Data Structures and Algorithms

Lecture 31

Aniket Basu Roy

2026-04-13 Mon

Contents

1 Agenda

- Proof of correctness of Dijkstra's algorithm
- Why negative edge weights break Dijkstra's algorithm

2 Proof of Correctness

2.1 Theorem

DIJKSTRA correctly computes $v.d = \delta(s, v)$ for all $v \in V$, provided all edge weights are non-negative.

2.2 Loop Invariant

At each iteration of the while-loop, for every vertex v already in S: $v.d = \delta(s, v)$.

2.3 Proof (induction on $|S|$)

Base case ($|S| = 1$): The first vertex extracted is s (smallest initial estimate, $s.d = 0$, all others ∞). Clearly $\delta(s, s) = 0 = s.d$.

Inductive step: Suppose every $v \in S$ satisfies $v.d = \delta(s, v)$. Let u be the next vertex extracted. Suppose for contradiction $u.d > \delta(s, u)$.

Let p be a shortest path $s \rightsquigarrow u$. Since $s \in S$ and $u \notin S$, path p must cross the cut $(S,\ V \setminus S)$. Let (x, y) be the first edge of p with $x \in S$ and $y \notin S$.

```
s --- ... --- x --> y --- ... --> u
(all in S)          (none in S)
```

1. By induction: $x.d = \delta(s, x)$.
2. When x was extracted, RELAX(x, y, w) was called: $y.d \leq x.d + w(x, y)$.
3. By optimal substructure on p, the subpath $s \rightsquigarrow x \to y$ is a shortest \$s\$-to-y path: $x.d + w(x, y) = \delta(s, x) + w(x, y) = \delta(s, y)$.
4. Hence $y.d \leq \delta(s, y)$. Combined with the upper-bound property ($y.d \geq \delta(s, y)$): $y.d = \delta(s, y)$.
5. Since all edge weights are non-negative, the subpath $y \rightsquigarrow u$ in p has non-negative weight: $\delta(s, y) \leq \delta(s, u)$.
6. Since EXTRACT-MIN chose u over y (both were in Q): $u.d \leq y.d$.

Combining steps 4–6:

$$u.d \leq y.d = \delta(s, y) \leq \delta(s, u).$$

The upper-bound property gives $u.d \geq \delta(s, u)$. Thus $u.d = \delta(s, u)$, contradicting our assumption. ■

2.4 Critical use of non-negative weights

Step 5 ($\delta(s, y) \leq \delta(s, u)$) requires that going further along p from y to u does not decrease the distance. This holds iff all edge weights on that subpath are non-negative. With a negative edge after y, the path to u could be cheaper than the path to y, invalidating the greedy choice.

3 Negative Edge Weights: Counterexample

3.1 Setup

```
s ---(1)---> a
|
+---(2)---> b ---(-3)---> a
```

Edges: $s \to a$ (weight 1), $s \to b$ (weight 2), $b \to a$ (weight -3).
True shortest-path distances:

$$\delta(s, a) = \min(1,\ 2 + (-3)) = -1, \qquad \delta(s, b) = 2.$$

3.2 Dijkstra's Execution

Step	Extracted	$s.d$	$a.d$	$b.d$	Action
Init	—	0	∞	∞	
1	s	0	1	2	Relax $s \to a$, $s \to b$
2	a	0	1	2	a finalized with $a.d = 1$; no outgoing edges
3	b	0	1	2	RELAX$(b, a, -3)$: $a.d$ would become -1, but $a \notin Q$

Result: $a.d = 1 \neq \delta(s, a) = -1$.

3.3 Why it fails

Dijkstra greedily finalizes a at distance 1 before the path $s \to b \to a$ (true cost -1) is discovered. Once a is extracted and removed from Q, RELAX can no longer update it.

The cheap route arrives "too late" — through a negative edge from a vertex with a larger d-value.

3.4 Fix: Bellman-Ford (Lecture 32)

Relax **all** edges repeatedly, $|V| - 1$ times. No early finalization.

4 Questions

- In the proof, step 3 uses optimal substructure of p to conclude $\delta(s, y) = \delta(s, x) + w(x, y)$. Why does this require (x, y) to be the **first** crossing edge?
- Give a second counterexample to Dijkstra's with negative edges, using a directed acyclic graph.
- For which graph structures does Dijkstra's give correct answers even with some negative edges?