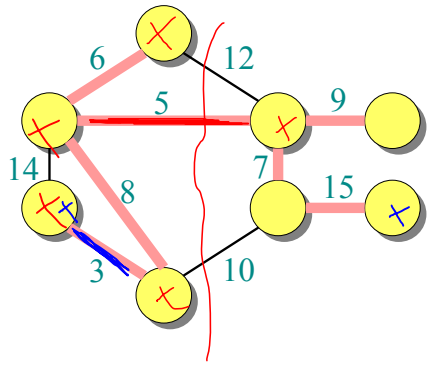
Input: A connected, undirected graph $G = (V, E)$ with weight function $w : E \to \mathsf{R}$.

- 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



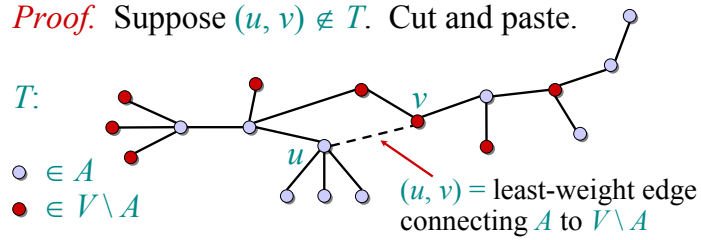
Hallmark for "greedy" algorithms

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, $(u, v) \in T$.

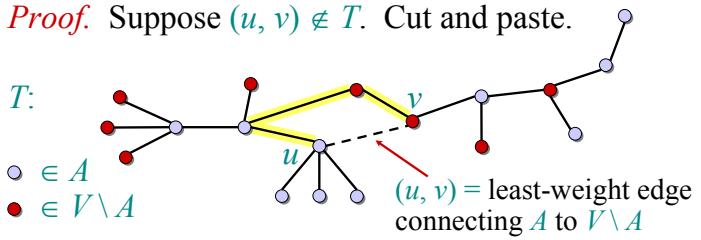
Proof of theorem

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



Proof of theorem

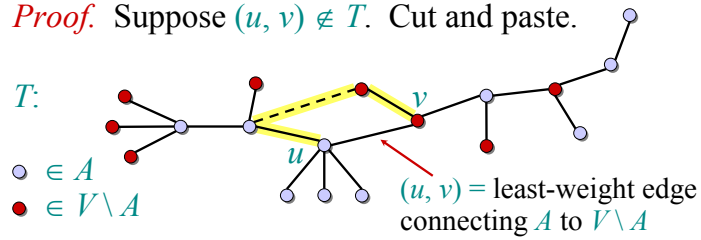
Proof. Suppose $(u, v) \notin T$. Cut and paste.



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

Proof of theorem

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

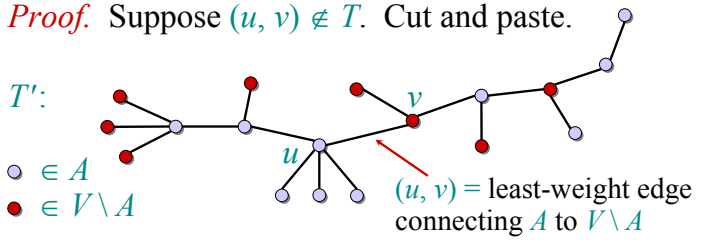


Consider the unique simple path from u to v in T.

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

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 $V \setminus A$.

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.

$Q \leftarrow V$ — set of vertices in the tree

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

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

while $Q \neq \varnothing$

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

    for each $v \in Adj[u]$

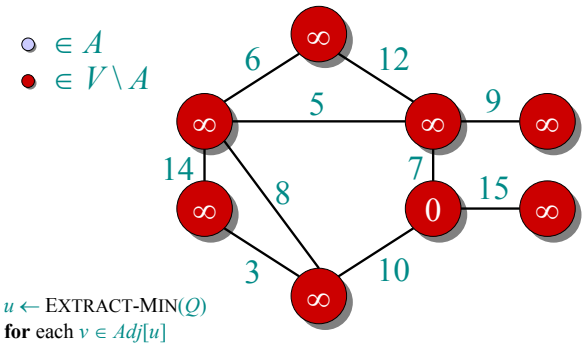
      do if $v \in Q$ and $w(u, v) < key[v]$

        then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY

          $\pi[v] \leftarrow u$

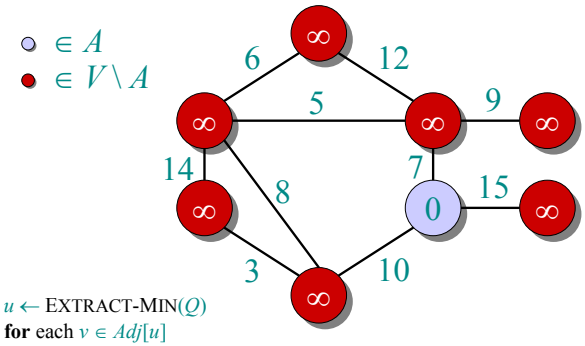
At the end, $\{(v, \pi[v])\}$ forms the MST.

