Algorithms

Shortest Paths

Shortest Paths

- Let G be a weighted graph. The **length** (or weight) of a path, P, is the sum of the weights of the edges of P.
- The **distance** from a vertex v to a vertex u in G, denoted d(u,v) is the length of a minimum length path(also called shortest path) from v to u, if such a path exists.
- Given as input a weighted graph, G = (V,E), and a distinguished vertex, s, we want to find the shortest weighted path from s to every other vertex in G. This is known as the **single source shortest paths** problem.
- One algorithm which can be used to compute the shortest path is Dijkstra's algorithm.

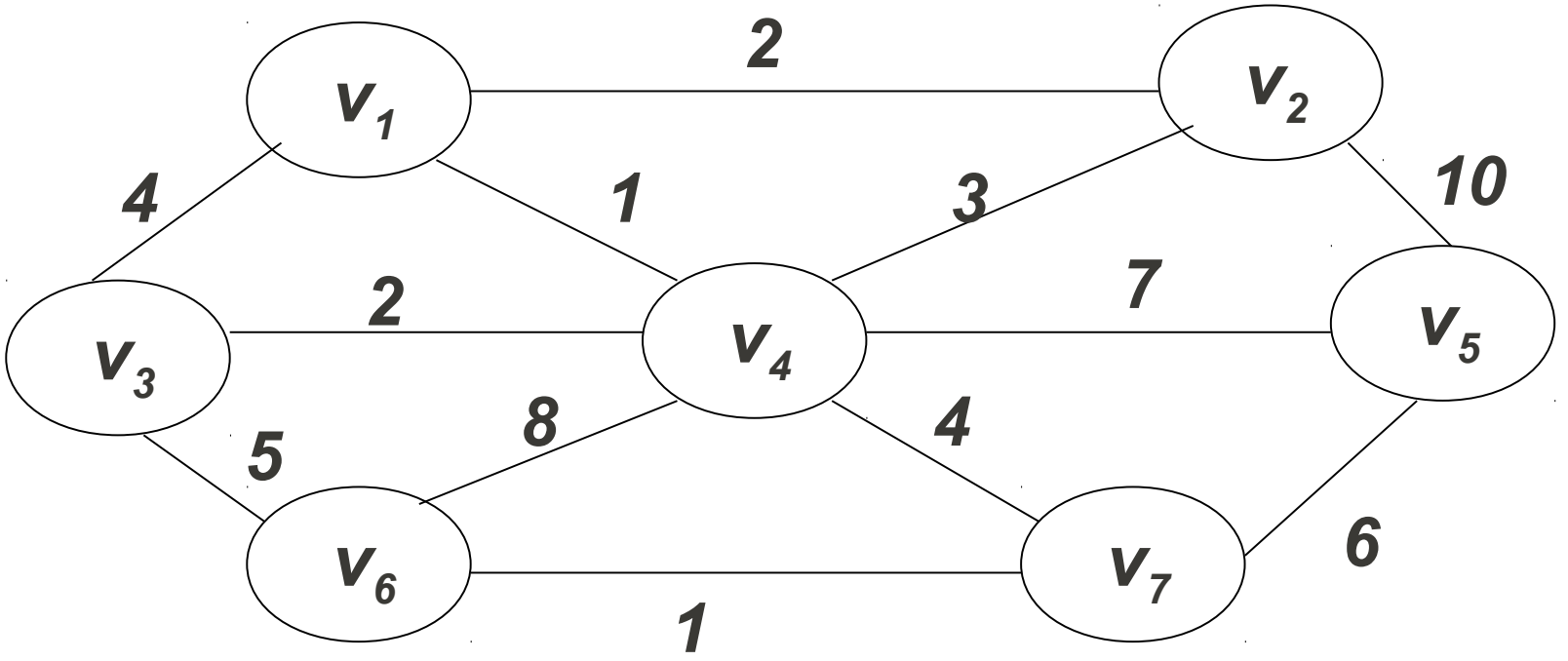
Dijkstra's Algorithm

- Dijkstra's algorithm is another example of a greedy algorithm
- It generates the shortest paths in stages.
- In each stage, a shortest path to a new destination vertex is generated
- The destination for the next shortest path is selected using the greedy criterion:
  From the vertices to which a shortest path has not been generated, select one that results in a least path length.

