Correctness of Dijkstra's algorithm

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Dijkstra's Algorithm

Function DIJKSTRA($G =< V, E, c, s >$)

{The input to the algorithm is a directed graph $G =< V, E >$, weighted by the cost function $c : E \rightarrow Z^{+}$; we assume that there are no zero-cost edges.}
for ($i = 1$ **to** n) **do**
  $d[i] = \infty$
end for
$d[s] = 0$
Organize the vertices into a heap Q, based on their d values.
$S \leftarrow \phi$.
while ($Q \neq \phi$) **do**
  $u \leftarrow$ EXTRACT-MIN(Q)
  for (each edge of the form $e = (u, v)$) **do**
    RELAX(e)
  end for
  $S \leftarrow S \cup \{u\}$
end while

Algorithm 1.1: Dijkstra's Alorithm for the Single Source Shortest Path problem with postive weights

Function RELAX($e = (u, v)$)

if ($d[v] > d[u] + c(u, v)$) **then**
  $d[v] = d[u] + c(u, v)$)
end if

Algorithm 1.2: Dijkstra's Alorithm for the Single Source Shortest Path problem with postive weights

2 Proof of Correctness

Let $\delta(v)$ denote the true shortest path distance of vertex v from the source s. Observe that Dijkstra's algorithm works by estimating an intial shortest path distance of ∞ from the source and gradually lowering this estimate.

Lemma 2.1 *If $d[v] = \delta(v)$ for any vertex v, at any stage of Dijkstra's algorithm, then $d[v] = \delta(v)$ for the rest of the algorithm.*

Proof: Clearly, $d[v]$ cannot become smaller than $\delta(v)$; likewise, the test condition in the RELAX() procedure will always fail. $\square$

Theorem 2.1 *Let $< v_1 = s, v_2, \ldots, v_n >$ denote the sequence of vertices extracted from the heap Q, by Dijkstra's algorithm. When vertex v_i is extracted from Q, $d[v_i] = \delta(v_i)$.*

Proof: Without loss of generality, we assume that every vertex is reachable from the source vertex s, either through a finite length path or an arc of length ∞.

Clearly, the claim is true for $v_1 = s$, since $d[s] = \delta(s) = 0$ and all edge weights are positive.

Assume that the claim is true for the first $k-1$ vertices, i.e., assume that for each $i = 2, 3, \ldots, k-1$, when vertex v_i is deleted from Q, $d[v_i] = \delta(v_i)$.

We focus on the situation, when vertex v_k as it is deleted from Q. As per the mechanics of Dijkstra's algorithm, $d[v_k] \leq d[v_j]$, $j = k+1, \ldots, n$. Observe that if the shortest path from $v_1 = s$ to v_k consisted entirely of vertices from the set $R = \{v_1, \ldots, v_{k-1}\}$, then $d[v_k] = \delta(v_k)$. (Why?) Assume that $\delta(v_k) < d[v_k]$. It follows that the shortest path from s to v_k involves vertices in the set $V - R$. Consider the first vertex $v_q \in V - R$, on the shortest path from s to v_k. Let v_p denote the vertex before v_q on this path; note that $v_p \in R$. Now, when v_p is deleted from Q, all its edges were relaxed, including the edge to v_q and therefore $d[v_q] = \delta(v_q)$. (See Lemma 2.1.) Since there are no zero-cost edges, $\delta(v_q) < \delta(v_k)$ and hence $d[v_q] < d[v_k]$. But this means that v_k could not have been chosen before v_q by Dijkstra's algorithm, contradicting the choice of v_k as a vertex for which $\delta(v_k) > d[v_k]$, when it is deleted from Q.

$\square$