Example of Prim's algorithm



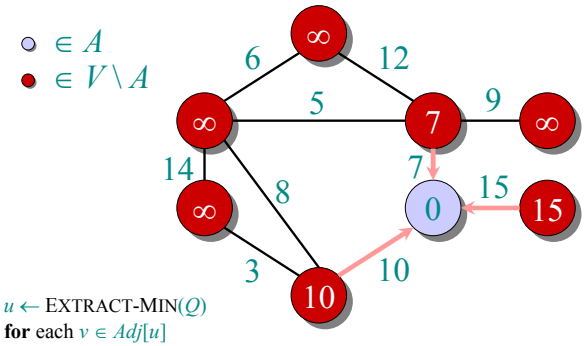
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



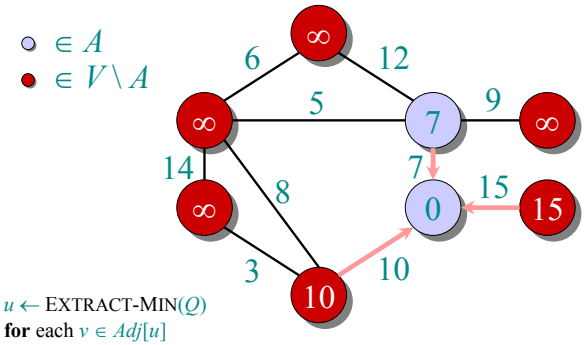
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



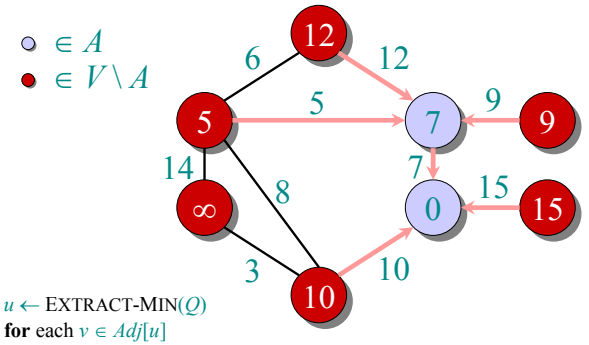
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



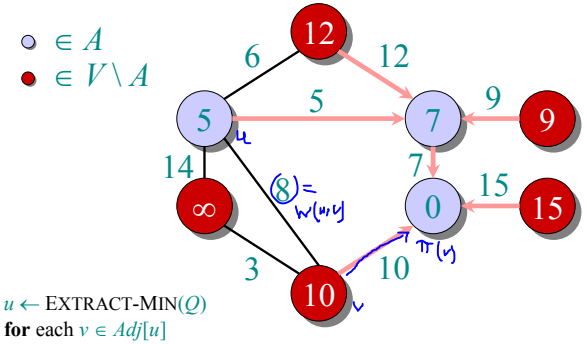
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



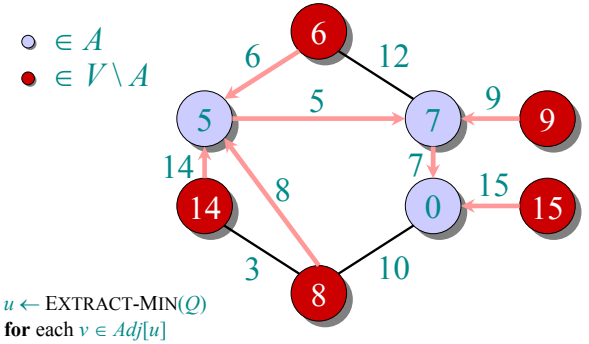
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



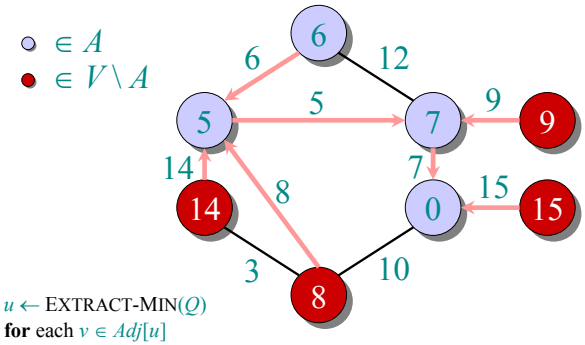
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



