

Minimum Spanning Trees

□ Problem

- Given a connected undirected weighted graph (G, w) with $G = (V, E)$, the goal of the minimum spanning tree (MST) problem is to find a spanning tree of the smallest cost.
- How to implement Prim's algorithm in $O((|V| + |E|) \cdot \log|V|)$ time?

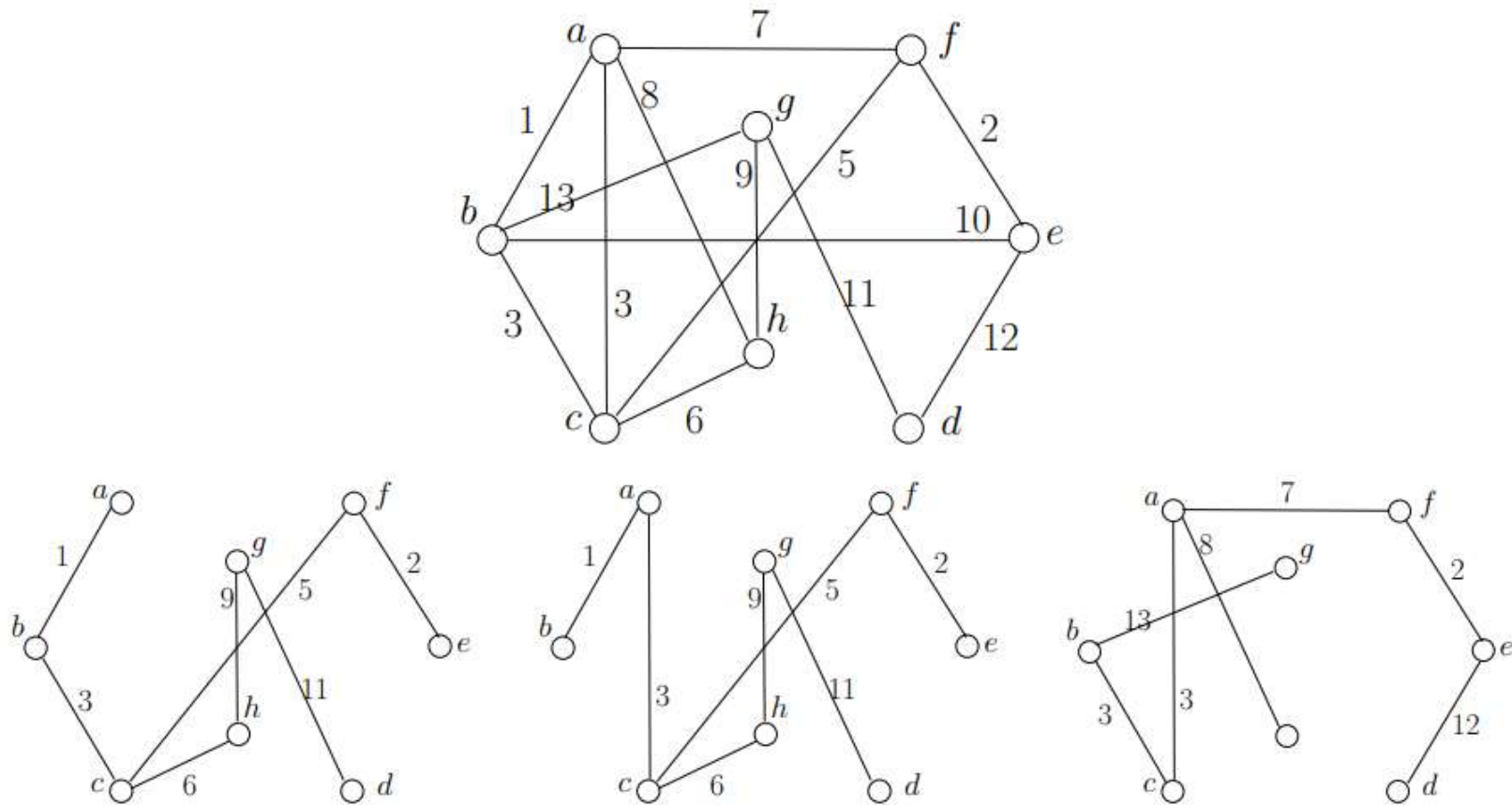
Let $G = (V, E)$ be a connected undirected graph. Let w be a function that maps each edge e of G to a positive integer $w(e)$ called the **weight** of e .

A spanning tree T is a tree satisfying the following conditions:

- The vertex set of T is V .
- Every edge of T is an edge in G .

The **cost** of T is the sum of the weights of all the edges in T .

Example



The second row shows three spanning trees. The cost of the first two trees is 37, and that of the right tree is 48.

Prim's algorithm

The algorithm grows a tree T_{mst} by including one vertex at a time.

At any moment, it divides the vertex set V into two parts:

- The set S of vertices that are already in T_{mst} .
- The set of other vertices: $V \setminus S$.

At the end of the algorithm, $S = V$.

If an edge connects a vertex in V and a vertex in $V \setminus S$, we call it an **cross edge**.

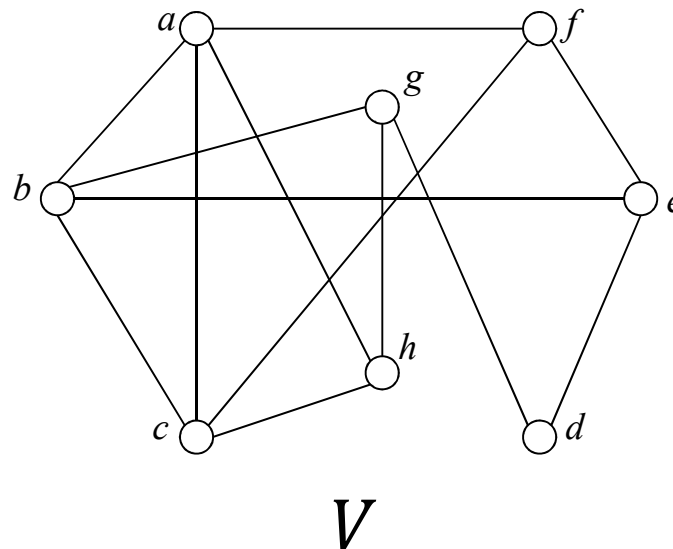
Prim's algorithm

The algorithm grows a tree T_{mst} by including one vertex at a time. At any moment, it divides the vertex set V into two parts:

- The set S of vertices that are already in T_{mst} .
- The set of other vertices: $V \setminus S$.

At the end of the algorithm, $S = V$.

If an edge connects a vertex in V and a vertex in $V \setminus S$, we call it a **cross edge**.



