ALGORITHM 1 Dijkstra's Algorithm.

procedure *Dijkstra*(G: weighted connected simple graph, with
all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$
where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
$L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all
other labels are ∞, and S is the empty set}
while $z \notin S$
$u :=$ a vertex not in S with $L(u)$ minimal
$S := S \cup \{u\}$
for all vertices v not in S
if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
{this adds a vertex to S with minimal label and updates the
labels of vertices not in S}
return $L(z)$ {$L(z) =$ length of a shortest path from a to z}

Example 2 illustrates how Dijkstra's algorithm works. Afterward, we will show that this algorithm always produces the length of a shortest path between two vertices in a weighted graph.

EXAMPLE 2 Use Dijkstra's algorithm to find the length of a shortest path between the vertices a and z in the weighted graph displayed in Figure 4(a).

Solution: The steps used by Dijkstra's algorithm to find a shortest path between a and z are shown in Figure 4. At each iteration of the algorithm the vertices of the set S_k are circled. A shortest path from a to each vertex containing only vertices in S_k is indicated for each iteration. The algorithm terminates when z is circled. We find that a shortest path from a to z is a, c, b, d, e, z, with length 13. ◂

Remark: In performing Dijkstra's algorithm it is sometimes more convenient to keep track of labels of vertices in each step using a table instead of redrawing the graph for each step.

Next, we use an inductive argument to show that Dijkstra's algorithm produces the length of a shortest path between two vertices a and z in an undirected connected weighted graph. Take as the inductive hypothesis the following assertion: At the kth iteration

(i) the label of every vertex v in S is the length of a shortest path from a to this vertex, and
(ii) the label of every vertex not in S is the length of a shortest path from a to this vertex that contains only (besides the vertex itself) vertices in S.

When $k = 0$, before any iterations are carried out, $S = \emptyset$, so the length of a shortest path from a to a vertex other than a is ∞. Hence, the basis case is true.

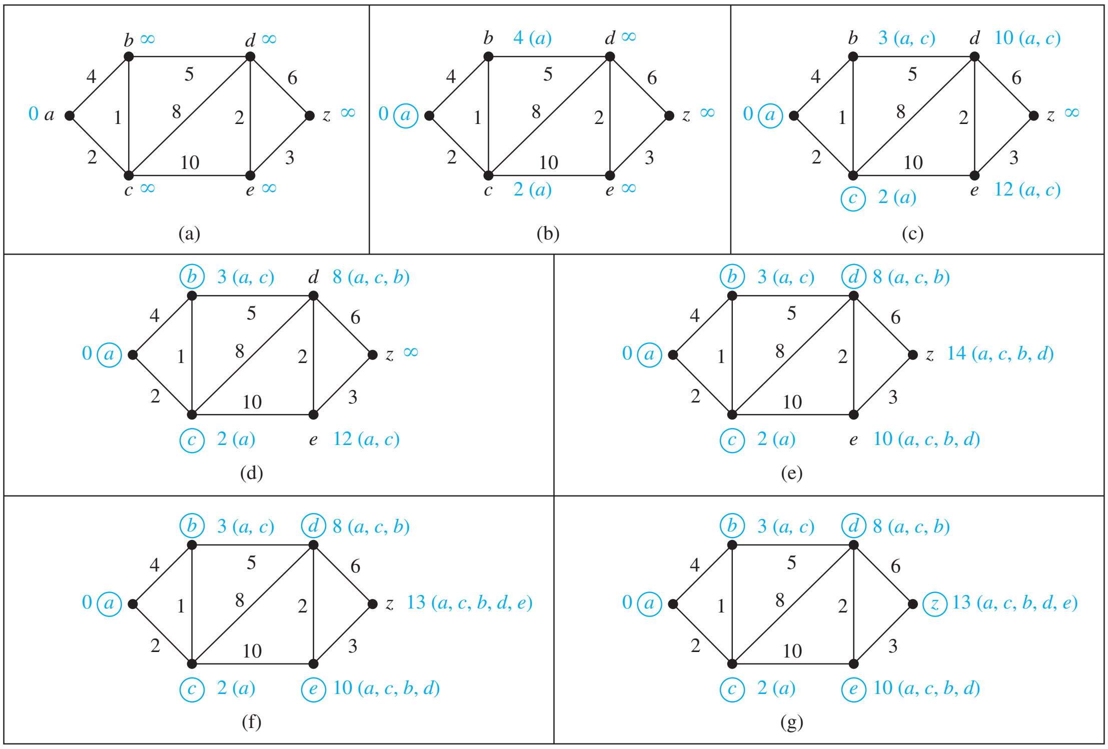

FIGURE 4 **Using Dijkstra's Algorithm to Find a Shortest Path from a to z.**

Assume that the inductive hypothesis holds for the kth iteration. Let v be the vertex added to S at the $(k + 1)$st iteration, so v is a vertex not in S at the end of the kth iteration with the smallest label (in the case of ties, any vertex with smallest label may be used).

From the inductive hypothesis we see that the vertices in S before the $(k + 1)$st iteration are labeled with the length of a shortest path from a. Also, v must be labeled with the length of a shortest path to it from a. If this were not the case, at the end of the kth iteration there would be a path of length less than $L_k(v)$ containing a vertex not in S [because $L_k(v)$ is the length of a shortest path from a to v containing only vertices in S after the kth iteration]. Let u be the first vertex not in S in such a path. There is a path with length less than $L_k(v)$ from a to u containing only vertices of S. This contradicts the choice of v. Hence, (i) holds at the end of the $(k + 1)$st iteration.

Let u be a vertex not in S after $k + 1$ iterations. A shortest path from a to u containing only elements of S either contains v or it does not. If it does not contain v, then by the inductive hypothesis its length is $L_k(u)$. If it does contain v, then it must be made up of a path from a to v of shortest possible length containing elements of S other than v, followed by the edge from v to u. In this case, its length would be $L_k(v) + w(v, u)$. This shows that (ii) is true, because $L_{k+1}(u) = \min\{L_k(u), L_k(v) + w(v, u)\}$.

We now state the thereom that we have proved.

THEOREM 1 Dijkstra's algorithm finds the length of a shortest path between two vertices in a connected simple undirected weighted graph.