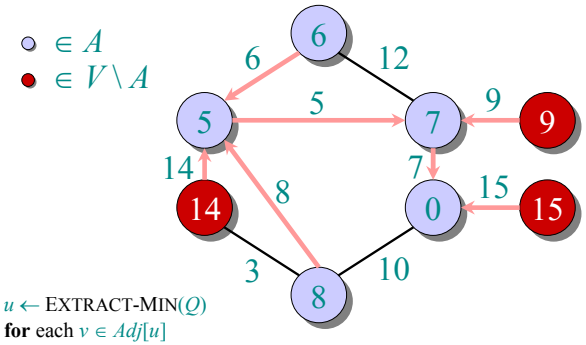
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



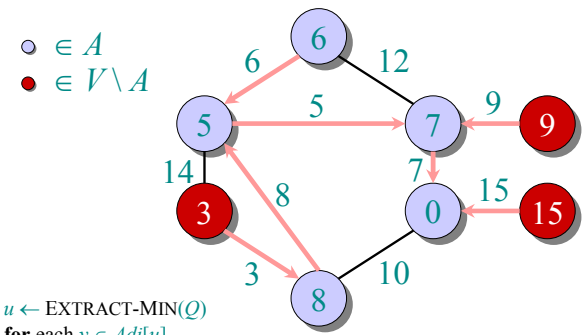
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



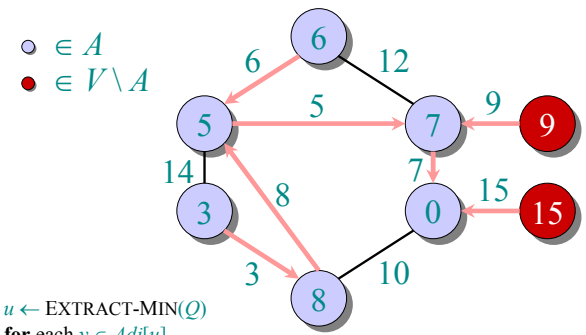
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



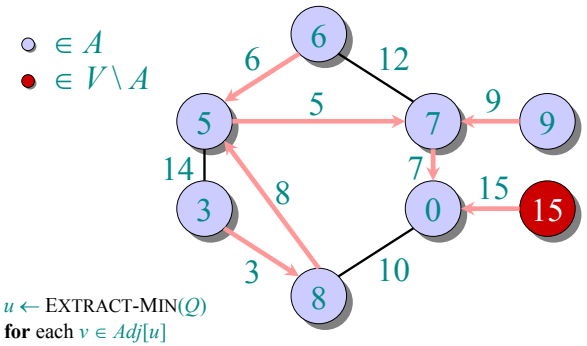
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



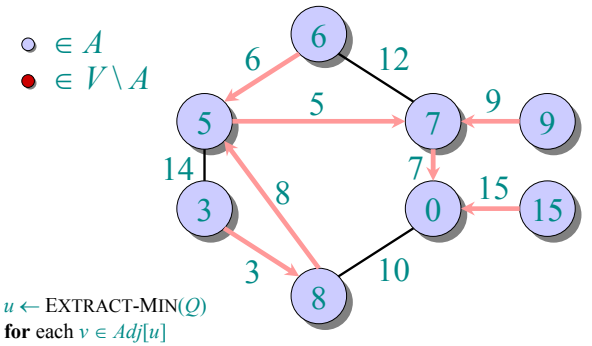
$u \leftarrow$ EXTRACT-MIN(Q)
for each $v \in Adj[u]$
  do if $v \in Q$ and $w(u, v) < key[v]$
    then $key[v] \leftarrow w(u, v)$ ▷ DECREASE-KEY
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



$u \leftarrow$ 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
      $\pi[v] \leftarrow u$

Example of Prim's algorithm



$u \leftarrow$ 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
      $\pi[v] \leftarrow u$

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 $s \in V$

while $Q \neq \varnothing$
  do $u \leftarrow$ EXTRACT-MIN(Q)

$|V|$ times; $degree(u)$ 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

Kruskal's algorithm

IDEA (again greedy):
Repeatedly pick edge with smallest weight as long as it does not form a cycle.