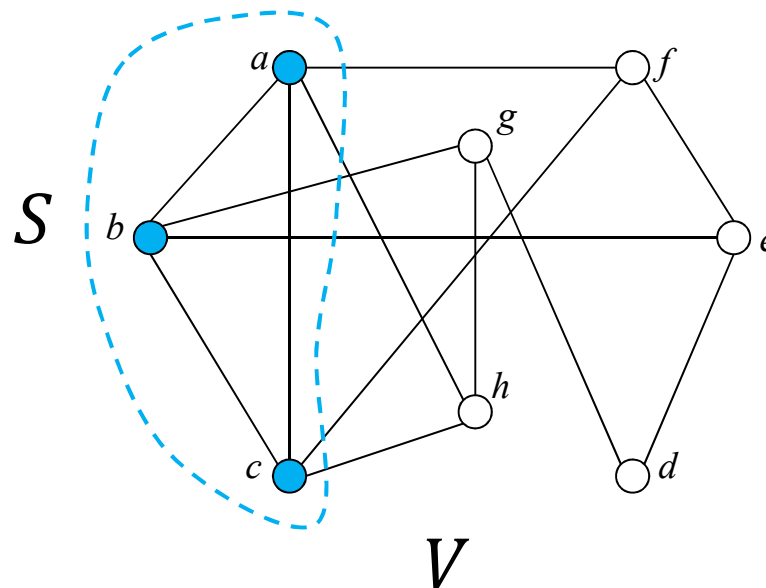
Prim's algorithm

The algorithm grows a tree T_{mst} by including one vertex at a time. At any moment, it divides the vertex set V into two parts:

- The set S of vertices that are already in T_{mst} .
- The set of other vertices: $V \setminus S$.

At the end of the algorithm, $S = V$.

If an edge connects a vertex in V and a vertex in $V \setminus S$, we call it a **cross edge**.



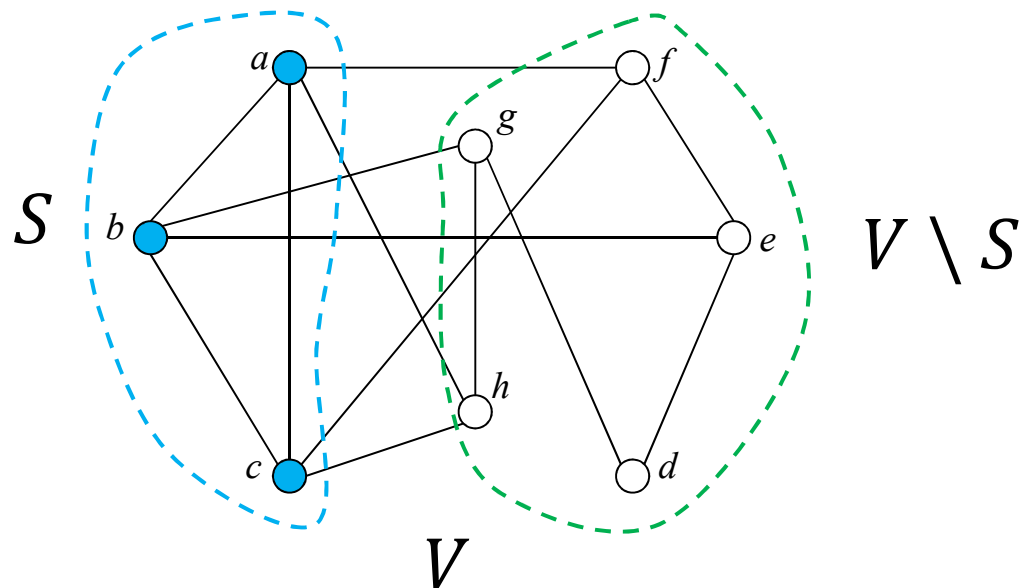
Prim's algorithm

The algorithm grows a tree T_{mst} by including one vertex at a time. At any moment, it divides the vertex set V into two parts:

- The set S of vertices that are already in T_{mst} .
- The set of other vertices: $V \setminus S$.

At the end of the algorithm, $S = V$.

If an edge connects a vertex in V and a vertex in $V \setminus S$, we call it a **cross edge**.



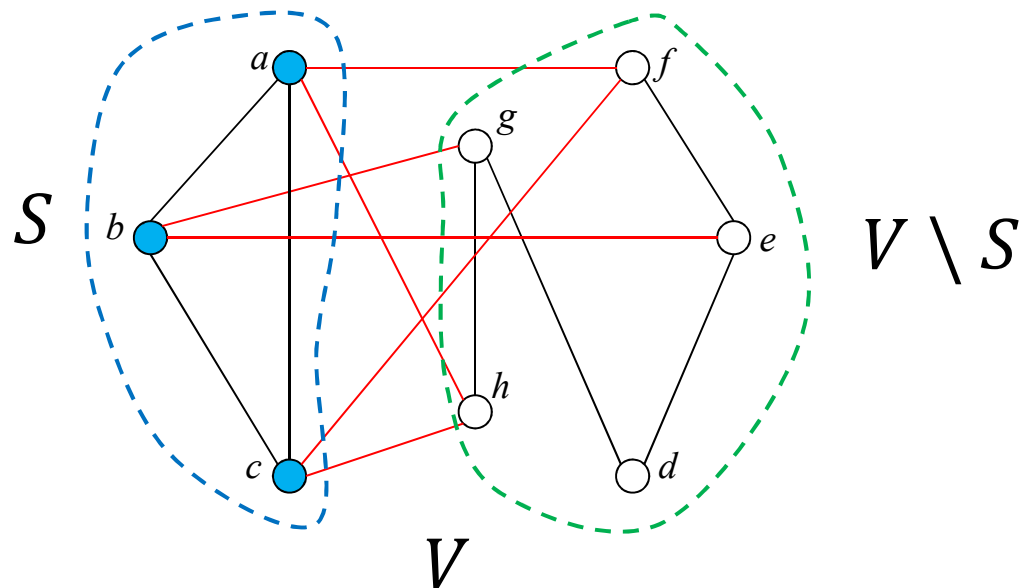
Prim's algorithm

The algorithm grows a tree T_{mst} by including one vertex at a time. At any moment, it divides the vertex set V into two parts:

- The set S of vertices that are already in T_{mst} .
- The set of other vertices: $V \setminus S$.

At the end of the algorithm, $S = V$.

If an edge connects a vertex in V and a vertex in $V \setminus S$, we call it a **cross edge**.



