Minimum Spanning Trees

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?

Undirected Weighted Graphs

Let $G = (V, E)$ be an undirected graph. Let w be a function that maps each edge of G to a positive integer value. Specifically, for each edge e, $w(e)$ is a positive integer value, which we call the weight of e.

An undirected weighted graph is defined as the pair (G, w).

We will denote an edge between vertices u and v in G as $\{u, v\}$—instead of (u, v)—to emphasize that the ordering of u, v does not matter.

We consider that G is connected, namely, there is a path between any two vertices in V.

Spanning Trees

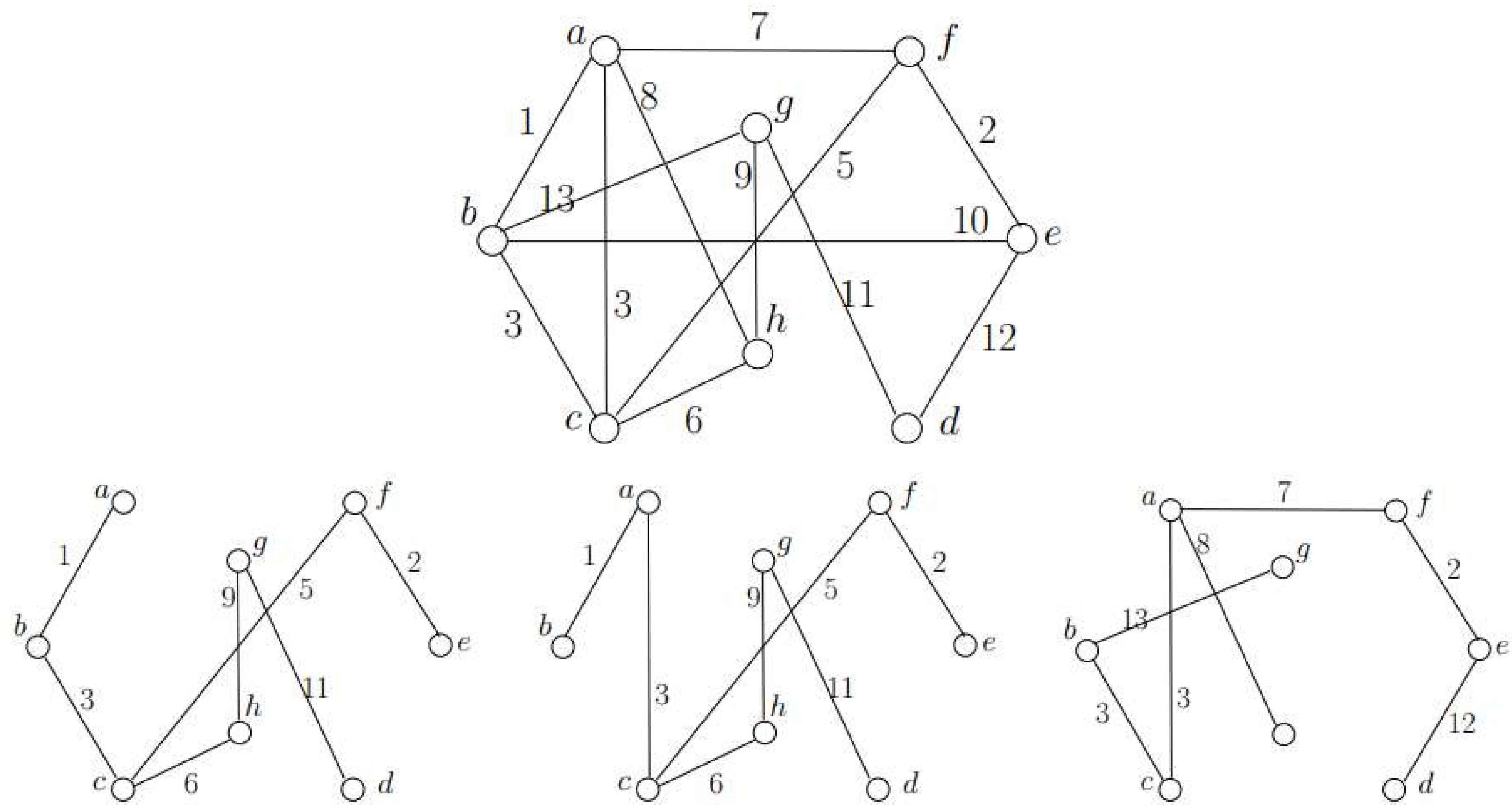
Remember that a tree is defined as a connected undirected graph with no cycles.

Given a connected undirected weighted graph (G, w) with $G = (V, E)$, 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 defined as the sum of the weights of all the edges in T (note that T must have $|V| - 1$ edges).

Example



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

The Minimum Spanning Tree 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.

Such a tree is called an MST of (G, w).

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 extension edge.

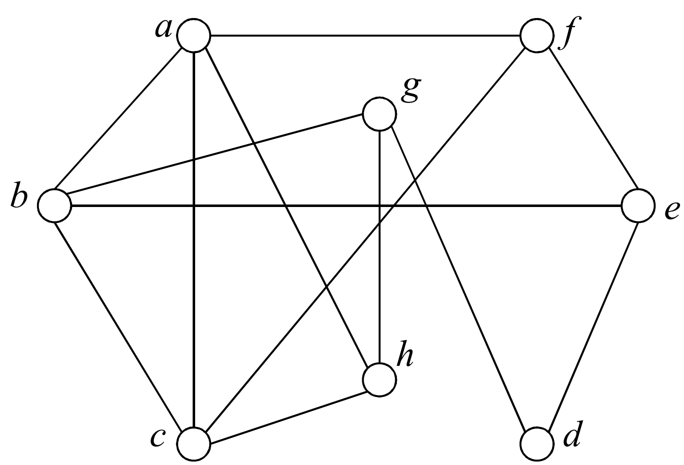
Prim's algorithm

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 an extension edge.



V

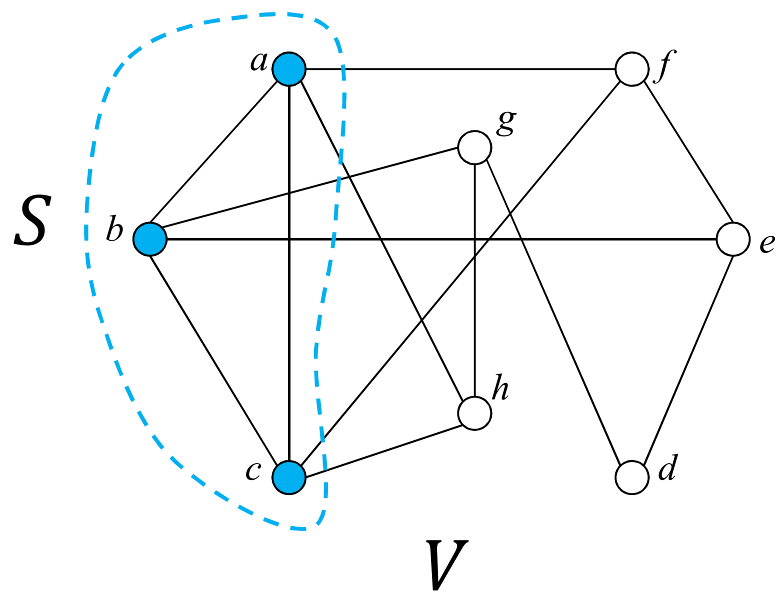
Prim's algorithm

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 extension edge.



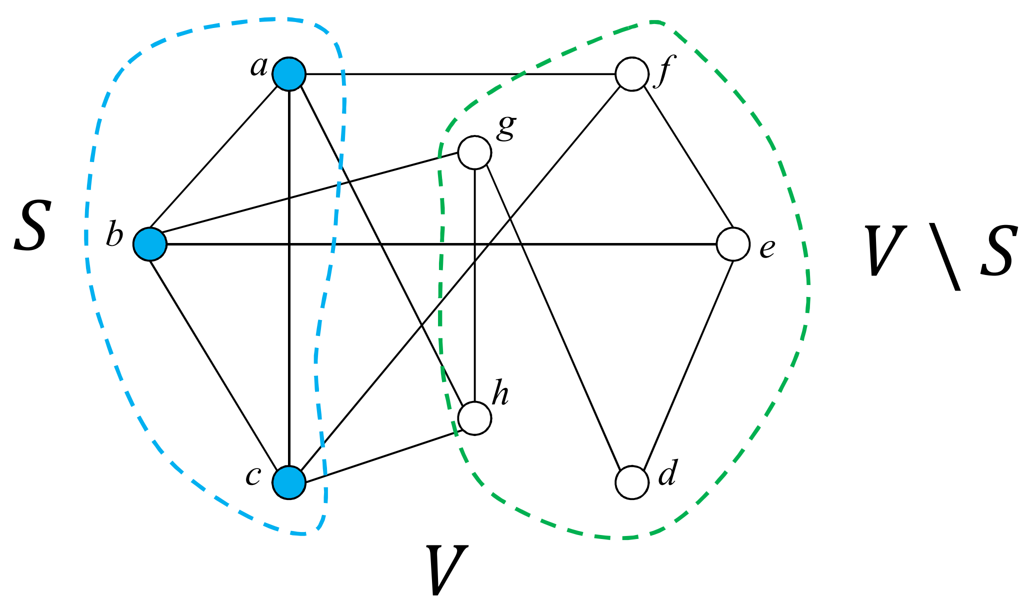
Prim's algorithm

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 an extension edge.



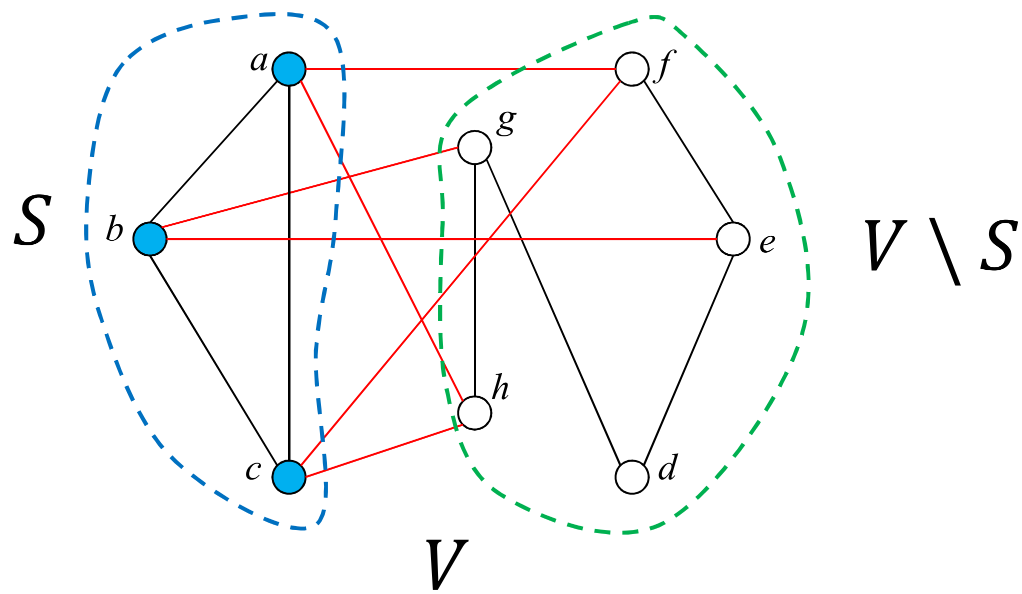
Prim's algorithm

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 an extension edge.



Prim's algorithm

At all times, the algorithm enforces the following **lightest extension principle**:

- For every vertex $v \in V \setminus S$, it remembers which extension edge of v has the smallest weight—referred to as the lightest extension edge of v, and denoted as *best-ext*(v).