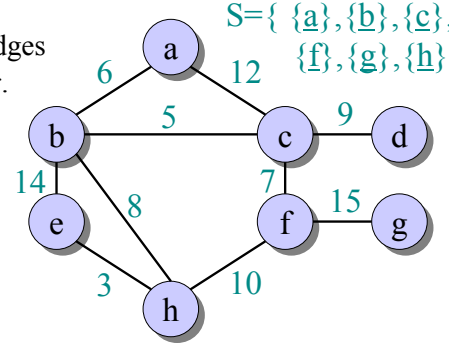
- The algorithm creates a set of trees (a **forest**)
- During the algorithm the added edges merge the trees together, such that in the end only one tree remains

- The correctness of this greedy strategy is not obvious and needs to be proven. (Proof skipped here.)

Example of Kruskal's algorithm

MST edges
a set repr.

S={ {a},{b},{c},{d},{e}
{f},{g},{h} }

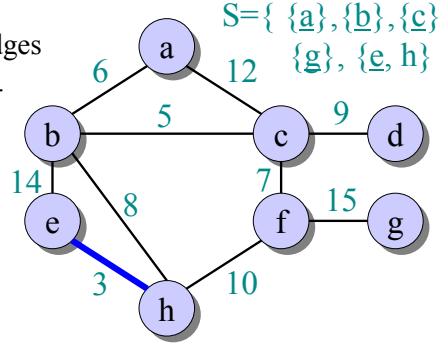


Every node is a single tree.

Example of Kruskal's algorithm

MST edges
a set repr.

S={ {a},{b},{c},{d},{f}
{g}, {e, h} }

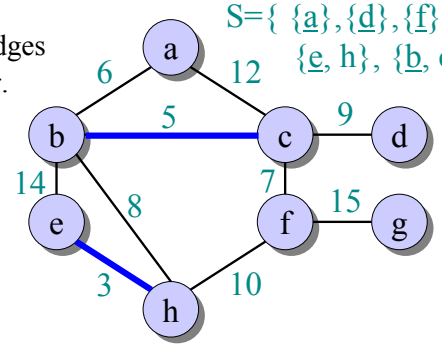


Edge 3 merged two singleton trees.

Example of Kruskal's algorithm

MST edges
a set repr.

S={ {a},{d},{f}, {g}
{e, h}, {b, c} }

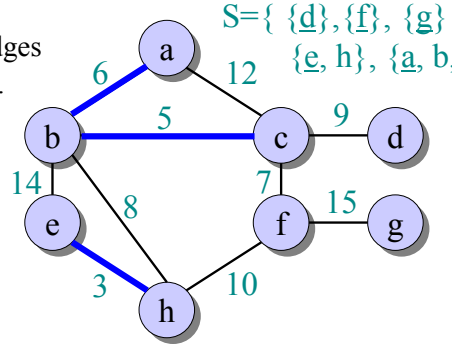


Example of Kruskal's algorithm

MST edges

a set repr.

S={ {d},{f}, {g}
{e, h}, {a, b, c} }

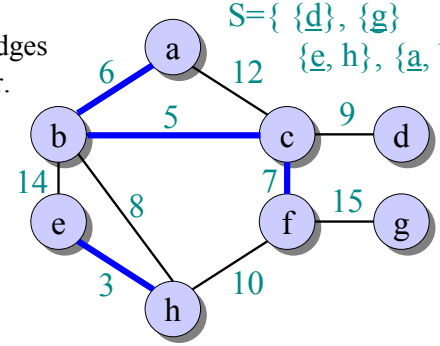


Example of Kruskal's algorithm

MST edges

a set repr.

S={ {d}, {g}
{e, h}, {a, b, c, f} }

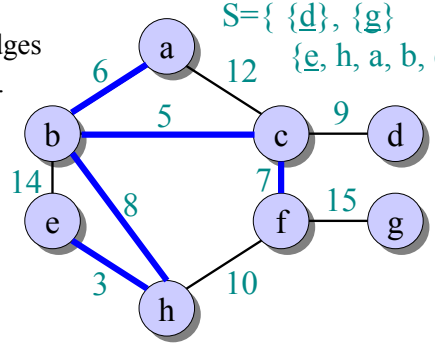


Example of Kruskal's algorithm

MST edges

a set repr.

S={ {d}, {g}
{e, h, a, b, c, f} }



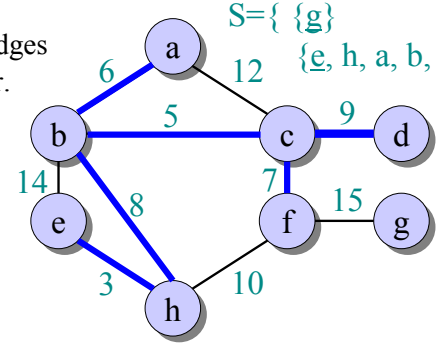
Edge 8 merged the two bigger trees.

Example of Kruskal's algorithm

MST edges

a set repr.