Implementing Prim's algorithm

To implement the algorithm efficiently, we will enforce the following **invariant**:

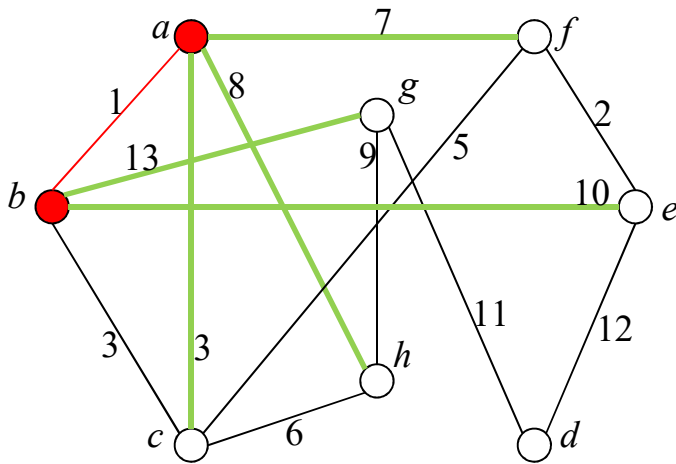
- For every vertex $v \in V \setminus S$, remember which cross edge of v has the smallest weight — refer to the edge as the **lightest cross edge** of v and denote it as *best-cross*(v).

Implementing Prim's algorithm

1. $\{u, v\}$ = an edge with the smallest weight among all edges.
2. Set $S = \{u, v\}$. Initialize a tree T_{mst} with only one edge $\{u, v\}$.
3. Enforce our invariant:
 - For every vertex z of $V \setminus S$
 - If z is a neighbor of u , but not of v
 $best-cross(z) = \text{edge } \{z, u\}$
 - If z is a neighbor of v , but not of u
 $best-cross(z) = \text{edge } \{z, v\}$
 - Otherwise
 $best-cross(z) = \text{the lighter edge between } \{z, u\} \text{ and } \{z, v\}$

Example

Edge $\{a, b\}$ is the lightest of all. So, in the beginning $S = \{a, b\}$. The MST now has one edge $\{a, b\}$.



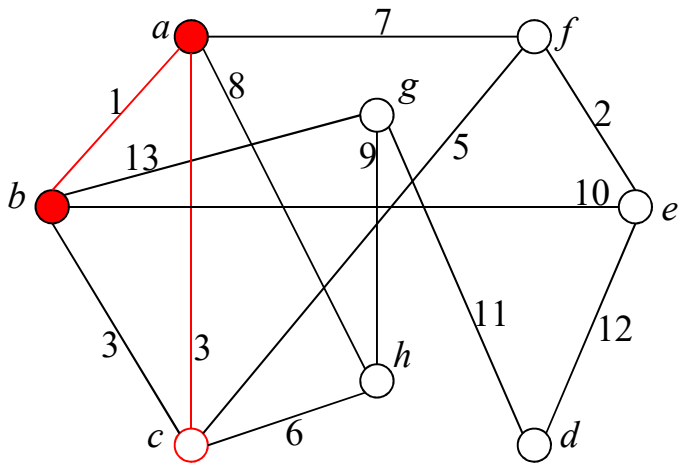
vertex v	best-cross and weight
a	n / a
b	n / a
c	$\{c, a\}, 3$
d	nil, ∞
e	$\{e, b\}, 10$
f	$\{a, f\}, 7$
g	$\{g, b\}, 13$
h	$\{a, h\}, 8$

Implementing Prim's algorithm

4. Repeat the following until $S = V$:
 5. Find a cross edge $\{u, v\}$ with the smallest weight
/* Without loss of generality, suppose $u \in S$ and $v \notin S$ */
 6. Add v into S , and add edge $\{u, v\}$ into T_{mst}
/* Next, restore the invariant. */
 7. for every edge $\{v, z\}$ of v :
 - If $z \notin S$ then
If $best\text{-}cross(z)$ is heavier than edge $\{v, z\}$ then
Set $best\text{-}cross(z) = \text{edge } \{v, z\}$

Example

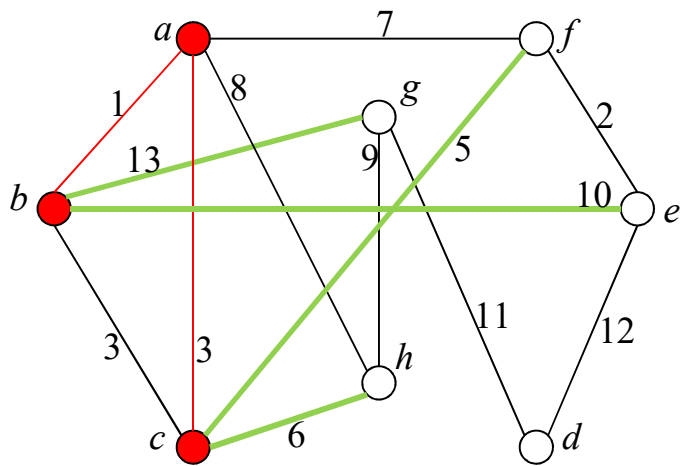
Edge $\{c, a\}$ is a lightest cross edge. So, we add c to S , which is now $S = \{a, b, c\}$. Add edge $\{c, a\}$ into the MST.



vertex v	best-cross and weight
a	n / a
b	n / a
c	$\{c, a\}, 3$
d	nil, ∞
e	$\{e, b\}, 10$
f	$\{a, f\}, 7$
g	$\{g, b\}, 13$
h	$\{a, h\}, 8$

Example

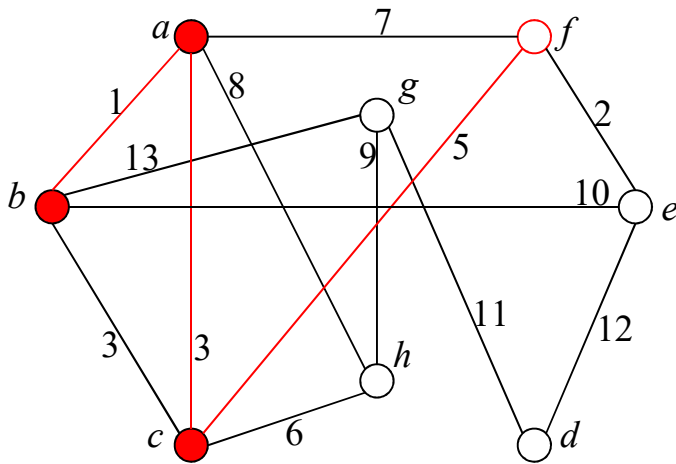
Restore the invariant.



vertex v	best-cross and weight
a	n / a
b	n / a
c	{c, a}, 3 => n / a
d	nil, ∞
e	{e, b}, 10
f	{a, f}, 7 => {c, f}, 5
g	{g, b}, 13
h	{a, h}, 8 => {c, h}, 6

