Dijkstra's Algorithm Verification

Yazhe Feng

March 14, 2019

1 Dijkstra's Algorithm

1.1 Pseudocode

Given input graph g and source node s with types:

g : Graph gsize weight
s : Node gsize

We denote (u, v) as an edge from node u to v, $weight(u, v)$ as the weight of edge (u, v). For any two nodes u, w that are not connected by an edge in the graph, we let $weight(u, w)$ equals infinity. We define $unexplored$ as the list of unexplored nodes, and $dist$ as the list storing distance from s to each node $n \in g$

(initially $unexplored$ contains all nodes in graph g)
$unexplored : List(Node\ \ gsize)$
$unexplored = \{v : v \in g\}$

(node value is used to index $dist$, initially distance of all nodes are infinity except the source node)
$dist : List\ \ weight$
$dist[s] = 0, dist[a] = infinity, \forall a \in g, a \neq s$

The Dijkstra's Algorithm runs as follows: Given graph g and source node s, $dist$ stores the distance value from s to all nodes in g calculated by the Dijkstra's algorithm, $dist[v]$ gives the corresponding distance value of v from s. We index $unexplored$ and $dist$ by the number of iterations. Specifically, denote u_i as the node being explored at the i^th iteration, and denote $dist_i$, $unexplored_i$ as the value of distance list and unexplored list at the beginning of the i^{th} iteration. Then during each iteration the Dijkstra's Algorithm calculates $dist, unexplored, explored$ as follows:

$\forall k \geq 1$
`choose` $u_k \in unexplored_k$ `and` $\forall u' \in unexplored_k, dist_k[u_k] \leq dist_k[u']$
$unexplored_{k+1} = unexplored_k - \{u_k\}$
$\forall v \in g$,
$\quad dist_{k+1}[v] = min(dist_k[v], (dist_k[u_k] + weight(u_k, v)))$

1.2 Assumptions

1. Weight of edges are non-negative
2. Distance value can only be zero, infinity, or summation of edge weights
3. All nodes n and edge e are valid: $n, e \in g$

2 Definition

Definition 2.1. Path
(We adopt the definition of path presented in the `Discrete Mathematics with Applications` book by `SUSANNA S. EPP`.)

A path from node v to w is a finite alternating sequence of adjacent vertices and edges of G, which does not contain any repeated edge or vertex. A path from v to w has the form:

$$ve_0v_0e_1v_2....v_{n-1}e_nw$$

where e_i is an edge in g with endpoints v_{i-1}, v_i. We denote the set of paths from v to w as $path(v, w)$.

Definition 2.2. Prefix of Path
Given a path from node v to w : $p(v, w) = ve_0v_0e_1v_2....v_{n-1}e_nw$, the prefix of this $v - w$ path is defined as the subsequence of $p(v, w)$ that starts with v and ends with some node $w' \in p(v, w)$ (w' is a vertex in the sequence $p(v, w)$).

Definition 2.3. Length of Path
The length of a path $p = ve_0v_0e_1v_2....v_{n-1}e_nw$ is the sum of the weights of all edges in p. We write:

$$length(p) = \sum weight(e_i), \forall e_i \in p.$$

Definition 2.4. Shortest Path
Denote $\Delta(s, v)$ as the shortest path from s to v, and $\delta(v)$ as the length of $\Delta(s, v)$. $\Delta(s, v)$ must fulfills:

$$\Delta(s, v) \in path(s, v)$$
and
$$\forall p' \in path(s, v),\ \delta(v) = length(\Delta(s, v)) \leq length(p')$$

3 Proof of Correctness

3.1 Proof of Termination

The inner for loop is guaranteed to terminate as the algorithm goes through each adjacent node exactly once. As the size of list `unexplored` decreases by one during each iteration of the while loop, the algorithm is guaranteed to terminate.

3.2 Proof of Correctness

Denote *explored* as the list of nodes in g but not in *unexplored*, i.e., *explored* stored all nodes whose neighbors have been updated by the algorithm. We index *explored* by the number of iterations, such that $explored_i$ denotes the value of *explored* at the beginning of the i^{th} iteration.

Lemma 3.1. Given any two nodes v, w, the prefix of the shortest path $\Delta(v, w)$ is also a shortest path.

