Fall 2019

Graphs - Part:2

CMPE 250 - Data Structures & Algorithms

Presenter: Meriç Turan

21 NOVEMBER 2019

Outline

- Depth-First Search (DFS)
- Dijkstra's Algorithm
- Prim's Minimum Spanning Tree (MST)

DFS DEMO

Dijkstra's Algorithm

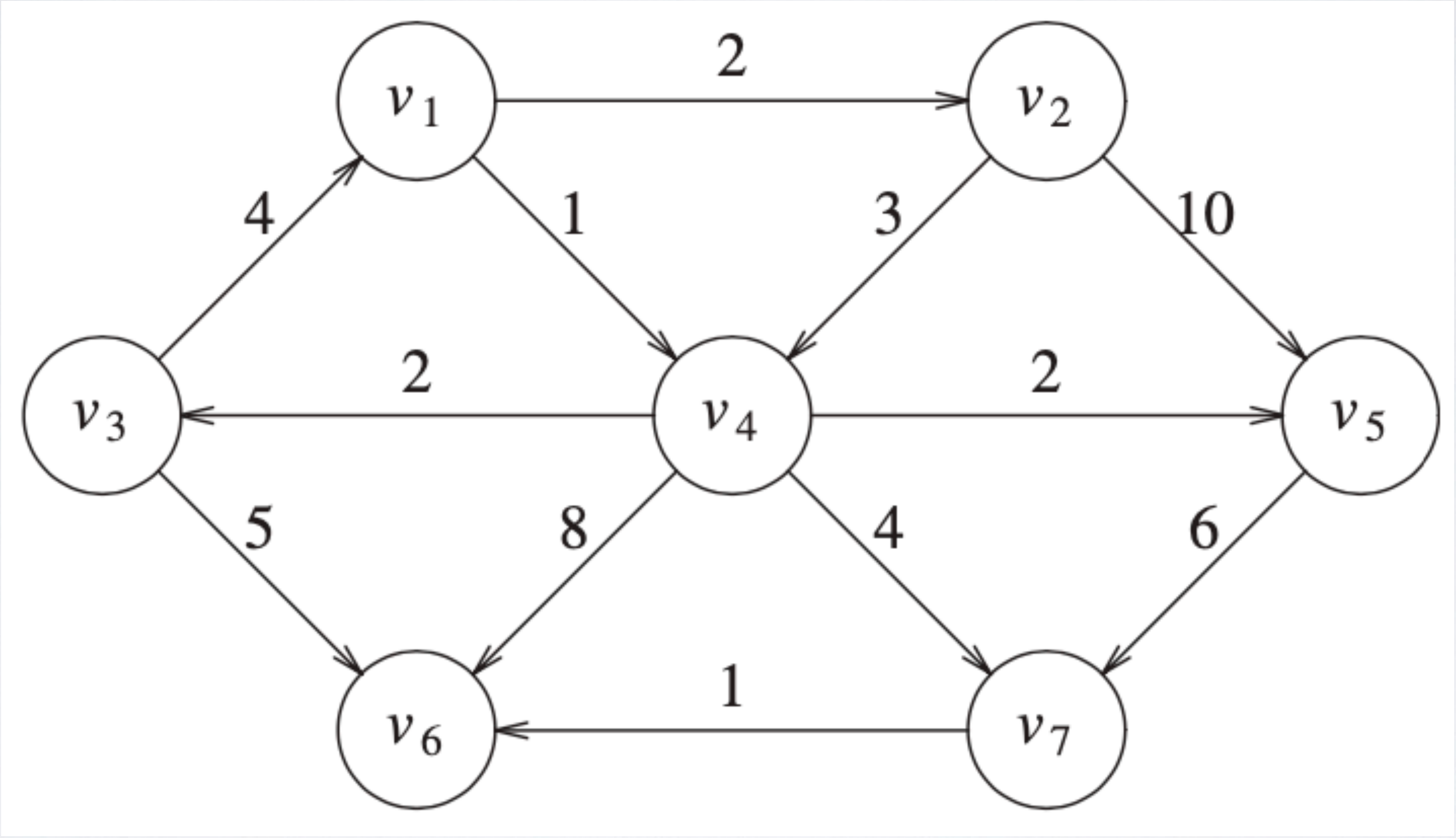
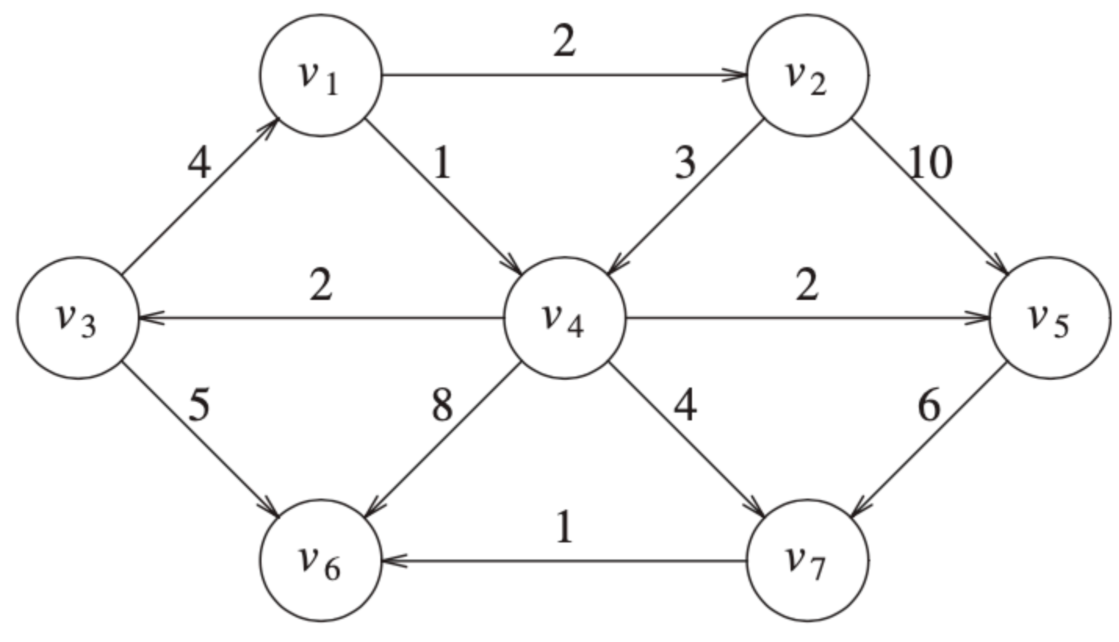