Example

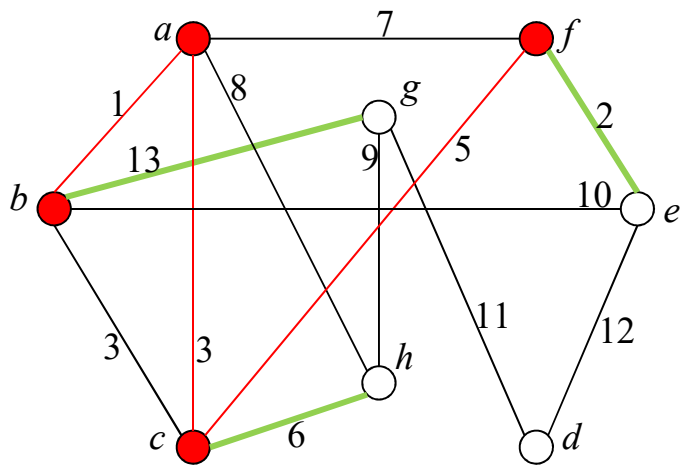
Edge $\{c, f\}$ is the lightest cross edge. So, we add f to S , which is now $S = \{a, b, c, f\}$. Add edge $\{c, f\}$ into the MST.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	nil, ∞
e	{e, b}, 10
f	{c, f}, 5
g	{g, b}, 13
h	{c, h}, 6

Example

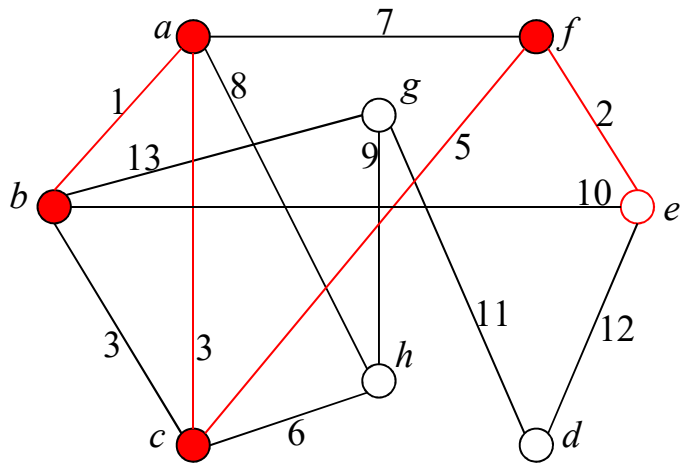
Restore the invariant.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	nil, ∞
e	{e, f}, 2
f	n / a
g	{g, b}, 13
h	{c, h}, 6

Example

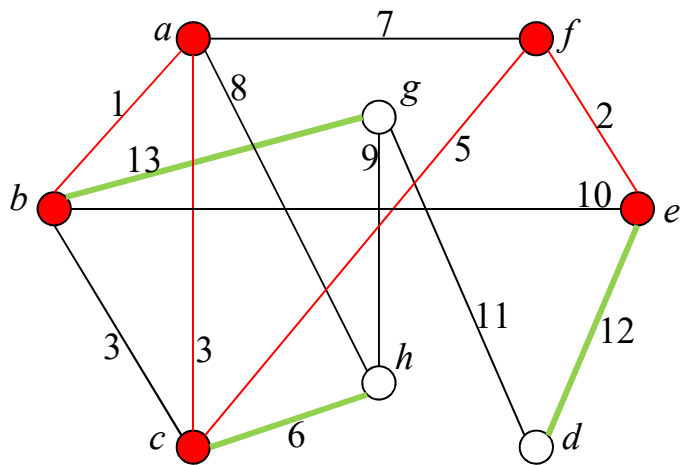
Edge $\{e, f\}$ is the lightest cross edge. So, we add e to S , which is now $S = \{a, b, c, f, e\}$. Add edge $\{e, f\}$ into the MST.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	nil, ∞
e	{e, f}, 2
f	n / a
g	{g, b}, 13
h	{c, h}, 6

Example

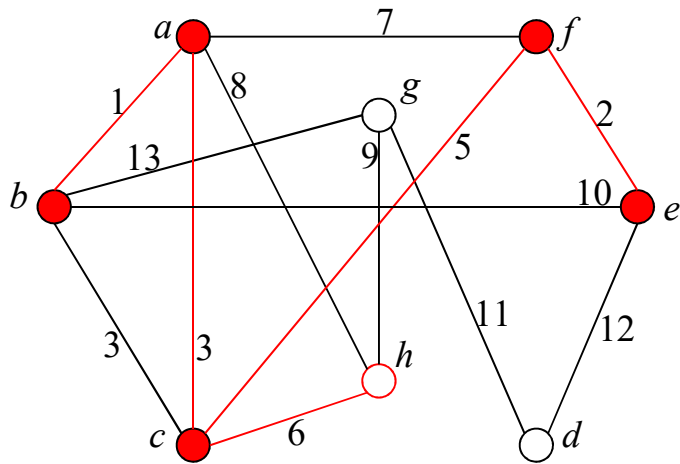
Restore the invariant.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	{e, d}, 12
e	n / a
f	n / a
g	{g, b}, 13
h	{c, h}, 6

Example

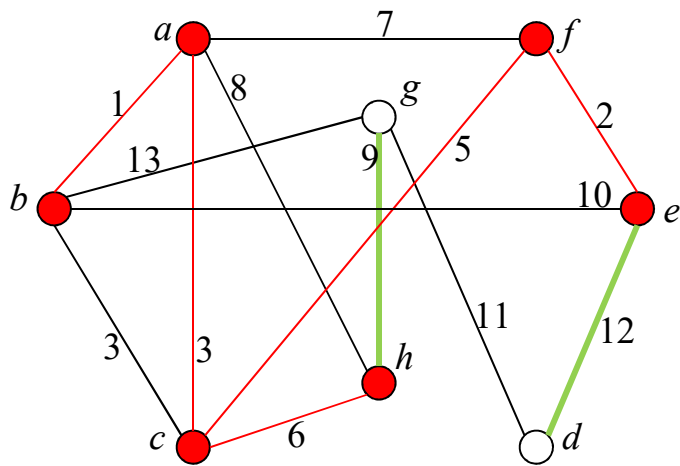
Edge $\{c, h\}$ is the lightest cross edge. So, we add h to S , which is now $S = \{a, b, c, f, e, h\}$. Add edge $\{c, h\}$ into the MST.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	$\{e, d\}, 12$
e	n / a
f	n / a
g	$\{g, b\}, 13$
h	$\{c, h\}, 6$