Proof. We will prove Lemma 3.1 by contradiction.
Consider any node q in the sequence of $\Delta(v, w)$, we have $\Delta(v, w) = ve_0v_0e_1v_2...v_iqv_j....v_{n-1}e_nw$. Suppose the prefix of $\Delta(v, w)$ from v to q, denote as $p(v, q)$, is not the shortest path from v to q. Then we know $p(v, q) = ve_0v_0e_1v_2...v_iq$ is a path from v to q and $length(p(v, q)) > length(\Delta(v, q))$.
Based on the definition of shortest path, we know:

$$length(\Delta(v, w)) \leq length(p), \forall p \in path(v, w)$$

Fenote the path after the node q as $p(q, w) = qv_j....v_{n-1}e_nw$, since $\Delta(v, w) = ve_0v_0e_1v_2...v_iqv_j....v_{n-1}e_nw$, then $\Delta(v, w) = p(v, q)+p(q, w)$, and that $length(\Delta(v, w)) = length(p(v, q))+length(p(q, w))$. Then we have:

$$length(\Delta(v, w)) = length(p(v, q)) + length(p(q, w)) \leq length(p), \forall p \in path(v, w)$$

Since $p(v, q)$ is not the shortest path from v to q by assumption, there exists another $v - w$ path $p'(v, w)$ such that:

$$\begin{aligned}
&p'(v, w) \in path(v, w) \\
&p'(v, w) = \Delta(v, q) + p(q, w) \\
&length(p'(v, w)) = length(\Delta(v, q)) + length(p(q, w)) \\
&\qquad\qquad\qquad\quad < length(p(v, q)) + length(p(q, w)) \\
&\text{i.e. } length(p'(v, w)) < length(\Delta(v, w))
\end{aligned}$$

Hence we have reached a contradiction. Thus by the principle of prove by contradiction, for any the prefix $p(v, q)$ of $\Delta(v, w)$ is the shortest path from v to q. Lemma 3.1 holds. □

Lemma 3.2. After the n^{th} iteration for $n \geq 1$, forall node $v \in explored_{n+1}$, if $dist_{n+1}[v] \neq infinity$, then $dist_{n+1}[v]$ is the length of some $s - v$ path, i.e, $path(s, v) \neq \emptyset$.

Proof. We will prove `Lemma 3.2` by inducting on the number of iterations.
Let P(n) be: After the n^{th} iteration, $n \geq 1$, for all node $v \in g$, if $dist_{n+1}[v] \neq infinity$, then $dist_{n+1}[v]$ is the length of some $s - v$ path.
Base Case: We shall show P(1) holds.
Based on the algorithm, initially $dist_1[s] = 0$ and for all node $v \in g, v \neq s, dist_1[v] = infinity$, then s is the only node whose distance value is not infinity. Based on the definition of path, the path from the source node s to itself is s, $path(s, s) = \{s\}$. Hence P(1) holds.
Inductive Hypothesis: Suppose $\forall i, 1 \leq i \leq k$, P(i) holds. That is, after the i^{th} iteration, $1 \leq i \leq k$, for all nodes $v \in g$, if $dist_{i+1}[v] \neq infinity$, then $dist_{n+1}[v]$ is the length of some $s - v$ path.
Inductive Step: We shall show P(k+1) holds.

For node u_{k+1} being explored during the $(k+1)^{th}$ iteration, based on the algorithm, $dist_{k+1}[u_{k+1}]$ is calculated as:

$$dist_{k+2}[u_{k+1}] = min(dist_{k+1}[u_{k+1}], dist_{k+1}[u_{k+1}] + weight(u_{k+1}, u_{k+1}))$$

Since the distance value from u_{k+1} to itself is 0, then $dist_{k+2}[u_{k+1}] = dist_{k+1}[u_{k+1}]$, and that $dist_{k+2}[u_{k+1}]$ and $dist_{k+1}[u_{k+1}]$ are the length of the same $s - u_{k+1}$ path if there exists one.
If $dist_{k+2}[u_{k+1}] \neq infinity$, then $dist_{k+1}[u_{k+1}] = dist_{k+2}[u_{k+1}] \neq infinity$. Since $k \leq k$ and $dist_{k+1}[u_{k+1}] \neq infinity$, then based on the inductive hypothesis, $dist_{k+1}[u_{k+1}]$ is the length of some $s - u_{k+1}$ path, and hence $dist_{k+2}[u_{k+1}]$ is the length of some $s - u_{k+1}$ path.

