Nikolas Mavrogeneiadis
August 9, 2025

Dijkstra Shortest Path Algorithm Proof

Suppose we want to find the shortest path from a starting node s to all other nodes in a directed graph with no negative weights. Set S contains all the nodes that we have already been processed (meaning we have calculated their minimum distance from s), and the set Q contains the rest. Each time we process a node, we check all its neighbors to see whether we can update their distance (relaxing edge) or not.

Algorithm 1 Dijkstra

Require: Graph $G(V, E, w), w : e \in E(G) \to R^+$
Ensure: $d[u]$: minimum distance from vertex s to all other vertices
1: $d[s] \leftarrow 0$
2: $d[v] \leftarrow \infty, \forall v \in V(G)$
3: $U \leftarrow \varnothing$
4: **while** $|U| < |V|$ **do**
5: $\quad u \leftarrow argmin\{d[v], v \in V(G) \backslash U\}$
6: $\quad U \leftarrow U + \{u\}$
7: $\quad$ **for all** $v \in N(u)$ **do**
8: $\quad\quad d[v] \leftarrow min(d[v], d[u] + w(u, v))$
9: $\quad$ **end for**
10: **end while**
11: return d

Theorem 1. *At the end of the algorithm 1, $d[v]$ contains the minimum distance from s to every other $v \in V(G)$.*

We will prove it by induction. Let $d(s, v)$ be the real distance of s to any other vertex v. We have to show that $d[v] = d(s, v)$ for each vertex $v \in V(G)$. It's obvious that $d(s, s) = d[s] = 0$. Consider now any iteration where $d(s, v) = d[v]$ holds for all the previous vertices added to U and algorithm chooses v^* to add to U. We will show that $d(s, v^*) = d[v^*]$. Now let any path P from s to v^* with length equal to $d(s, v^*)$. Assume, for sake of contradiction, that $d[v^*] > d(s, v^*)$ (1). Because $s \in U$ and $v^* \notin U$ there exists at least one edge (x, y) on P that $x \in U$ and $y \notin U$ (could be $x = s$ or $y = v^*$). We pick the first such edge. By induction it follows that $d[x] = d(s, x)$. Providing that weights are non negative and P contains a $s - x$ path (plus the edge (x, y)), it holds that $d[x] + w(x, y) \leqslant d(s, v^*)$ (2). Moreover, we calculated $d[y]$, from line 8, as $d[y] = min(d[y], d[x] + w(x, y))$ after putting x to U. This means that $d[y] \leqslant d[x] + w(x, y)$ (3). Hence, $d[y] \leqslant d(s, v^*)$ from (2) and (3). But this would mean that $d[y] < d[v^*]$. This is a contradiction, because we assumed that the algorithm picked v^* (line 5) at the beginning of the iteration and not y. See the next page for a possible part of an input graph. Complexity is $O(|V|^2)$ because we do $|V|$ iterations (line 4) and argmin takes $O(|V|)$ (line 5). We can do better $(O(|V| + |E|log|E|))$ using a heap.

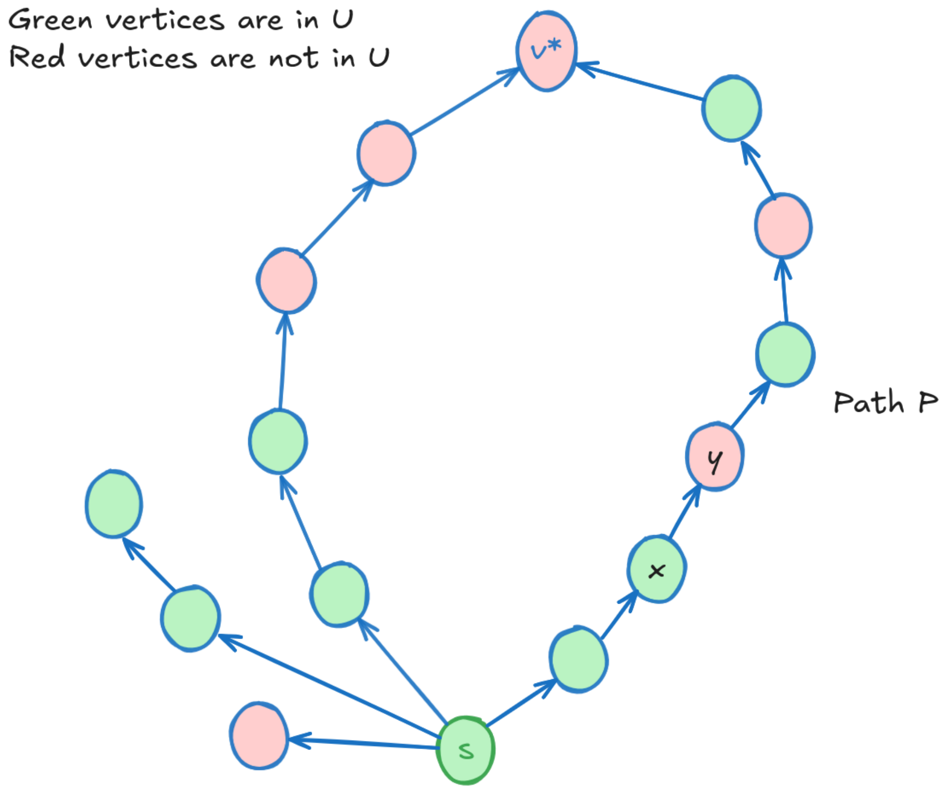

Figure 1: An example of a path P. Note that it may leave U and enter U again.