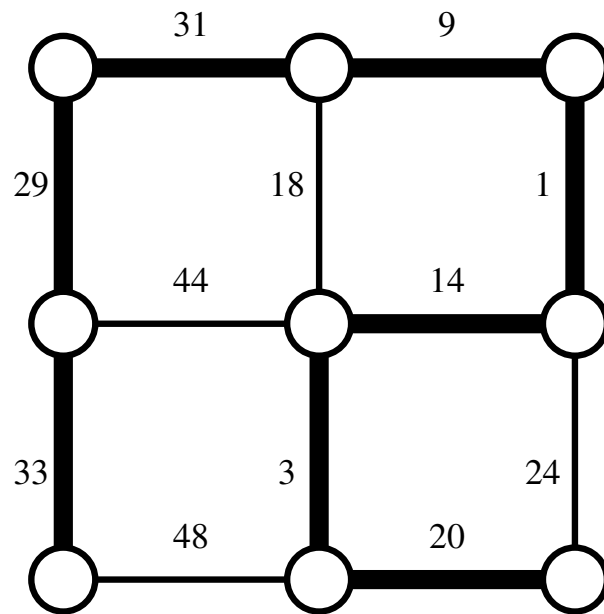


# Minimum Spanning Trees

Given a connected, weighted, undirected graph  $G(V, E)$ , for each edge  $(u, v) \in E$ , there is a weight  $w(u, v)$  associated with it. The **Minimum Spanning Tree problem** (MST) in  $G$  is to find a **spanning tree**  $T(V, E')$  such that the weighted sum of the edges in  $T$  is minimized, i.e.

$$\begin{aligned} \text{minimize } w(T) &= \sum_{(u,v) \in E'} w(u,v), \\ \text{where } E' &\subseteq E. \end{aligned}$$

The diagram below shows a graph,  $G$ , of nine vertices and 12 weighted edges. The bold edges form the edges of the MST,  $T$ . Adding up the weights of the MST edges, we get  $w(T) = 140$ .



## 1 Approaches to Finding MSTs

- Kruskal's algorithm (Boruvka, 1926)

- Prim's algorithm (1956)
- Guan's algorithm (1974)

## 2 Generic Algorithm for MSTs

The greedy strategy is captured by the following generic algorithm which grows the minimum spanning tree one edge at a time.

The algorithm maintains a subset  $A$  of the set  $E$  of edges, which is the edge set of the MST so far. At each step, an edge  $(u, v) \in E$  is added to  $A$  if it is not in  $A$  and it does not violate the condition that no cycle in  $A$  is formed after the addition of it into  $A$ .

```

GENERIC-MST( $G, w$ )
1   $A \leftarrow \emptyset$ 
2  while the edges in  $A$  do not form an MST
3      do find an edge  $(u, v) \in E - A$  that has the
.         minimum weight and this edge, along with the
.         edges in  $A$ , does not form a cycle.
4   $A \leftarrow A \cup \{(u, v)\}$ 
5  return  $A$ 

```

Evidently, the gist of the MST algorithm is finding an edge  $(u, v) \in E - A$  that has the minimum weight and that adding this edge to  $A$  does not result in a cycle in  $A$ . To describe to do this, the following terms are introduced.

- **cut** – A cut is a partition of vertices into two parts; those in a set  $S$  and those in the set  $V - S$  (i.e. not in  $S$ ).
- **edge crossing** – An edge that connects a vertex in  $S$  to a vertex in  $V - S$ .
- **light edge** – The edge crossings with the minimum weight. Note that there may be more than one such light edges if two or more edge crossings have the minimum weight.

**Theorem** Let  $G(V, E)$  be a connected undirected weighted graph with a real-valued weight function  $w$  defined on  $E$ . Let  $A$  be a subset of  $E$  that is included in some MST for  $G$ , let  $(S, V - S)$  be any cut of  $G$  that respects  $A$ , and let  $(u, v)$  be a light edge crossing  $(S, V - S)$ . Since  $(u, v)$  connects two vertices from two distinct sets  $S$  and  $V - S$ , adding  $(u, v)$  to  $A$  will not result in loops.

## 3 Kruskal's Algorithm

In Kruskal's algorithm, the set  $A$  is a forest initially. The safe edge added to  $A$  is always a least weight edge (light edge) in the graph that connected two

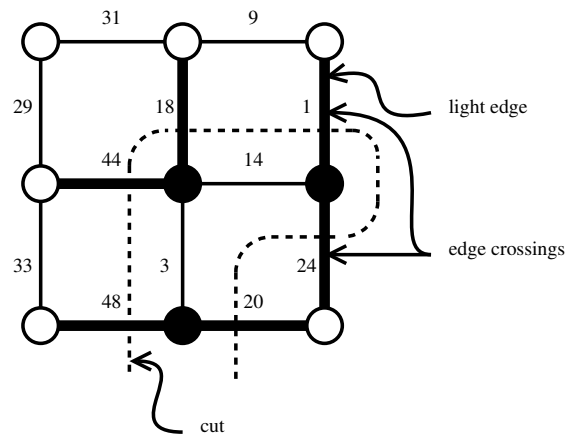


Figure 1: Graph showing a cut. Vertices in  $S$  are shaded black. Edge crossings are indicated by bold lines. The light edge in this example has a weight of 1.