Based on the algorithm, we have $dist_{k+2}[v] = min(dist_{k+1}[v], dist_{k+1}[u_{k+1}] + weight(u_{k+1}, v))$. There are two cases:

- `Case 1:` $dist_{k+1}[v] < dist_{k+1}[u_{k+1}] + weight(u_{k+1}, v)$.
  In this case, $dist_{k+2}[v] = dist_{k+1}[v]$. Then if $dist_{k+2}[v] \neq infinity$, we have $dist_{k+1}[v] \neq infinity$, and that $dist_{k+2}[v]$ and $dist_{k+1}[v]$ are the length of the same $s - v$ path if there exists one. Since $dist_{k+1}[v] \neq infinity$, the inductive hypothesis implies that $dist_{k+1}[v]$ is the length of some $s - v$ path, hence $dist_{k+2}[v]$ is the length of some $s - v$ path. P(k+1) holds.

- `Case 2:` $dist_{k+1}[v] \geq dist_{k+1}[u_{k+1}] + weight(u_{k+1}, v)$
  Under this case, $dist_{k+2}[v] = dist_{k+1}[u_{k+1}] + weight(w, v)$. If $dist_{k+2}[v] \neq infinity$, then it follows that $dist_{k+1}[u_{k+1}] = dist_{k+2}[v] - weight(u_{k+1}, v) \neq infinity$. Then the inductive hypothesis implies that $dist_{k+1}[u_{k+1}]$ must be the length of some $s - u_{k+1}$ path, denote as $p(s, u_{k+1})$. Since there is an edge $(u_{k+1}, v) \in g$, then $dist_{k+2}[v] = dist_{k+1}[u_{k+1}] + weight(u_{k+1}, v)$ must be the length of the $s - v$ path through u_{k+1}. P(k+1) holds.

Hence P(k+1) holds for u_{k+1} and all nodes $v \in g$ other than u_{k+1}. By the principle of prove by induction, P(n) holds. `Lemma 3.2` proved. □

Lemma 3.3. For any node $v \in g$, if after the i^{th} iteration, $dist_{i+1}[v] = \delta(v)$, then for each proceeding j^{th} iteration, $j > i$, $dist_{j+1}[v] = dist_{i+1}[v] = \delta(v)$.

Proof. We will prove `Lemma 3.3` by induction on the number iterations after the i^{th} iteration.
Let P(n) be: For any node $v \in g$, if after the i^{th} iteration, $dist_{i+1}[v] = \delta(v)$, then for the $(i+n)^{th}$ iteration, $n \geq 1$, $dist_{i+n+1}[v] = dist_{i+1}[v] = \delta(v)$
Base Case: We shall show P(1) holds.
During the $(i+1)^{th}$ iteration, suppose u_{i+1} is the node being explored, then $dist_{i+2}[v]$ is calculated as:

$$dist_{i+2}[v] = min(dist_{i+1}[v], dist_{i+1}[u_{i+1}] + weight(u_{i+1}, v))$$

If $(u_{i+1}, v) \in g$, $dist_{i+1}[u_{i+1}]$ is the length of some $s - u_{i+1}$ path, then $(dist_{i+1}[u_{i+1}] + weight(u_{i+1}, v))$ is the length of some $s - v$ path. Since $dist_{i+1}[v] = \delta(v)$, based on the definition of shortest path, $dist_{i+1}[v] \leq dist_{i+1}[u_{i+1}] + weight(u_{i+1}, v)$, hence $dist_{i+2}[v] = dist_{i+1}[v] = \delta(v)$.

If u_{i+1} does not have an edge to v, $weight(u_{i+1}, v) = \infty$, we have:

$$\begin{aligned} dist_{i+2}[v] &= min(dist_{i+1}[v], dist_{i+1}[u_{i+1}] + weight(u_{i+1}, v)) \\ &= min(dist_{i+1}[v], dist_{i+1}[u_{i+1}] + \infty) \\ &= dist_{i+1}[v] = \delta(v). \end{aligned}$$

Hence $dist_{i+2}[v] = dist_{i+1}[v] = \delta(v)$. P(1) holds.