Prim's algorithm

1. Let $\{u, v\}$ be 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 the lightest extension principle:
   - For every vertex z of $V \setminus S$
     - If z is a neighbor of u, but not of v

       best-ext(z) = edge $\{z, u\}$
     - If z is a neighbor of v, but not of u

       best-ext(z) = edge $\{z, v\}$
     - Otherwise

       best-ext(z) = the lighter edge between $\{z, u\}$ and $\{z, v\}$

Prim's algorithm

4. Repeat the following until $S = V$:

 5. Get an extension 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, we restore the lightest extension principle. */

 7. for every edge $\{v, z\}$ of v :

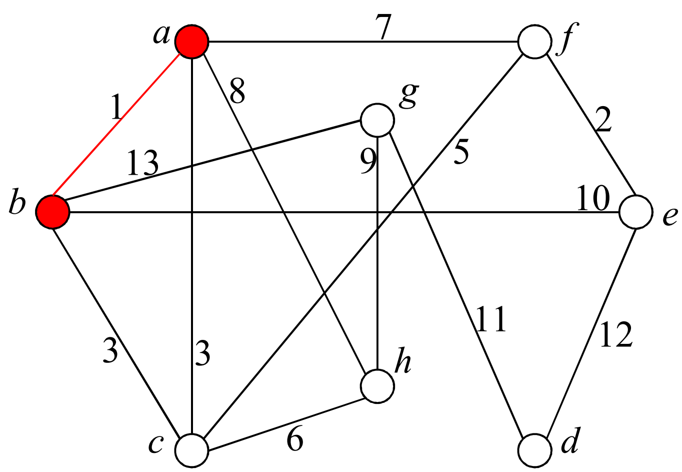
 - If $z \notin S$ then

 If *best-ext*(z) is heavier than edge $\{v, z\}$ then

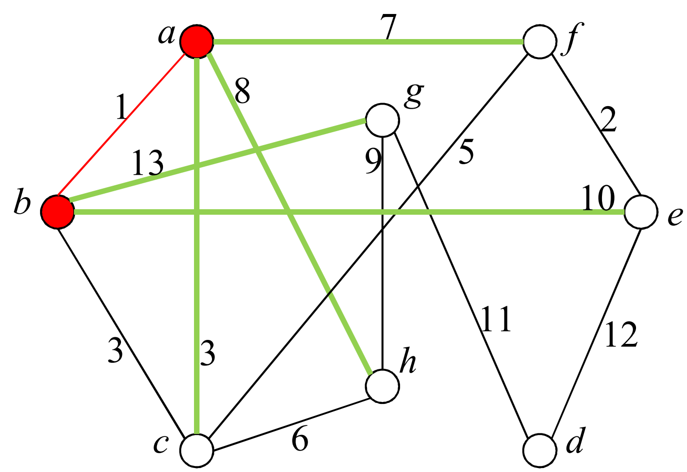
 Set *best-ext*(z) = edge $\{v, z\}$

Example

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



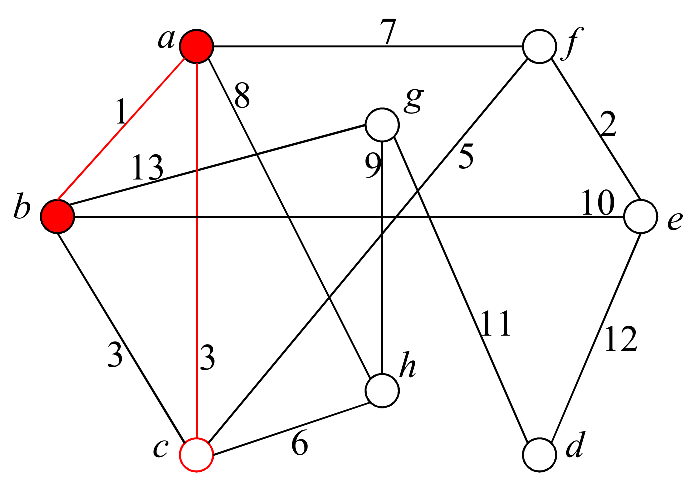
Example



vertex v	best-ext(v) 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

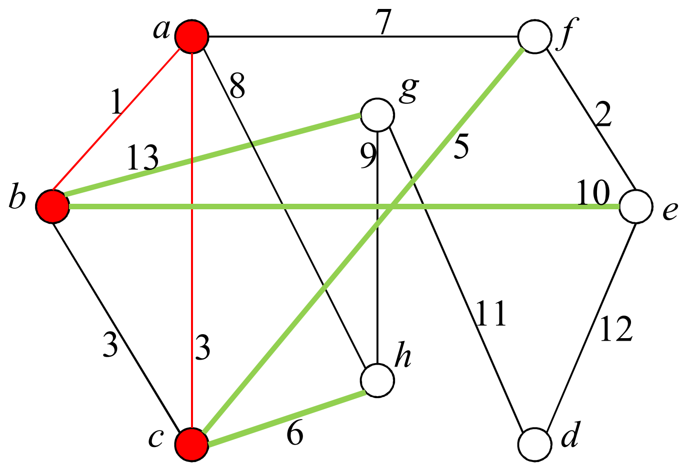
Edge $\{c, a\}$ is the lightest extension edge. So, we add c to S, which is now $S = \{a, b, c\}$. Add edge $\{c, a\}$ into the MST.



vertex v	best-ext(v) 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

Note: Edges $\{c, a\}$ and $\{c, b\}$ have the same weight. Either of them can be *best-ext*(c).

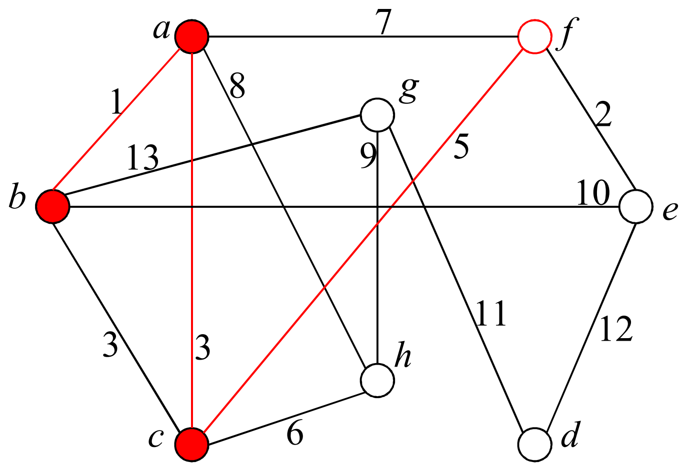
Example



vertex v	best-ext(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

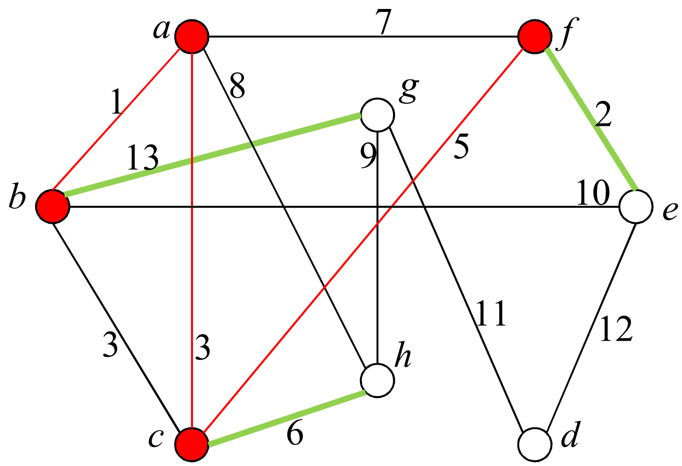
Example

Edge $\{c, f\}$ is the lightest extension 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-ext(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

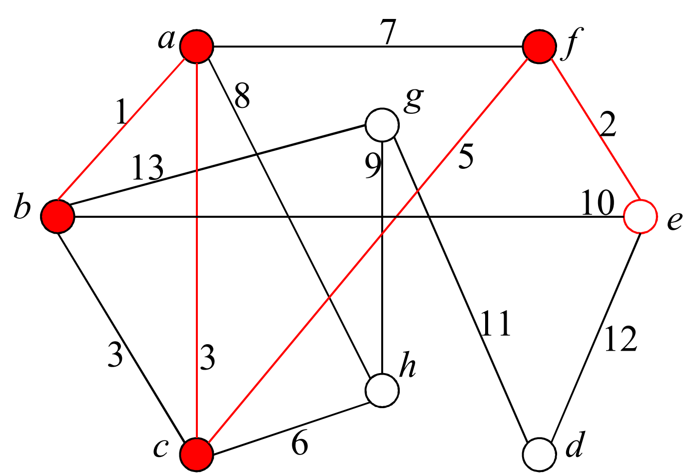
Example



vertex v	best-ext(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

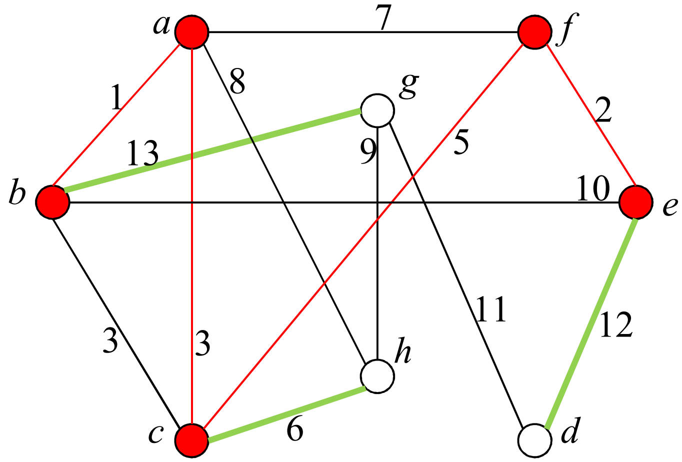
Example

Edge $\{e, f\}$ is the lightest extension 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-ext(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

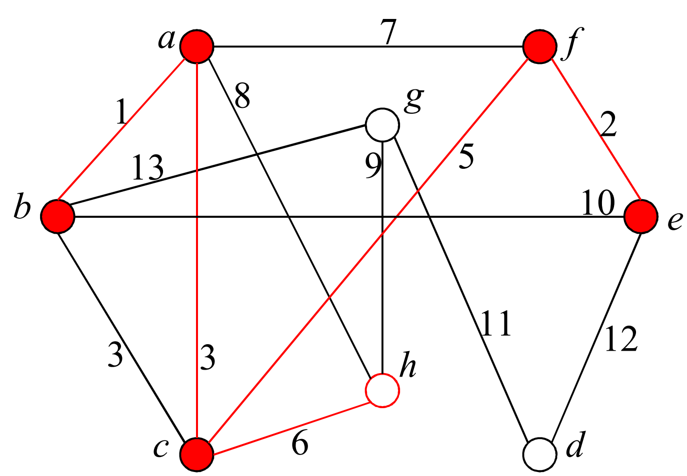
Example



vertex v	best-ext(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

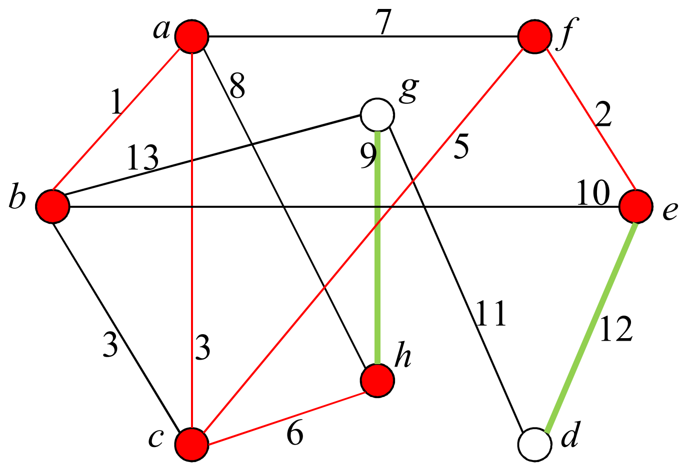
Example

Edge $\{c, h\}$ is the lightest extension 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-ext(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