distinct components. Let  $C_1$  and  $C_2$  denote two trees that are connected by  $(u, v)$  and  $(u, v)$  be the minimum weighted edge between them, then add  $(u, v)$  to  $A$ . This procedure continues until all vertices in  $V$  are included by a single tree.

```

KRUSKAL-MST( $G, w$ )
1   $A \leftarrow \emptyset$ 
2  for each vertex  $v \in V$ 
3    do MAKE-SET( $v$ )
4  Sort the edges of  $E$  by nondecreasing weight  $w$ 
5  for each edge  $(u, v) \in E$  in order by nondecreasing weight
6    do if FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ )
7      then  $A \leftarrow A \cup \{(u, v)\}$ 
8          UNION( $u, v$ )
9  return  $A$ 

```

The running time is  $O(m \log n)$  where  $\alpha(m, n) = \log m = O(\log n)$ .

The running time of Kruskal's algorithm: If use forest to represent the disjoint sets and use union by rank and path compression, its running time is  $O(|E| \log |E|) = O(m \log n)$  since  $\alpha(n) = O(\log n)$ .

## 4 Prim's Algorithm

Prim's algorithm operates much like Dijkstra's shortest path algorithm.

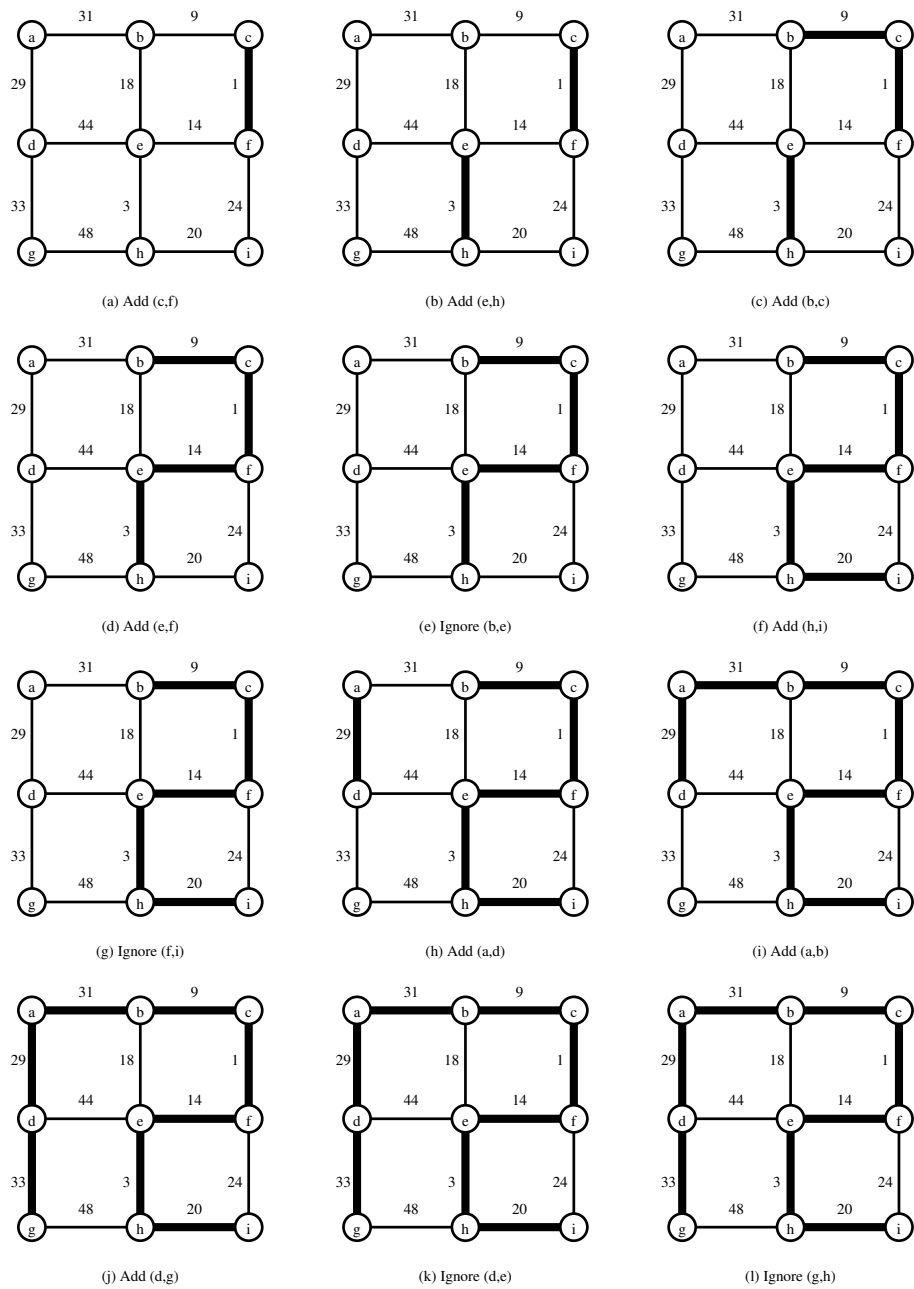


Figure 2: Snapshots for each step of the KRUSKAL-MST function on a graph with nine vertices and 12 edges. Edges in  $A$  are marked using bold lines. Notice that edges are considered in order of nondecreasing weights. Edges that link two vertices in the same component are ignored (Steps (e), (g), (k), and (l)). A vertex  $u$  is in the same component as  $v$  if there is a path from  $u$  to  $v$  using only the edges in  $A$ .