Figure: A directed graph (v_1 is source).

Dijkstra's Algorithm



v	*known*	d_v	p_v
v_1	F	0	0
v_2	F	∞	0
v_3	F	∞	0
v_4	F	∞	0
v_5	F	∞	0
v_6	F	∞	0
v_7	F	∞	0

Initial configuration table

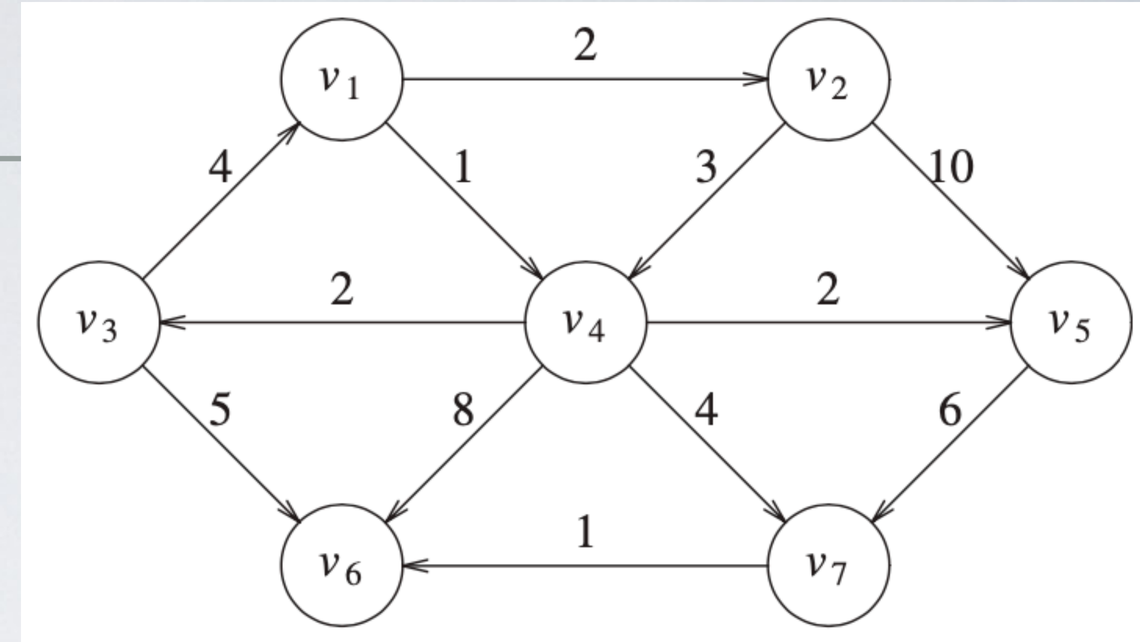
v	*known*	d_v	p_v
v_1	T	0	0
v_2	F	2	v_1
v_3	F	∞	0
v_4	F	1	v_1
v_5	F	∞	0
v_6	F	∞	0
v_7	F	∞	0

After v_1 is declared known

v	*known*	d_v	p_v
v_1	T	0	0
v_2	F	2	v_1
v_3	F	3	v_4
v_4	T	1	v_1
v_5	F	3	v_4
v_6	F	9	v_4
v_7	F	5	v_4

After v_4 is declared known

Dijkstra's Algorithm



v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	F	3	v_4
v_4	T	1	v_1
v_5	F	3	v_4
v_6	F	9	v_4
v_7	F	5	v_4

After v_2 is declared known

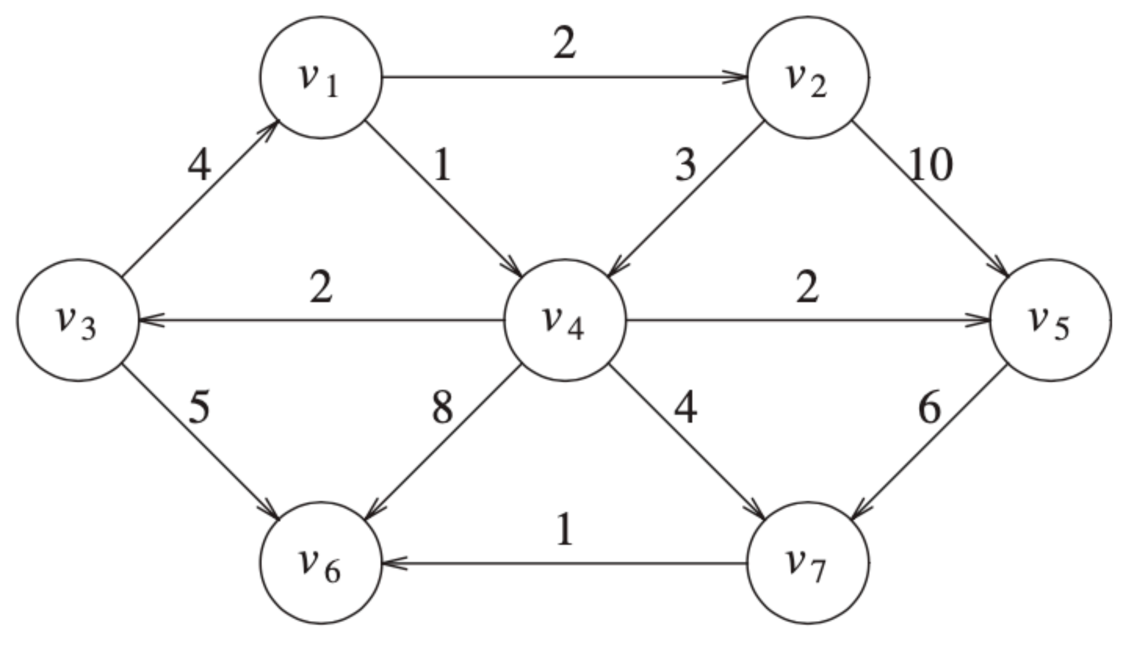
v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	3	v_4
v_4	T	1	v_1
v_5	T	3	v_4
v_6	F	8	v_3
v_7	F	5	v_4