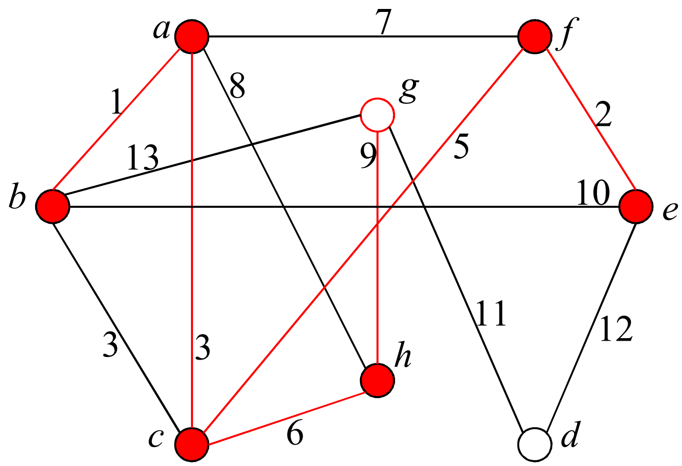
Example



vertex v	best-ext(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

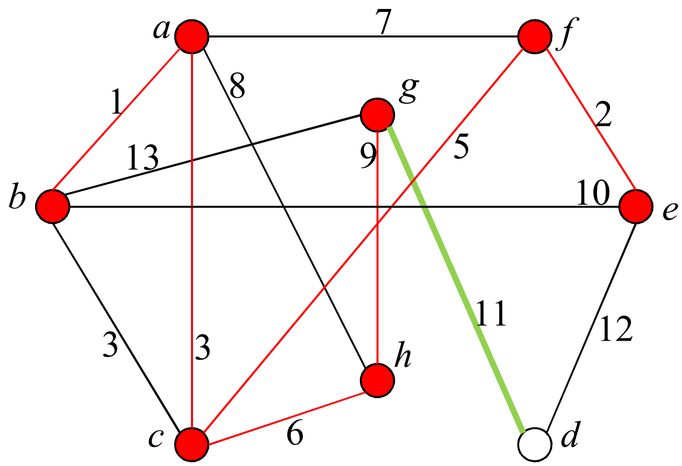
Example

Edge $\{g, h\}$ is the lightest extension 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-ext(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

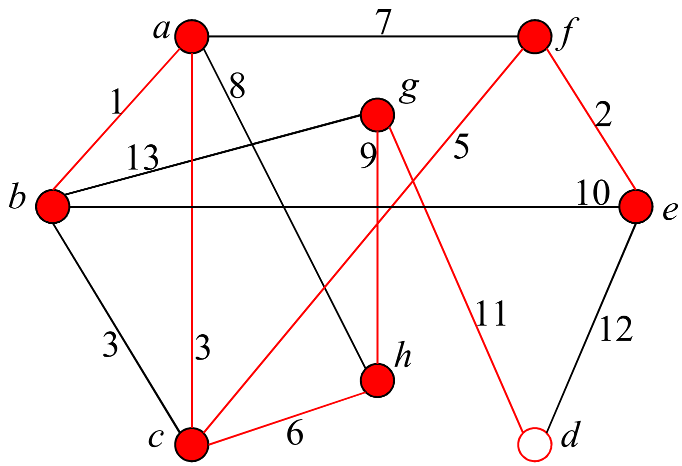
Example



vertex v	best-ext(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

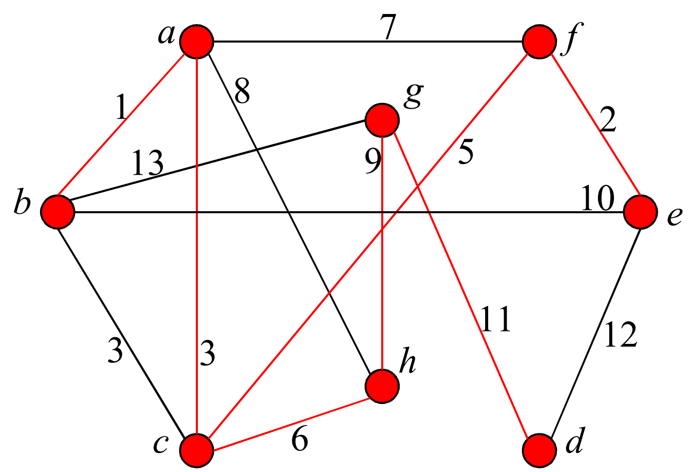
Finally, edge $\{d, g\}$ is the lightest extension edge. So, we add d to S, which is now $S = \{a, b, c, f, e, h, g, d\}$. Add edge $\{d, g\}$ into the MST.



vertex v	best-ext(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-ext(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

It remains to discuss how to design a good data structure to support all the operations we need. Let us consider a stand-alone problem which captures the essence of what we need.

Let S be a set of integer pairs of the form (id, key). Design a data structure that supports the following operations:

- Insert: add a new pair (id, key) to S (you can assume that S does not already have a pair with the same id).
- Delete: given an integer t, delete the pair (id, key) from S where $t = id$, if such a pair exists.
- DeleteMin: remove from S the pair with the smallest key, and return it.

The structure must consume $O(n)$ space, and support all operations in $O(\log n)$ time where $n = |S|$.

Binary Search Tree (BST)

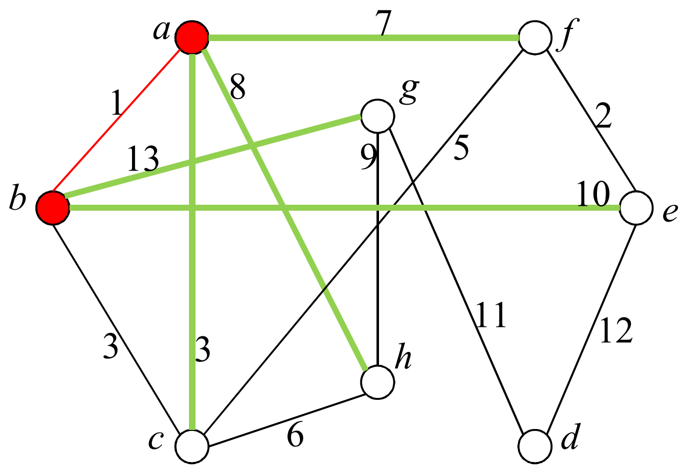
Balanced BST guarantees:

- $O(n)$ space consumption.
- $O(\log n)$ time per predecessor query (hence, also per dictionary lookup).
- $O(\log n)$ time per insertion
- $O(\log n)$ time per deletion

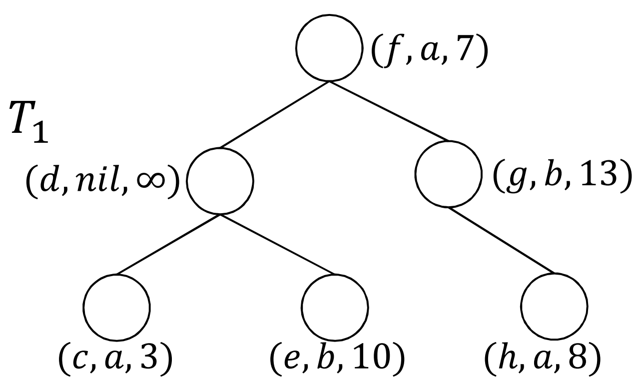
where $n = |S|$. Note that all the above complexities hold in the worst case.

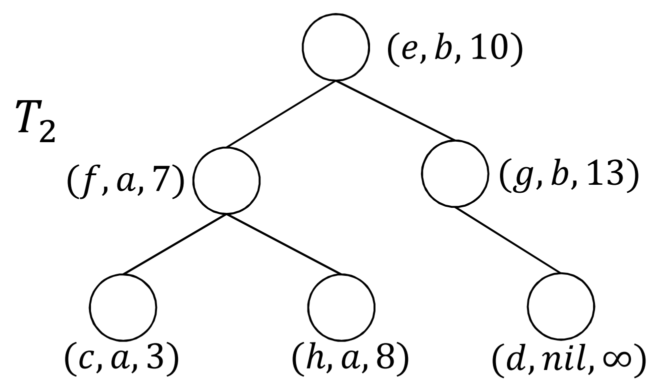
Example

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



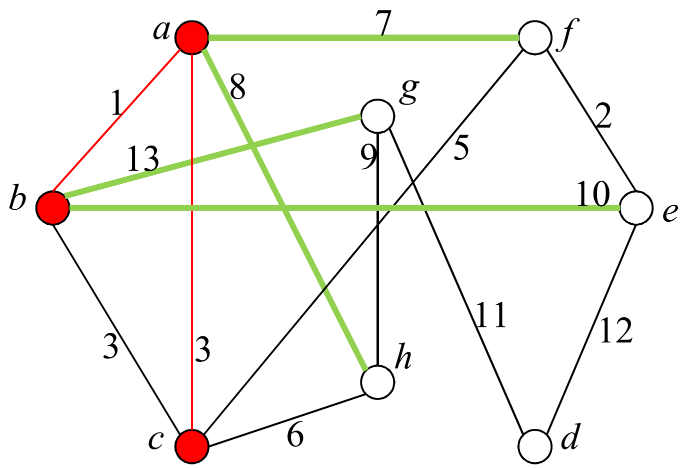
Insert all the vertices in $V \setminus S$ to construct BST which needs at most $O(|V| \cdot \log|V|)$ time.



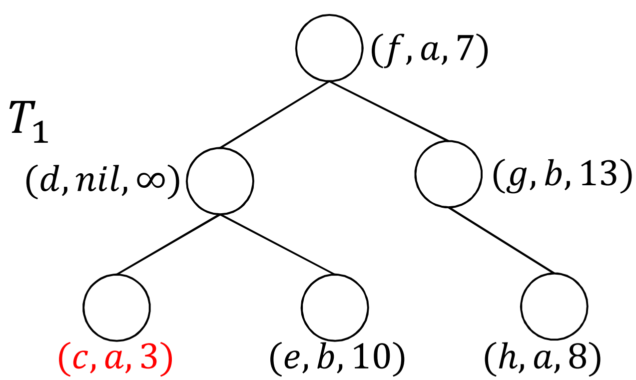


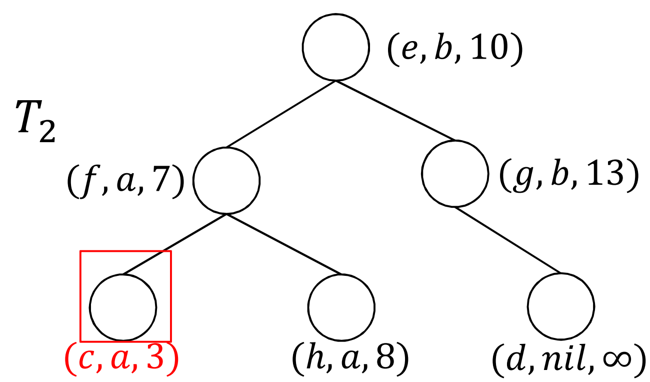
Example

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



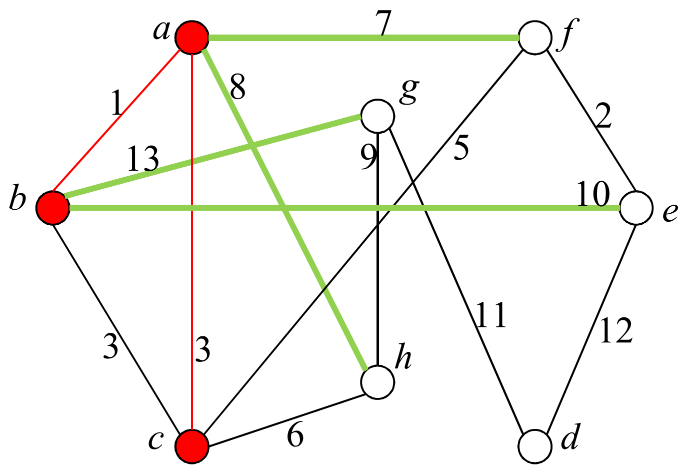
Perform DeleteMin to obtain $(c, a, 3)$ in $O(\log|V|)$ time.



