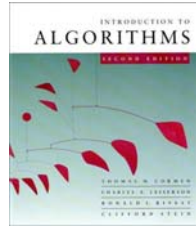




# CS 5633 -- Spring 2008



## 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  $w : E \rightarrow \mathbb{R}$ .

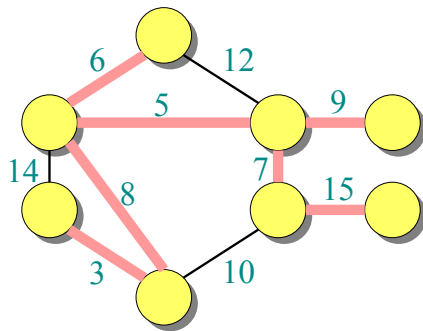
- For simplicity, assume that all edge weights are distinct. (CLRS covers the general case.)

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

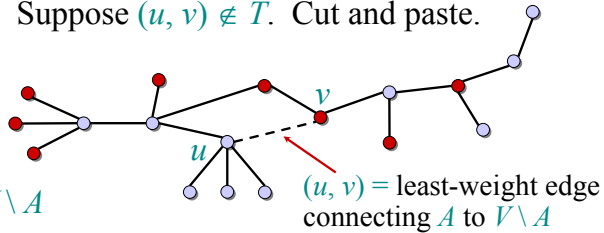


## Proof of theorem

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

$T$ :

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



4/3/08

CS 5633 Analysis of Algorithms

5

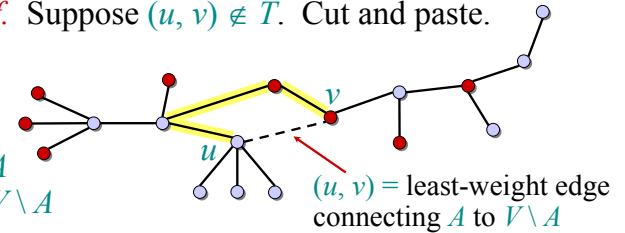


## Proof of theorem

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

$T$ :

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



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

4/3/08

CS 5633 Analysis of Algorithms

6

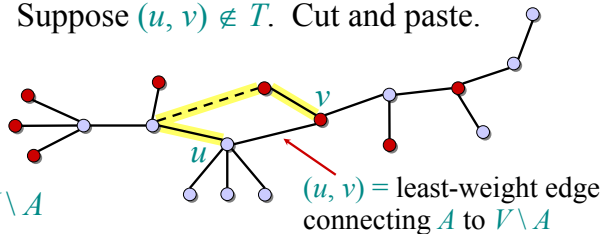


## Proof of theorem

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

$T$ :

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



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  $V \setminus A$ .

4/3/08

CS 5633 Analysis of Algorithms

7

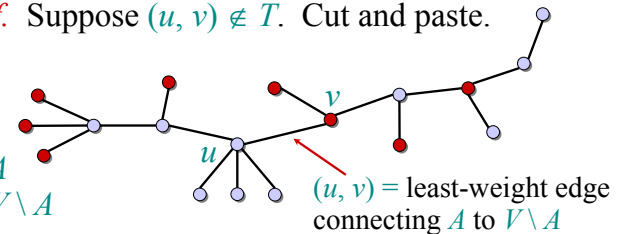


## Proof of theorem

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

$T'$ :

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



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  $V \setminus A$ .

A lighter-weight spanning tree than  $T$  results. □

4/3/08

CS 5633 Analysis of Algorithms

8



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

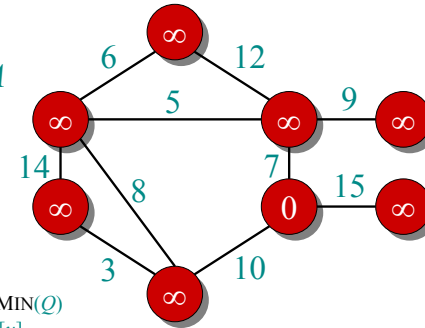
```

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



# Example of Prim's algorithm

$\circ \in A$   
 $\bullet \in V \setminus A$



