Algorithms

Minimum Spanning Trees

Minimum Spanning Trees

- Suppose we wish to connect all of the computers in a new office building using the least amount of cable. We can model the problem using a weighted graph G, whose vertices represent the computers, where the weight w((v,u)) of edge (v,u) is equal to the amount of cable needed to connect computer v to computer u.

- Given a weighted undirected graph G, we are interested in finding a tree, T, that contains all the vertices in G and minimizes the sum

$$w(T) = \sum_{(v,u) \in T} w(v,u)$$

Minimum Spanning Trees

- A tree such as this that contains every vertex of a connected graph is said to be a **spanning tree**, and the problem of computing a spanning tree, T, with smallest total weight is known as the **minimum spanning tree (MST) problem**.
- The MST problem is an example of an **optimization problem**.
- In an optimization problem we are given a set of constraints and an optimization function.
- Solutions that satisfy the constraints are called **feasible solutions**.

Minimum Spanning Trees

- A feasible solution for which the optimization function has the best possible value is called an **optimal solution**.
- There are many different algorithms for solving the MST problem of which we will look at 2
- Both of these algorithms are applications of the **greedy method**.

The Greedy Method

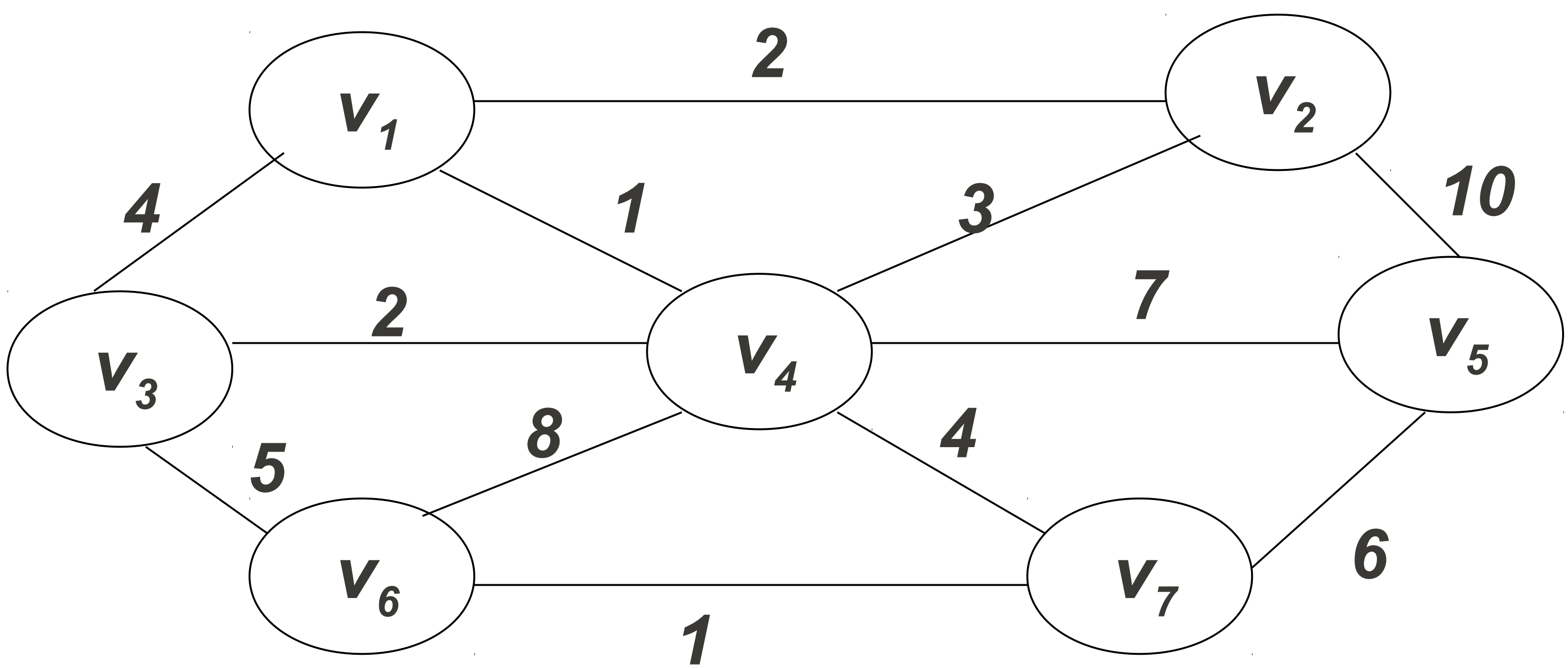
- In the greedy method we attempt to construct an optimal solution in stages.
- At each stage we make a decision that appears to be the best (under some criterion) at that stage.
- A decision made in one stage is not changed in a later stage, so each decision will assure feasibility.
- The criterion used to make the greedy decision at each stage is called the greedy criterion.

The Greedy Method

- The sequence starts from some well understood starting condition, and computes the cost for that initial condition.
- Iteratively make additional choices by identifying the decision that achieves the best from all of the choices that are currently possible.
- The greedy method does not always lead to an optimal solution but works well for some algorithms.

