Greedy Strategy

Introduction

Greedy approach suggests constructing a solution through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution to the problem is reached. On each step the choice made must be

- Feasible: Solution which satisfy the problems constraint.
- Locally optimal: Best local choice among all feasible choices
- Irrevocable: Once decision made it cannot be changed on subsequent steps of the algorithm

Spanning Tree

Spanning Tree:

Connected acyclic subgraph(or tree), that contains all the vertices of the graph.

Minimum Cost Spanning tree(MCST):

MCST of a weighted connected graph is its spanning tree of the smallest weight, where the weight of a tree is defined as the sum of the weights on all its edges.

Kruskal’s Algorithm

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.

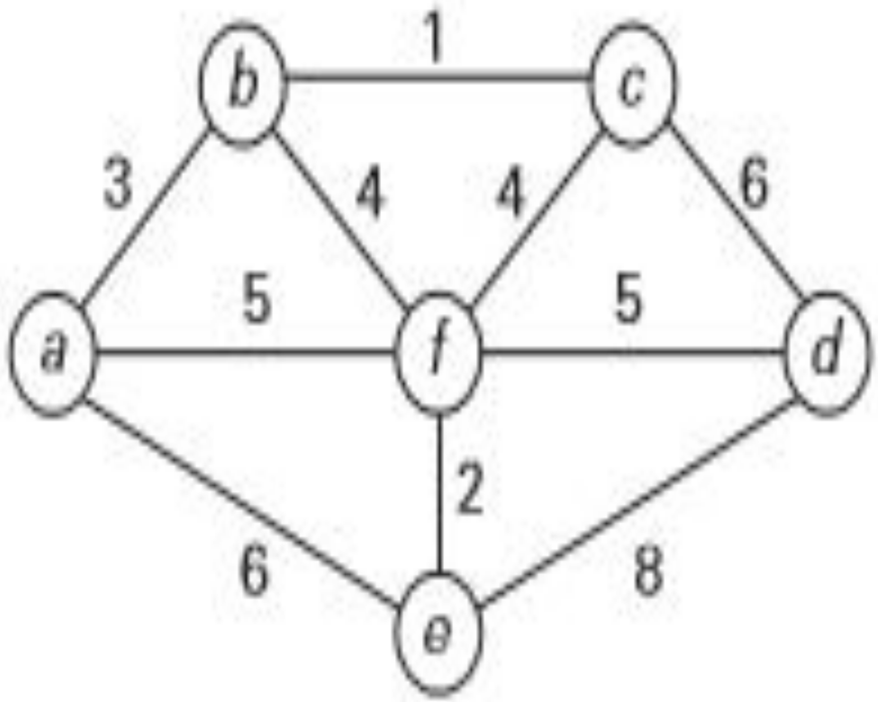
The algorithm begins by sorting the graph’s edges in nondecreasing order of their weights. Then, starting with the empty subgraph, it scans this sorted list, adding the next edge on the list to the current subgraph if such an inclusion does not create a cycle and simply skipping the edge otherwise.

Kruskal's Algorithm

ALGORITHM *Kruskal(G)*

//Kruskal's algorithm for constructing a minimum spanning tree
//Input: A weighted connected graph $G = \langle V, E \rangle$
//Output: E_T, the set of edges composing a minimum spanning tree of G
sort E in nondecreasing order of the edge weights $w(e_{i_1}) \le \cdots \le w(e_{i_{|E|}})$
$E_T \leftarrow \varnothing$; $\quad ecounter \leftarrow 0$ //initialize the set of tree edges and its size
$k \leftarrow 0$ //initialize the number of processed edges
while $ecounter < |V| - 1$ **do**
    $k \leftarrow k + 1$
    if $E_T \cup \{e_{i_k}\}$ is acyclic
        $E_T \leftarrow E_T \cup \{e_{i_k}\}$; $\quad ecounter \leftarrow ecounter + 1$
return E_T

