

Course «Algorithms» · Lecture 11

Greedy Algorithms: Dijkstra’s Algorithm

Textbook: A. Levitin, Section 9.3

Today’s question

What is the **shortest way from here to everywhere else?**

- Dijkstra’s algorithm: almost Prim’s code, one number changed
- a full example with path recovery
- why it is correct, and exactly where it breaks

Single-source shortest paths

A graph with **nonnegative** weights and a source s. Find $\delta(s, v)$, the shortest distance to every v, and the paths.

- maps and navigation, network routing (OSPF), flight planning
- Floyd solves all pairs in $\Theta(n^3)$; for one source there is a faster, **greedy** way

The idea: grow a tree of final distances

- Tree vertices: shortest distance is **final**
- Outside vertex u: tentative distance **d[u]** and predecessor **p[u]**

Greedy step: take the outside vertex with the smallest d, make it final, add it to the tree.

Relaxing the edges of the new vertex v

For each neighbor u: is going through v shorter than what we had?

$$d[v] + w(v, u) < d[u] \quad \Rightarrow \quad d[u] \leftarrow d[v] + w(v, u), \quad p[u] \leftarrow v$$

The algorithm

```
Dijkstra(G, s)
 for each vertex v: d[v] ← ∞; p[v] ← nil
 d[s] ← 0
 Q ← all vertices with keys d
 while Q is not empty do
  v ← ExtractMin(Q)    // d[v] is final
  for each neighbor u of v still in Q do
   if d[v] + w(v, u) < d[u]
    d[u] ← d[v] + w(v, u); p[u] ← v
    DecreaseKey(Q, u, d[u])
```

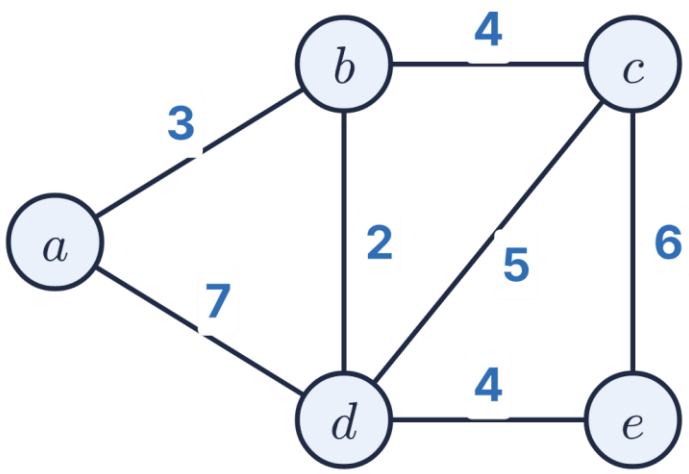
Compare with Prim (Lecture 10)

The only difference is the key.

$$\text{Prim}: \quad \text{key}[u] = w(v, u) \qquad\qquad \text{Dijkstra}: \quad d[u] = d[v] + w(v, u)$$

Prim: the weight of **one edge**. Dijkstra: the length of the **whole path** from s.

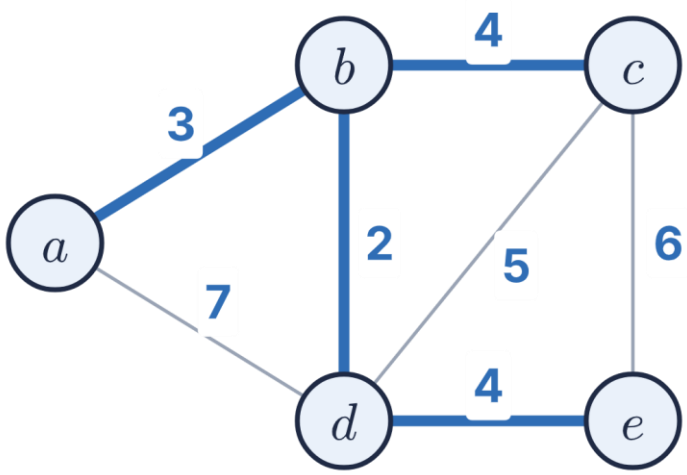
Example: source a



Step	Final	Tentative d (via)
0	a (0)	b: 3 (a), d: 7 (a), c: ∞, e: ∞
1	b (3)	c: 7 (b), d: 5 (b), e: ∞
2	d (5)	c: 7 (b), e: 9 (d)
3	c (7)	e: 9 (d)
4	e (9)	—

Step 1: via b, d costs 3 + 2 = 5 instead of the direct 7. Via d, c would cost 10 and via c, e would cost 13: no change.

Recovering the paths



Follow p[·] back from the target, then reverse: e ← d ← b ← a

To	Path	Length
b	a → b	3
d	a → b → d	5
c	a → b → c	7
e	a → b → d → e	9

The predecessor array already is the shortest-path tree.