Example

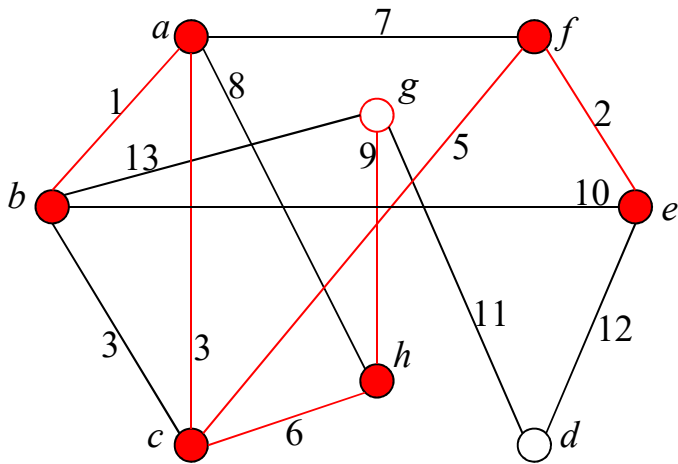
Restore the invariant.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	{e, d}, 12
e	n / a
f	n / a
g	{g, h}, 9
h	n / a

Example

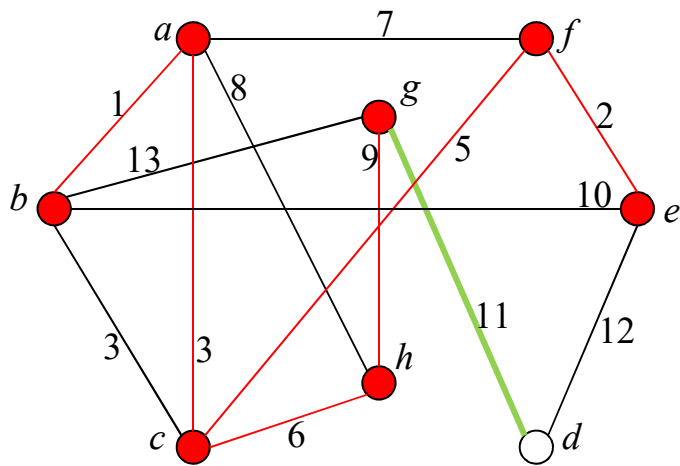
Edge $\{g, h\}$ is the lightest cross edge. So, we add g to S , which is now $S = \{a, b, c, f, e, h, g\}$. Add edge $\{g, h\}$ into the MST.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	{e, d}, 12
e	n / a
f	n / a
g	{g, h}, 9
h	n / a

Example

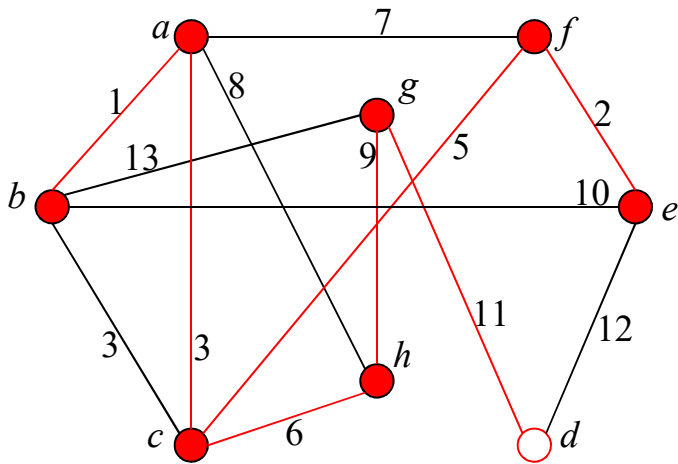
Restore the invariant.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	{d, g}, 11
e	n / a
f	n / a
g	n / a
h	n / a

Example

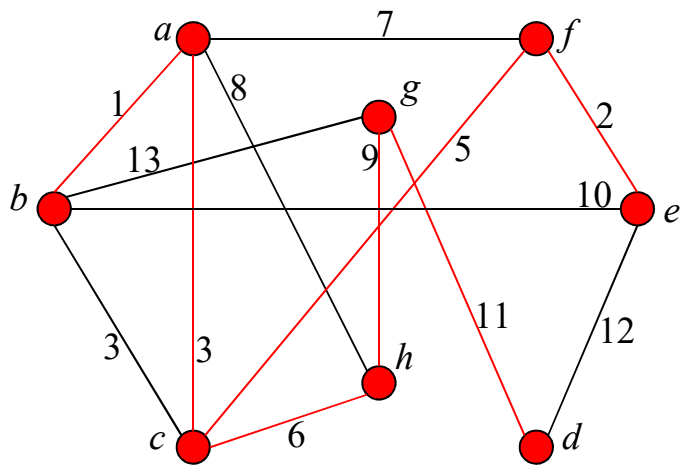
Finally, edge $\{d, g\}$ is the lightest cross edge. So, we add d to S , which is now $S = \{a, b, c, f, e, h, g, d\}$. Add edge $\{d, g\}$ into the MST.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	$\{d, g\}, 11$
e	n / a
f	n / a
g	n / a
h	n / a

Example

We have obtained our final MST.



vertex v	best-cro(v) and weight
a	n / a
b	n / a
c	n / a
d	n / a
e	n / a
f	n / a
g	n / a
h	n / a

Data structure

For a fast implementation, we need a good data structure.

Let P be a set of n tuples of the form $(id, weight, data)$. Design a data structure to support the following operations:

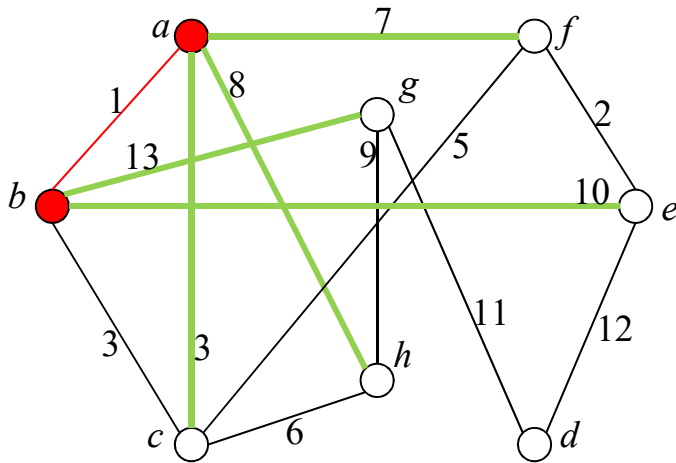
- ✓ **Find**: given an integer t , find the tuple $(id, weight, data)$ from P where $t = id$; return nothing if the tuple does not exist.
- ✓ **Insert**: add a new tuple $(id, weight, data)$ to P .
- ✓ **Delete**: given an integer t , delete the tuple $(id, weight, data)$ from P where $t = id$.
- ✓ **DeleteMin**: remove from P the tuple with the smallest weight.