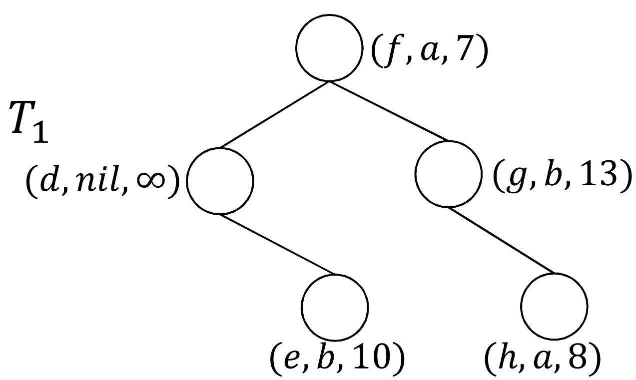


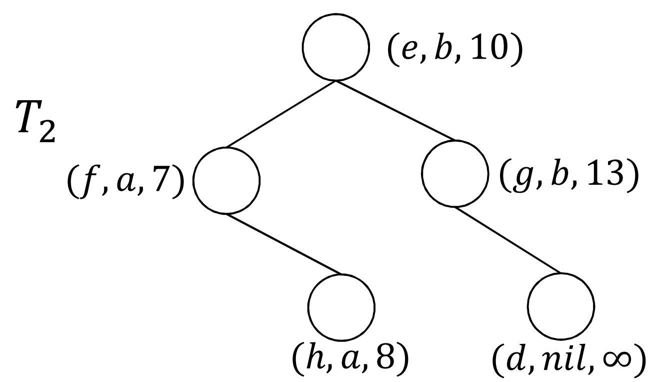
Example

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

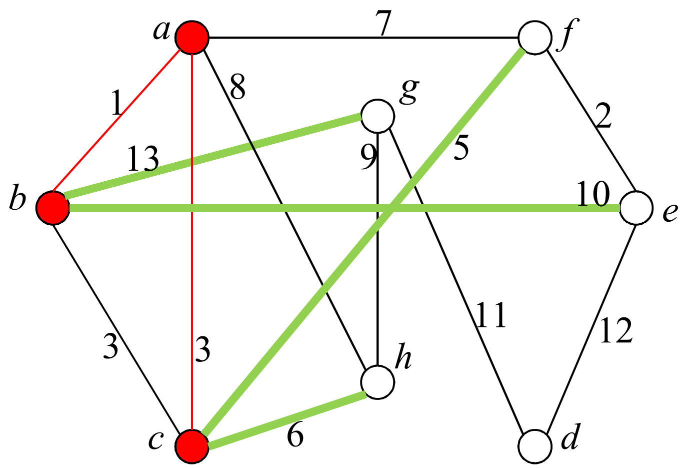


Perform DeleteMin to obtain $(c, a, 3)$ in $O(\log|V|)$ time.

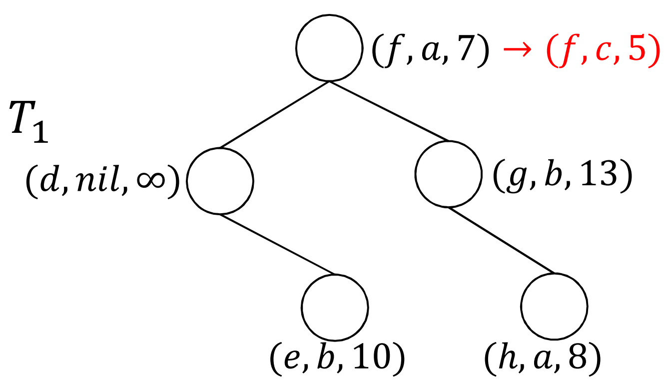


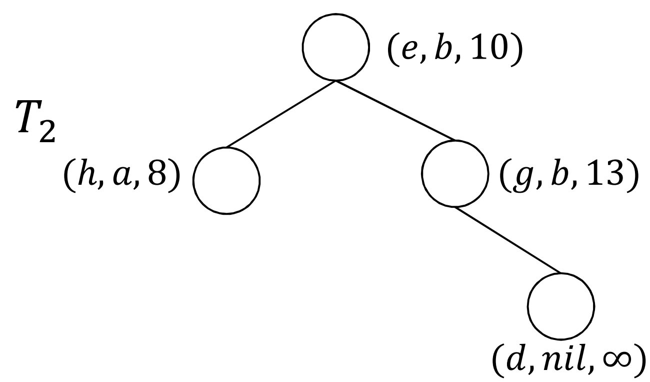


Example

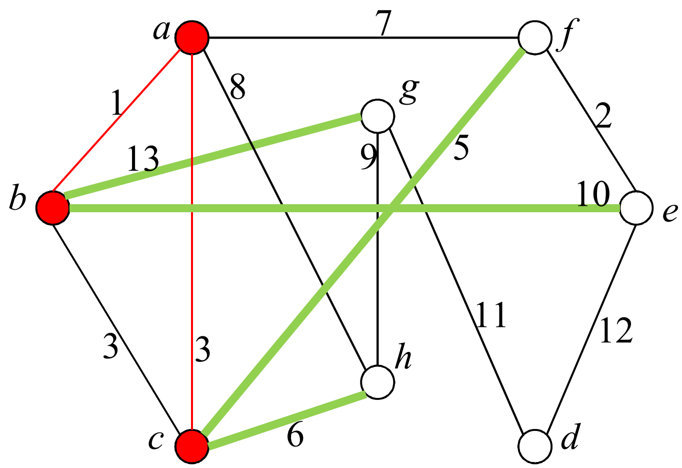


For edge (u, c) where $u \notin S$,
$u = f$, perform Delete and Insert
in $O(\log|V|)$ time.

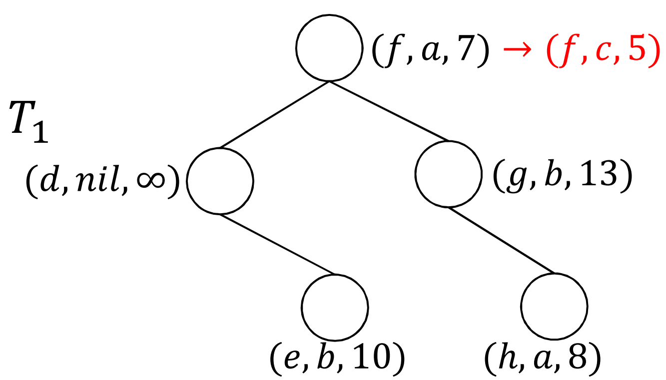


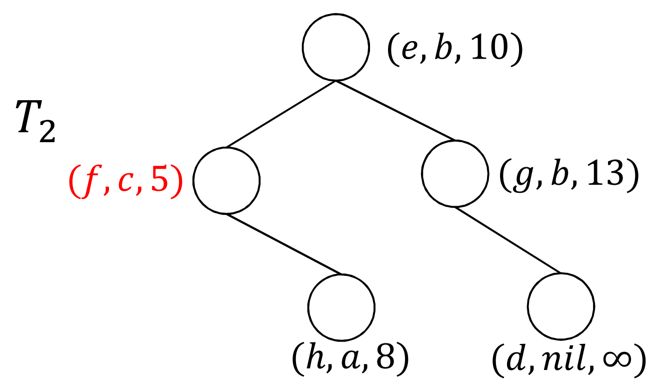


Example

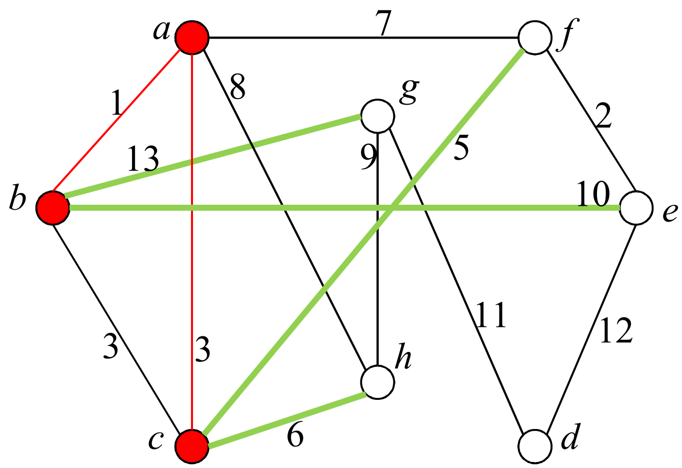


For edge (u, c) where $u \notin S$,
$u = f$, perform Delete and Insert in $O(\log|V|)$ time.

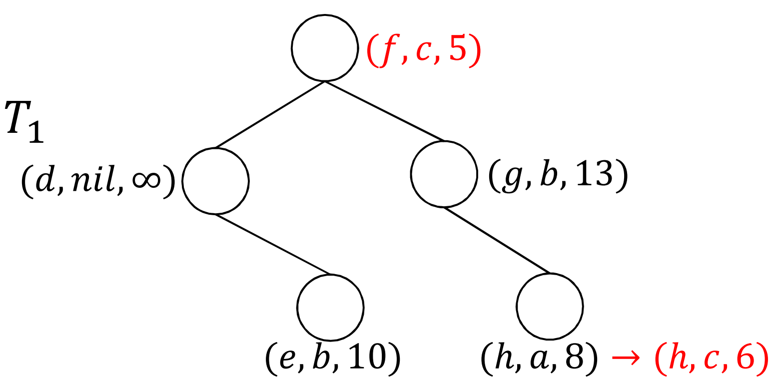


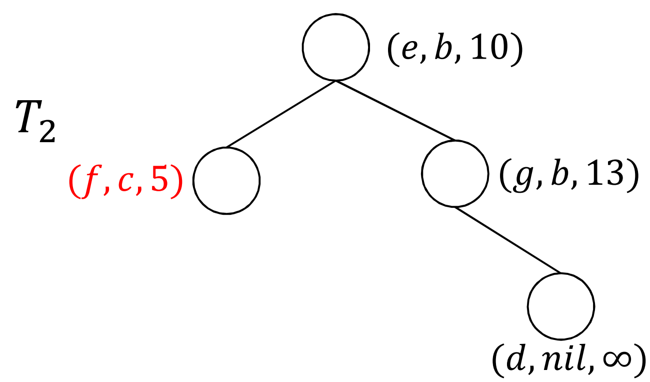


Example

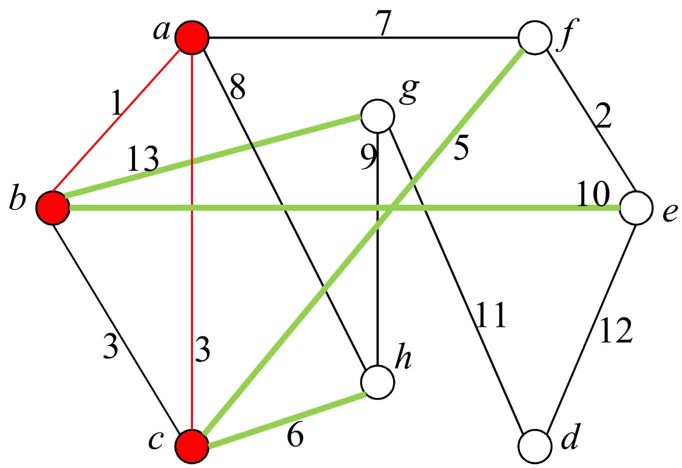


For edge (u, c) where $u \notin S$, $u = h$, perform Delete and Insert in $O(\log|V|)$ time.