```

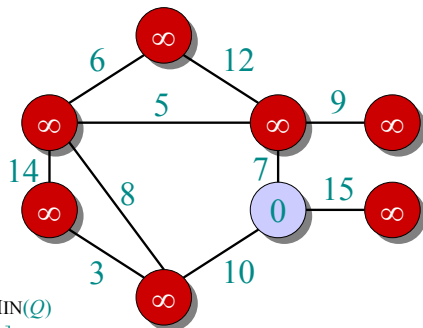
 $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
for each  $v \in \text{Adj}[u]$ 
  do if  $v \in Q$  and  $w(u, v) < key[v]$ 
    then  $key[v] \leftarrow w(u, v)$   $\triangleright$  DECREASE-KEY
     $\pi[v] \leftarrow u$ 

```



# Example of Prim's algorithm

$\circ \in A$   
 $\bullet \in V \setminus A$



```

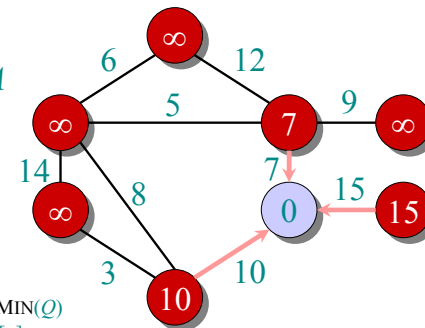
 $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
for each  $v \in \text{Adj}[u]$ 
  do if  $v \in Q$  and  $w(u, v) < key[v]$ 
    then  $key[v] \leftarrow w(u, v)$   $\triangleright$  DECREASE-KEY
     $\pi[v] \leftarrow u$ 

```



# Example of Prim's algorithm

$\circ \in A$   
 $\bullet \in V \setminus A$



```

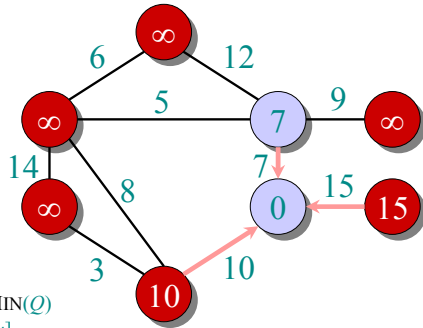
 $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
for each  $v \in \text{Adj}[u]$ 
  do if  $v \in Q$  and  $w(u, v) < key[v]$ 
    then  $key[v] \leftarrow w(u, v)$   $\triangleright$  DECREASE-KEY
     $\pi[v] \leftarrow u$ 

```



# Example of Prim's algorithm

- $\circ \in A$
- $\bullet \in V \setminus A$



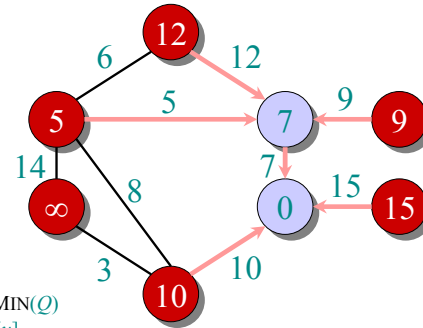
```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```



# Example of Prim's algorithm

- $\circ \in A$
- $\bullet \in V \setminus A$



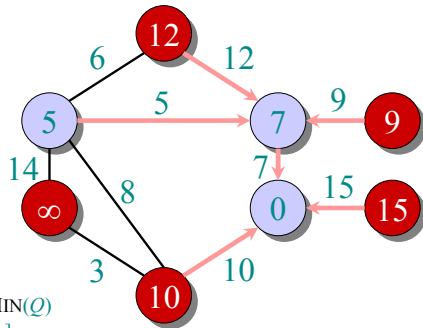
```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```



# Example of Prim's algorithm

- $\circ \in A$
- $\bullet \in V \setminus A$



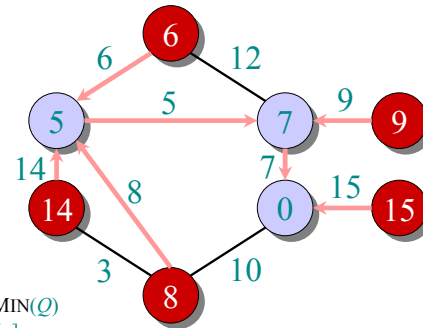
```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```



# Example of Prim's algorithm

- $\circ \in A$
- $\bullet \in V \setminus A$



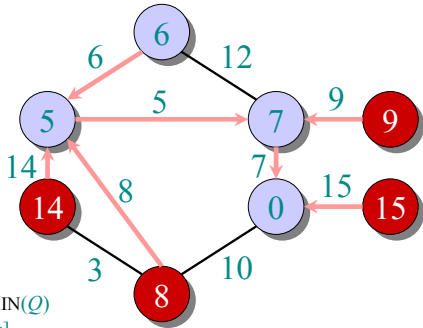
```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```



# Example of Prim's algorithm

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



