# Contents



% Chapter 06 — Shortest Paths (Dijkstra & Bellman–Ford) % CS161 Reader (Plotkin F10) %

# 1 Chapter 06 — Shortest Paths

Shortest-path algorithms are relaxation machines: they maintain **upper bounds** and improve them until no improvement is possible. Dijkstra is fast but needs **nonnegative weights**; Bellman–Ford is slower but handles negative edges and detects negative cycles.

**Primary patterns:** relaxation, invariant.

**Executable witnesses:** `cs161lab.algorithms.sssp.dijkstra`, `cs161lab.algorithms.sssp.bellman_ford`.

## 1.1 6.1 Problem definition

Let $G = (V, E)$ be a directed graph with weights $w : E \to \mathbb{R}$ and a source $s$. Define

$$\delta(s, v) = \min_{p: s \rightsquigarrow v} w(p),$$

where $w(p)$ sums edge weights along $p$; if no path exists, $\delta(s, v) = +\infty$.

If there is a reachable negative cycle, shortest paths are ill-defined for vertices reachable from it (distance $-\infty$).

We maintain estimates $d[v]$ and predecessors $\pi[v]$.

## 1.2 6.2 Relaxation and the upper-bound invariant

### 1.2.1 Definition (Relaxation)

Relaxing edge $(u, v)$ sets

$$d[v] \leftarrow \min\{d[v],\ d[u] + w(u, v)\}.$$

### 1.2.2 Invariant 6.1 (Upper-bound invariant)

At all times,

$$d[v] \geq \delta(s, v) \quad \forall v.$$

**Proof.** True at initialization. If $d[u] \geq \delta(s, u)$ then $d[u] + w(u, v) \geq \delta(s, u) + w(u, v) \geq \delta(s, v)$, so taking a min preserves the inequality. □

### 1.2.3 Lemma 6.2 (Witness paths via predecessors)

Whenever $d[v]$ is finite, predecessors reconstruct an $s \to v$ path of weight $d[v]$.

**Proof.** Set $\pi[v] = u$ exactly when relaxation improves $d[v]$ using $u$. Following predecessors yields a path. □

## 1.3 6.3 Dijkstra (nonnegative weights)

Assume $w(e) \geq 0$ for all edges.

### 1.3.1 Algorithm

Initialize $d[s] = 0$, others $+\infty$. Maintain finalized set $S = \emptyset$. Repeatedly extract $u \notin S$ with minimum $d[u]$, add to $S$, relax outgoing edges.

**Trace events:** `finalize`, `relax`.

### 1.3.2 Theorem 6.3 (Finalization lemma)

When Dijkstra finalizes a vertex $u$, then $d[u] = \delta(s, u)$.

**Proof.** Let $S$ be finalized vertices before choosing $u$. Consider a shortest path from $s$ to $u$ and let $(x, y)$ be the first edge crossing from $S$ to $V \setminus S$. When $x$ was finalized, $(x, y)$ was relaxed, so

$$d[y] \leq d[x] + w(x, y) = \delta(s, x) + w(x, y) = \delta(s, y).$$

With Invariant 6.1, $d[y] = \delta(s, y)$. Since remaining edges on the path are nonnegative, $\delta(s, u) \geq \delta(s, y) = d[y]$. As $u$ minimizes $d$ outside $S$, $d[u] \leq d[y] \leq \delta(s, u)$. With Invariant 6.1, $d[u] \geq \delta(s, u)$, hence equality. □

### 1.3.3 Corollary 6.4

At termination, Dijkstra outputs correct distances for all vertices reachable from $s$.

### 1.3.4 Complexity

Using a binary heap: $O((n + m) \log n)$.

## 1.4 6.4 Bellman–Ford (negative edges allowed)

### 1.4.1 Algorithm

Initialize $d[s] = 0$, others $+\infty$. Repeat $n - 1$ passes: relax every edge. Do one extra pass; if any edge relaxes, report a reachable negative cycle.

**Trace events:** `iteration`, `relax`, `neg_cycle`.

### 1.4.2 Lemma 6.5 (k-edge optimality)

After $k$ full passes, $d[v]$ is at most the shortest-path weight among all $s \to v$ paths using at most $k$ edges.

**Proof (induction).** Each pass allows paths that extend a $k$-edge best path by one edge. □

### 1.4.3 Theorem 6.6 (Correctness without negative cycles)

If there is no reachable negative cycle, then after $n-1$ passes, $d[v] = \delta(s, v)$ for all $v$.

**Proof.** A shortest path can be chosen simple, hence has at most $n-1$ edges. Apply Lemma 6.5 with $k = n-1$ and combine with Invariant 6.1. □

### 1.4.4 Theorem 6.7 (Negative-cycle detection)

If an edge can still be relaxed after $n-1$ passes, then a reachable negative cycle exists.

**Proof.** A strict improvement implies an improving walk of length at least $n$ edges, hence repeats a vertex; the repeated segment is a negative cycle. □

## 1.5 Appendix — Trace events as proof witnesses

- Dijkstra `finalize` events correspond to Theorem 6.3 checkpoints.
- Bellman–Ford `iteration(k)` tracks the Lemma 6.5 induction parameter.
- `neg_cycle` is a concrete witness of a violated "no negative cycle" assumption.