We can obtain a structure of $O(n)$ space that supports all operations in $O(\log n)$ time. See Problem 4 of Regular Exercise 4.

Data structure operations

Edge $\{a, b\}$ is the lightest of all. $S = \{a, b\}$.

P



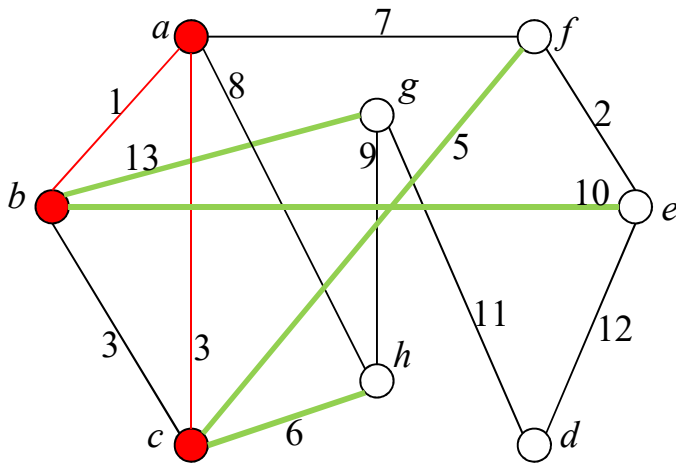
vertex	weight	best-cross
c	3	{c, a}
d	∞	nil
e	10	{e, b}
f	7	{a, f}
g	13	{g, b}
h	8	{a, h}

6 (*id, weight, data*) insertions into P .

In general, $|V| - 2$ insertions in $O(|V| \cdot \log|V|)$ time.

Data structure operations

Edge $\{c, a\}$ is the lightest cross edge. So, we add c to S , which is now $S = \{a, b, c\}$. Add edge $\{c, a\}$ into the MST.



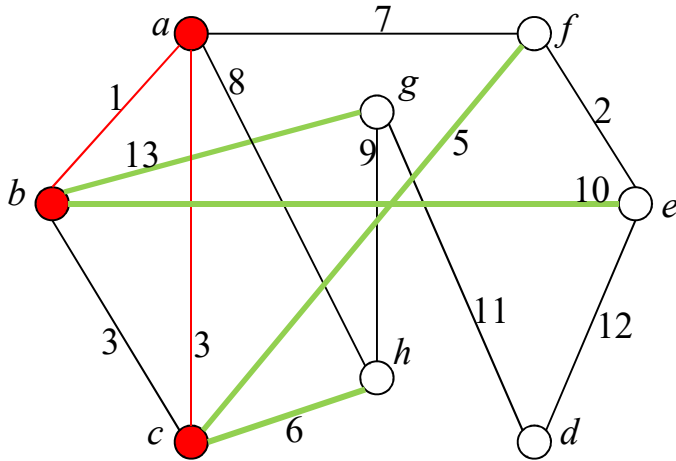
P

vertex	weight (key)	best-cross
ϵ	3	$\{c, a\}$
d	∞	nil
e	10	$\{e, b\}$
f	7	$\{a, f\}$
g	13	$\{g, b\}$
h	8	$\{a, h\}$

Perform DeleteMin to obtain $\{c, a\}$ in $O(\log|V|)$ time.

Data structure operations

Restore the invariant.



<i>P</i>		
vertex	weight	best-cross
d	∞	nil
e	10	{e, b}
f	7 => 5	{a, f} => {c, f}
g	13	{g, b}
h	8 => 6	{a, h} => {c, h}

For edge $\{c, b\}$, perform a find op. using the id of $b \Rightarrow b$ has no tuple in P .

For edge $\{c, a\}$, perform a find op. $\Rightarrow a$ has no tuple in P .

For edge $\{c, f\}$, perform a find op. $\Rightarrow f$ has a tuple with weight 7.

As $\{c, f\}$ is lighter, delete $(f, 7, \{a, f\})$ from P and insert $(f, 5, \{c, f\})$.

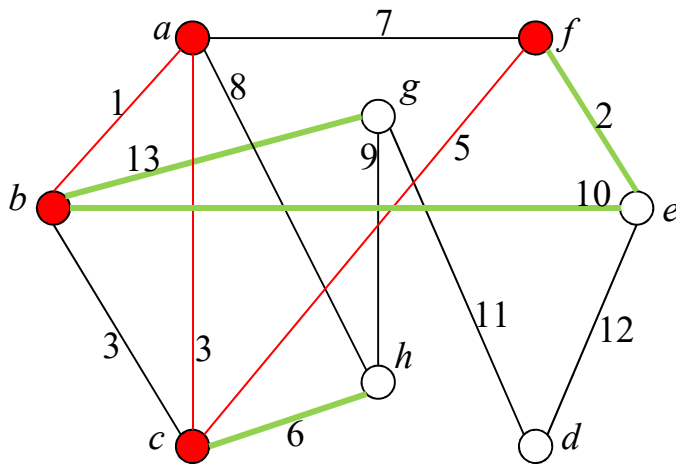
For edge $\{c, h\}$, perform a find op. $\Rightarrow h$ has a tuple with weight 8.

As $\{c, h\}$ is lighter, delete $(h, 8, \{a, h\})$ from P and insert $(h, 6, \{c, h\})$.

Time: $O(d_c \log|V|)$ time where d_c is the degree of c .

Data structure operations

Edge $\{c, f\}$ is the lightest cross edge. So, we add f to S , which is now $S = \{a, b, c, f\}$. Add edge $\{c, f\}$ into the MST.



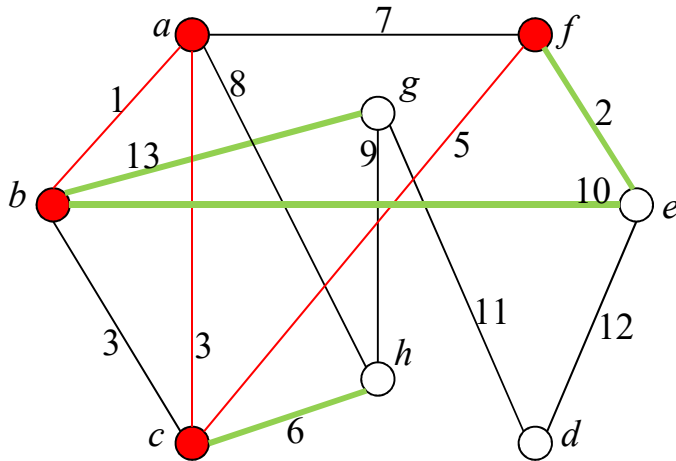
P

vertex	weight	best-cross
d	∞	Nil
e	10	{e, b}
f	5	{e, f}
g	13	{g, b}
h	6	{c, h}

Perform DeleteMin to obtain $\{f, c\}$ in $O(\log|V|)$ time.