Example 1



Example 1

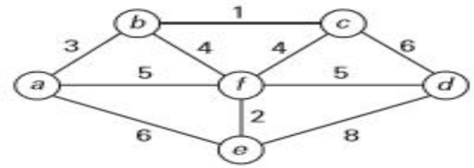
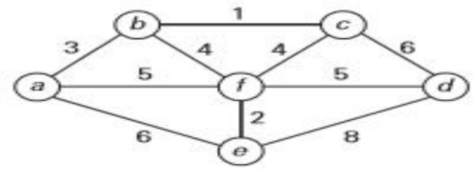
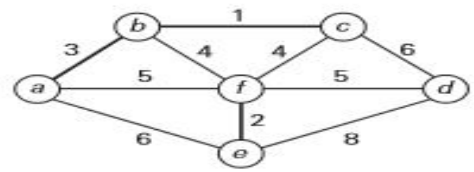
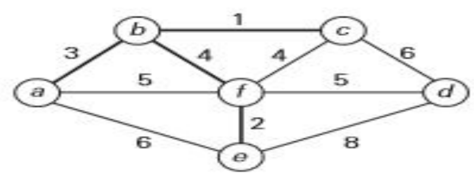
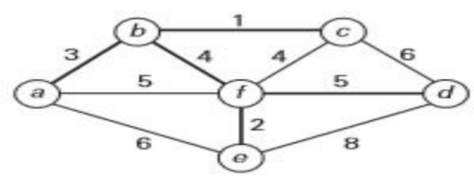
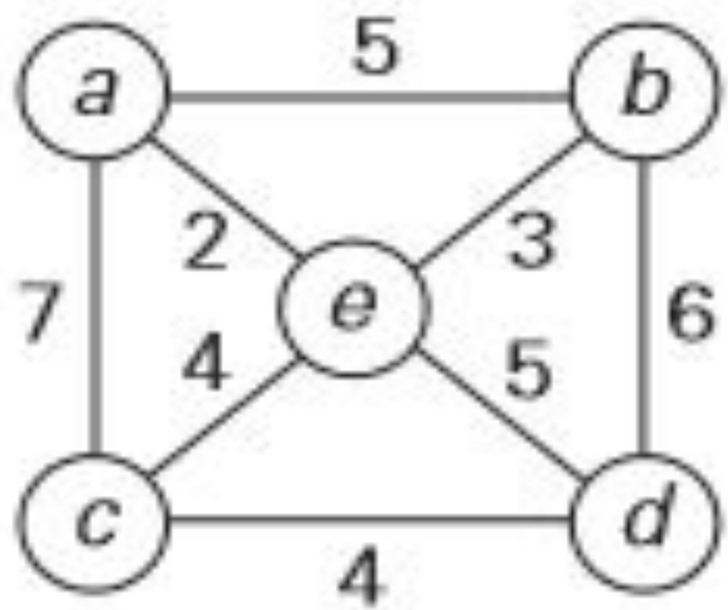
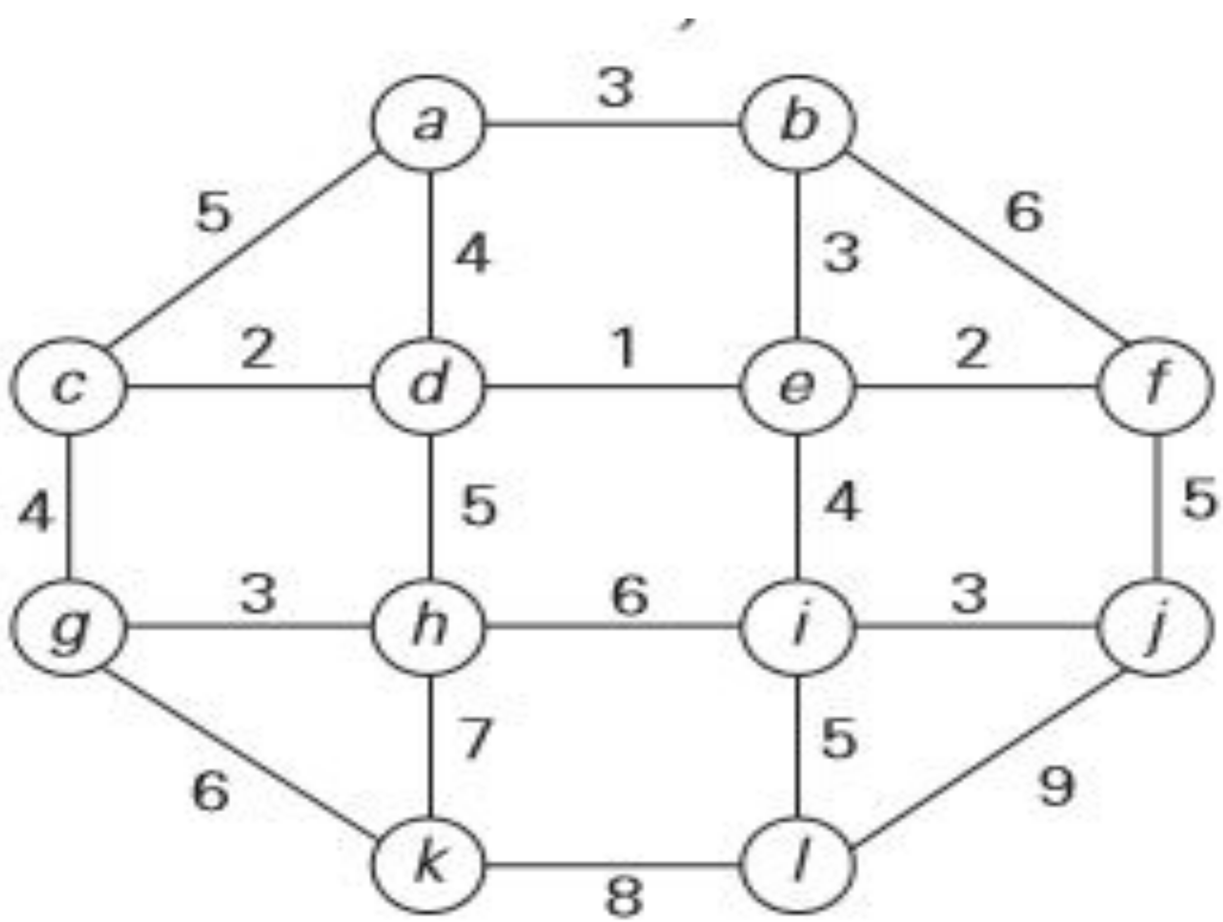
Tree edges	Sorted list of edges	Illustration
	bc 1 ef 2 ab 3 bf 4 cf 4 af 5 df 5 ae 6 cd 6 de 8	
bc 1	bc 1 **ef** 2 ab 3 bf 4 cf 4 af 5 df 5 ae 6 cd 6 de 8	
ef 2	bc 1 ef 2 **ab** 3 bf 4 cf 4 af 5 df 5 ae 6 cd 6 de 8	
ab 3	bc 1 ef 2 ab 3 **bf** 4 cf 4 af 5 df 5 ae 6 cd 6 de 8	
bf 4	bc 1 ef 2 ab 3 bf 4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

FIGURE 9.5 Application of Kruskal's algorithm. Selected edges are shown in bold.

Example 2



Example 3



Applications

Given n points, connect them in the cheapest possible way so that there will be a path between every pair of points.It has direct applications to the design of all kinds of networks— including communication, computer, transportation, and electrical—by providing the cheapest way to achieve connectivity. It identifies clusters of points in data sets. It has been used for classification purposes in archeology, biology, sociology, and other sciences. It is also helpful for constructing approximate solutions to more difficult problems such the traveling salesman problem