```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```

4/3/08

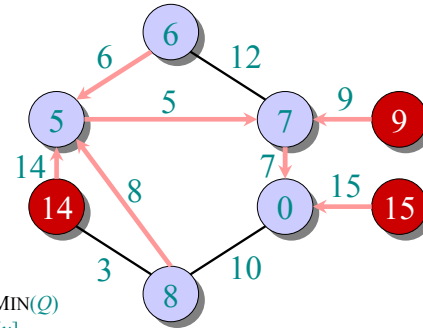
CS 5633 Analysis of Algorithms

17



# Example of Prim's algorithm

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



```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```

4/3/08

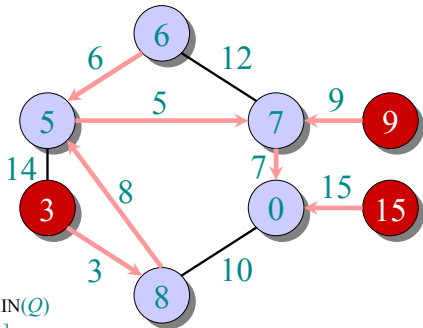
CS 5633 Analysis of Algorithms

18



# Example of Prim's algorithm

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



```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```

4/3/08

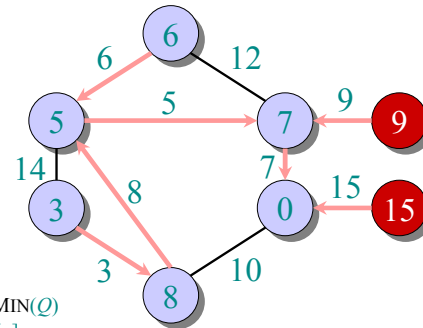
CS 5633 Analysis of Algorithms

19



# Example of Prim's algorithm

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



```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```

4/3/08

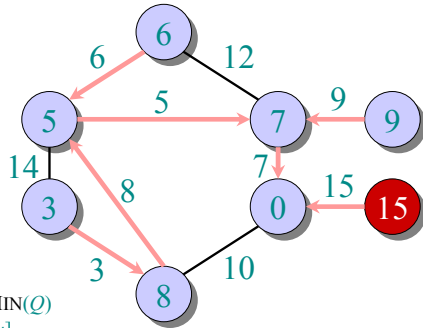
CS 5633 Analysis of Algorithms

20



# Example of Prim's algorithm

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



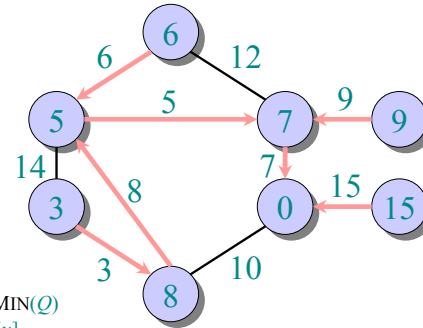
```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```



# Example of Prim's algorithm

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



```

u ← EXTRACT-MIN(Q)
for each v ∈ Adj[u]
  do if v ∈ Q and w(u, v) < key[v]
    then key[v] ← w(u, v) ▷ DECREASE-KEY
    π[v] ← u
  
```



# Analysis of Prim

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

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

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



# Analysis of Prim (continued)

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

$Q$	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	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):

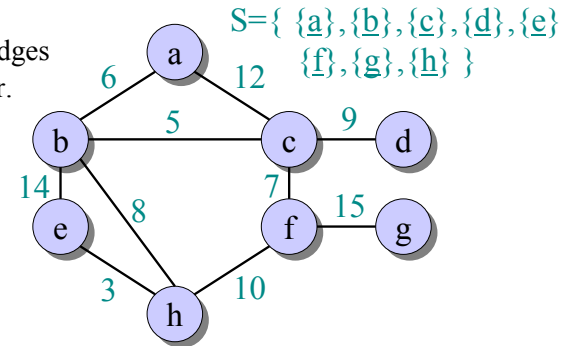
Repeatedly pick edge with smallest weight as long as it does not form a cycle.

- 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.