Data structure operations

Restore the invariant.



P		
vertex	weight	best-cross
d	∞	Nil
e	10 \Rightarrow 2	$\{e, b\} \Rightarrow \{e, f\}$
g	13	$\{g, b\}$
h	6	$\{c, h\}$

For edge $\{f, a\}$, perform a find op. using the id of $a \Rightarrow a$ has no tuple in P .

For edge $\{f, c\}$, perform a find op. $\Rightarrow c$ has no tuple in P .

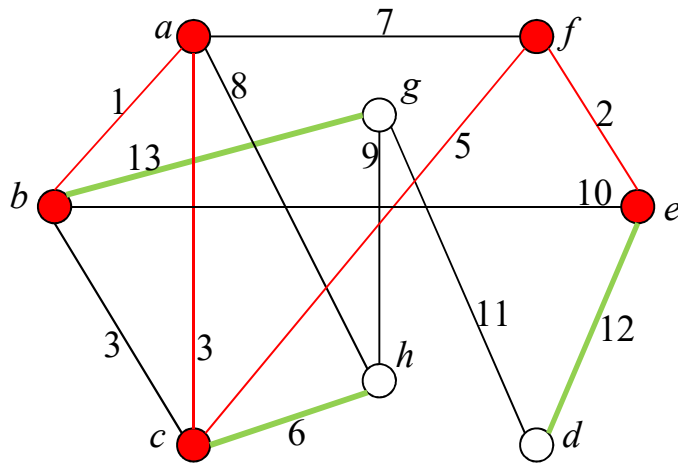
For edge $\{f, e\}$, perform a find op. $\Rightarrow e$ has a tuple with weight 2.

As $\{f, e\}$ is lighter, delete $(e, 10, \{e, b\})$ from P and insert $(e, 2, \{e, f\})$.

Time: $O(d_f \log|V|)$ time where d_f is the degree of f .

Data structure operations

Edge $\{e, f\}$ is the lightest cross edge. So, we add e to S , which is now $S = \{a, b, c, f, e\}$. Add edge $\{e, f\}$ into the MST.

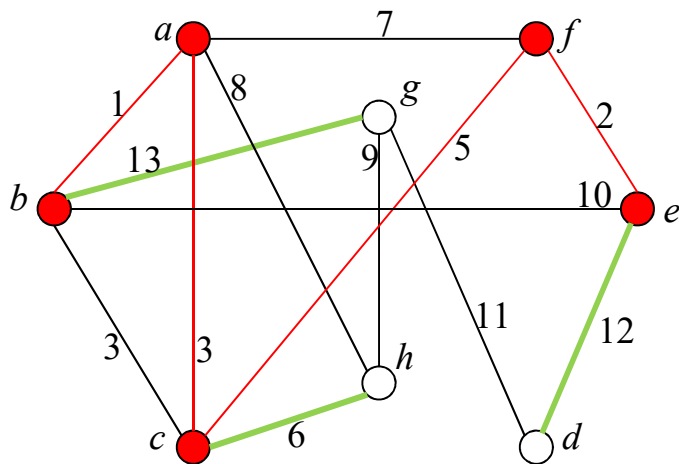


<i>P</i>		
vertex	weight	best-cross
d	∞	Nil
e	2	{e, f}
g	13	{g, b}
h	6	{c, h}

Perform DeleteMin to obtain $\{e, f\}$ in $O(\log|V|)$ time.

Data structure operations

Restore the invariant.



P

vertex	weight	best-cross
d	$\infty \Rightarrow 12$	Nil $\Rightarrow \{e,d\}$
g	13	$\{g, b\}$
h	6	$\{c, h\}$

For edge $\{e, f\}$, perform a find op. using the id of $f \Rightarrow f$ has no tuple in P .

For edge $\{e, b\}$, perform a find op. $\Rightarrow b$ has no tuple in P .

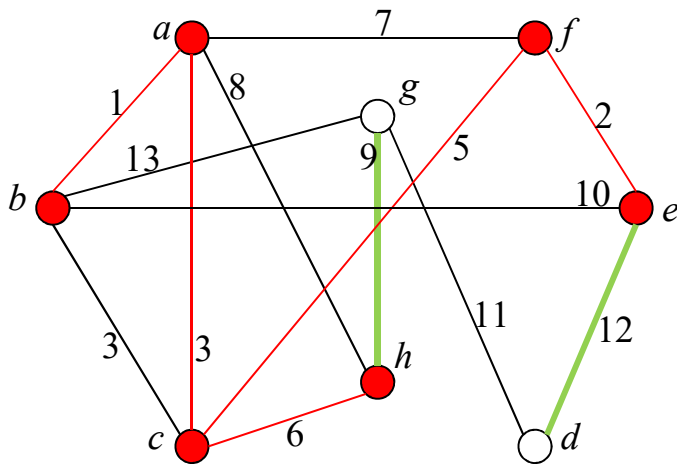
For edge $\{e, d\}$, perform a find op. $\Rightarrow d$ has a tuple with weight ∞ .

As $\{e, d\}$ is lighter, delete (d, ∞, Nil) from P and insert $(d, 12, \{e, d\})$.

Time: $O(d_e \log|V|)$ time where d_e is the degree of e .

Data structure operations

Edge $\{c, h\}$ is the lightest cross edge. So, we add h to S , which is now $S = \{a, b, c, f, e, h\}$. Add edge $\{c, h\}$ into the MST.



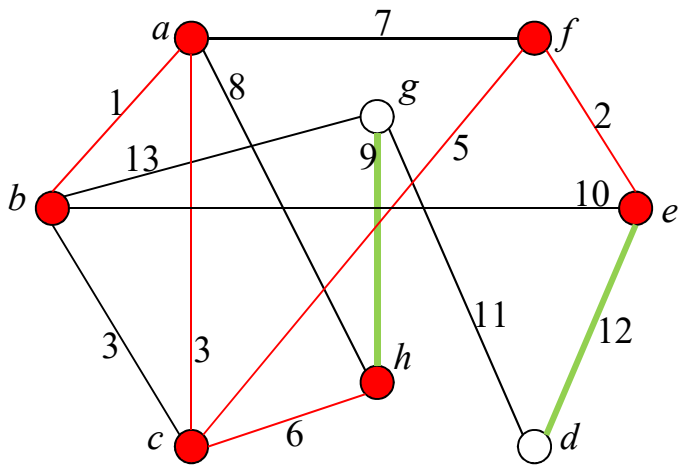
P

vertex	weight	best-cross
d	12	{e,d}
g	13	{g, b}
h	6	{e, h}

Perform DeleteMin to obtain $\{c, h\}$ in $O(\log|V|)$ time.

Example

Restore the invariant.