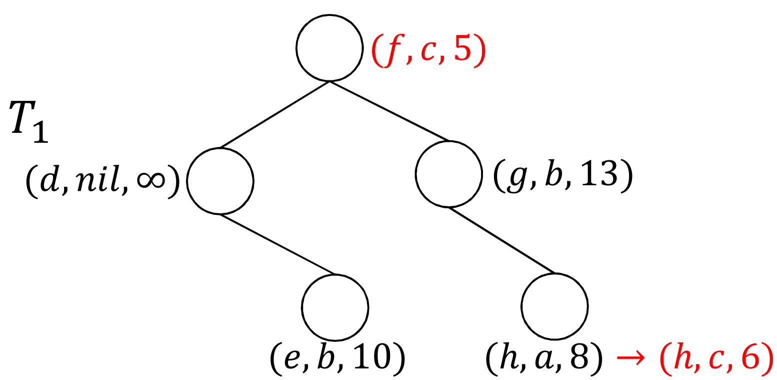


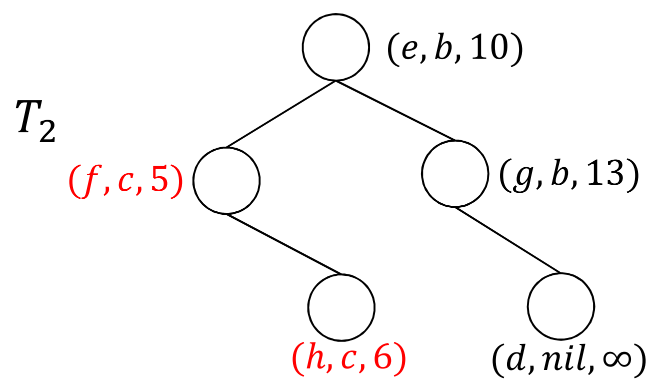


Example

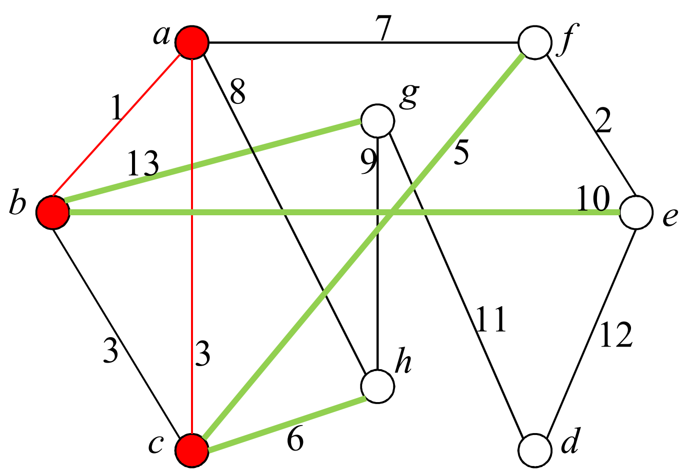


For edge (u, c) where $u \notin S$,
$u = h$, perform Delete and Insert in $O(\log|V|)$ time.

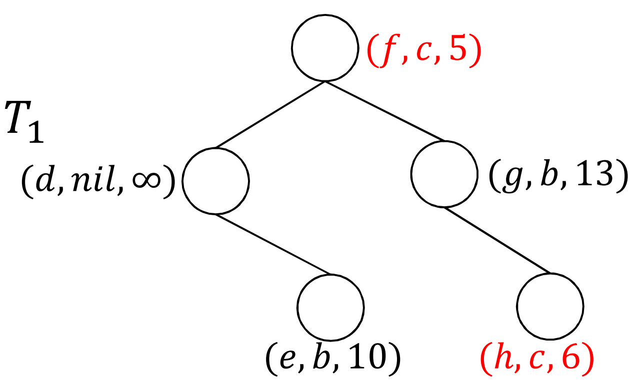


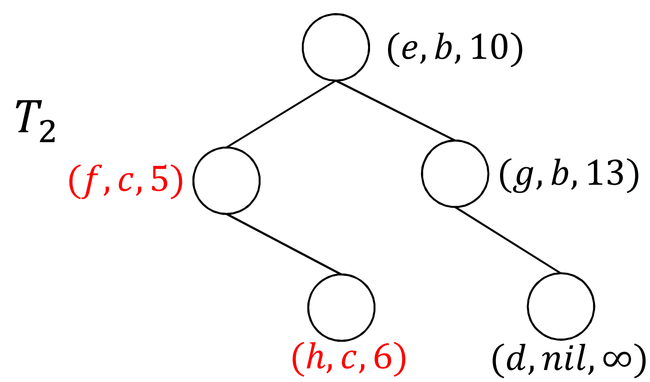


Example



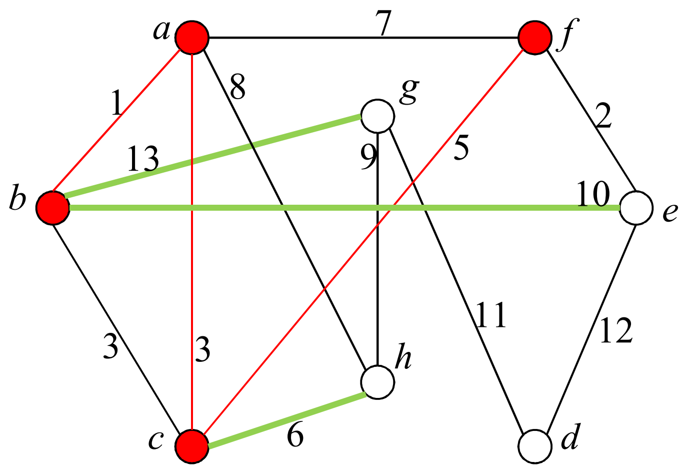
For each edge (u, c) where $u \notin S$, perform Delete and Insert in $O(d_c \cdot \log|V|)$ time, where d_c is the number of edges of c.



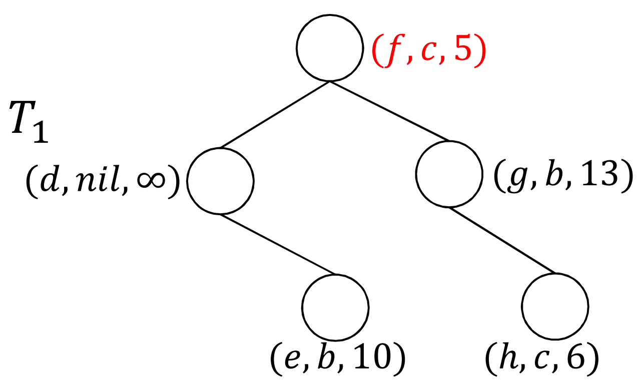


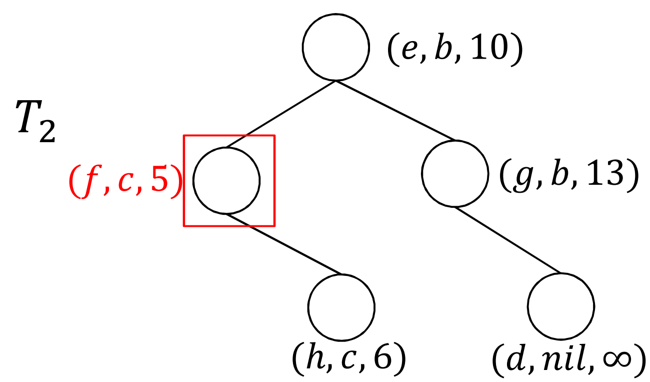
Example

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



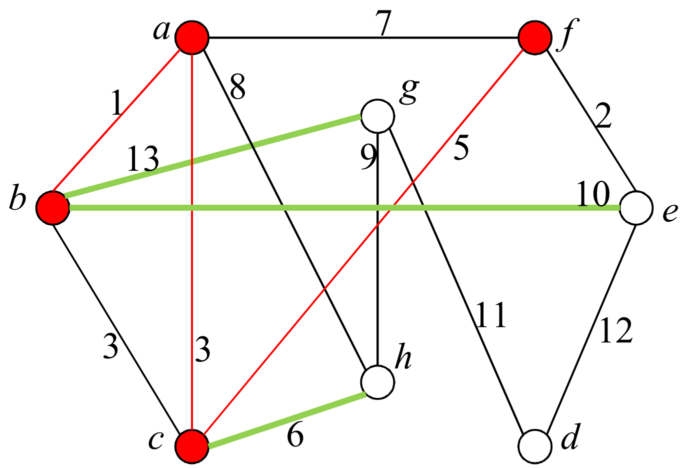
Perform DeleteMin to obtain $(f, c, 5)$ in $O(\log|V|)$ time.



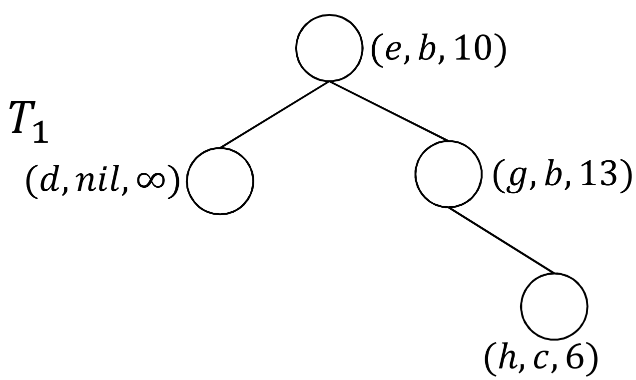


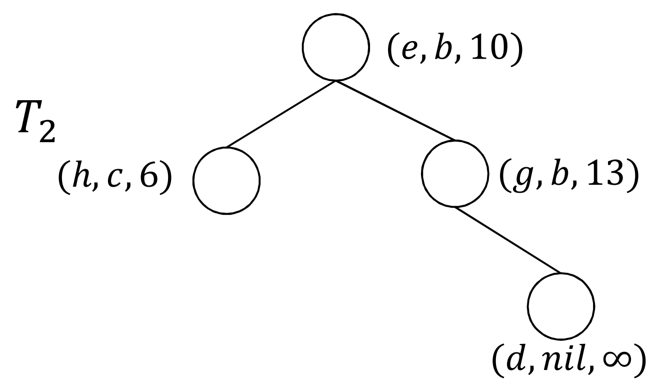
Example

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

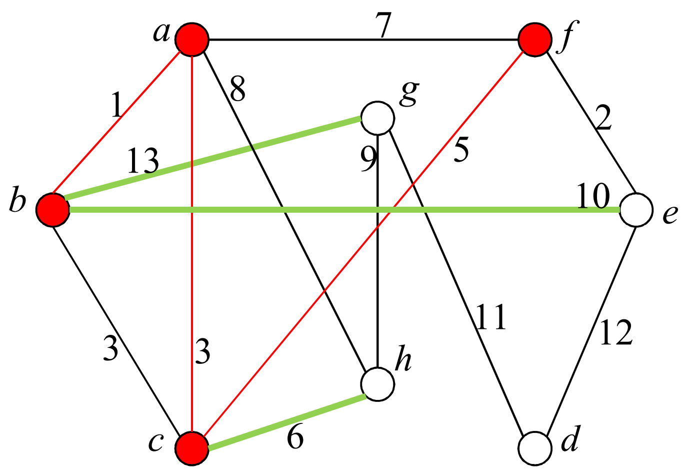


Perform DeleteMin to obtain $(f, c, 5)$ in $O(\log|V|)$ time.

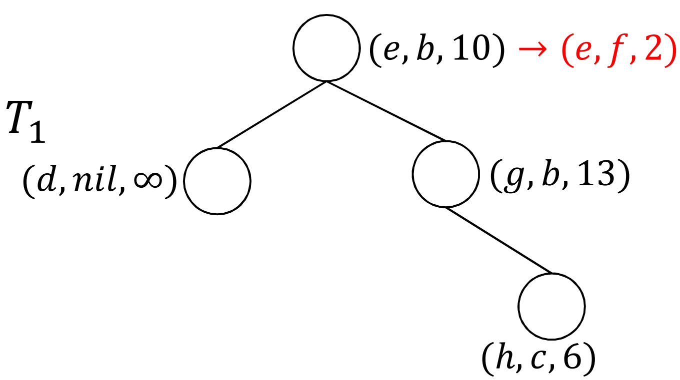


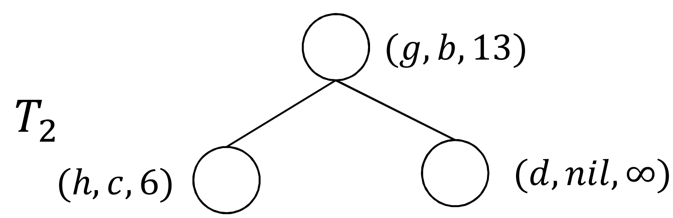


Example

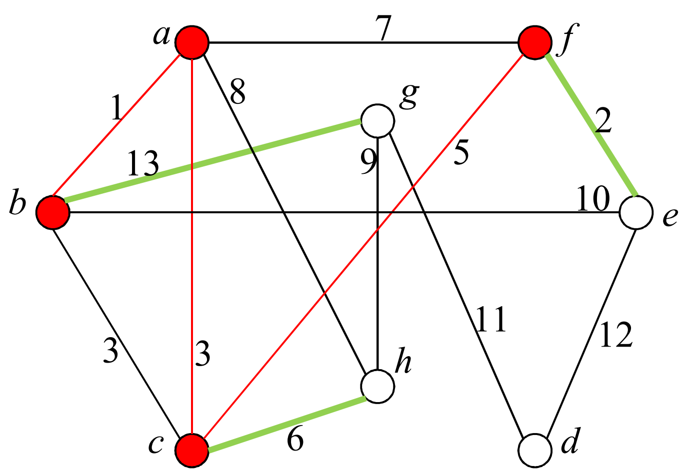


For edge (u, f) where $u \notin S$, $u = e$, perform Delete and Insert in $O(\log|V|)$ time.

