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.

Prim’s Algorithm

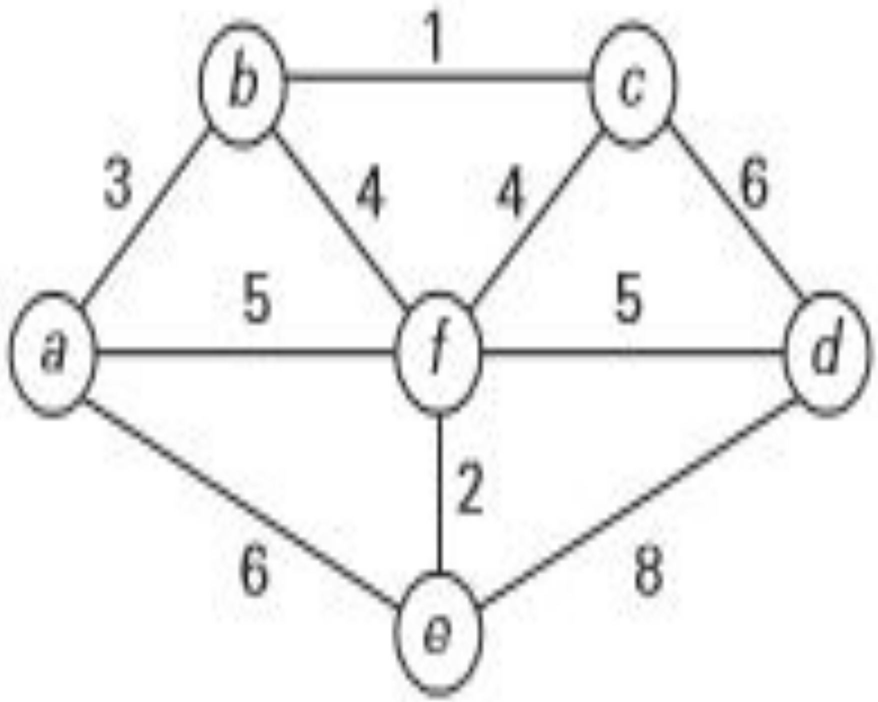
Prim’s algorithm constructs a minimum spanning tree through a sequence of expanding subtrees. The initial subtree in such a sequence consists of a single vertex selected arbitrarily from the set V of the graph’s vertices. On each iteration, the algorithm expands the current tree in the greedy manner by simply attaching to it the nearest vertex not in that tree.(By the nearest vertex, we mean a vertex not in the tree connected to a vertex in the tree by an edge of the smallest weight. Ties can be broken arbitrarily.) The algorithm stops after all the graph’s vertices have been included in the tree being constructed.

Prim's Algorithm

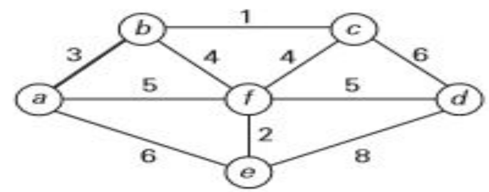
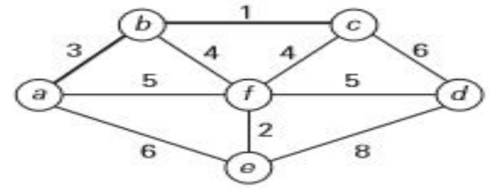
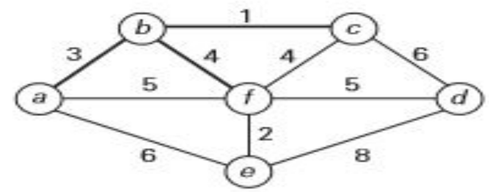
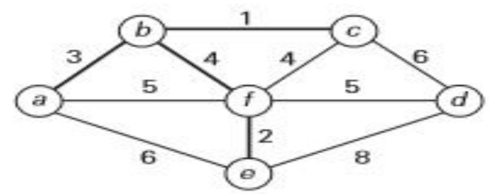
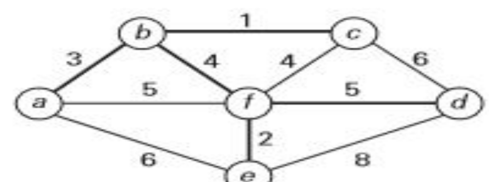
ALGORITHM *Prim(G)*