Inductive Hypothesis: Suppose P(k) holds, that is, if after the i^{th} iteration, $dist_{i+1}[v] = \delta(v)$, then for the $(i+k)^{th}$ iteration, $n \geq 1$, $dist_{i+k+1}[v] = dist_{i+1}[v] = \delta(v)$.

Inductive Step: We shall show P(k+1) holds.
Based on the algorithm, for $dist_{i+k+2}[v]$, we have:

$$dist_{i+k+2}[v] = min(dist_{i+k+1}[v], dist_{i+k+1}[u_{i+k+1}] + weight(u_{i+k+1}, v))$$

If u_{i+k+1} does not have an edge to v, then $weight(u_{i+k+1}, v) = \infty$, we have:

$$\begin{aligned} dist_{i+k+2}[v] &= min(dist_{i+k+1}[v], dist_{i+k+1}[u_{i+k+1}] + weight(u_{i+k+1}, v)) \\ &= min(dist_{i+k+1}[v], dist_{i+k+1}[u_{i+k+1}] + \infty) \end{aligned}$$

Based on our inductive hypothesis, $dist_{i+k+1}[v] = dist_{i+1}[v] = \delta(v)$, then if $weight(u_{i+k+1}, v) = \infty$, we have $dist_{i+k+2}[v] = dist_{i+k+1}[v] = \delta(v)$.
If $weight(u_{i+k+1}, v) \neq \infty$, $dist_{i+k+1}[u_{i+k+1}]$ is the length of some $s - u_{i+k+1}$ path, then $(dist_{i+k+1}[u_{i+1}] + weight(u_{i+k+1}, v))$ is the length of some $s - v$ path. Based on the definition of shortest path distance, $dist_{i+k+1}[v] = \delta(v) \leq (dist_{i+k+1}[u_{i+1}] + weight(u_{i+k+1}, v))$. Hence:

$$\begin{aligned} dist_{i+k+2}[v] &= min(dist_{i+k+1}[v], dist_{i+k+1}[u_{i+k+1}] + weight(u_{i+k+1}, v)) \\ &= min(\delta(v), dist_{i+k+1}[u_{i+k+1}] + weight(u_{i+k+1}, v)) \\ &= \delta(v) = dist_{i+1}[v] \end{aligned}$$

Thus P(k+1) holds. By the principle of prove by induction, P(n) holds. `Lemma 3.3` proved. □

Lemma 3.4. Forall nodes $v \in g, v' \in explored_{n+1}$, $dist_{n+1}[v] \leq dist_n[v'] + weight(v', v)$.

Proof. We will prove Lemma 3.4 by inducting on the number n.
Let P(n) be: for $n \geq 1$, forall nodes $v \in g, v' \in explored_{n+1}$, $dist_{n+1}[v] \leq dist_n[v'] + weight(v', v)$
Base Case: We shall show P(1) holds.
Based on the algorithm, $dist_1[s] = 0$, and for all node $v \in g$ other than s, $dist_1[v] = \infty$. Since $explored_2$ only contains s, then $dist_2[s] = 0 \leq dist_1[s] + weight(s, s) = 0$. For all node $v \in g$ other than s, $dist_2[v] = min(dist_1[v], dist_1[s] + weight(s, v)) = min(\infty, 0 + weight(s, v)) = min(\infty, weight(s, v))$. If s has an edge to v, then $weight(s, v) < \infty$, $dist_2[v] = weight(s, v) \leq dist_1[s] + weight(s, v)$. If s does not have an edge to v, then $weight(s, v) = \infty$, $dist_2[v] = \infty \leq dist_1[s] + weight(s, v)$. Hence P(1) holds.

Induction Hypothesis: Suppose P(k) holds for $k > 1$. That is, for $k > 1$, forall nodes $v \in g, v' \in explored_{k+1}$, $dist_{k+1}[v] \leq dist_k[v'] + weight(v', v)$

Inductive Step: we shall show P(k+1) holds.

