

Dijkstra's Algorithm Verification

Yazhe Feng

February 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) . 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:

```
while (unexplored is not Nil) {
  (At the  $k^{th}$  iteration of the while loop)
  choose  $u \in unexplored$  s.t.  $\forall u' \in unexplored, dist[u] \leq dist[u']$ 
  let unexplored' be the list after removing  $u$  from unexplored
  for( $\forall v \in g$  s.t.  $(u, v) \in g$ ) {
    (At the  $p^{th}$  iteration of this for loop)
    if( $dist[u] + weight(u, v) < dist[v]$ ) {
      let  $dist' = dist$  with  $dist'[v] = dist[u] + weight(u, v)$ 
    }
    input the new  $dist'$  to the  $(p+1)^{th}$  iteration of the for loop
  }
  input the new unexplored' and  $dist'$  to the  $(k+1)^{th}$  iteration of the while loop
}
```

1.2 Assumptions

1. Weight of edges are non-negative
2. 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:

$$ve_0v_0e_1v_2\dots v_{n-1}e_nw$$

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. Length of Path

The length of a path $p = ve_0v_0e_1v_2\dots v_{n-1}e_nw$ 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 2.3. Shortest Path

Denote $\Delta(s, v)$ as the shortest path from s to v . $\Delta(s, v)$ must fulfill:

$$\Delta(s, v) \in path(s, v) \quad \text{and} \quad \forall p' \in path(s, v), length(\Delta(s, v)) \leq length(p')$$

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

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 . 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 also denote $\delta(v)$ as the distance value of the shortest path from s to node $v \in g$.

Lemma (1). During each iteration of the algorithm, for all node $v \in explored$, we have:

1. $\delta(v) < \delta(v'), \forall v' \in unexplored$.
2. $dist[v] = \delta(v)$

Proof. We will prove this by inducting on the number of iterations.

Let $P(n)$ be: During the n^{th} iteration of the algorithm, $n > 1$, for all nodes $v \in explored$ of length, (1) $\delta(v) < \delta(v'), \forall v' \in unexplored, v \neq v'$, and (2) $\delta(v) = dist[v]$.

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 $dist[s] = 0$. Hence during the first iteration, only s is removed from *unexplored* and added to *explored*. Since all edge weights are non-negative, then $dist[s] = \delta(s)$ and $\delta(s) < \delta(v') \forall v' \neq s$. Hence $P(1)$ holds.

Inductive Hypothesis: Suppose $P(k)$ holds for all $k > 1$. That is, during the k^{th} iteration of the algorithm for all $k > 1$, for all $v \in explored$, (1) $\delta(v) < \delta(v'), \forall v' \in unexplored, v \neq v'$, and (2) $\delta(v) = dist[v]$.

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

Suppose v is the node added into *explored* during the $(k+1)^{th}$ iteration. We need to show (1) $\delta(v) < \delta(v'), \forall v' \in unexplored, v' \neq v$, and (2) $dist[v] = \delta(v)$.

1. $\delta(v) \leq \delta(v'), \forall v' \in unexplored, v' \neq v$

We will prove (1) by contradiction. Suppose there exists $w \in unexplored$, such that $\delta(w) < \delta(v)$. Since $w \in unexplored$ and $v \in explored$, then based on the algorithm, $dist[v] < dist[w]$ holds for the $(k+1)^{th}$ iteration. Since $\delta(v) \leq dist[v]$ and $dist[v] < dist[w]$, then $\delta(v) < dist[w]$ holds for the $(k+1)^{th}$ iteration ([a]).

Assume w' is the node just before w in $\Delta(s, w)$ (Definition 2.3). Then we have:

$$\delta(w) = dist[w'] + weight(w', w)$$

Since $\delta(w) < \delta(v)$, then:

$$\begin{aligned} \delta(w) &< \delta(v) \\ dist[w'] + weight(w', w) &< \delta(v) \\ dist[w'] &< \delta(v) \end{aligned}$$

Since $dist[w'] < \delta(v)$ and $\delta(v) \leq dist[v]$, then $dist[w'] < dist[v]$. Thus based on the algorithm, the node w' must have been explored before v , i.e. $w' \in explored$. Since w' has an edge (w', w) to w , then the algorithm must have compared $(dist[w'] + weight(w', w))$ with the current $dist[w]$ before the $(k+1)^{th}$ iteration and chose $dist[w]$. Thus it must be $(dist[w'] + weight(w', w)) \geq dist[w]$, i.e. $\delta(w) \geq dist[w]$. Since $\delta(v) > \delta(w)$ and $\delta(w) \geq dist[w]$, then $\delta(v) > dist[w]$, which contradicts with [a]. Hence by the principle of prove by contradiction, (1) holds for the $(k+1)^{th}$ iteration.

2. $dist[v] = \delta(v)$

Suppose $dist[v]$ is associated with path $p \in path(s, v)$ during the $(k + 1)^{th}$ iteration, and assume the shortest path from s to v is some path $p' \in path(s, v)$ different than p , $length(p') = \delta(v) < dist[v]$. Suppose v' is the node just before v in p' .

$$\delta(v) = dist[v'] + weight(v', v)$$

Since all edge weights are non-negative, then $dist[v'] < \delta(v)$. Based on (1), since $\delta(v) < \delta(w) \forall w \in unexplored$, then v' must be in *explored*. Since v' is in *explored* and has an edge to v , then the algorithm must have compared $dist[v'] + weight(v', v)$ to the current $dist[v]$ and chose $dist[v]$. Hence it must be $dist[v'] + weight(v', v) \geq dist[v]$, i.e. $\delta(v) \geq dist[v]$, which contradicts with [b]. Hence by the principle of prove by contradiction, p is the shortest path from s to v , and that $dist[v] = \delta(v)$.

□

Proof. Prove of Correctness

By applying Lemma (1) to each 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 $n \in g$.

□