Kruskal's Algorithm

- Kruskal's Algorithm is one that can be used to construct a minimum spanning tree for a graph.
- For a graph which contains n vertices, Kruskal's algorithm selects the n - 1 edges one at a time using the greedy criterion:
  From the remaining edges, select a least-cost edge that does not result in a cycle when added to the set of already selected edges



Kruskal's Algorithm

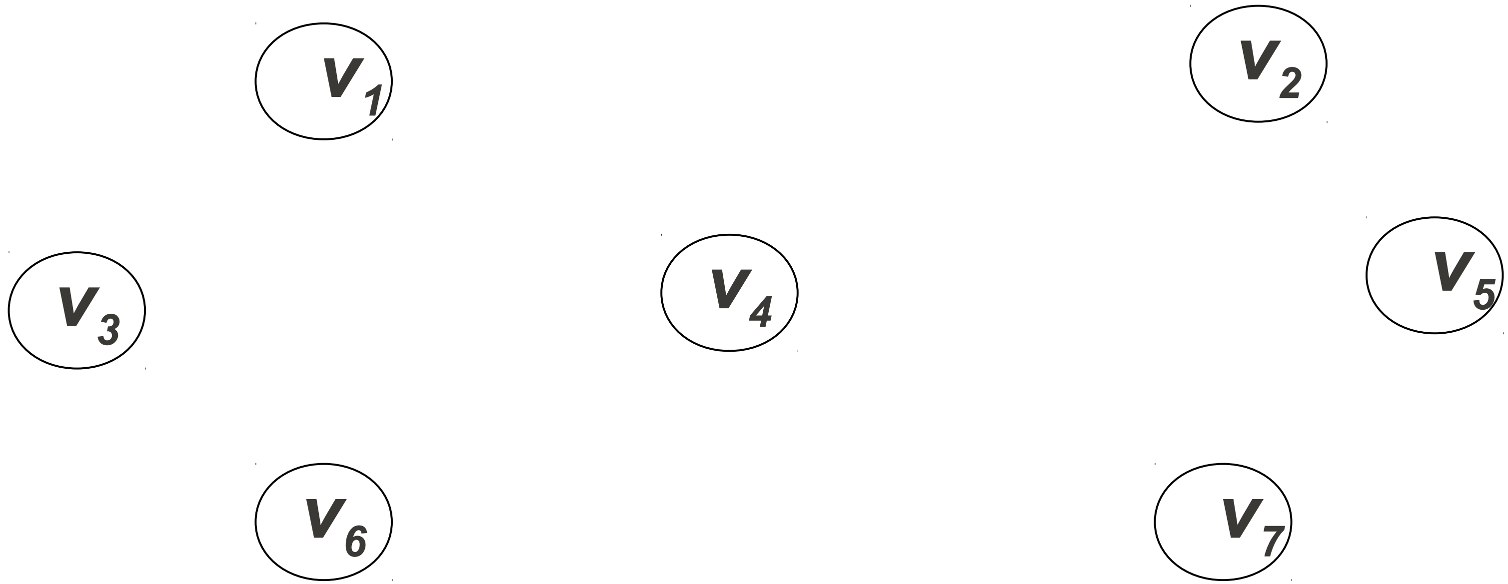
Edge	Weight
(v_1, v_4)	1
(v_6, v_7)	1
(v_1, v_2)	2
(v_3, v_4)	2
(v_2, v_4)	3
(v_1, v_3)	4
(v_4, v_7)	4
(v_3, v_6)	5
(v_5, v_7)	6
(v_4, v_5)	7
(v_4, v_6)	8
(v_2, v_5)	10

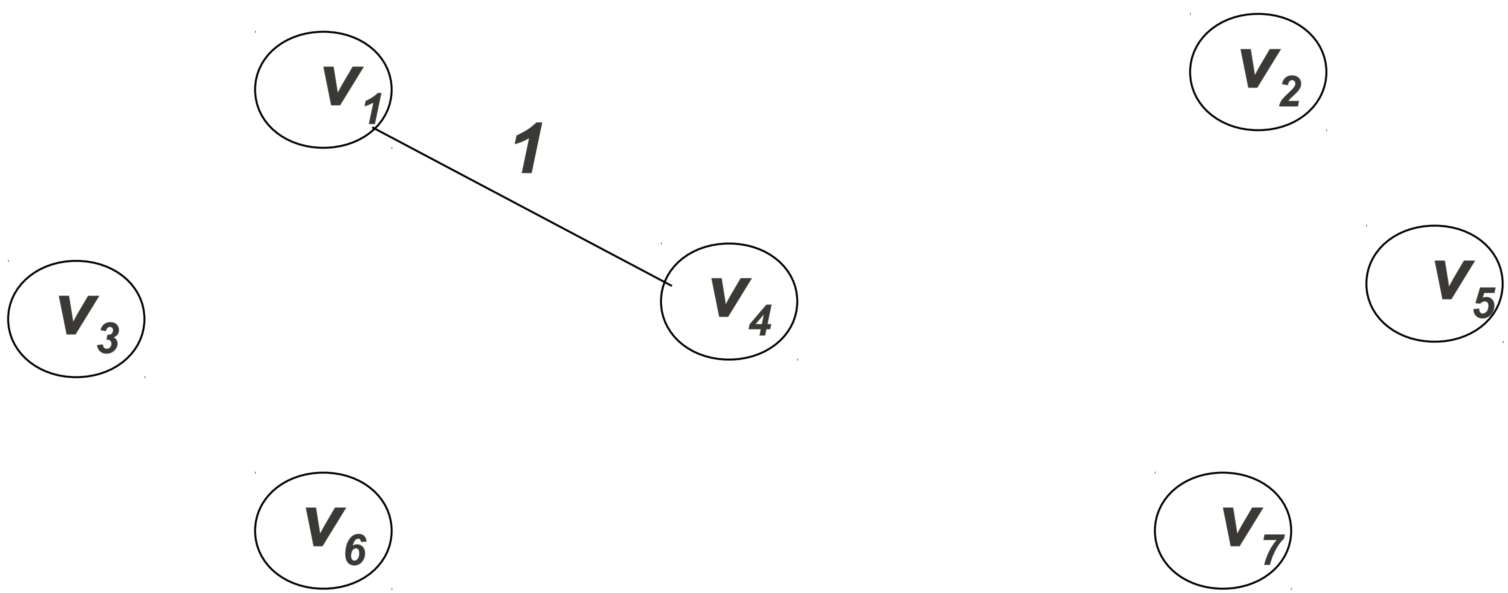
Kruskal's Algorithm

- We start by choosing the edge with the smallest weight (we can choose either (v_1, v_4) or (v_6, v_7). We will choose (v_1, v_4).
- Next, from the remaining edges, we will choose the edge with the smallest weight that does not create a cycle, i.e. (v_6, v_7) and so on