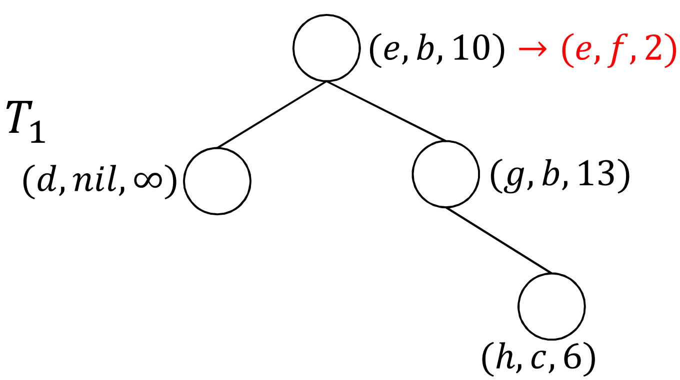


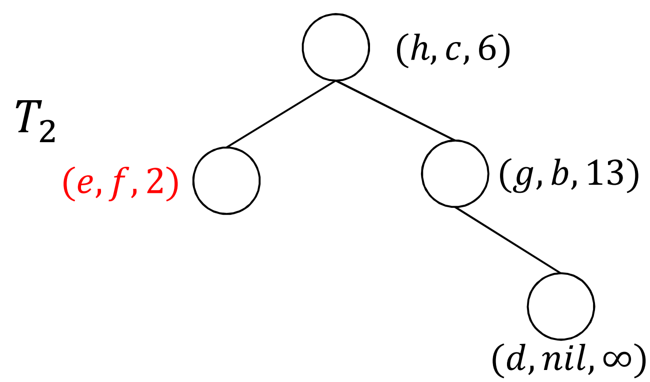


Example

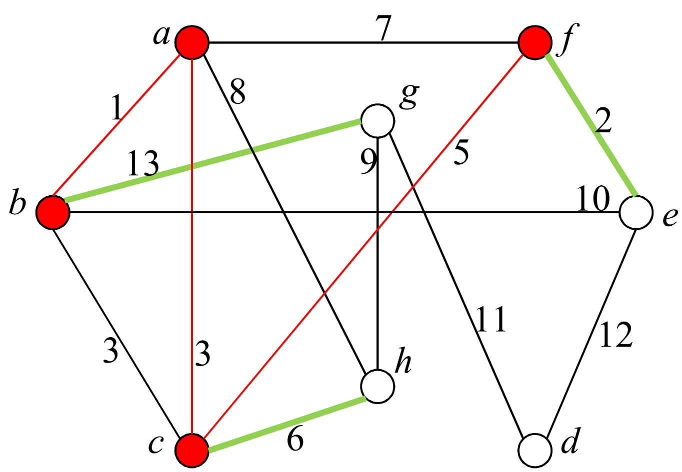


For edge (u, f) where $u \notin S$, $u = e$, perform Delete and Insert in $O(\log|V|)$ time.

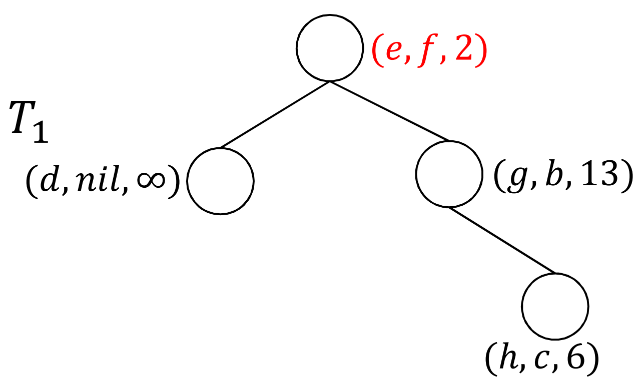


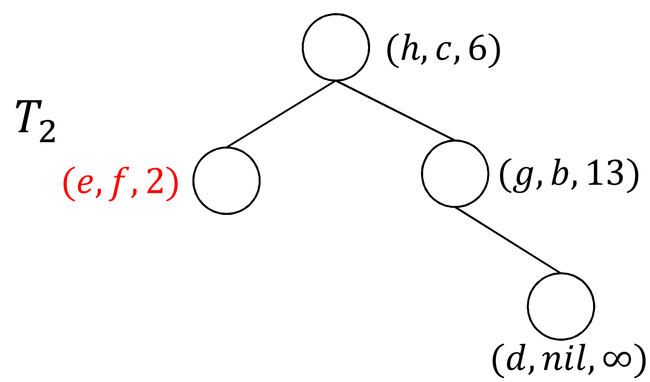


Example



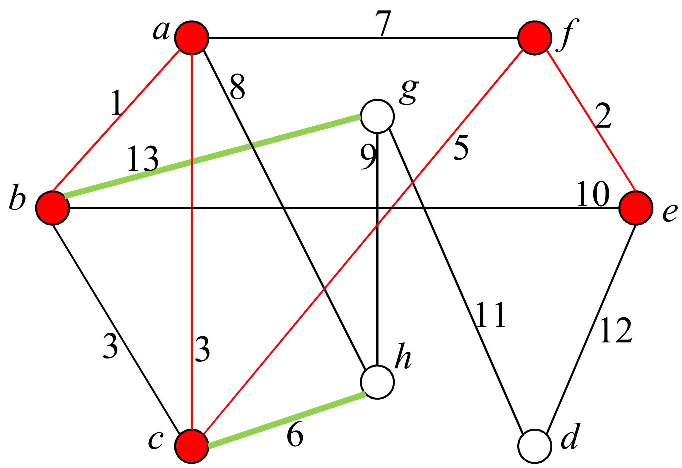
For each edge (u, f) where $u \notin S$, perform DecreaseKey with u and (u, f) in $O(d_f \cdot \log|V|)$ time, where d_f is the number of edges of f.





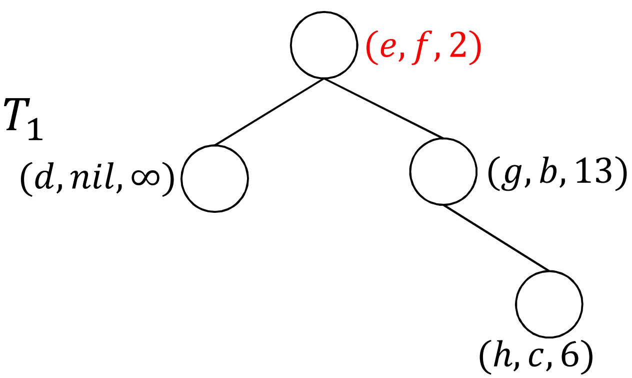
Example

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

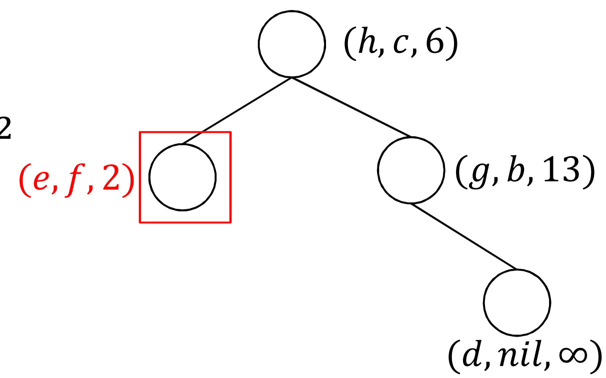


Perform DeleteMin to obtain $(e, f, 2)$ in $O(\log|V|)$ time.

T_1

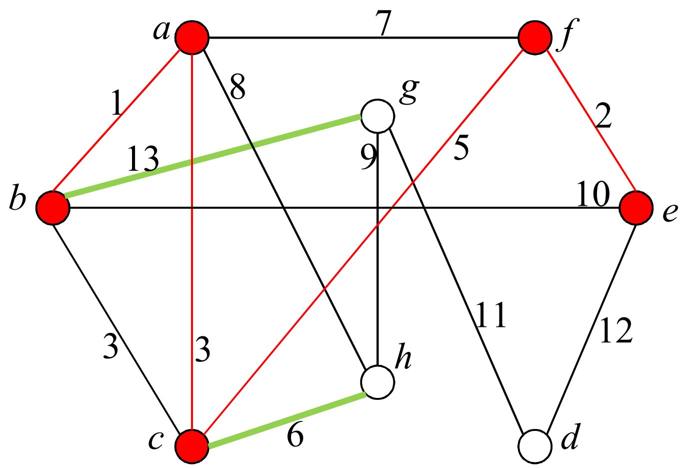


T_2

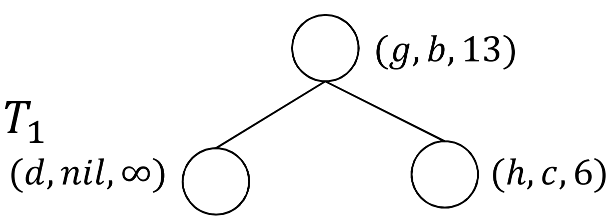


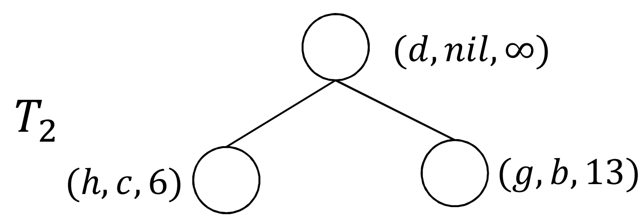
Example

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

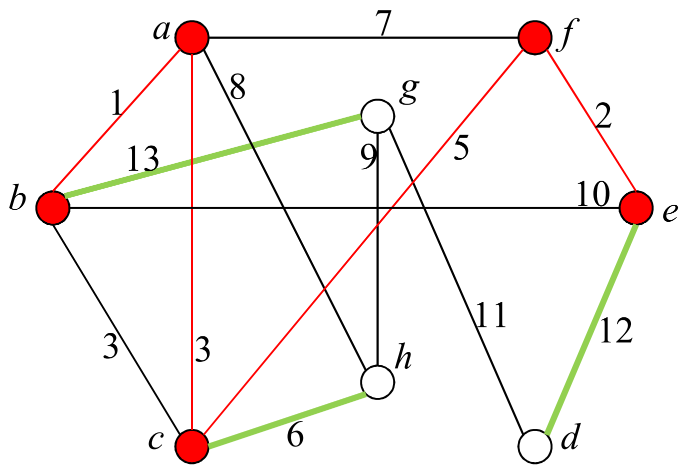


Perform DeleteMin to obtain $(e, f, 2)$ in $O(\log|V|)$ time.



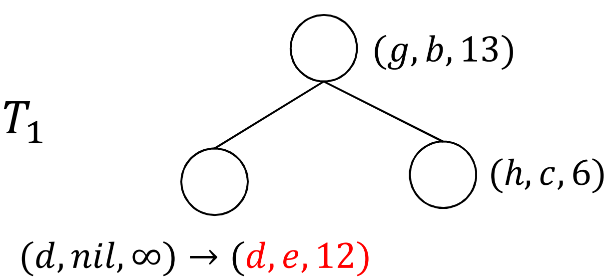


Example

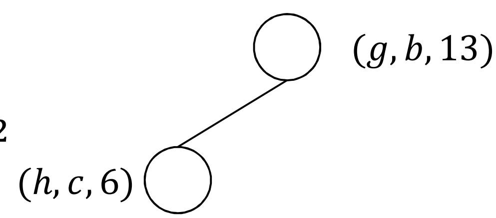


For edge (u, e) where $u \notin S$,
$u = d$, perform Delete and Insert
in $O(\log|V|)$ time.