After v_5 and v_3 is declared known

v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	3	v_4
v_4	T	1	v_1
v_5	T	3	v_4
v_6	F	6	v_7
v_7	T	5	v_4

After v_7 is declared known

Dijkstra's Algorithm



v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	3	v_4
v_4	T	1	v_1
v_5	T	3	v_4
v_6	T	6	v_7
v_7	T	5	v_4

After v_6 is declared known, and algorithm terminates

DIJKSTRA DEMO

Minimum Spanning Tree

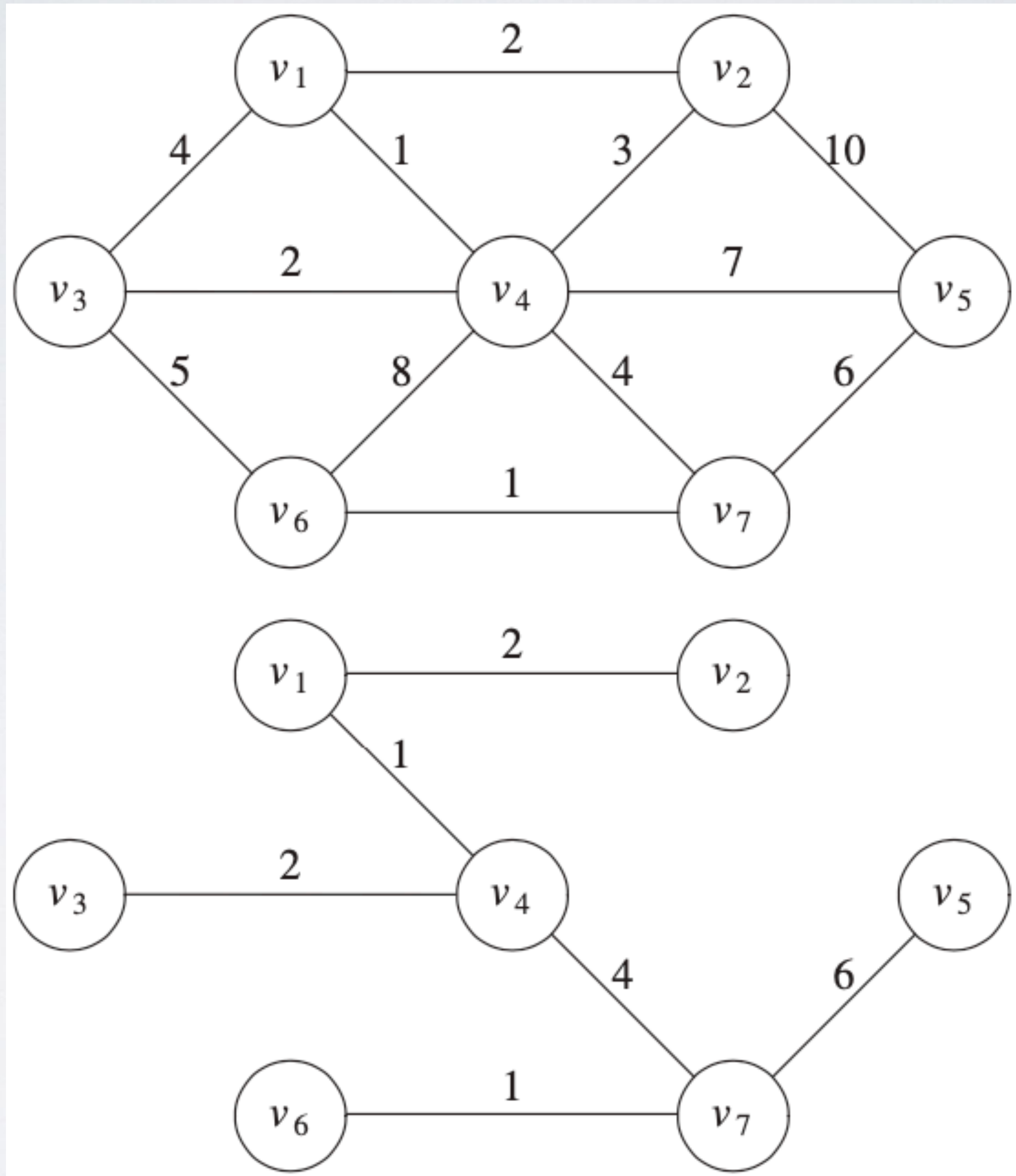


Figure: A graph G and its minimum spanning tree.

Prim's Minimum Spanning Tree

- One way to compute a minimum spanning tree is to grow the tree in successive stages.
- In each stage, one node is picked as root, and we add an edge, and thus an associated vertex, to the tree.
- 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.
- In each step, one edge and one vertex is added to the tree.