//Prim'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
$V_T \leftarrow \{v_0\}$ //the set of tree vertices can be initialized with any vertex
$E_T \leftarrow \varnothing$
for $i \leftarrow 1$ **to** $|V| - 1$ **do**
    find a minimum-weight edge $e^* = (v^*, u^*)$ among all the edges (v, u)
    such that v is in V_T and u is in $V - V_T$
    $V_T \leftarrow V_T \cup \{u^*\}$
    $E_T \leftarrow E_T \cup \{e^*\}$
return E_T

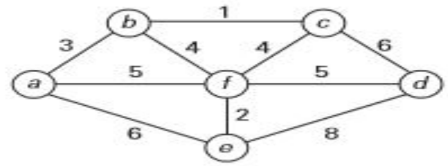
Example 1

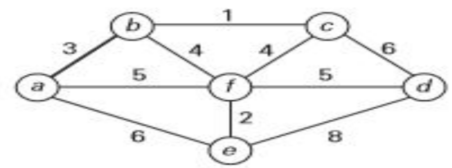
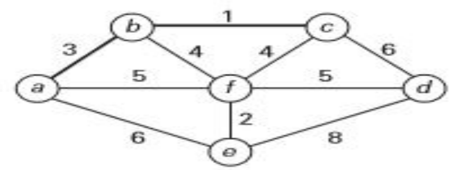
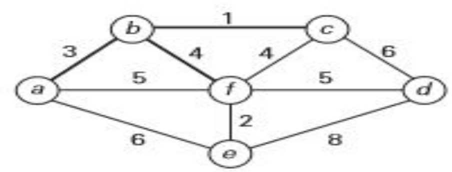
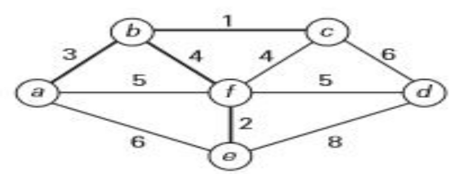
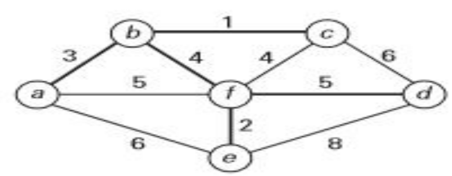


Solutions for example 1

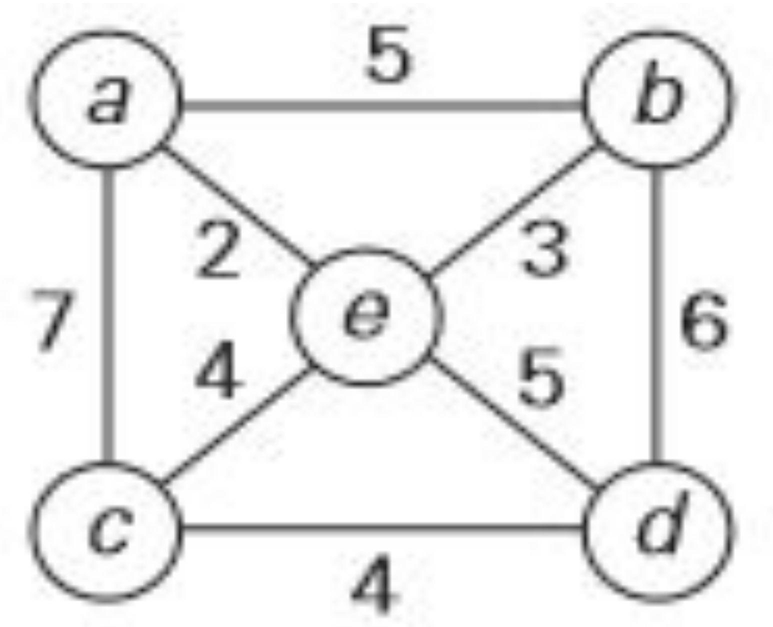
Tree vertices	Remaining vertices	Illustration
a(−, −)	**b(a, 3)** c(−, ∞) d(−, ∞) e(a, 6) f(a, 5)	
b(a, 3)	**c(b, 1)** d(−, ∞) e(a, 6) f(b, 4)	
c(b, 1)	d(c, 6) e(a, 6) **f(b, 4)**	
f(b, 4)	d(f, 5) **e(f, 2)**	
e(f, 2)	**d(f, 5)**	
d(f, 5)		