Suppose w is the node being explored during the $(k+1)^{th}$ iteration, then forall node $v \in g$, we have:

$$dist_{k+2}[v] = min(dist_{k+1}[v], dist_{k+1}[w] + weight(w, v))$$

$\square$

Lemma 3.5. Assume g is a connected graph. Forall node $v \in explored_{n+1}$:

1. $dist_{n+1}[v] < \infty$
2. $\delta(v) \leq \delta(v'), \forall v' \in unexplored_{n+1}$.
3. $dist_{n+1}[v] = \delta(v)$

Proof. We will prove `Lemma 3.4` by inducting on the number of iterations.

Let P(n) be: For a connected graph g, for $n \geq 1$, forall node $w \in explored_{n+1}$: (L1) $dist_{n+1}[w] < \infty$; (L2) $\delta(w) \leq \delta(w'), \forall w' \in unexplored_{n+1}$; (L3) $dist_{n+1}[w] = \delta(w)$.

Base Case: We shall show P(1) holds
Based on the algorithm, during the first iteration, the node with minimum distance value is the source node s with $dist_1[s] = 0$. Hence during the first iteration, only s is removed from $unexplored_1$ and added to $explored_2$. Since $dist_2[s] = 0 < \infty$, then (L1) holds for P(1). Since all edge weights are non-negative, then the shortest distance value from s to s is indeed 0, hence $dist_2[s] = 0 = \delta(s)$ and $\delta(s) \leq \delta(v')$, $\forall v' \in unexplored_2$. Thus (L2) and (L3) holds for P(1). Hence P(1) holds.

Induction Hypothesis: Suppose P(i) is true for all $1 \leq i \leq k$. That is, forall $1 < i \leq k$, forall node $w \in explored_{i+1}$: (L1) $dist_{i+1}[w] < \infty$; (L2) $\delta(w) \leq \delta(w'), \forall w' \in unexplored_{i+1}$; (L3) $dist_{i+1}[w] = \delta(w)$;

Inductive Step: We shall show P(k+1) holds. That is, forall node $w \in explored_{k+2}$, (L1) $dist_{k+2}[w] \neq \infty$; (L2) $\delta(w) \leq \delta(w'), \forall w' \in unexplored_{k+2}$; (L3) $dist_{k+2}[w] = \delta(w)$;

Suppose u_{k+1} is the node added into *explored* during the $(k+1)^{th}$ iteration, then $explored_{k+2} = explored_{k+1} \cup \{u_{k+1}\}$. We will show that (L1)(L2) and (L3) holds for all nodes in $explored_{k+1}$ in `Part (a)`, and `Part (b)` proves (L1)(L2)(L3) holds for u_{k+1}, so that the statements holds forall nodes in $explored_{k+2}$.

- Part(a): `WTP: After the` $(k+1)^{th}$ `iteration,` $\forall w \in explored_{k+1}$`, (L1)(L2)(L3) holds.`

  Consider each node $q \in (explored_{k+1} \cap explored_{k+2}) = explored_{k+1}$, q must be explored before the $(k+1)^{th}$ iteration. Suppose q is explored during the i^{th} iteration for some $i < k+1$, then based on our induction hypothesis, $dist_{i+1}[q] = \delta(q)$, and $\delta(q) \leq \delta(q'), \forall q' \in unexplored_{i+1}$.

  `Proof of (L2)`: Based on the algorithm, for each iteration, the algorithm explores exactly one node and never revisits any explored nodes. For each node $q \in explored_{k+1}$ mentioned above, since q is explored before the $(k+1)^{th}$ iteration, then $unexplored_{k+1} \subseteq unexplored_{i+1}$. Since $\delta(q) \leq \delta(q'), \forall q' \in unexplored_{i+1}$, and $unexplored_{i+1}$ includes all node in $unexplored_{k+1}$, then $\delta(q) \leq \delta(q'), \forall q' \in unexplored_{k+1}$. (L2) holds for $explored_{k+1}$.

`Proof of (L3)`: Since for each node $q \in explored_{k+1}$, the induction hypothesis implies that $dist_{k+1}[q] = \delta(q)$, then `Lemma 3.3` imples that $dist_{k+2}[q] = dist_{k+1}[q] = \delta(q)$. (L3) holds for $explored_{k+1}$.

