Data Structures and Algorithms

Lecture 30

Aniket Basu Roy

2026-04-10 Fri

Contents

1 Agenda

1.1 Single Source Shortest Paths

- Edge-weighted graphs, problem definition

- Optimal substructure, relaxation
- Dijkstra's algorithm: definition and example
- Time complexity

2 Edge-Weighted Graphs

A **weighted directed graph** $G = (V, E, w)$ where $w : E \to \mathbb{R}$.

The **weight of a path** $p = \langle v_0, v_1, \ldots, v_k \rangle$:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

Shortest-path weight from s to v:

$$\delta(s, v) = \begin{cases} \min\{w(p) : s \xrightarrow{p} v\} & \text{if } v \text{ is reachable from } s, \\ \infty & \text{otherwise.} \end{cases}$$

A **shortest path** from s to v is any path p with $w(p) = \delta(s, v)$.

2.1 Single Source Shortest Paths (SSSP)

Input: Weighted directed graph $G = (V, E, w)$, source vertex $s \in V$.

Output: $\delta(s, v)$ and a shortest path, for every $v \in V$.

3 Optimal Substructure

3.1 Lemma

Subpaths of shortest paths are shortest paths.

Formally: If $p = \langle v_0, \ldots, v_k \rangle$ is a shortest \$v$_0$\$-to-v_k path, then for any $0 \le i \le j \le k$, the subpath $\langle v_i, \ldots, v_j \rangle$ is a shortest \$v$_i$\$-to-v_j path.

3.2 Proof

Cut-and-paste: if a shorter path q from v_i to v_j existed, replacing p_{ij} with q in p gives a shorter \$v$_0$\$-to-v_k path — contradiction. ■

3.3 Triangle Inequality

$$\delta(s, v) \le \delta(s, u) + w(u, v) \quad \text{for every } (u, v) \in E.$$

4 Relaxation

Each vertex v stores an estimate $v.d$ (upper bound on $\delta(s, v)$) and a predecessor $v.\pi$.

```
INITIALIZE-SINGLE-SOURCE(G, s)
1.  for each v in G.V
2.      v.d  = infinity
3.      v.pi = NIL
4.  s.d = 0

RELAX(u, v, w)
1.  if v.d > u.d + w(u, v)
2.      v.d  = u.d + w(u, v)
3.      v.pi = u
```

RELAX checks: "Can I improve the estimate for v by routing through u?"

All SSSP algorithms reduce to choosing which edges to relax and in what order.

4.1 Upper-Bound Property

After any sequence of RELAX calls: $v.d \geq \delta(s, v)$ for all v. (RELAX only sets $v.d \leftarrow u.d + w(u, v) \geq \delta(s, u) + w(u, v) \geq \delta(s, v)$.)

5 Dijkstra's Algorithm

Assumption: All edge weights are non-negative: $w(u, v) \geq 0$ for all $(u, v) \in E$.

Greedy strategy: Maintain a set S of vertices with finalized shortest-path distances. Repeatedly extract the vertex $u \notin S$ with the smallest estimate $u.d$, add it to S, and relax all edges leaving u.

```
DIJKSTRA(G, w, s)
1.  INITIALIZE-SINGLE-SOURCE(G, s)
2.  S = {}
3.  Q = G.V                          // min-priority queue keyed by v.d
4.  while Q != {}
5.      u = EXTRACT-MIN(Q)
6.      S = S union {u}
```

```
7.      for each v in G.Adj[u]
8.          RELAX(u, v, w)
```

Analogy with Prim-Jarník MST: replace "minimum incident edge weight" by "minimum path-distance estimate $u.d + w(u, v)$".

6 Example

Graph: $V = \{s, t, x, y, z\}$ (CLRS Figure 22.6).

```
Edges:  s->t (10),  s->y (5),
        t->x (1),   t->y (2),
        y->t (3),   y->x (9),  y->z (2),
        x->z (4),
        z->x (6),   z->s (7)
```

Step	Extracted	$s.d$	$t.d$	$x.d$	$y.d$	$z.d$
Init	—	0	∞	∞	∞	∞
1	s	0	10	∞	5	∞
2	y	0	8	14	5	7
3	z	0	8	13	5	7
4	t	0	8	9	5	7
5	x	0	8	9	5	7

Final: $\delta(s, s) = 0$, $\delta(s, y) = 5$, $\delta(s, z) = 7$, $\delta(s, t) = 8$, $\delta(s, x) = 9$.

7 Time Complexity

Let $n = |V|$, $m = |E|$. Same analysis as Prim-Jarník.

Operation	# Calls	Cost per call	Total
EXTRACT-MIN	n	$O(\log n)$	$O(n \log n)$
RELAX (DECREASE-KEY)	$\leq m$	$O(\log n)$	$O(m \log n)$

Using a binary min-heap:

$$\boxed{T_{\text{Dijkstra}} = O(m \log n)}$$

(For connected graphs $m \geq n - 1$, so the $m \log n$ term dominates.)

8 Questions

- Trace Dijkstra's on the example above and draw the resulting shortest-path tree.
- How should Dijkstra's be adapted for undirected graphs with non-negative weights?
- What loop invariant does Dijkstra's maintain about the set S?