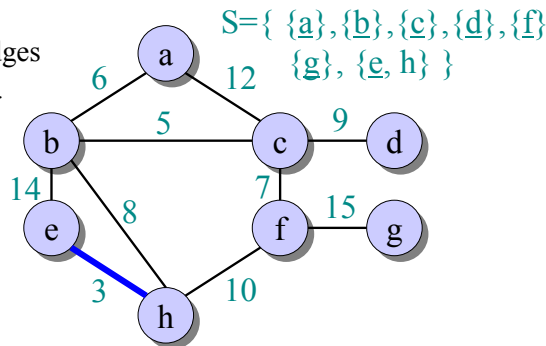


Every node is a single tree.



# Example of Kruskal's algorithm

— MST edges  
a set repr.

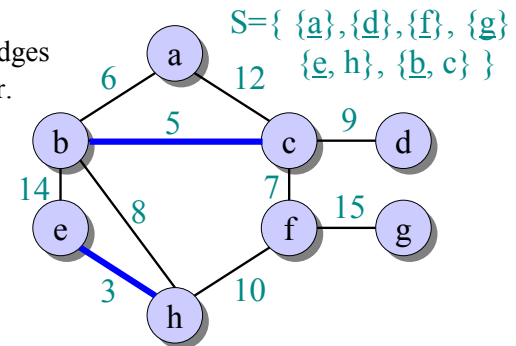


Edge 3 merged two singleton trees.



# Example of Kruskal's algorithm

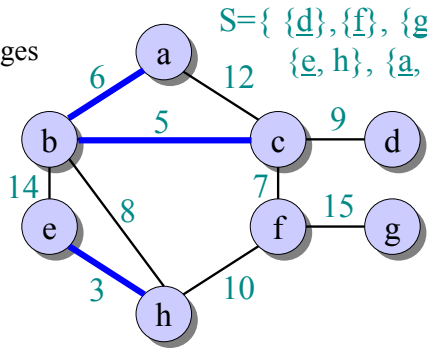
— MST edges  
a set repr.





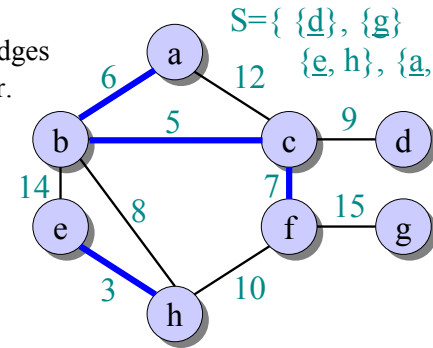
## Example of Kruskal's algorithm

— MST edges  
a set repr.



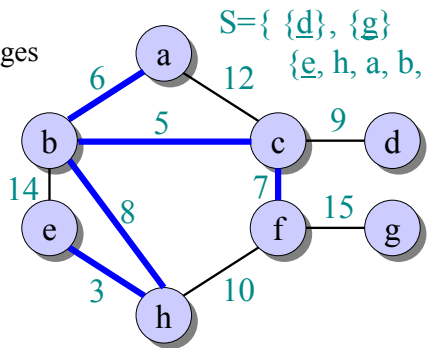
## Example of Kruskal's algorithm

— MST edges  
a set repr.



## Example of Kruskal's algorithm

— MST edges  
a set repr.

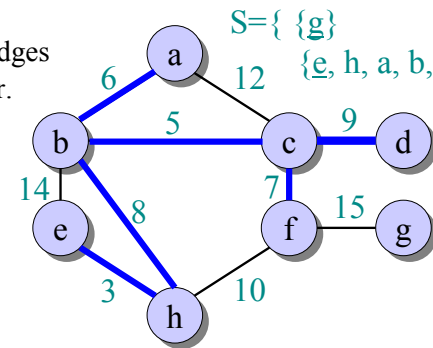


Edge 8 merged the two bigger trees.



## Example of Kruskal's algorithm

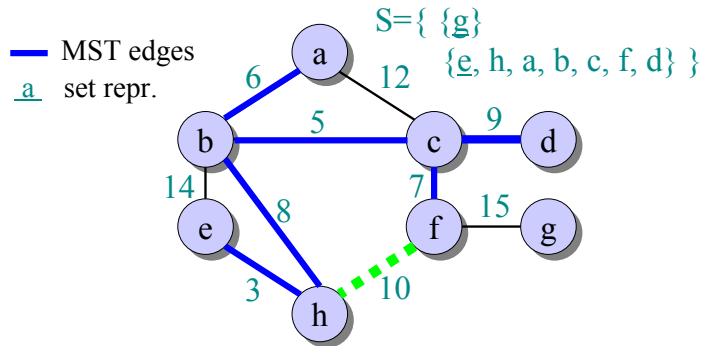
— MST edges  
a set repr.







