Shortest Paths Problem

**Problem**: find shortest paths in a directed graph with edge *lengths* (the Google maps problem)

Let's Formalize the Problem

- Directed graph $G = (V, E)$ with edge lengths $\ell(e) > 0$
- Define length of path $P$ consisting of edges $e_1, e_2, \ldots, e_k$ as $\ell(P) = \ell(e_1) + \ell(e_2) + \ldots + \ell(e_k)$
- Starting node $s$
- Let $d(v)$ be the length of shortest $s \rightarrow v$ path.
- **Problem**: Can we efficiently find $d(v)$ for all nodes $v \in V$?

Shortest Paths Problem

Suppose all edges have integer length. Can we use BFS to solve this problem?
Shortest Paths Problem

Idea: keep track of the “wavefront”
- \(d'(v)\) — best tentative arrival time so far for node \(v\)
- \(d(v)\) — actual arrival time

What’s required to keep track of the wavefront?
- Find next arrival: find node \(v\) with smallest \(d'(v)\)
- Set arrival time: \(d(v) = d'(v)\)
- Update \(d'(v)\) for neighbors of \(v\) if they get better “offers”

What data structure supports find smallest and update values?
Priority queue.

Dijkstra’s Algorithm

Set \(A = V\)
Set \(d'(v) = \infty\) for all nodes
Set \(d'(s) = 0\)
while \(A\) not empty do
  Extract node \(v\) ∈ \(A\) with smallest \(d'(v)\) value
  Set \(d(v) = d'(v)\)
  for all edges \((v, w)\) where \(w\) ∈ \(A\) do
    if \(d(v) + \ell(v, w) < d'(w)\) then
      \(d'(w) = d(v) + \ell(v, w)\)
    end if
  end for
end while

Running Time?

Use heap-based priority queue for \(A\)
Set \(A = V\)
Set \(d'(v) = \infty\) for all nodes
Set \(d'(s) = 0\)
while \(A\) not empty do
  Extract node \(v\) ∈ \(A\) with smallest \(d'(v)\) value
  Set \(d(v) = d'(v)\)
  for all edges \((v, w)\) where \(w\) ∈ \(A\) do
    if \(d(v) + \ell(v, w) < d'(w)\) then
      \(d'(w) = d(v) + \ell(v, w)\)
    end if
  end for
end while

- \(n\) extract-min operations. \(O(n \log n)\)
- \(m\) update-key operations. \(O(m \log n)\)
- Total: \(O((m + n) \log n)\)
Finding the Actual Path

Keep track of prev(v) = node that last updated arrival time 

d'(v) = predecessor in shortest path

Set A = V
Set d'(v) = \infty for all nodes
Set prev(v) = null
Set d'(s) = 0
while A not empty do
Extract node v ∈ A with smallest d'(v) value
Set d(v) = d'(v)
for all edges (v, w) where w ∈ A do
if d(v) + ℓ(v, w) < d'(w) then
    d'(w) = d(v) + ℓ(v, w)
    prev(w) = v
end if
end for
end while

Proof of Correctness

Let S = V \ A be the set of explored nodes at any point in the algorithm—those v for which we have assigned d(v)

Observation: for v \notin S, the value d'(v) is the minimum value 
d(u) + ℓ(u, v) over all edges (u, v) where u ∈ S, v \notin S.

Claim (invariant): for v ∈ S, the value d(v) is the length of the shortest s ⇝ v-path

Proof (by induction)

Base case: Initially S = {s} and d(s) = 0. ✓

Induction step:

Assume the invariant is true after the kth execution of the while loop, when |S| = k.
Let v be the next node added to S, and let (u, v) be the preceding edge. Then d'(u) = d(u) + d(u, v), and d'(u) ≤ d'(x) or any node x.
Let P_s be the shortest s ⇝ u path, which has length d(u)
Let P = P_s \cup (u, v) be the path found by Dijkstra, which has length ℓ(P) = d'(v) = d(u) + ℓ(u, v)
Consider any other s ⇝ v path P'. We'll argue that P is already longer than P_s by the time it first leaves S.
Let (x, y) be the first edge in P with x ∈ S, y \notin S, and let P' be the subpath of P from s to x
Then,
ℓ(P) ≥ ℓ(P') + ℓ(x, y) ≥ d(x) + ℓ(x, y) ≥ d'(y) ≥ d'(v) = ℓ(P_v)