`Proof of (L1)`: Since the induction hypothesis implies that $\forall q \in explored_{k+1}, dist_{k+1}[q] < \infty$, and the proof of (L3) above shows that $dist_{k+2}[q] = dist_{k+1}[q]$, then $dist_{k+2}[q] < \infty$. (L1) holds for $explored_{k+1}$.

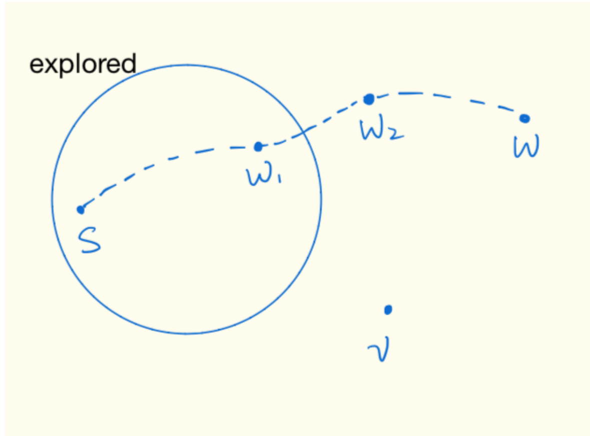
Hence we have proved that both (1) and (2) holds for all nodes in $explored_{k+1}$.

- Part(b): (L1)(L2)(L3) holds for $\{u_{k+1}\}$.
  Specifically, we want to show: (L1) $dist_{k+2}[u_{k+1}] < \infty$; (L2) $\delta(u_{k+1}) \leq \delta(v'), \forall v' \in unexplored_{k+2}$, and (L2) $dist_{k+2}[u_{k+1}] = \delta(u_{k+1})$.

  1. (L1) $dist_{k+2}[u_{k+1}] \neq \infty$
     Since g is a connected graph, then s must have a path to u_{k+1}. Since u_{k+1} is the node currently being explored, then we know there must exists a $s - u_{k+1}$ path, denote as $p(s, u_{k+1})$, such any node proceeding u_{k+1} in $p(s, u_{k+1})$ are explored before u_{k+1}, i.e., in $explored_{k+1}$. Denote the node right before u_{k+1} in $p(s, u_{k+1})$ as u', $u' \in explored_{k+1}$. The induction hypothesis implies that $dist_{k+1}[u'] < \infty$. Since u' is right before u_{k+1} in $p(s, u_{k+1})$, then based on the algorithm, $dist_{k+2}[u_{k+1}] \leq dist_{k+1}[u'] + weight(u', u_{k+1})$. Since u' has an edge to u_{k+1}, then $weight(u', u_{k+1}) < \infty$, and $dist_{k+1}[u'] < \infty$, then $dist_{k+2}[u_{k+1}] \leq dist_{k+1}[u'] + weight(u', u_{k+1}) < \infty + weight(u', u_{k+1}) = \infty$, i.e, $dist_{k+1}[u_{k+1}] < \infty$. (L1) holds for u_{k+1}.

  2. (L2) $\delta(u_{k+1}) \leq \delta(v'), \forall v' \in unexplored_{k+2}$

     We will prove (L2) by contradiction. Suppose there exists $w \in unexplored_{k+2}$, such that $\delta(u_{k+1}) > \delta(w)$.
     Based on proof of (L1) above, we know $dist_{k+2}[u_{k+1}] < \infty$. Thus `Lemma 3.2` implies that $dist_{k+2}[u_{k+1}]$ is the length of some $s - v$ path. Based on the definition of shortest path, $\delta(u_{k+1}) \leq dist_{k+2}[u_{k+1}]$. Since $\delta(u_{k+1}) > \delta(w)$ and $\delta(u_{k+1}) \leq dist_{k+2}[u_{k+1}]$, then we have $\delta(w) < dist_{k+2}[u_{k+1}]$([NE1]).
     Consider the shortest path $\Delta(s, w)$ from s to w, $\delta(w) = length(\Delta(s, w))$. Since $w \notin explored_{k+2}$, then there must exists some node in $\Delta(s, w)$ that are not in $explored_{k+2}$. Suppose the first node along $\Delta(s, w)$ that is not in the $explored_{k+2}$ list is w_2, and the node right before w_2 in the s to w_2 subpath is w_1, thus $w_1 \in explored_{k+2}$. The image below illustrates this construction:



Denote the subpath from s to w_1 in $\Delta(s, w)$ as $p(s, w_1)$, subpath from s to w_2 in $\Delta(s, w)$ as $p(s, w_2)$, and subpath w_2 to w as $p(w_2, w)$. Based on `Definition 2.2 Prefix of Path`, $p(s, w_1)$ is a prefix of $\Delta(s, w)$. Since $p(s, w_1)$ is the prefix of the shortest $s - w$ path, then based on `Lemma 3.1`, $p(s, w_1)$ is the shortest path from s to w_1, $\Delta(s, w_1) = p(s, w_1)$, $length(p(s, w_1)) = \delta(w_1)$. Similarly, since $p(s, w_2) = p(s, w_1) + (w_1, w_2)$, then $p(s, w_2)$ is a prefix of $\Delta(s, w)$, and hence `Lemma 3.1` implies that $p(s, w_2)$ is the shortest path from s to w_2. Then we have:

$$\begin{aligned}
\Delta(s, w_2) &= p(s, w_2) = p(s, w_1) + (w_1, w_2) \\
\delta(w_2) &= length(\Delta(s, w_2)) \\
&= length(p(s, w_2)) \\
&= length(p(s, w_1)) + weight(w_1, w_2) \\
&= \delta(w_1) + weight(w_1, w_2)([E1])
\end{aligned}$$

For $\Delta(s, w)$ we have:

$$\begin{aligned}
\delta(w) &= length(p_w) \\
&= length(p(s, w_1)) + weight(w_1, w_2) + length(p(w_2, w)) \\
&= \delta(w_1) + weight(w_1, w_2) + length(p(w_2, w))
\end{aligned}$$

Since all edge weights are non-negative, then:

$$\delta(w_2) = \delta(w_1) + weight(w_1, w_2) \le \delta(w) \text{ ([E2])}$$

Since $w_1 \in explored_{k+2}$, there are two cases to consider: $w_1 = u_{k+1}$ and $w_1 \ne u_{k+1}$. We will prove P(k+1) under both cases below.

`Case 1:` $w_1 = u_{k+1}$
When $w_1 = u_{k+1}$, then substitude w_1 by u_{k+1} in [E2], we have:

$$\delta(w_1) + weight(w_1, w_2) = \delta(u_{k+1}) + weight(u_{k+1}, w_2) \le \delta(w)$$
i.e. $\delta(u_{k+1}) \le \delta(w)$

which contradicts with our assumption that $\delta(u_{k+1}) > \delta(w)$. Hence by the principle of prove by contradiction, $\delta(u_{k+1}) < \delta(w)$. (L2) holds for u_{k+1}.

`Case 2`: $w_1 \neq u_{k+1}$

Since $w_1 \in explored_{k+2}$ and $w_1 \neq u_{k+1}$, w_1 is explored before the $(k+1)^{th}$ iteration. i.e., $w_1 \in explored_{k+1}$. Suppose w_1 is being explored during the i^{th} iteration, $i < k+1$, then based on the algorithm, the value of $dist_{i+1}[w_1]$ is calculated as:

$$\begin{aligned} dist_{i+1}[w_1] &= min(dist_i[w_1], dist_i[w_1] + weight(w_1, w_1)) \\ &= min(dist_i[w_1], dist_i[w_1] + 0) \\ &= min(dist_i[w_1], dist_i[w_1]) \\ &= dist_i[w_1] \end{aligned}$$

Since the induction hypothesis implies that $dist_{i+1}[w_1] = \delta(w_1)$, then $dist_i[w_1] = \delta(w_1)$.
Since w_1 has an edge to w_2, then $dist_{i+1}[w_2]$ must have been updated according as follows:

$$\begin{aligned} dist_{i+1}[w_2] &= min(dist_i[w_2], dist_i[w_1] + weight(w_1, w_2)) \\ &= min(dist_i[w_2], \delta(w_1) + weight(w_1, w_2)) \end{aligned}$$