Prim's Minimum Spanning Tree

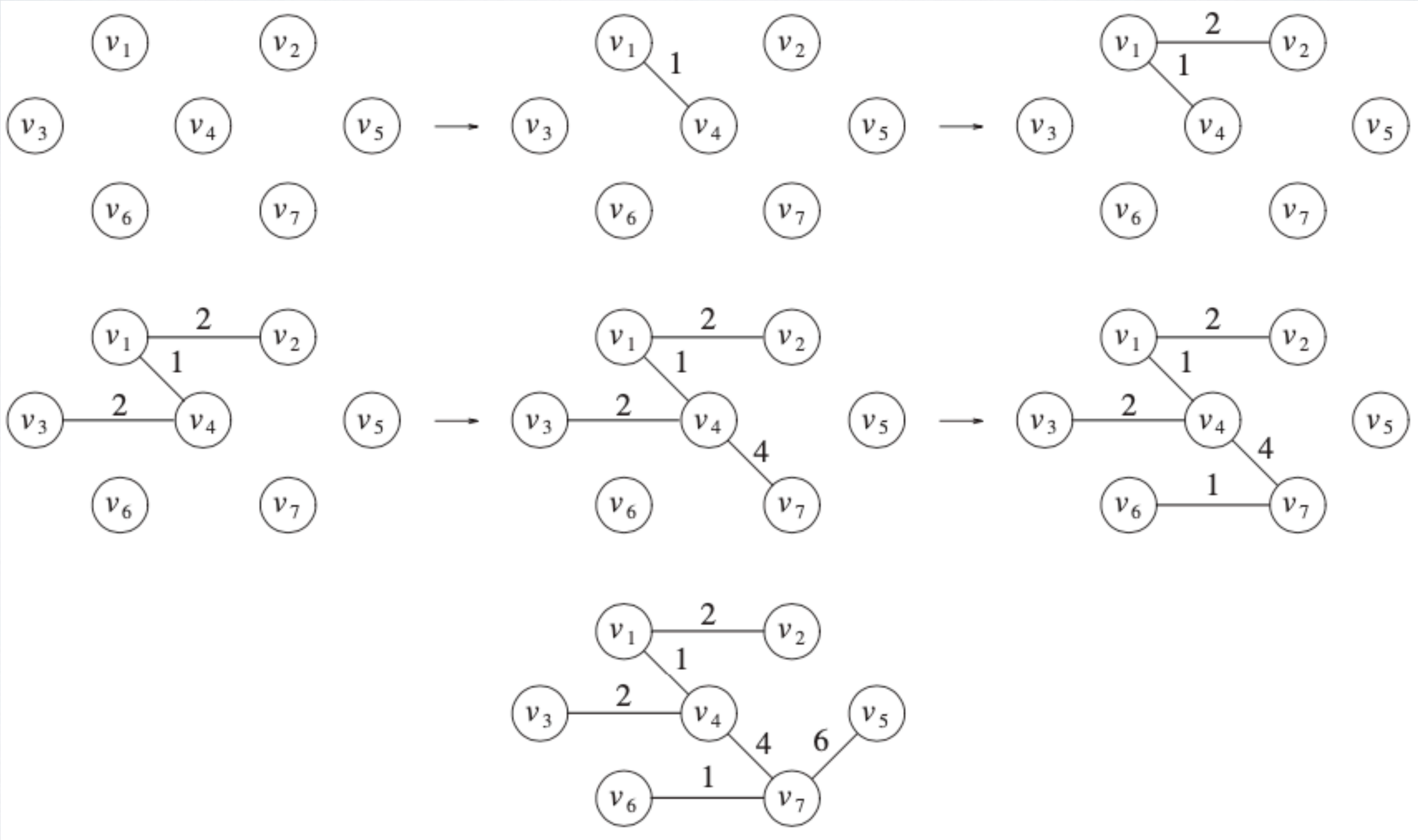
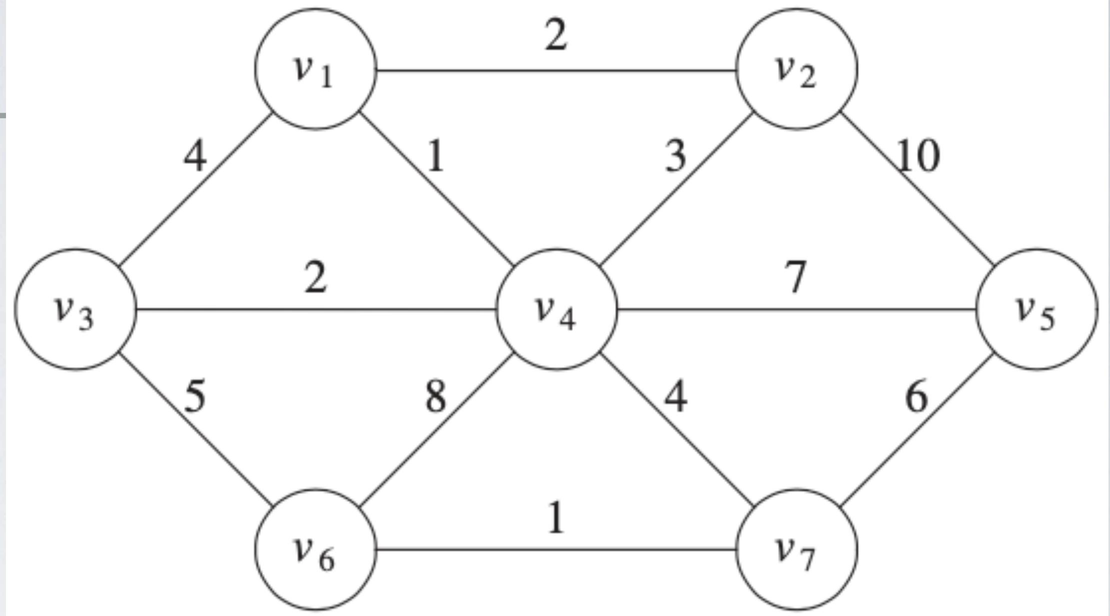


Figure: Prim's algorithm after each step.

Prim's Minimum Spanning Tree

- We can see that Prim's algorithm is essentially identical to Dijkstra's algorithm for shortest paths.
- As before, for each vertex we keep values d_v and p_v and an indication of whether it is *known* or *unknown*.
- d_v is the weight of the shortest edge connecting v to a known vertex, and p_v, as before, is the last vertex to cause a change in d_v.
- Rest of the algorithm is exactly the same, with the exception that since the definition of d_v is different, so is the update rule.
- For this problem, update rule is even simpler than before: After a vertex, v, is selected, for each unknown w adjacent to v, $d_w = \min(d_w, c_{w,v})$.