S={ {g}
{e, h, a, b, c, f, d} }



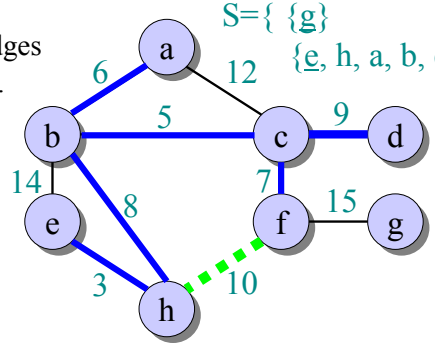
Example of Kruskal's algorithm

MST edges

a set repr.

S={ {g}

{e, h, a, b, c, f, d} }



Skip edge 10 as it would cause a cycle.

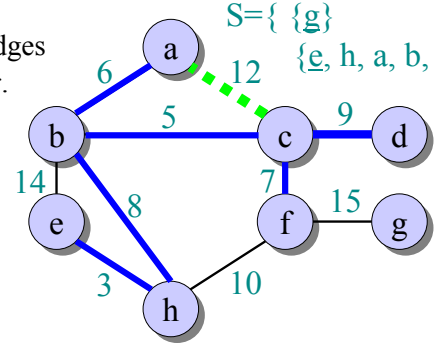
Example of Kruskal's algorithm

MST edges

a set repr.

S={ {g}

{e, h, a, b, c, f, d} }



Skip edge 12 as it would cause a cycle.

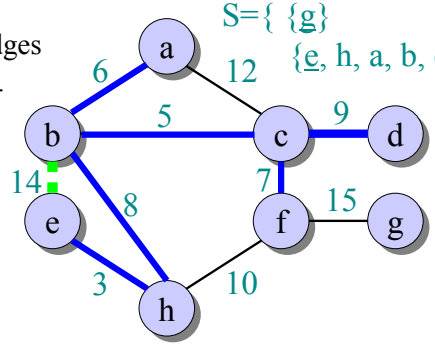
Example of Kruskal's algorithm

MST edges

a set repr.

S={ {g}

{e, h, a, b, c, f, d} }



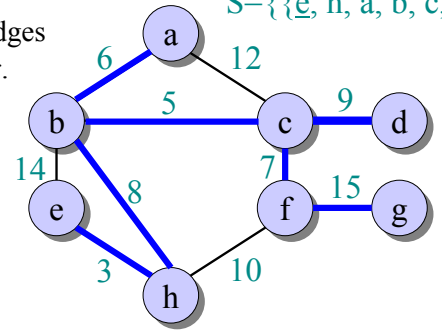
Skip edge 14 as it would cause a cycle.

Example of Kruskal's algorithm

MST edges

a set repr.

S={{e, h, a, b, c, f, d, g} }



Disjoint-set data structure (Union-Find)

- Maintains a dynamic collection of *pairwise-disjoint* sets $\mathsf{S} = \{S_1, S_2, \ldots, S_r\}$.
- Each set S_i has one element distinguished as the **representative** element.
- Supports operations:
  - $O(1)$ • MAKE-SET(x): adds new set $\{x\}$ to S
  - $O(\alpha(n))$ • UNION(x, y): replaces sets S_x, S_y with $S_x \cup S_y$
  - $O(\alpha(n))$ • FIND-SET(x): returns the representative of the set S_x containing element x
- $1 < \alpha(n) < \log^*(n) < \log(\log(n)) < \log(n)$

Kruskal's algorithm

IDEA: Repeatedly pick edge with smallest weight as long as it does not form a cycle.

```
                S ← ∅   ▷ S will contain all MST edges
O(|V|)          for each v ∈ V do MAKE-SET(v)
O(|E|log|E|)    Sort edges of E in non-decreasing order according to w
O(|E|)          For each (u,v) ∈ E taken in this order do
O(α(|V|))       { if FIND-SET(u) ≠ FIND-SET(v)   ▷ u,v in different trees
                    S ← S ∪ {(u,v)}
                    UNION(u,v)   ▷ Edge (u,v) connects the two trees
```

Runtime: $O(|V|+|E|\log|E|+|E|\alpha(|V|)) = O(|E| \log |E|)$

MST algorithms

- Prim's algorithm:
  - Maintains one tree
  - Runs in time $O(|E| \log |V|)$, with binary heaps.
- Kruskal's algorithm:
  - Maintains a forest and uses the disjoint-set data structure
  - Runs in time $O(|E| \log |E|)$
- Best to date: Randomized algorithm by Karger, Klein, Tarjan [1993]. Runs in expected time $O(|V| + |E|)$