Example 1

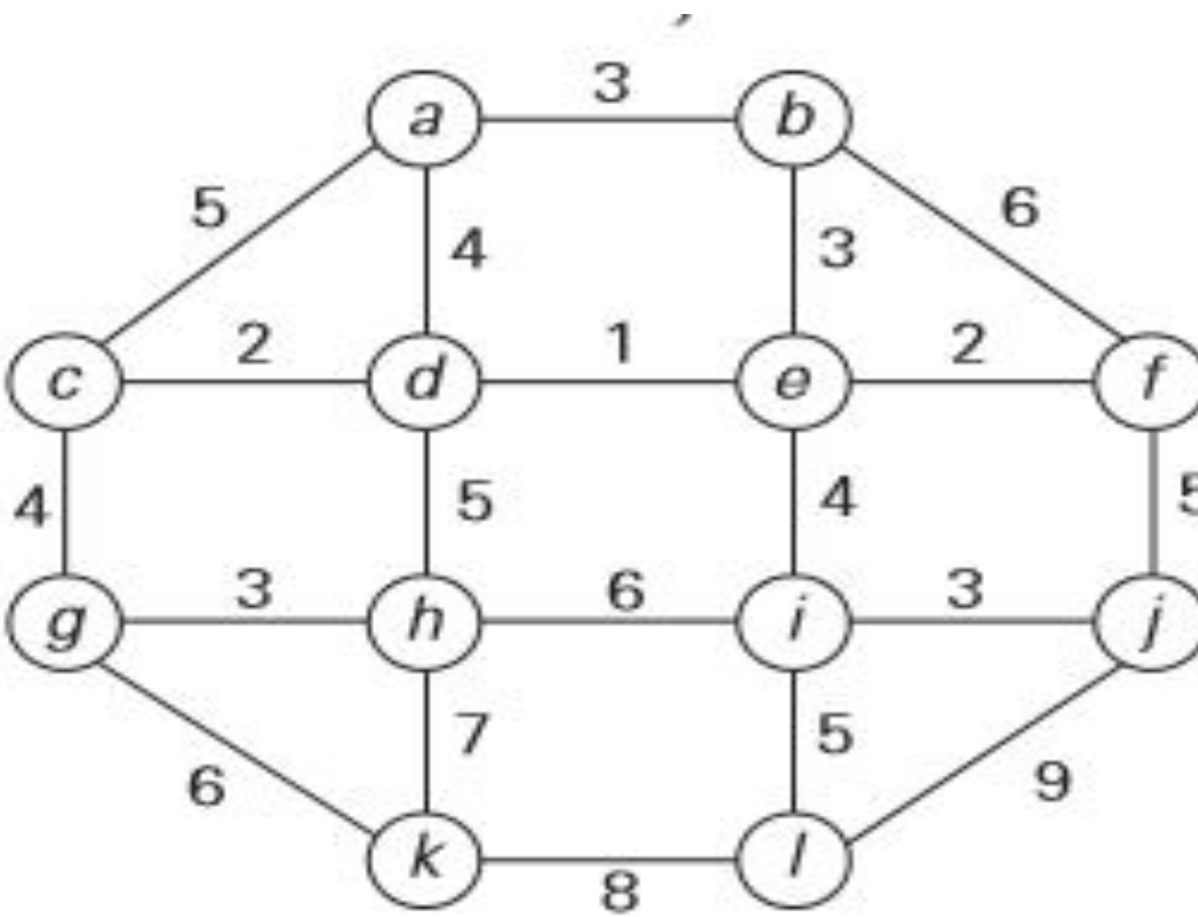


Tree vertices	Remaining vertices	Illustration
a(−, −)	**b(a, 3)** c(−, ∞) d(−, ∞) e(a, 6) f(a, 5)	
b(a, 3)	**c(b, 1)** d(−, ∞) e(a, 6) f(b, 4)	
c(b, 1)	d(c, 6) e(a, 6) **f(b, 4)**	
f(b, 4)	d(f, 5) **e(f, 2)**	
e(f, 2)	**d(f, 5)**	
d(f, 5)		

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