Dijkstra's Algorithm

- We begin with the trivial path from the source vertex to itself. This path has no edges and has a length of 0.
- In each stage of the greedy algorithm, the next shortest path is generated. This next shortest path is the shortest possible one edge extension of an already generated shortest path.

Dijkstra's Algorithm



- Using the above graph we choose v_1 as our starting vertex.
- We now make this node the reference node and mark it as known.
- We maintain a table of distances d_v and vertices which cause a change to d_v which we call p_v

Dijkstra's Algorithm

Initial table configuration

vertex	***known***	d_v	p_v
v_1	No	0	0
v_2	No	-	0
v_3	No	-	0
v_4	No	-	0
v_5	No	-	0
v_6	No	-	0
v_7	No	-	0

Dijkstra's Algorithm

- We now calculate the distance from v_1 to all of its adjacent nodes which are not known and update the table

vertex	*known*	d_v	p_v
v_1	Yes	0	0
v_2	No	2	v_1
v_3	No	4	v_1
v_4	No	1	v_1
v_5	No	-	0
v_6	No	-	0
v_7	No	-	0

Dijkstra's Algorithm

- We now mark v_4 as known and recalculate the table

vertex	***known***	d_v	p_v
v_1	Yes	0	0
v_2	No	2	v_1
v_3	No	3	v_4
v_4	Yes	1	v_1
v_5	No	3	v_4
v_6	No	9	v_4
v_7	No	5	v_4

Dijkstra's Algorithm

- We now mark v_2 as known and recalculate the table

vertex	***known***	d_v	p_v
v_1	Yes	0	0
v_2	Yes	2	v_1
v_3	No	3	v_4
v_4	Yes	1	v_1
v_5	No	3	v_4
v_6	No	9	v_4
v_7	No	5	v_4

Dijkstra's Algorithm

- We now mark v_3 as known and recalculate the table

vertex	***known***	d_v	p_v
v_1	Yes	0	0
v_2	Yes	2	v_1
v_3	Yes	3	v_4
v_4	Yes	1	v_1
v_5	No	3	v_4
v_6	No	8	v_3
v_7	No	5	v_4

Dijkstra's Algorithm

- We now mark v_5 as known and recalculate the table

vertex	***known***	d_v	p_v
v_1	Yes	0	0
v_2	Yes	2	v_1
v_3	Yes	3	v_4
v_4	Yes	1	v_1
v_5	Yes	3	v_4
v_6	No	8	v_3
v_7	No	5	v_4