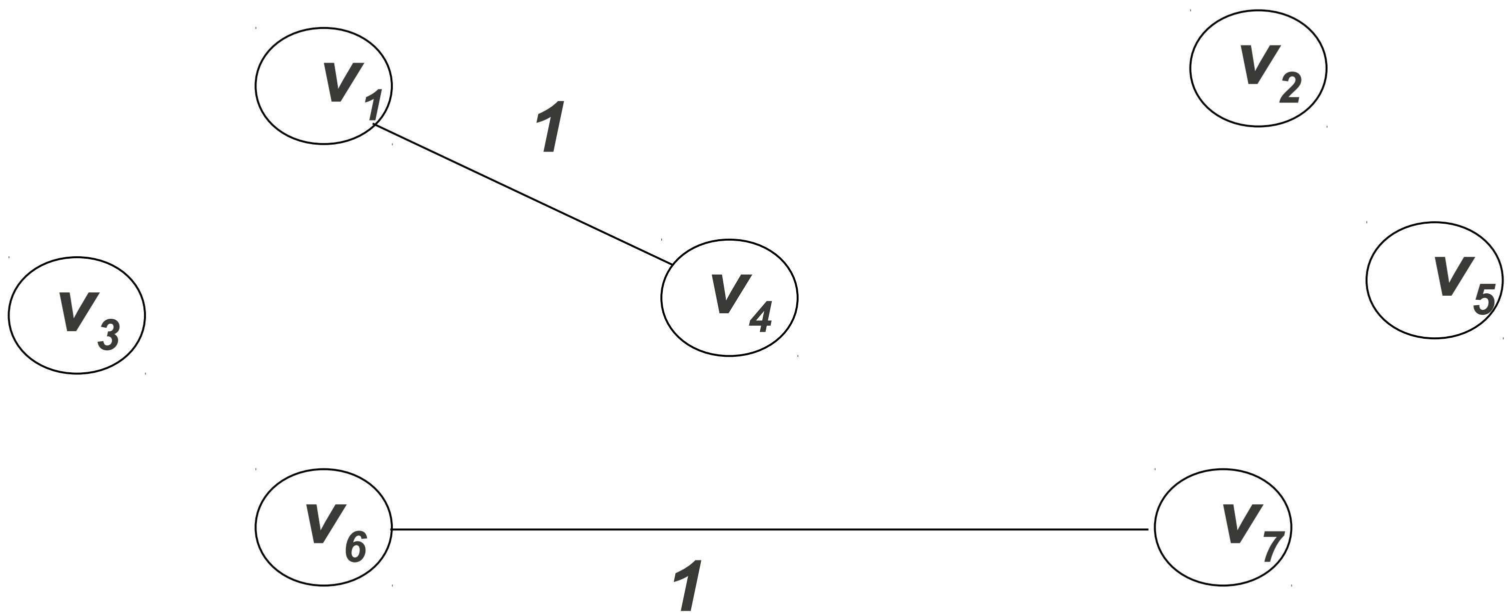
Edge	***Weight***	***Action***
(v_1, v_4)	1	Accepted
(v_6, v_7)	1	Accepted
(v_1, v_2)	2	Accepted
(v_3, v_4)	2	Accepted
(v_2, v_4)	3	Rejected
(v_1, v_3)	4	Rejected
(v_4, v_7)	4	Accepted
(v_3, v_6)	5	Rejected
(v_5, v_7)	6	Accepted