Based on [E1] we know that $\delta(w_2) = \delta(w_1) + weight(w_1, w_2)$, then $dist_{i+1}[w_2] = min(dist_i[w_2], \delta(w_2))$. If $dist_i[w_2] = \infty$, then $dist_{i+1}[w_2] = min(dist_i[w_2], \delta(w_2)) = \delta(w_2)$. If $dist_i[w_2] \neq \infty$, then based on `Lemma 3.2`, $dist_i[w_2]$ is the length of some $s - w_2$ path. Since $\delta(w_2) \leq length(p), \forall p \in path(s, w_2)$, then $dist_{i+1}[w_2] = min(dist_i[w_2], \delta(w_2)) = \delta(w_2)$. Hence in either cases, we conclude that $dist_{i+1}[w_2] = \delta(w_2)$.
Since $dist_{i+1}[w_2] = \delta(w_2)$ and $i < k+1$, then based on `Lemma 3.3`, we have $dist_{k+1}[w_2] = dist_{i+1} = \delta(w_2)$. Based on [E2], $\delta(w_2) < \delta(w)$, then $dist_{k+1}[w_2] < \delta(w)$ [NE2]. Combining with [NE1], we have:

$$\begin{aligned} &\delta(w) < dist_{k+1}[u_{k+1}] \text{ (from [NE1])} \\ &dist_{k+1}[w_2] < \delta(w) \end{aligned}$$

Hence $dist_{k+1}[w_2] < dist_{k+1}[u_{k+1}]$ [NE2].
Based on our assumption, at the beginning of the $(k+1)^{th}$ generation, $u_{k+1}, w_2 \notin explored_{k+1}$ and u_{k+1} is selected by the algorithm, then we must have $dist_{k+1}[w_2] \geq dist_{k+1}[u_{k+1}]$, which contradicts with [NE2]. Hence by the principle of prove by contradiction, there does not exsist $w \in unexplored_{k+2}$, such that $\delta(u_{k+1}) > \delta(w)$, i.e. $\delta(u_{k+1}) \leq \delta(w), \forall w \in unexplored_{k+2}$. Hence (L2) holds for u_{k+1}.

3. (L3) $dist_{k+2}[u_{k+1}] = \delta(u_{k+1})$

   We will prove this by contradiction.
   Since (L1) proves $dist_{k+2}[u_{k+1}] \neq \infty$, then `Lemma 3.2` implies that $dist_{k+2}[u_{k+1}]$ is the length of some $s - u_{k+1}$ path, denote as p. Suppose there is a $s - u_{k+1}$ path p' that's shorter than p, i.e, $dist_{k+2}[u_{k+1}] > \delta(v') + weight(v', v)$[NE3]. Suppose the node right before u_{k+1} in p' is v', then there are two cases: (1) $v' \in explored_{k+2}$.

`Case(1):` $v' \in explored_{k+2}$

Suppose v' is explored during the j^{th} iteration, $j < k+1$. Then the induction hypothesis indicates that $dist_{j+1}[v'] = \delta(v')$. Since $dist_{j+1}[v'] = min(dist_j[v'], dist_j[v'] + weight(v', v')) = dist_j[v']$, then $dist_j[v'] = \delta(v')$. Then `Lemma 3.3` implies that $dist_{k+2}[v'] = \delta(v')$. Since $\delta(v') + weight(v', v) < dist_{k+2}[u_{k+1}]$, then $dist_{k+2}[v'] + weight(v', v) < dist_{k+2}[u_{k+1}]$, which contradicts with `Lemma 3.4`. By the principle of prove by contradiction, $dist_{k+2}[u_{k+1}] = \delta(u_{k+1})$

Since we have proved both (1) and (2) forall nodes in $explored_{k+1}$ after the $(k+1)^{th}$ iteration, P(k+1) holds. Then by the principle of prove by induction, `Lemma 3.4` holds. □

Proof. **Prove of Correctness**

By applying `Lemma 3.4` to the last iteration of the algorithm, we obtained that for all nodes n in the explored list, $dist[n]$ is indeed the shortest path distance value from source s to n, hence Dijkstra's algorithm indeed calculates the shortest path distance value from the source s to each node $n \in g$. □