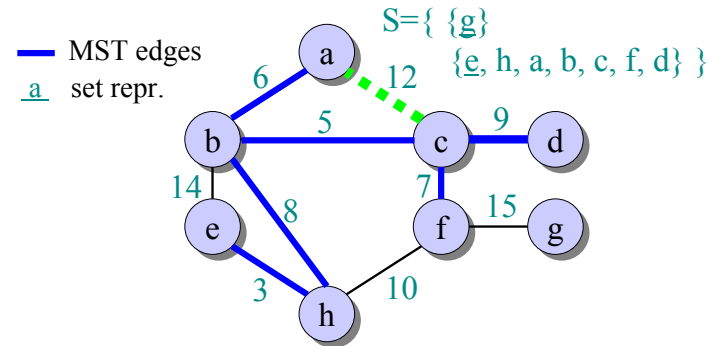
## Example of Kruskal's algorithm



Skip edge 10 as it would cause a cycle.



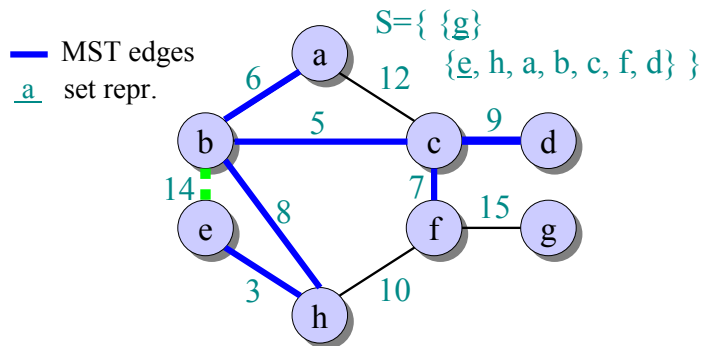
## Example of Kruskal's algorithm



Skip edge 12 as it would cause a cycle.



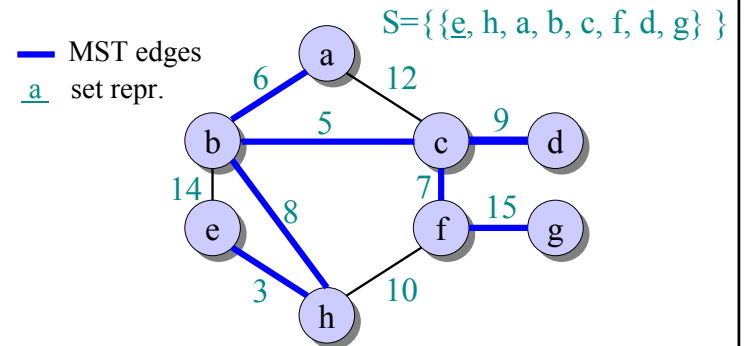
## Example of Kruskal's algorithm



Skip edge 14 as it would cause a cycle.



## Example of Kruskal's algorithm





## Disjoint-set data structure (Union-Find)

- Maintains a dynamic collection of *pairwise-disjoint* sets  $\mathbf{S} = \{S_1, S_2, \dots, S_r\}$ .
- Each set  $S_i$  has one element distinguished as the **representative** element.
- Supports operations:
  - $O(1)$  • MAKE-SET( $x$ ): adds new set  $\{x\}$  to  $\mathbf{S}$
  - $O(\alpha(n))$  • 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.

$S \leftarrow \emptyset$  ▶  $S$  will contain all MST edges

$O(|V|)$  for each  $v \in V$  do MAKE-SET( $v$ )

$O(|E| \log |E|)$  Sort edges of  $E$  in non-decreasing order according to  $w$

$O(|E|)$  For each  $(u, v) \in E$  taken in this order do

$O(\alpha(|V|))$   $\left\{ \begin{array}{l} \text{if FIND-SET}(u) \neq \text{FIND-SET}(v) \text{ } \blacktriangleright u, v \text{ in different trees} \\ S \leftarrow S \cup \{(u, v)\} \\ \text{UNION}(u, v) \text{ } \blacktriangleright \text{Edge } (u, v) \text{ connects the two trees} \end{array} \right.$

**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  $O(|V| + |E|)$