KRUSKAL'SALGORITHM

Kruskal's algorithm looks at a minimum spanning tree of a weighted connected graph G= {V, E} as an acyclic subgraph with |V| − 1 edges for which the sum of the edge weights is the smallest. the algorithm constructs a minimum spanning tree as an expanding sequence of subgraphs that are always acyclic but are not necessarily connected on the intermediate stages of the algorithm.

The algorithm begins by sorting the graph's edges in nondecreasing order of their weights. Then, starting with the empty subgraph, it scans this sorted list, adding the next edge on the list to the current subgraph if such an inclusion does not create a cycle and simply skipping the edge otherwise.

Kruskal's algorithm looks at a minimum spanning tree of a weighted connected graph G = (V, E) as an acyclic subgraph with |V| − 1 edges for which the sum of the edge weights is the smallest.

ALGORITHM *Kruskal(G)*

//Kruskal's algorithm for constructing a minimum spanning tree

//Input: A weighted connected graph $G = (V, E)$

//Output: E_T, the set of edges composing a minimum spanning tree of G

sort E in nondecreasing order of the edge weights $w(e_{i1}) \le . . . \le w(e_{i|E|})$ $E_T \leftarrow \Phi$; $ecounter \leftarrow 0$ //initialize the set of tree edges and itssize

$K \leftarrow 0$ //initialize the number of processededges

while $ecounter < |V| - 1$ **do**

$k \leftarrow k + 1$

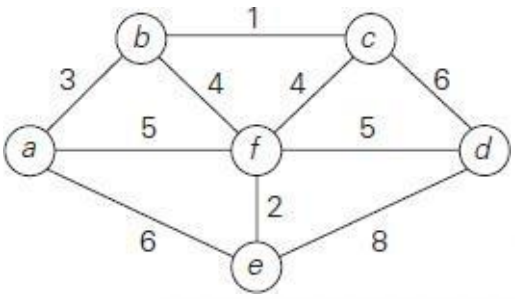
if $E_T \cup \{e_{ik}\}$ is acyclic

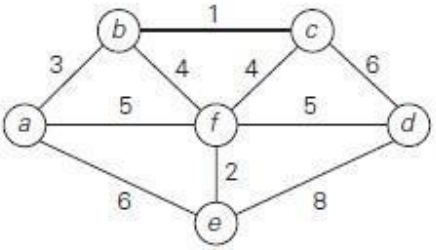
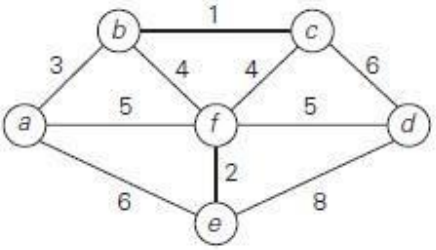
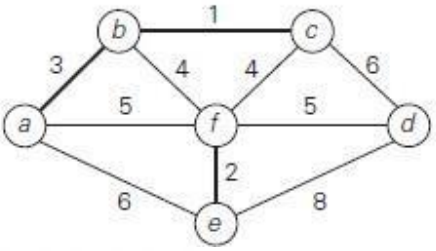
$E_T \leftarrow E_T \cup \{e_{ik}\}$; $ecounter \leftarrow ecounter + 1$

return E_T

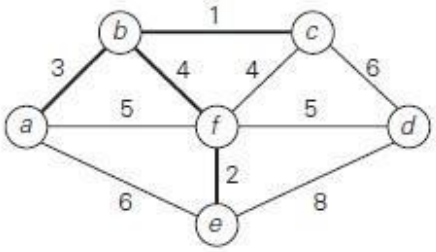
The initial forest consists of |V | trivial trees, each comprising a single vertex of the graph. The final forest consists of a single tree, which is a minimum spanning tree of the graph. On each iteration, the algorithm takes the next edge (u, v) from the sorted list of the graph's edges, finds the trees containing the vertices u and v, and, if these trees are not the same, unites them in a larger tree by adding the edge (u, v).

Fortunately, there are efficient algorithms for doing so, including the crucial check for whether two vertices belong to the same tree. They are called union-find algorithms. With an efficient union-find algorithm, the running time of Kruskal's algorithm will be $O(|E| \log |E|)$.

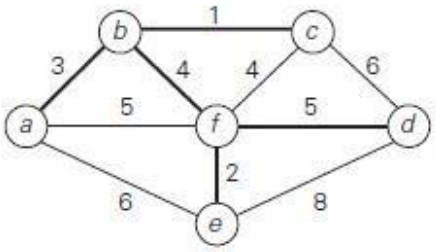


Tree edges	Sorted list of edges	Illustration
	bc ef ab bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
bc 1	bc **ef** ab bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
ef 2	bc ef **ab** bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	

ab		bc	ef	ab	**bf**	cf	af	df	ae	cd	de
3		1	2	3	4	4	5	5	6	6	8



bf		bc	ef	ab	bf	cf	af	**df**	ae	cd	de
4		1	2	3	4	4	5	5	6	6	8



df
5

FIGURE 3.15 Application of Kruskal's algorithm. Selected edges are shown in bold.