Dijkstra's Algorithm

- We now mark v_7 as known and recalculate the table

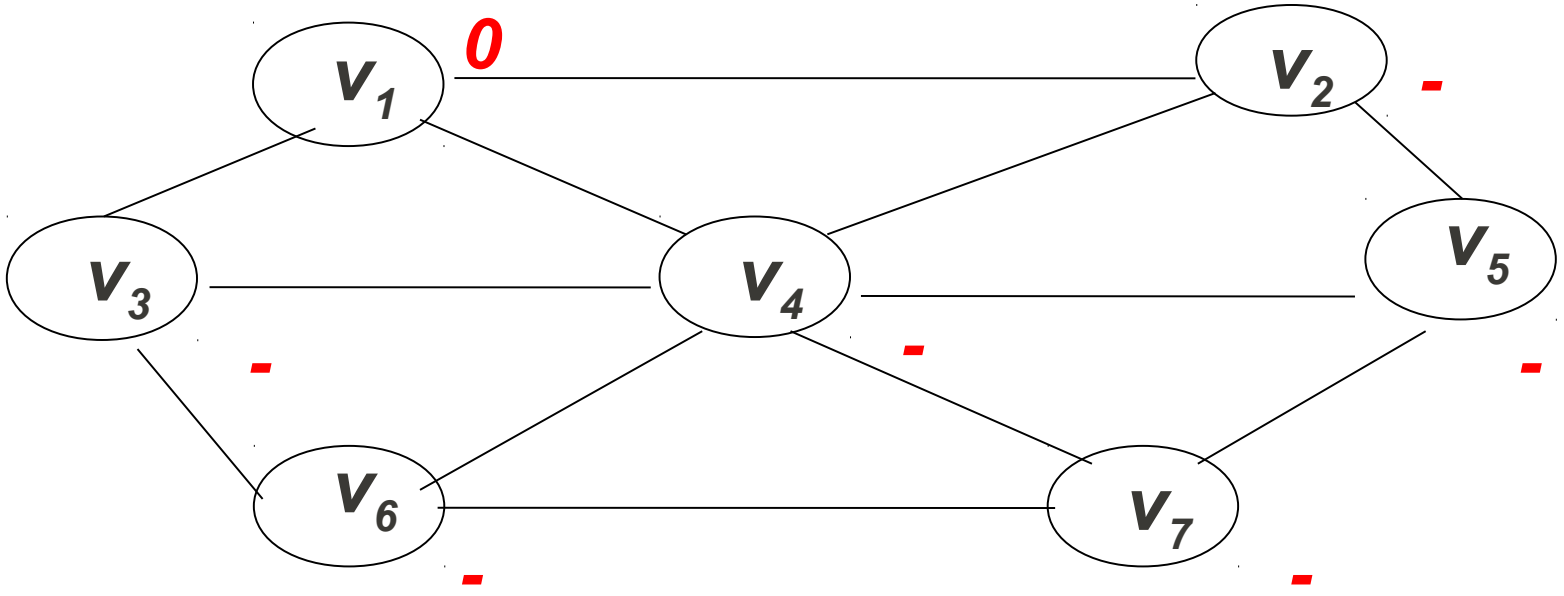
vertex	***known***	d_v	p_v
v_1	Yes	0	0
v_2	Yes	2	v_1
v_3	Yes	3	v_4
v_4	Yes	1	v_1
v_5	Yes	3	v_4
v_6	No	6	v_7
v_7	Yes	5	v_4