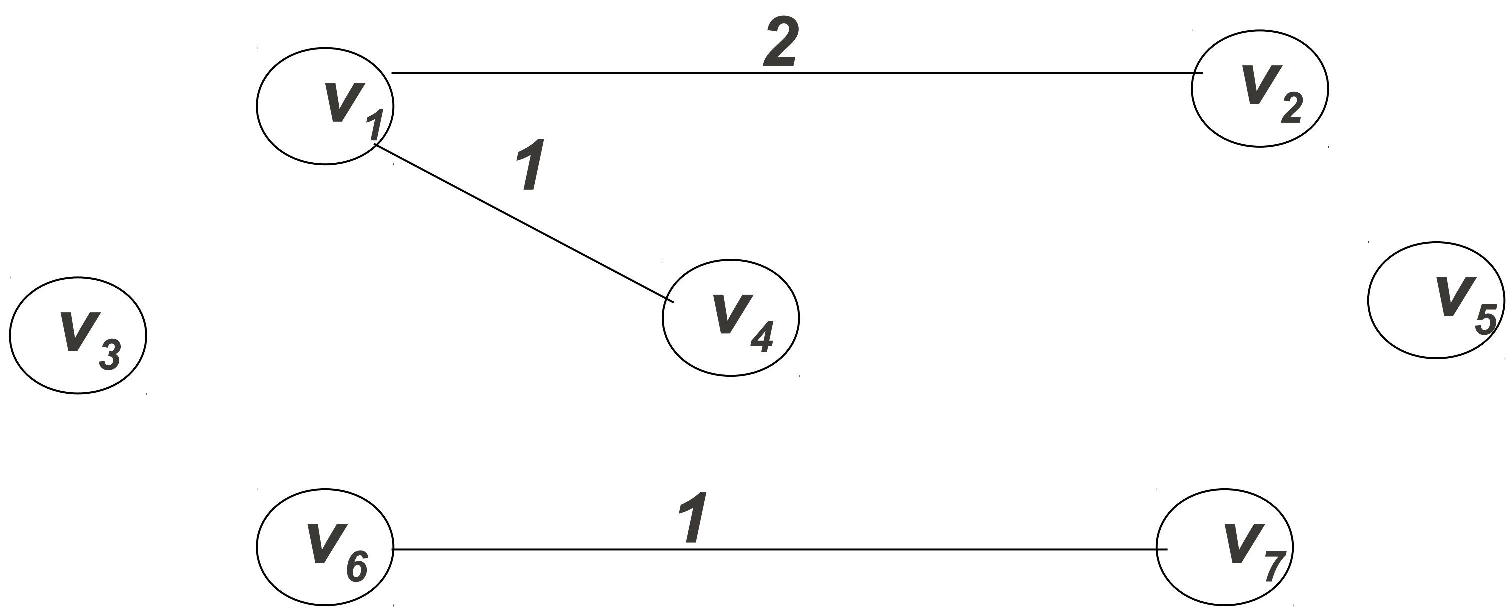
Kruskal’s Algorithm



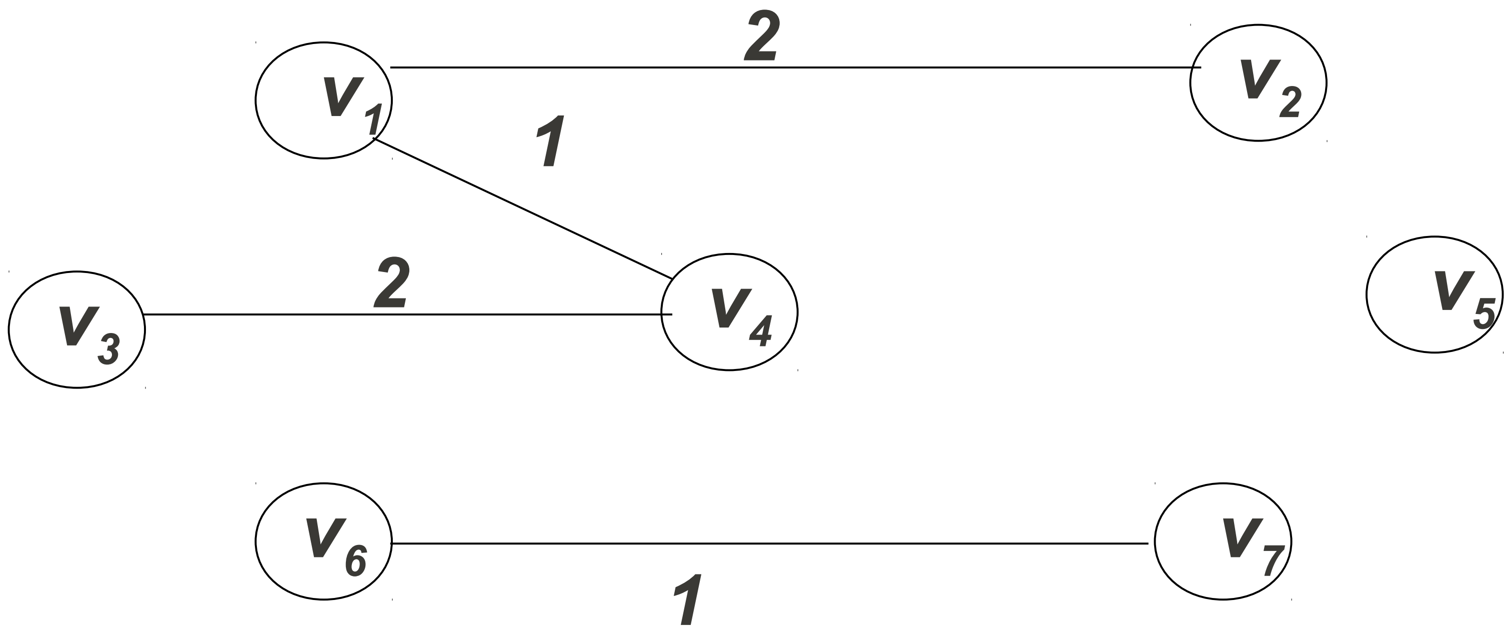


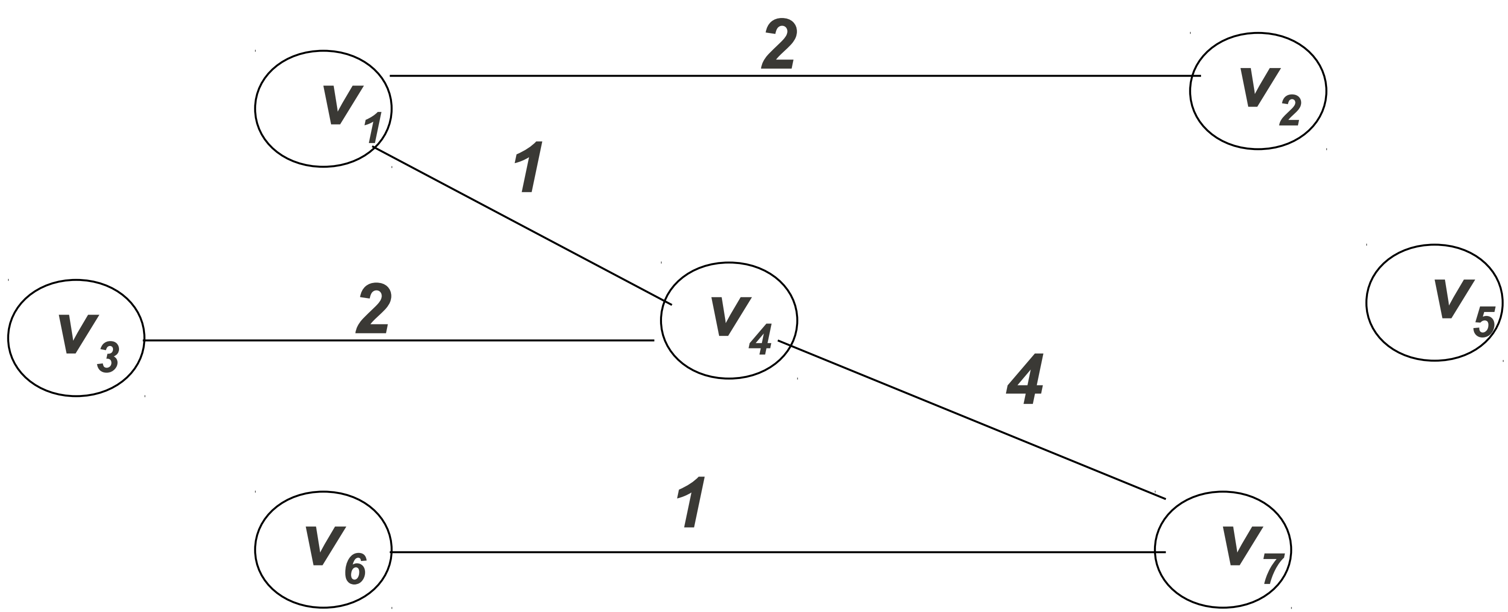
Kruskal's Algorithm



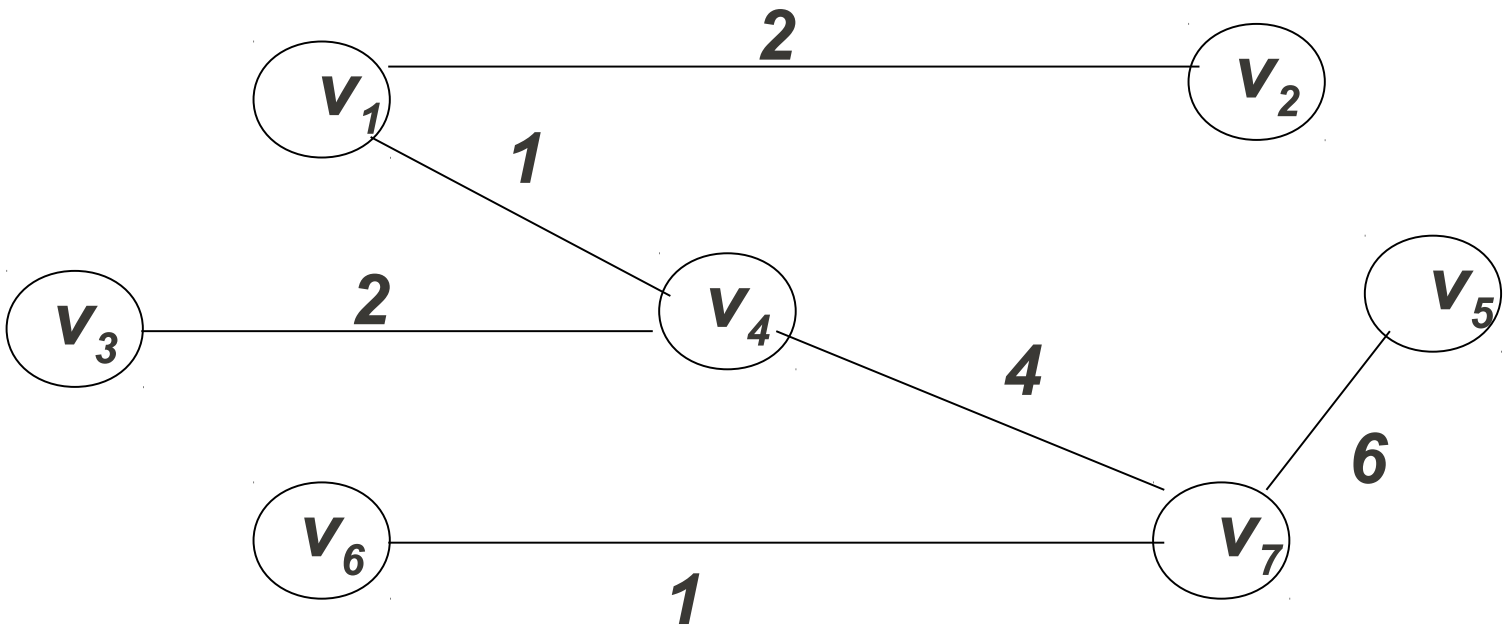


Kruskal's Algorithm





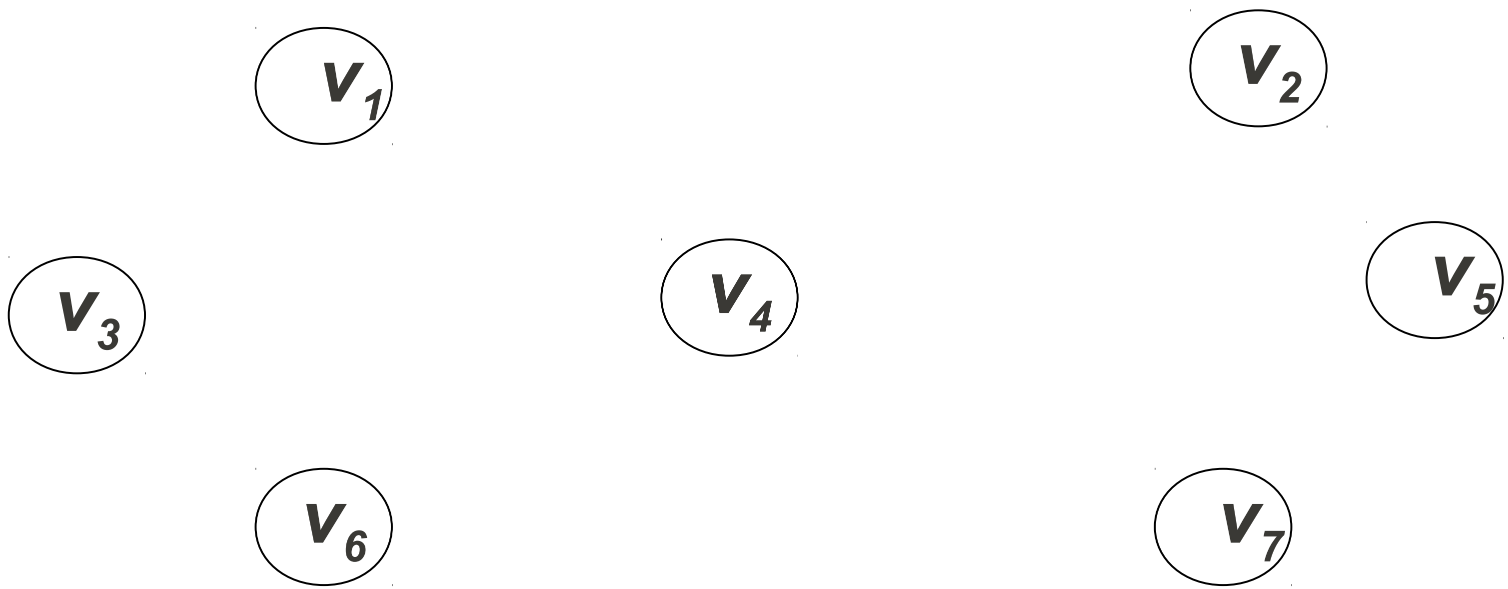