Prim's algorithm has the property that the edges in  $A$  always form a single tree. The tree starts at an arbitrary vertex  $r$  and grows until it covers all the vertices in  $V$ . Let  $S$  be the vertex set of  $T$  so far, at each step we try to find an edge  $(u, v) \in S \times (V - S) \cap E$  and the weight of the edge is the minimum one, add the edge into  $A$  and add  $v$  to  $S$ .

```

PRIM-MST( $G, w, r$ )
1   $Q \leftarrow V$ 
2  for each  $u \in Q$ 
3      do  $key[u] \leftarrow \infty$ 
4   $key[r] \leftarrow 0$ 
5   $p[r] \leftarrow \text{NIL}$ 
6  while  $Q \neq \emptyset$ 
7      do  $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
8          for each  $v \in \text{Adj}[u]$ 
9              do if  $v \in Q$  and  $w(u, v) < key[v]$ 
10                 then  $p[v] \leftarrow u$ 
11                     $key[v] \leftarrow w(u, v)$ 

```

The performance of the algorithm above depends on how we implement the priority queue  $Q$ . Show the correctness of the Prim algorithm.

## 5 Guan's Algorithm

Start from the original graph  $G$ , find a simple cycle in the current graph and delete the edge with the maximum weight from the cycle until the resulting graph contains no cycles. Thus, the resulting graph is an MST of  $G$ .

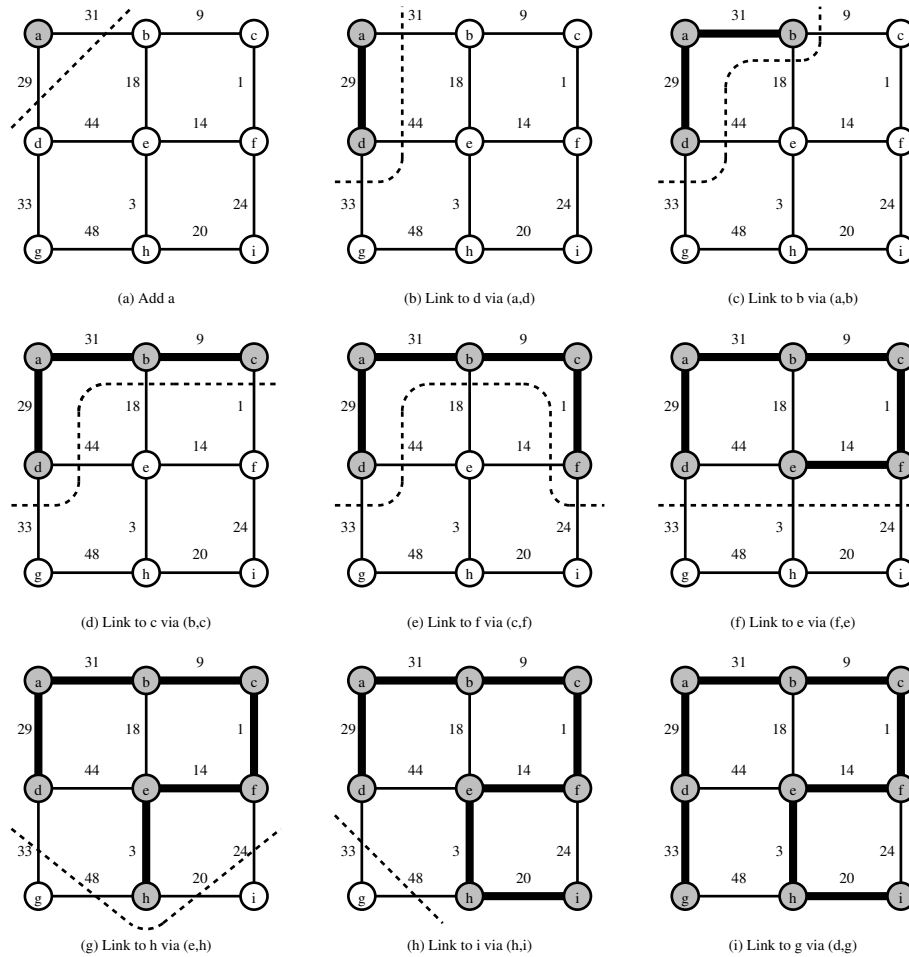


Figure 3: Growing a MST using PRIM-MST. At step (a), the root of the MST is initialised to vertex  $a$ . At each intermediate step, the dotted-line indicates the *cut* dividing the vertices in the MST (shaded) and those not in the MST (not shaded). Notice that the MST is grown along *light edge* edges only.