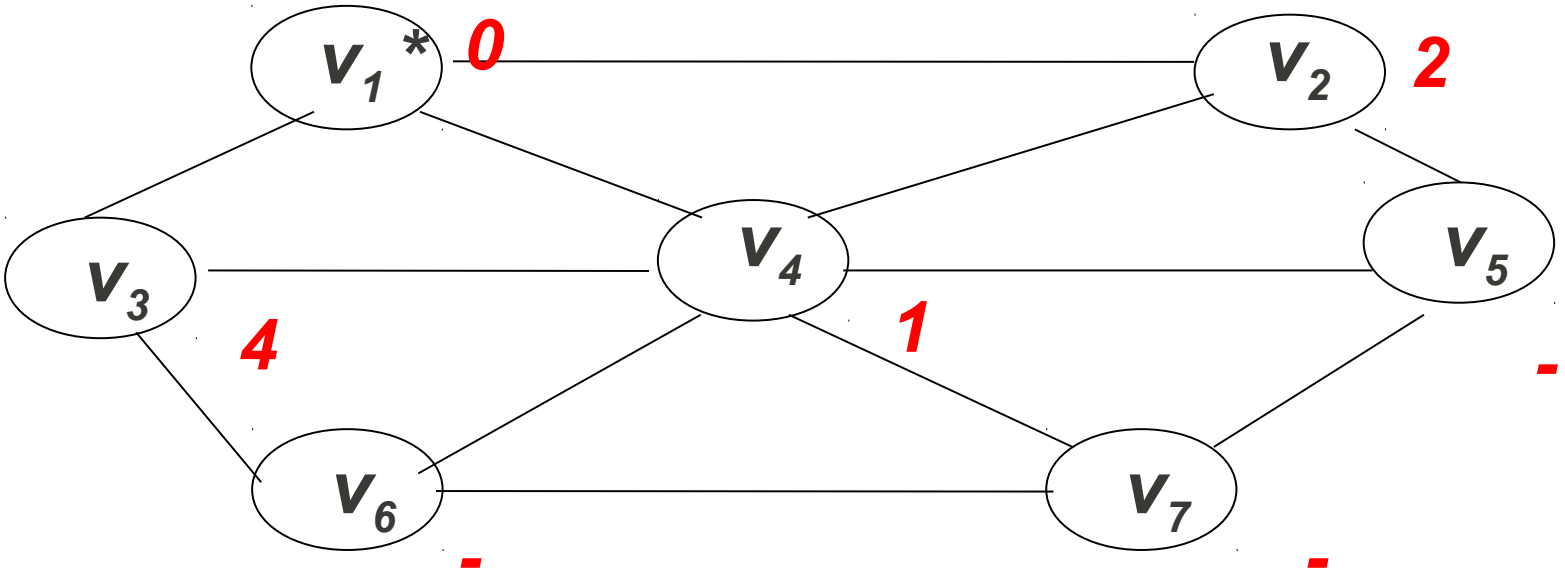
Dijkstra's Algorithm

- We now mark v_6 as known

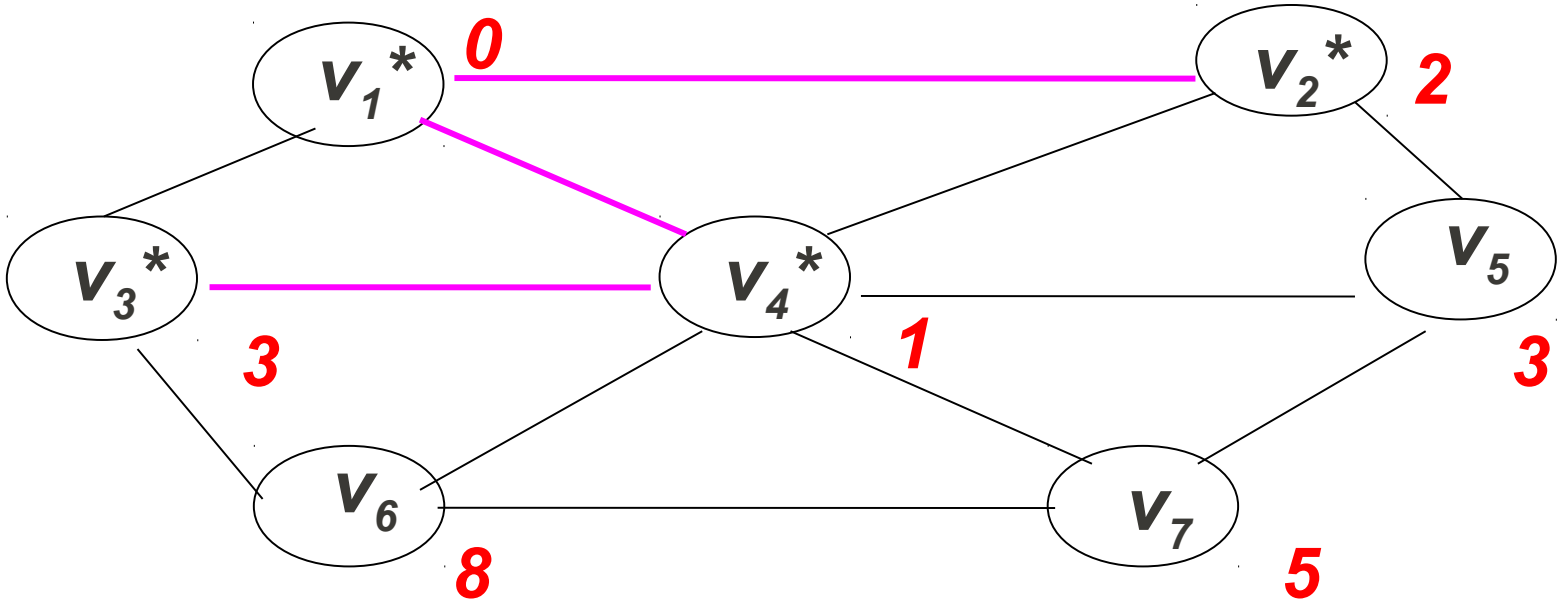
vertex	***known***	d_v	p_v
v_1	Yes	0	0
v_2	Yes	2	v_1
v_3	Yes	3	v_4
v_4	Yes	1	v_1
v_5	Yes	3	v_4
v_6	Yes	6	v_7
v_7	Yes	5	v_4

