Dijkstra’s Algorithm for Finding the Shortest Path Through a Weighted Graph
E. L. Lady
(December 1, 1999)

The way the algorithm works is to put labels on a growing number of vertices. A label on a vertex \( v \) will have two parts: a length \( L(v) \) and a pointer back to another vertex. \( L(v) \) represents the length of the shortest known path (to date) from \( a \) to \( v \), and the pointer shows the vertex that precedes \( v \) on that path. At each step of the algorithm we will have:

A set of vertices \( S \). Vertices in \( S \) will all have permanent labels.

One vertex \( u \) which is the newest vertex added to \( S \).

Vertices which are one step away from \( S \) will all have provisional labels.

Vertices more than one step away from \( S \) are unlabeled, or we can label them with \( L(v) = \infty \).

As we go through the algorithm, at each step the following things will be true:

(1) For vertices \( v \in S \), \( L(v) \) is the best possible. (This is why the labels on vertices in \( S \) are permanent.)

(2) For vertices \( v \) one step away from \( S \), \( L(v) \) is the length of the shortest possible path from \( a \) to \( v \) that stays inside \( S \) up until the last edge: however there may be shorter paths if we use more vertices outside \( S \).

Algorithm: At each step, Dijkstra’s algorithm does two things:

(a) It chooses a new vertex \( u \) to add to \( S \).

(b) It updates the labels on all the vertices which are one step away from (the new) \( S \).

We will describe how it does these two things and at the same time we will prove by induction that assertions (1) and (2) are true at each step. (Note: In what follows, when we speak of the “smallest” value chosen from a certain set, we mean that there is no smaller possible value, but perhaps some other values could be equal.)

(a) If we are just starting the algorithm, we choose \( u \) to be \( a \) and we set \( L(a) = 0 \). At all the later steps, we choose \( u \) to be one step away from \( S \) and having the smallest possible \( L(u) \). (\( L(u) \) will be already given from the previous step.) We then adjoin \( u \) to \( S \).
To see that assertion (1) will still be true for the new $S$, we need to see that $L(u)$ is the length of the shortest possible path from $a$ to $u$. This is certainly true at beginning of the algorithm, when $u = a$. At the later steps, we know that $L(u)$ is the length of the shortest possible path that stays completely in $S$. (This is because (2) is true for the old $S$.) But assertion (2) (for the old $S$) also tells us that any path from $a$ to $u$ that goes outside $S$ already becomes at least as long as $L(u)$ at the moment it leaves $S$. In other words, if some path from $a$ to $u$ goes outside $S$, and $v$ is the first vertex on this path which is outside $S$, then we know from assertion (2) that the length of the part of the path from $a$ to $v$ is already at least as long as $L(v)$, and $L(v) \geq L(u)$. Thus no path from $a$ to $u$ could be shorter than $L(u)$.

(b) Now we need to update the labels on all vertices $v$ which are one step away from the new $S$. We actually only need to worry about the vertices adjacent to $u$. If $v$ is adjacent to $u$ and either $v$ is still unlabeled or if the existing $L(v)$ is larger than $L(u) + w(u,v)$, then we set $L(v)$ equal to $L(u) + w(u,v)$ and change the pointer on $v$ to point back to $u$.

Now we need to see that assertion (2) is true for these new labels. If we are at the very first step (so that $S = \{a\}$), then this is clear. At the later steps, notice that what we’ve done is to check whether any path to $v$ with $u$ as its next to the last vertex is shorter than the paths we already know about. So there are two things we need to rule out: i) that some vertex $w$ which is not adjacent to $u$ should have been given a new label; ii) that with the new $S$, for some vertex $v$ adjacent to $u$ there is now an even shorter path from $a$ to $v$ which lies in $S$ except for the last edge, and which does not have $u$ as the next to the last vertex.

Okay, first consider objection i). Suppose $w$ is one step away from $S$ and consider a path from $a$ to $w$ that lies inside $S$ until the very last edge, and suppose that next to the last vertex is $v$, where $v \in S$ and $v \neq u$. Then $v$ lies in the old $S$, and we already know from assertion (1) that the path we previously had from $a$ to $v$ was the shortest conceivable path; therefore monkeying around with this path can get us nowhere.

Now consider objection ii): The answer here is really the same as the answer to objection i): There’s no point in considering paths to $v$ that don’t have $u$ as their next to the last vertex, because the old label on $v$ already shows the best that can possibly be achieved that way.

Therefore assertions (1) and (2) are true at every step. Since there are a finite number of vertices, after enough steps we finally have $z \in S$, and assertion (1) then tells us that the label on $z$ gives the length of the shortest possible path from $a$ to $z$. Furthermore, by starting at $z$ and tracing backwards using the pointer part of the labels, we actually see what this shortest path is.