<i>P</i>		
vertex	weight	best-cross
d	12	{e,d}
g	13 => 9	{g, b} => {g,h}

For edge $\{h, a\}$, perform a find op. using the id of $a \Rightarrow a$ has no tuple in P .

For edge $\{h, c\}$, perform a find op. $\Rightarrow c$ has no tuple in P .

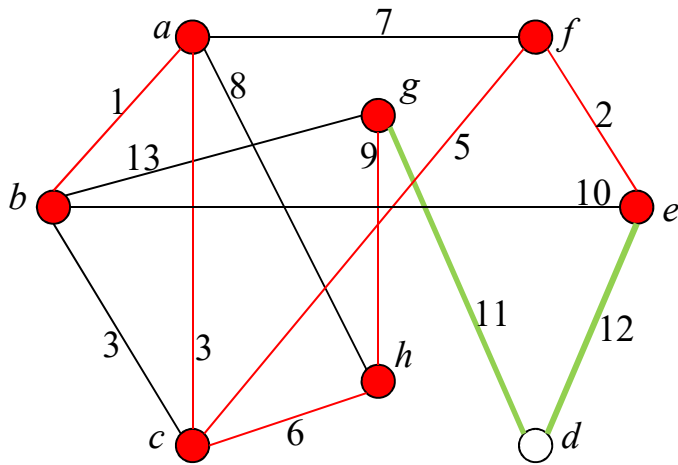
For edge $\{h, g\}$, perform a find op. $\Rightarrow g$ has a tuple with weight 13.

As $\{h, g\}$ is lighter, delete $(g, 13, \{g, b\})$ from P and insert $(g, 9, \{g, h\})$.

Time: $O(d_h \log|V|)$ time where d_h is the degree of h .

Example

Edge $\{g, h\}$ is the lightest cross edge. So, we add g to S , which is now $S = \{a, b, c, f, e, h, g\}$. Add edge $\{g, h\}$ into the MST.

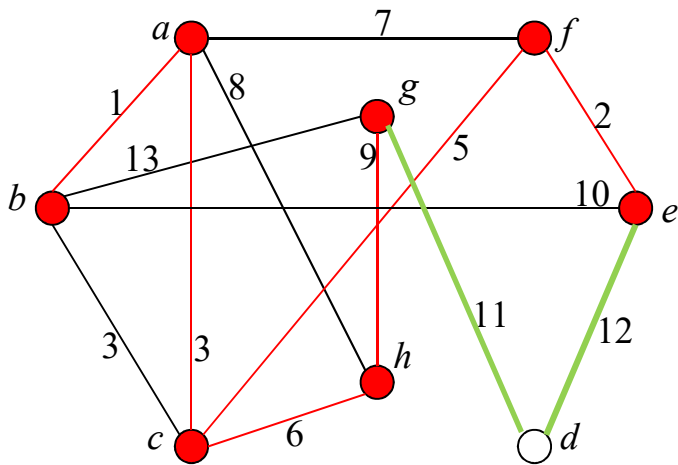


<i>P</i>		
vertex	weight	best-cross
d	12	{e,d}
g	9	{g,h}

Perform DeleteMin to obtain $\{g, h\}$ in $O(\log|V|)$ time.

Example

Restore the invariant.



<i>P</i>		
vertex	weight	best-cross
d	12=>11	{e,d}=>{g,d}

For edge $\{g, b\}$, perform a find op. using the id of $b \Rightarrow b$ has no tuple in P .

For edge $\{g, h\}$, perform a find op. $\Rightarrow h$ has no tuple in P .

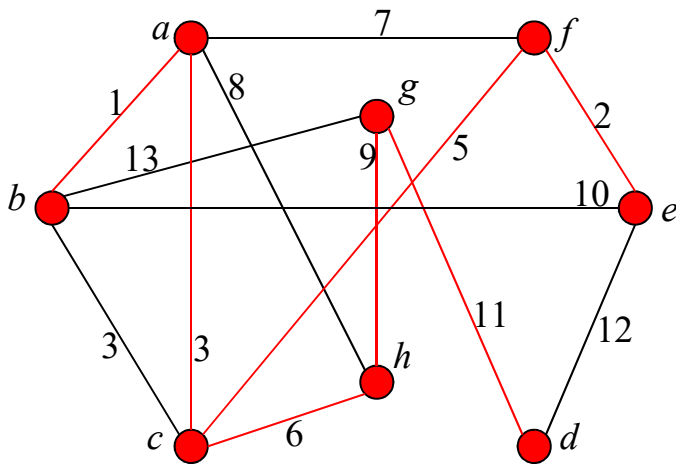
For edge $\{g, d\}$, perform a find op. $\Rightarrow d$ has a tuple with weight 12.

As $\{g, d\}$ is lighter, delete $(d, 12, \{e, d\})$ from P and insert $(g, 11, \{g, d\})$.

Time: $O(d_g \log|V|)$ time where d_g is the degree of g .

Example

Finally, edge $\{g, d\}$ is the lightest cross edge. So, we add d to S , which is now $S = \{a, b, c, f, e, h, g, d\}$. Add edge $\{g, d\}$ into the MST.

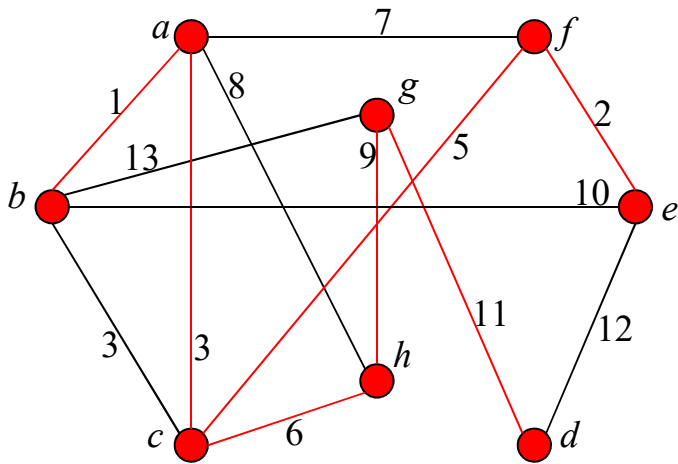


<i>P</i>		
vertex	weight	best-cross
d	11	$\{g, d\}$

Perform DeleteMin to obtain $\{g, d\}$ in $O(\log|V|)$ time.

Example

We have obtained our final MST.



Total time:

$$\begin{aligned} &O(|V| \cdot \log|V| + \sum_{v \in V} \log|V| + \\ &\sum_{v \in V} d_v \log|V|) \\ &= O((2|V| + 2|E|) \cdot \log|V|) \\ &= O((|V| + |E|) \cdot \log|V|) \end{aligned}$$