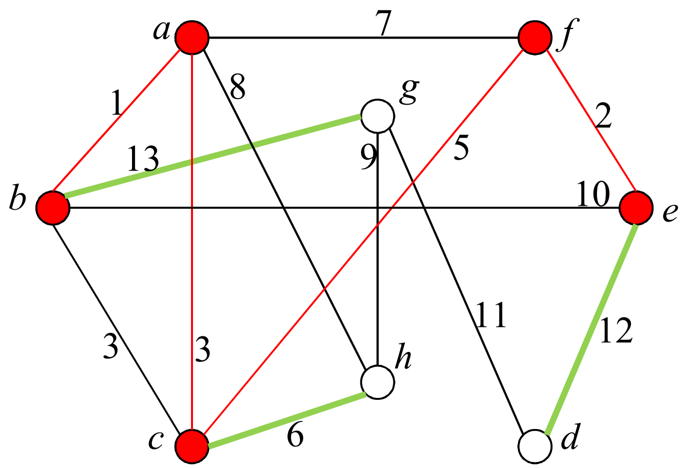
T_1



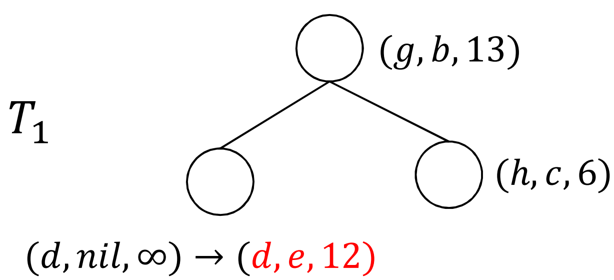
T_2

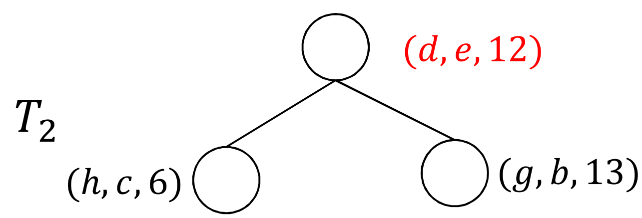


Example

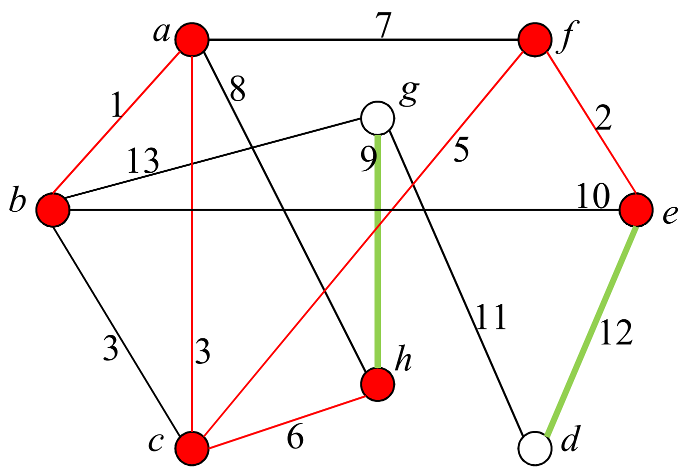


For edge (u, e) where $u \notin S$,
$u = d$, perform Delete and Insert
in $O(\log|V|)$ time.

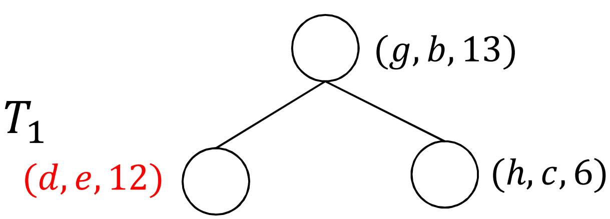


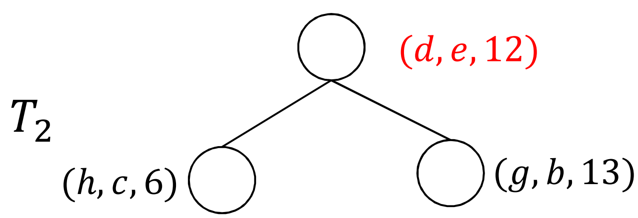


Example



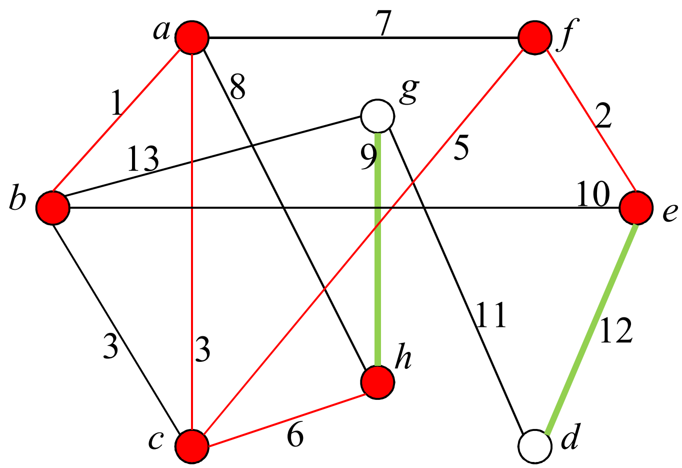
For each edge (u, e) where $u \notin S$, perform DecreaseKey with u and (u, e) in $O(d_e \cdot \log|V|)$ time, where d_e is the number of edges of e.



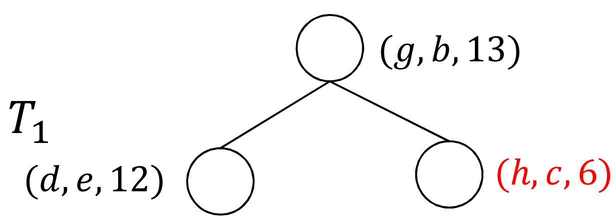


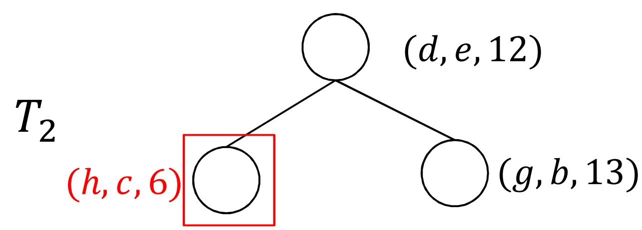
Example

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



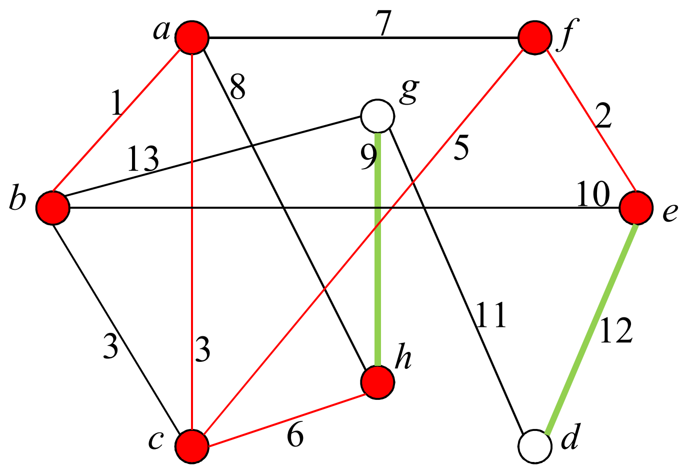
Perform DeleteMin to obtain $(h, c, 6)$ in $O(\log|V|)$ time.



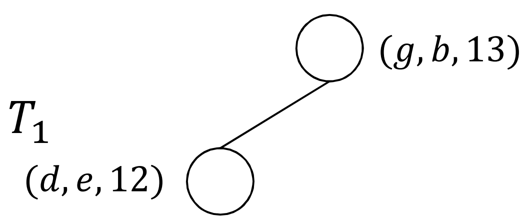


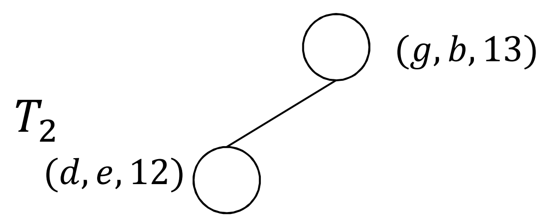
Example

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

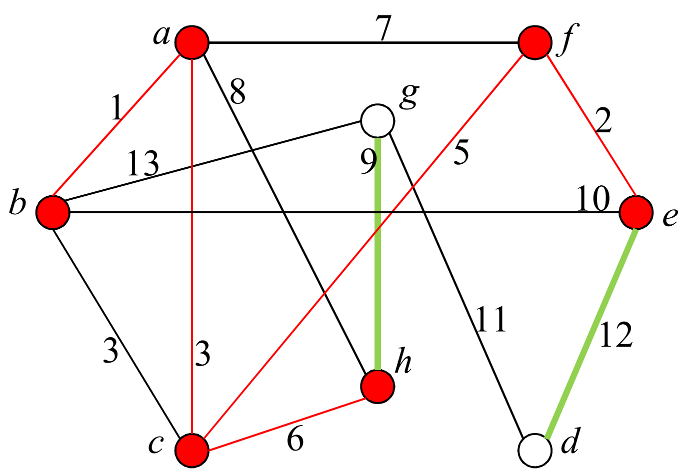


Perform DeleteMin to obtain $(h, c, 6)$ in $O(\log|V|)$ time.

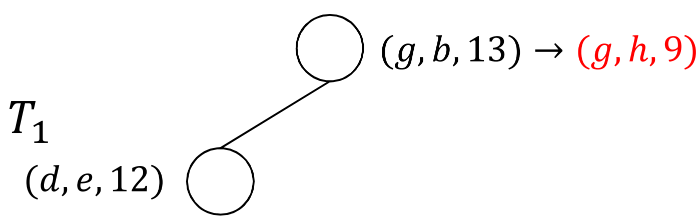


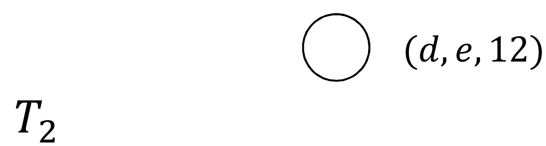


Example

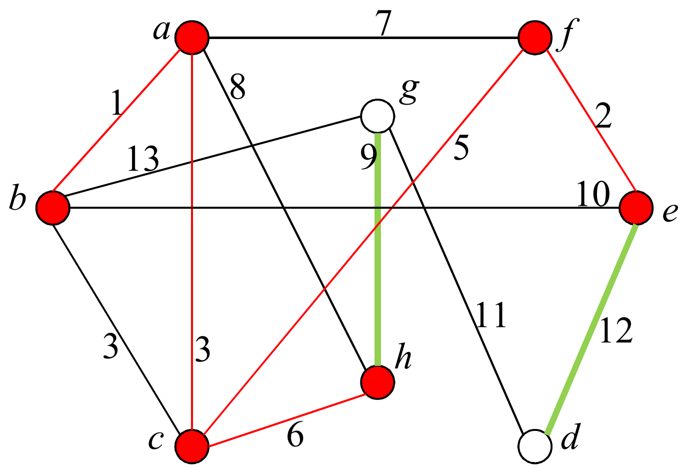


For edge (u, h) where $u \notin S$,
$u = g$, perform Delete and Insert
in $O(\log|V|)$ time.

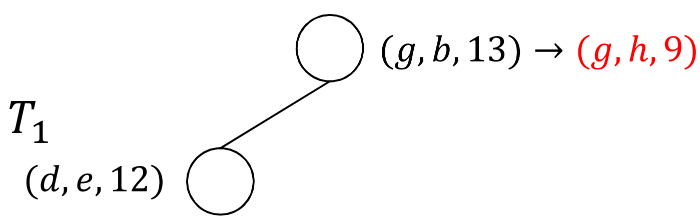


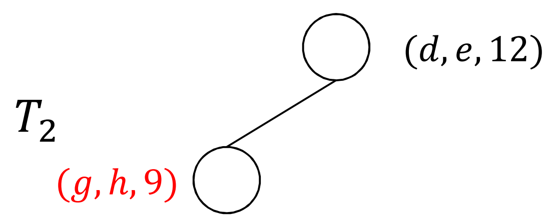


Example

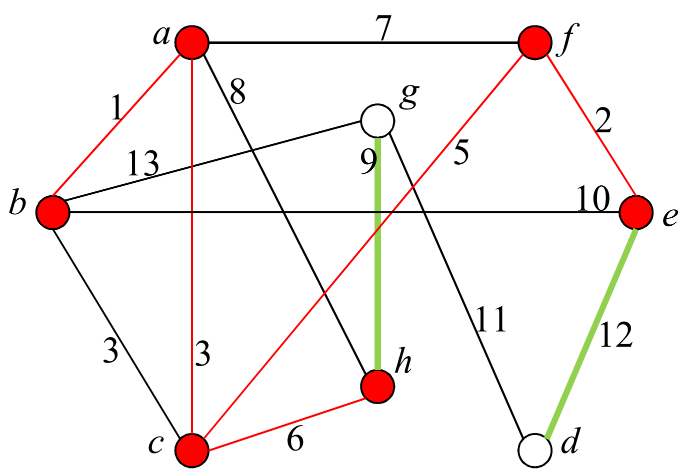


For edge (u, h) where $u \notin S$,
$u = g$, perform Delete and Insert
in $O(\log|V|)$ time.

