Proof of Correctness: Dijkstra's Algorithm for Company Cost Minimization

1 Algorithm Overview

Dijkstra's algorithm maintains:

- dist$[v]$: the current best-known distance from s to node v
- S: the set of nodes whose shortest path distances have been finalized
- Q: a min-priority queue of nodes not yet in S

The algorithm proceeds as follows:

Algorithm 1 Dijkstra's Algorithm

Initialize $\text{dist}[s] = 0$ and $\text{dist}[v] = \infty$ for all $v \neq s$
Initialize $S = \emptyset$ and $Q = V$
while $Q \neq \emptyset$ **do**
    $u \leftarrow$ node in Q with minimum $\text{dist}[u]$
    Remove u from Q and add u to S
    for each neighbor v of u where $v \in Q$ **do**
        $\text{newDist} \leftarrow \text{dist}[u] + \text{company\_cost}(u, v)$
        if $\text{newDist} < \text{dist}[v]$ **then**
            $\text{dist}[v] \leftarrow \text{newDist}$
            Update v's position in priority queue Q
        end if
    end for
end while

2 Key Assumption

Claim 1. *All edge weights are non-negative: company_cost$(u, v) \geq 0$ for all $(u, v) \in E$.*

This assumption is critical for the correctness of Dijkstra's algorithm, as demonstrated in Part B where negative edge weights cause the algorithm to produce incorrect results.

3 Proof of Correctness

The correctness could be proved by induction, using the greedy stays ahead technique.

Theorem 1. *When Dijkstra's algorithm terminates, for all nodes $v \in V$, dist$[v]$ equals the true shortest path distance from s to v.*

Proof. We prove by induction on the size of set S (the set of finalized nodes).

Claim: At each iteration, for every node $u \in S$, dist$[u]$ equals the true shortest path distance from s to u.

Base Case: Initially, the start vertex is the only visited vertex. In ther words, $S = \{s\}$ and dist$[s] = 0$. The shortest path from s to itself has length 0, so the claim holds.

Inductive Step: Assume the claim holds when $|S| = k$ for some $k \geq 1$. We show it holds when we add the $(k+1)$th node to S.

Let u be the next node selected by Dijkstra (the node in Q with minimum dist$[u]$). Let $d^*(u)$ be the true shortest path distance from s to u. In other words, we need to show dist$[u] = d^*(u)$.

Step 1: Show dist$[u] \geq d^*(u)$

By the algorithm's construction, dist$[u]$ represents the length of some path from s to u (found via relaxation steps). Since $d^*(u)$ is the minimum over all possible paths:

$$\text{dist}[u] \geq d^*(u)$$

Step 2: Show dist$[u] \leq d^*(u)$

Consider any shortest path P from s to u. We can decompose P as:

$$s \rightsquigarrow x \to y \rightsquigarrow u$$

where:

- $s \rightsquigarrow x$ is the portion of the path entirely within S
- $x \in S$ is the last node in S on this path
- $y \notin S$ is the first node outside S on this path
- $y \rightsquigarrow u$ is the remaining portion of the path

By the inductive hypothesis, dist$[x] = d^*(x)$ (since $x \in S$).

When x was added to S, the algorithm relaxed edge (x, y), so:

$$\text{dist}[y] \leq \text{dist}[x] + \text{company\_cost}(x, y) = d^*(x) + \text{company\_cost}(x, y)$$

Since P is a shortest path, the portion from s to y along P has length:

$$d^*(y) = d^*(x) + \text{company\_cost}(x, y)$$

Therefore:

$$\text{dist}[y] \leq d^*(y)$$

Now, since y appears on a shortest path to u, and all edge weights are non-negative:

$$d^*(u) = d^*(y) + \text{cost of path } (y \rightsquigarrow u) \geq d^*(y)$$

The inequality holds because the path from y to u has non-negative length.

Combining the inequalities:

$$\text{dist}[u] \leq \text{dist}[y] \leq d^*(y) \leq d^*(u)$$

The first inequality holds because u was selected over y from the priority queue, meaning $\text{dist}[u] \leq \text{dist}[y]$.

Conclusion: From Step 1 and 2, we have:

$$d^*(u) \leq \text{dist}[u] \leq d^*(u)$$

Therefore, $\text{dist}[u] = d^*(u)$, and the claim is maintained.

By induction, when the algorithm terminates with $S = V$, all nodes have their correct shortest path distances. $\square$

4 Conclusion

We have proven that Dijkstra's algorithm correctly computes shortest paths from a source node to all other nodes in a graph with non-negative edge weights. The greedy choice of always selecting the unfinalized node with minimum distance stays ahead of any alternative approach, guaranteeing optimality.