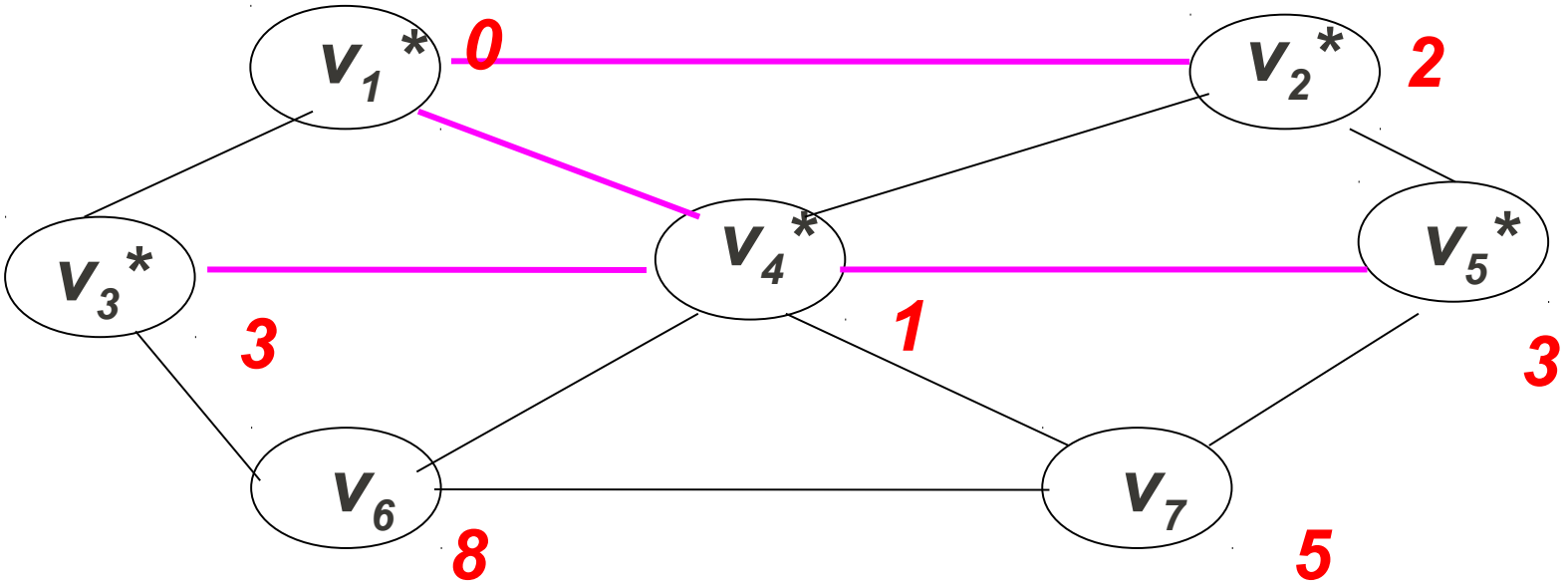
Dijkstra's Algorithm





Dijkstra's Algorithm





Dijkstra's Algorithm