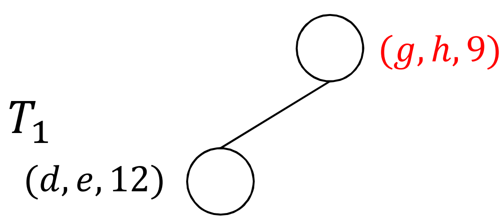


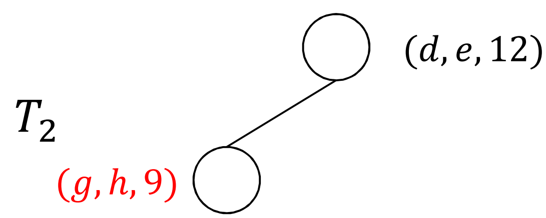


Example



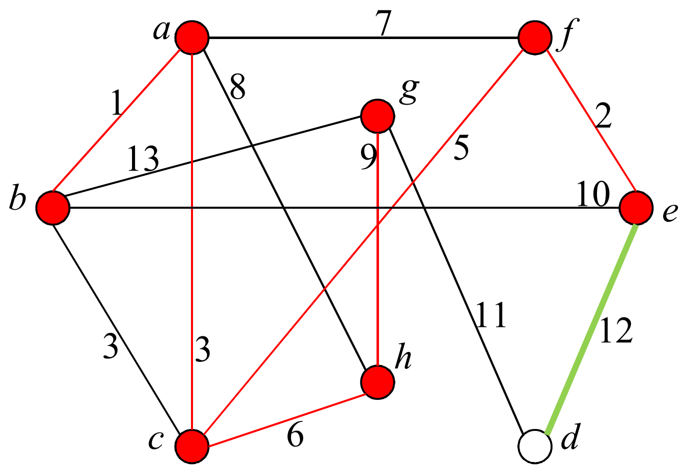
For each edge (u, h) where $u \notin S$, perform DecreaseKey with u and (u, h) in $O(d_h \cdot \log|V|)$ time, where d_h is the number of edges of h.



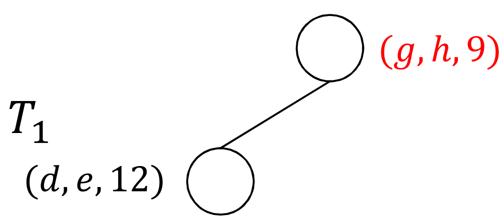


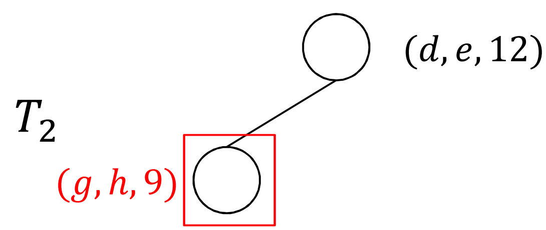
Example

Edge $\{g, h\}$ is the lightest extension 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.



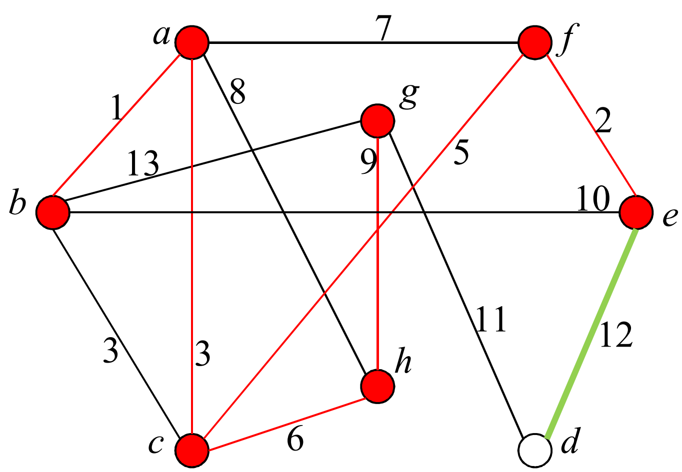
Perform DeleteMin to obtain $(g, h, 9)$ in $O(\log|V|)$ time.





Example

Edge $\{g, h\}$ is the lightest extension 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.



Perform DeleteMin to obtain $(g, h, 9)$ in $O(\log|V|)$ time.

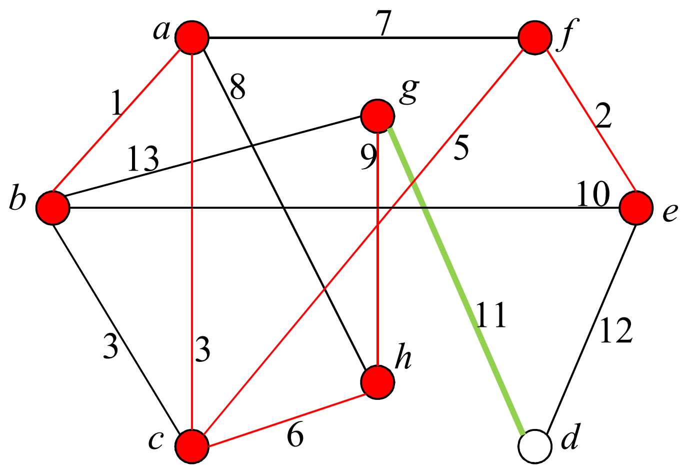
$(d, e, 12)$

T_1

$(d, e, 12)$

T_2

Example



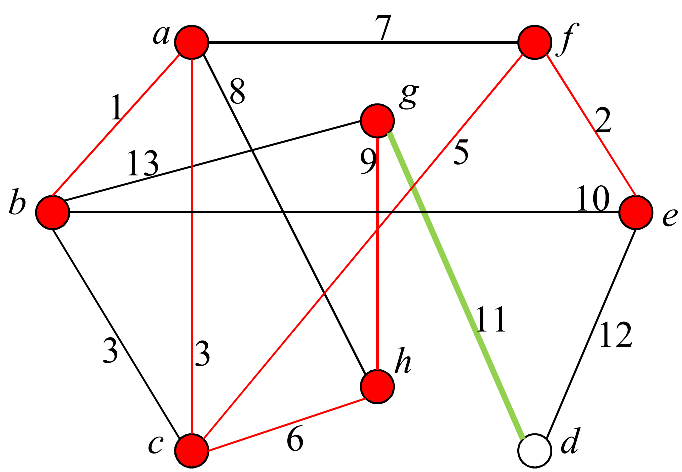
For edge (u, g) where $u \notin S$, $u = d$, perform Delete and Insert in $O(\log|V|)$ time.

$(d, e, 12) \rightarrow (d, g, 11)$

T_1

T_2

Example

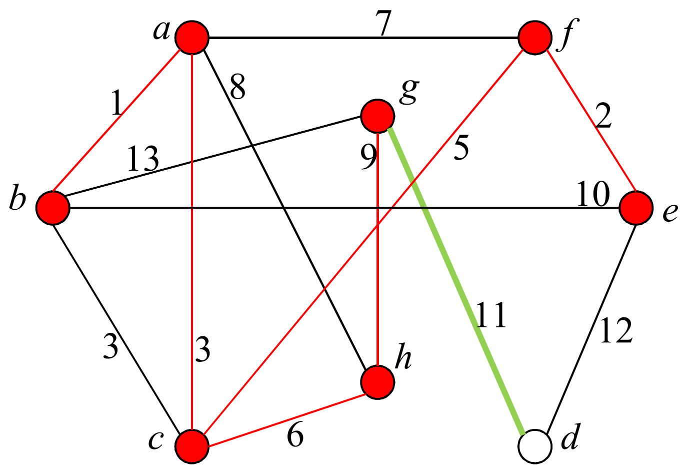


For edge (u, g) where $u \notin S$,
$u = d$, perform Delete and Insert in $O(\log|V|)$ time.

T_1 $(d, e, 12) \rightarrow (d, g, 11)$

T_2 $(d, g, 11)$

Example



For each edge (u, g) where $u \notin S$, perform DecreaseKey with u and (u, g) in $O(d_g \cdot \log|V|)$ time, where d_g is the number of edges of g.

$(d, e, 12) \to (d, g, 11)$

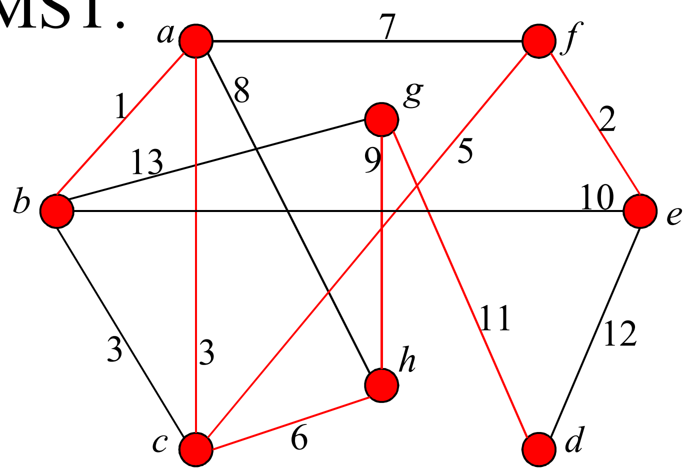
T_1

$(d, g, 11)$

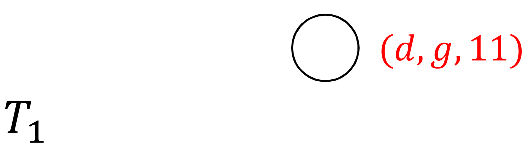
T_2

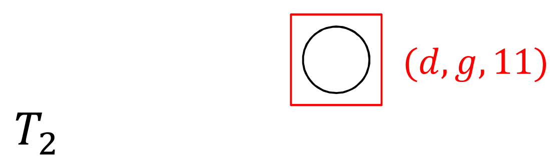
Example

Finally, edge $\{d, g\}$ is the lightest extension 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.



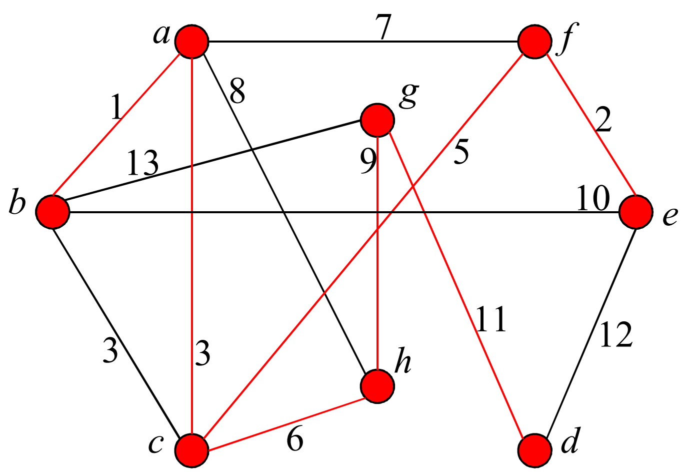
Perform DeleteMin to obtain $(d, g, 11)$ in $O(\log|V|)$ time.





Example

We have obtained our final MST.



Total time consumption:

$$|V| \cdot \log|V| + \sum_{v \in V} \log|V|$$

$$+ \sum_{v \in V} d_v \log|V|$$

$$= O((2|V| + 2|E|) \cdot \log|V|)$$

$$= O((|V| + |E|) \cdot \log|V|)$$

T_1 $\emptyset$

T_2 $\emptyset$