Kruskal's Algorithm

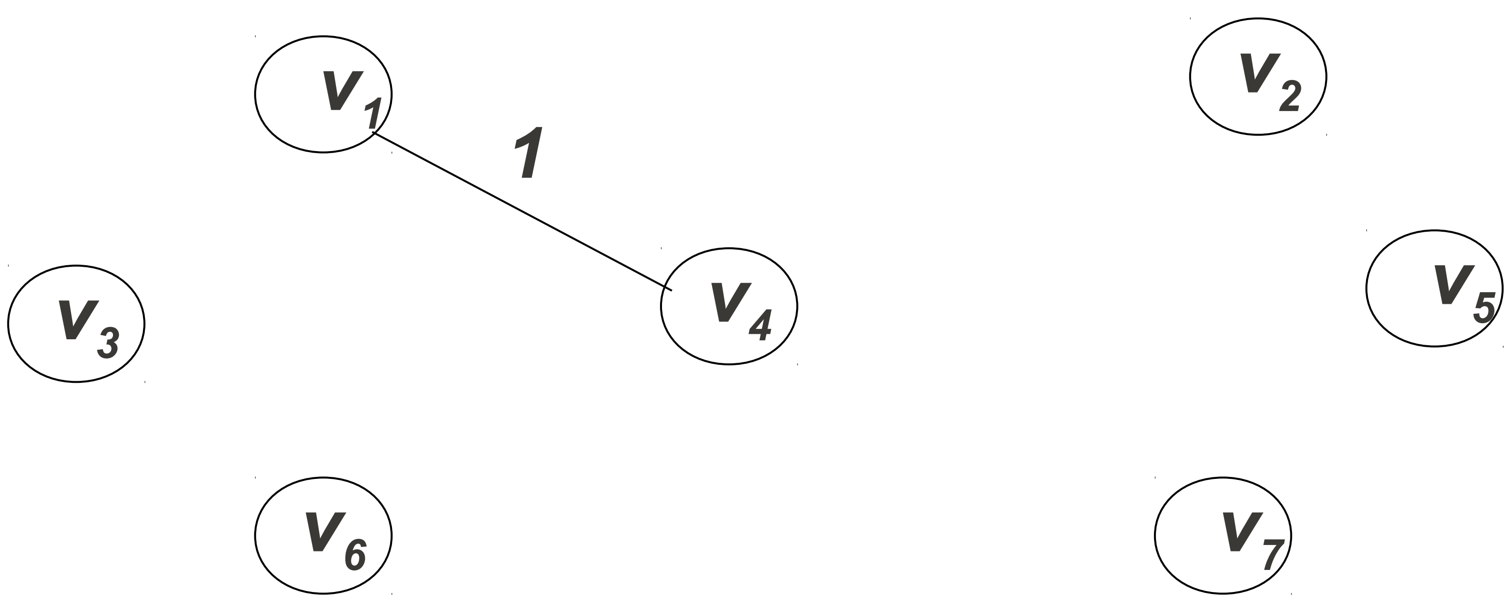


Prim's Algorithm

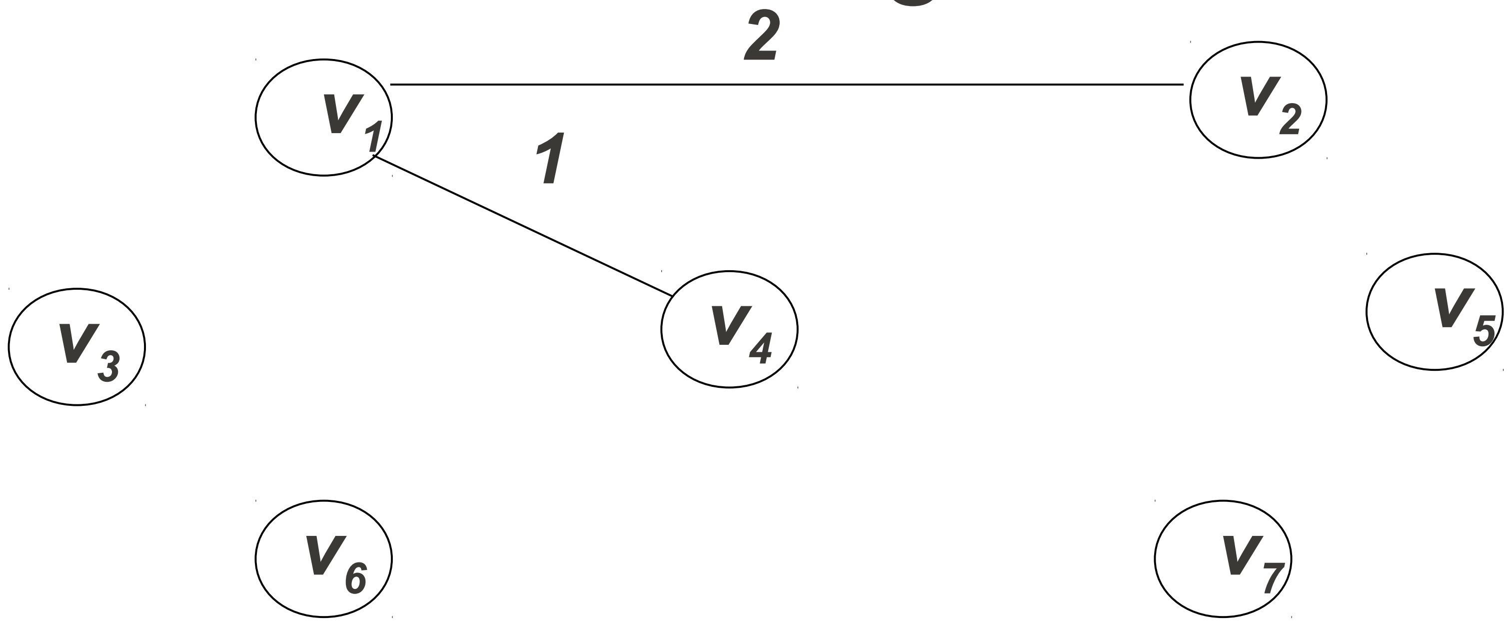
- Prim's algorithm, like Kruskal's, constructs the minimum spanning tree by selecting edges one at a time.
- The greedy criterion used to determine the next edge to select is: *From the remaining edges, select a least-cost edge whose addition to the set of selected edges forms a tree*
- Prim's algorithm begins with a tree that contains a single vertex
- At any point in the algorithm, we can see that we have a set of vertices that have already been included in the tree, the rest of the vertices have not.
- The algorithm then finds, at each stage, a new vertex to add to the tree by choosing the edge (u,v) such that the cost of (u,v) is the smallest among all edges where u is in the tree and v is not.