Why is it correct?

$$\text{when } v \text{ is extracted}: \quad d[v] = \delta(s, v)$$

d[v] is always the length of some real path, so it is never below the truth. The only risk is d[v] > δ(s, v).

Proof by contradiction

Let v be the **first** extracted vertex with d[v] > δ(s, v). On a true shortest path to v, let y be the first vertex outside the tree and x the one before it.

x was final, (x, y) was relaxed:

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

Weights ≥ 0: a prefix is no longer than the whole path

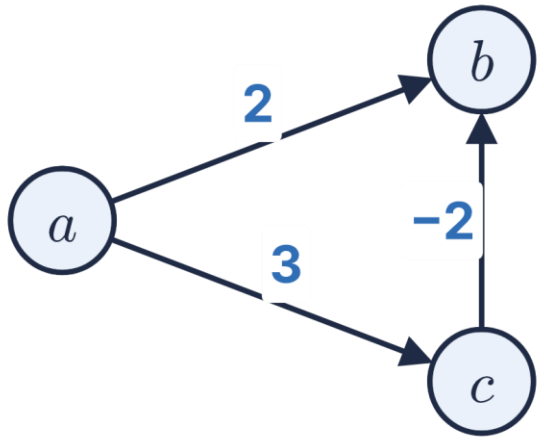
$$\delta(s, y) \le \delta(s, v)$$

Together (so y ≠ v):

$$d[y] = \delta(s, y) \le \delta(s, v) < d[v]$$

But v was chosen as the minimum while y was in the queue: d[v] ≤ d[y]. Contradiction. ■

Negative weights break it



1. d[b] = 2, d[c] = 3
2. b is smallest: final with 2
3. c final with 3; c → b would give 1, but b is already final

True distance δ(a, b) = 3 + (−2) = **1**, Dijkstra says **2**, with no warning. Use Bellman–Ford or Floyd.

How fast is it?

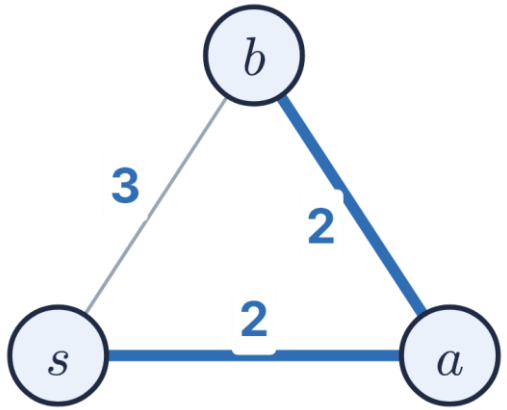
Implementation	Time
Array, adjacency matrix	$\Theta(\|V\|^2)$
Binary heap, adjacency lists	$\Theta(\|E\| \log \|V\|)$
Fibonacci heap	$\Theta(\|E\| + \|V\| \log \|V\|)$

- Same analysis as Prim: $|V|$ extractions, $\leq |E|$ decrease-keys
- From every vertex on a sparse graph: beats Floyd's $\Theta(|V|^3)$
- One target t? Stop as soon as t is extracted

$$\Theta(|V| \cdot |E| \log |V|)$$

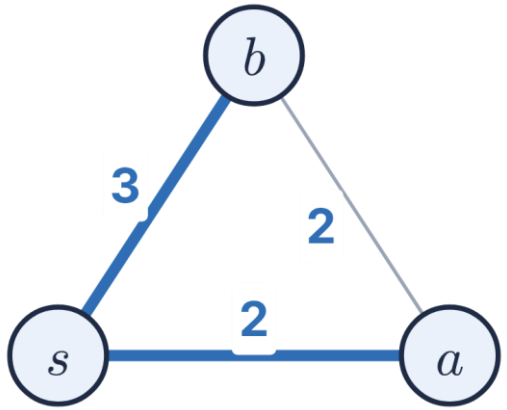
Same code, different trees

Prim from s



Total weight 4: the MST

Dijkstra from s



b at distance 3, not 4

Common mistakes

- **Negative weights:** wrong answer, no warning
- **Prim's key instead of Dijkstra's:** one edge vs. the whole path
- **Updating final vertices:** relax only those still in the queue

Summary

- Finalize the vertex with the smallest d, then relax its edges
- Paths come from the predecessors p[u]: a shortest-path tree
- Correctness needs nonnegative weights: a prefix is never longer
- $\Theta(|V|^2)$ with an array, $\Theta(|E| \log |V|)$ with a heap
- Same code as Prim, different key, different tree

Review questions

1. Trace Dijkstra from s on s → a (4), s → b (1), b → a (2), a → t (1), b → t (6). Give all distances and the path to t.
2. State and prove the invariant behind Dijkstra's algorithm. Where exactly are nonnegative weights used?
3. Give a graph with a negative edge on which Dijkstra fails. What does it output, and what is the truth?