Prim's Minimum Spanning Tree



v	*known*	d_v	p_v
v_1	F	0	0
v_2	F	∞	0
v_3	F	∞	0
v_4	F	∞	0
v_5	F	∞	0
v_6	F	∞	0
v_7	F	∞	0

Initial configuration table

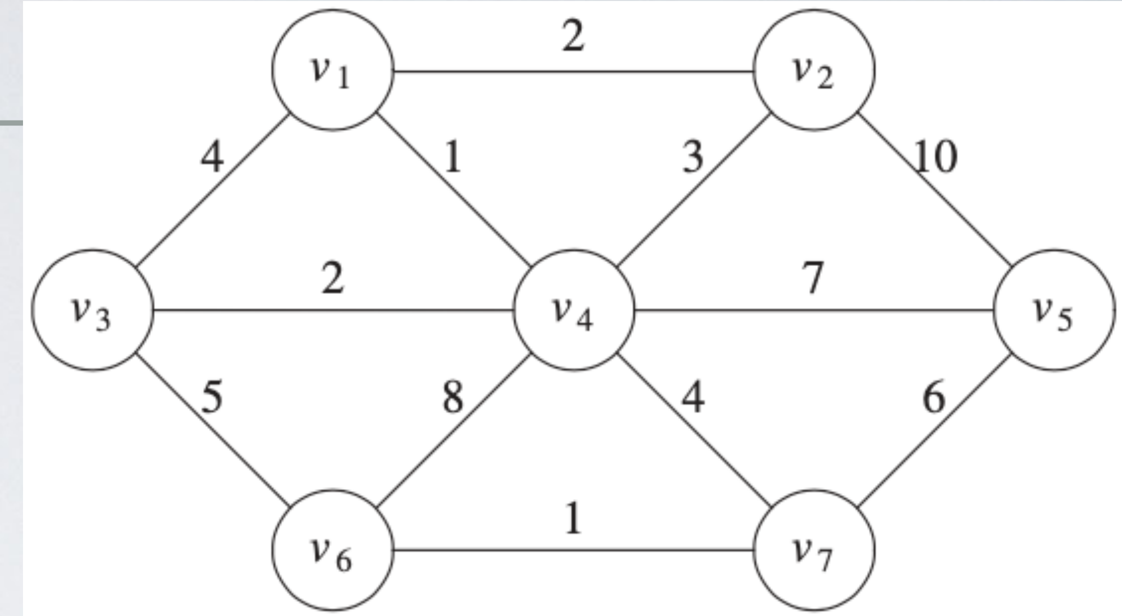
v	*known*	d_v	p_v
v_1	T	0	0
v_2	F	2	v_1
v_3	F	4	v_1
v_4	F	1	v_1
v_5	F	∞	0
v_6	F	∞	0
v_7	F	∞	0

After v_1 is declared known

v	*known*	d_v	p_v
v_1	T	0	0
v_2	F	2	v_1
v_3	F	2	v_4
v_4	T	1	v_1
v_5	F	7	v_4
v_6	F	8	v_4
v_7	F	4	v_4

After v_4 is declared known

Prim's Minimum Spanning Tree



v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	2	v_4
v_4	T	1	v_1
v_5	F	7	v_4
v_6	F	5	v_3
v_7	F	4	v_4

After v_2 and v_3 is declared known

v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	2	v_4
v_4	T	1	v_1
v_5	F	6	v_7
v_6	F	1	v_7
v_7	T	4	v_4

After v_7 is declared known

v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	2	v_4
v_4	T	1	v_1
v_5	T	6	v_7
v_6	T	1	v_7
v_7	T	4	v_4

After v_6 and v_5 is declared known, and Prim's algorithm terminates

Prim's Minimum Spanning Tree

- Edges in the spanning tree can be read from the table:
  - (v2, v1), (v3, v4), (v4, v1), (v5, v7), (v6, v7), (v7, v4).
  - Total cost is 16.
- Entire implementation of this algorithm is virtually identical to that of Dijkstra's algorithm, and everything that was said about the analysis of Dijkstra's algorithm applies here.
- Be aware that Prim's algorithm runs on undirected graphs, so when coding it, remember to put every edge in two adjacency lists.
- Running time is $O(|V|^2)$ without heaps, which is optimal for dense graphs, and $O(|E| \log |V|)$ using binary heaps, which is good for sparse graphs.

PRIM'S MST DEMO