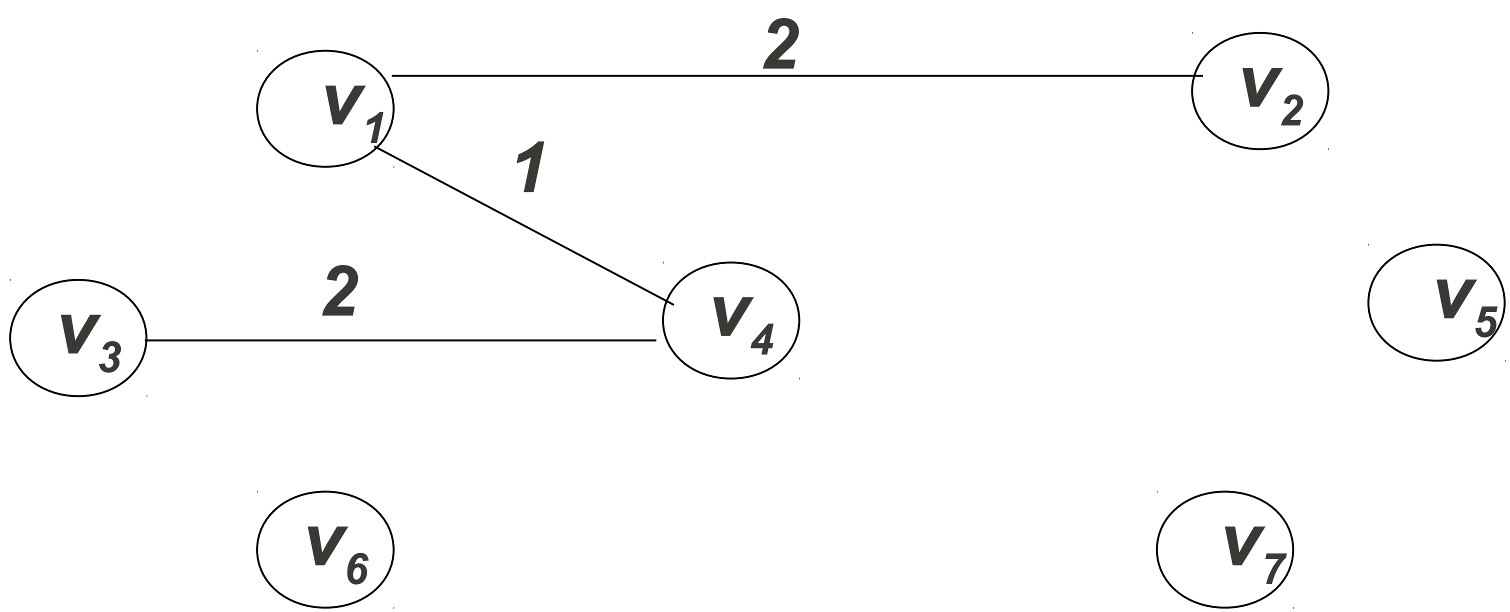
Prim’s Algorithm



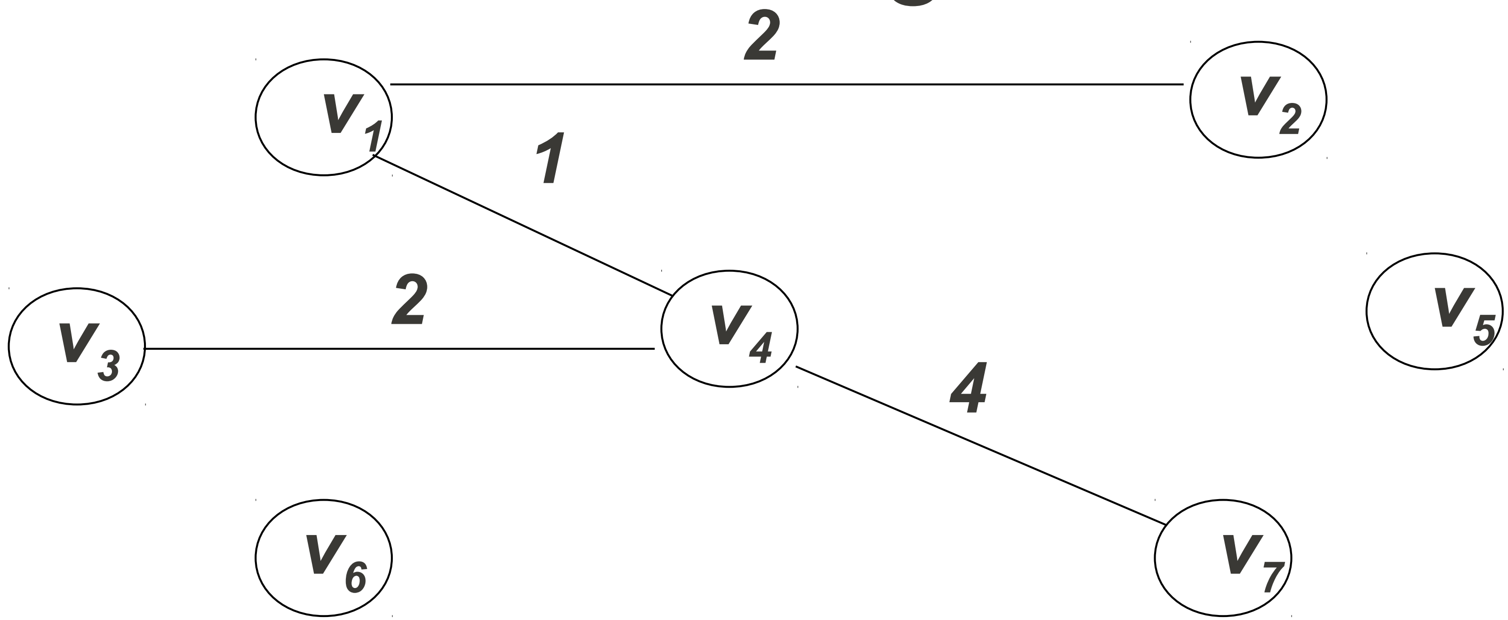


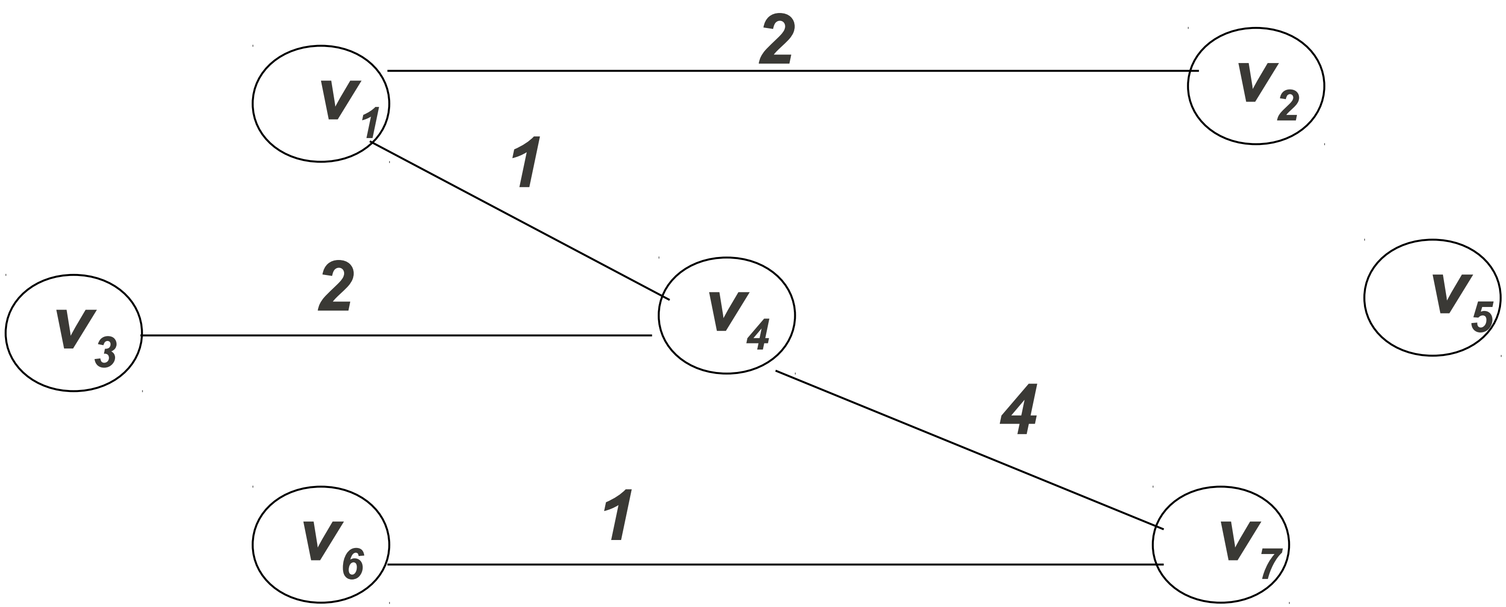
Prim's Algorithm





Prim's Algorithm





Prim's Algorithm