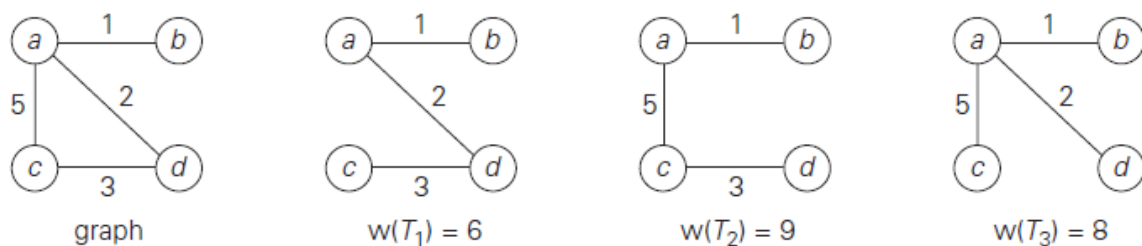


Text Book: Introduction to the Design and Analysis of Algorithms Author: Anany Levitin 2 nd Edition

Unit-4

5. Prim's Algorithm

A **spanning tree** of an undirected connected graph is its connected acyclic subgraph (i.e., a tree) that contains all the vertices of the graph. If such a graph has weights assigned to its edges, a **minimum spanning tree** 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. The **minimum spanning tree problem** is the problem of finding a minimum spanning tree for a given weighted connected graph.



Graph and its spanning trees, with T_1 being the minimum spanning tree.

Applications:

1. The MST has applications in many practical situations: given n points, connect them in the cheapest possible way so that there will be a path between every pair of points.
2. 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.
3. It has been used for classification purposes in archeology, biology, sociology, and other sciences.
4. It is also helpful for constructing approximate solutions to more difficult problems such the traveling salesman problem

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.
- Since the algorithm expands a tree by exactly one vertex on each of its iterations, the total number of such iterations is $n - 1$, where n is the number of vertices in the graph.
- The tree generated by the algorithm is obtained as the set of edges used for the tree expansions.

Pseudocode

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
 $E_T \leftarrow \emptyset$ 
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  $E_T$ 
```

Time Complexity

How efficient is Prim's algorithm? The answer depends on the data structures chosen for the graph itself and for the priority queue of the set $V - VT$ whose vertex priorities are the distances to the nearest tree vertices.

In particular, if a graph is represented by its weight matrix and the priority queue is implemented as an unordered array, the algorithm's running time will be in $\Theta(|V|^2)$. Indeed, on each of the $|V| - 1$ iterations, the array implementing the priority queue is traversed to find and delete the minimum and then to update, if necessary, the priorities of the remaining vertices.

If a graph is represented by its adjacency lists and the priority queue is implemented as a min-heap, the running time of the algorithm is in $O(|E| \log |V|)$.

This is because the algorithm performs $|V| - 1$ deletions of the smallest element and makes $|E|$ verifications and, possibly, changes of an element's priority in a min-heap of size not exceeding $|V|$. Each of these operations, as noted earlier, is a $O(\log |V|)$ operation. Hence, the running time of this implementation of Prim's algorithm is in

$$(|V| - 1 + |E|)O(\log |V|) = O(|E| \log |V|)$$

because, in a connected graph, $|V| - 1 \leq |E|$.