Single-Source Shortest Paths

Given a directed or undirected graph with non-negative “lengths” of edges/ arcs (= numeric labels), and given a source node \( s \), find for each node \( v \) the shortest “distance” (= least sum of labels) of any path from \( s \) to \( v \).

Dijkstra’s Algorithm

Grows a region of settled nodes whose shortest distance from \( s \) is known.

- Inductive computation: For each node \( v \), \( \text{dist}(v) \) is the length of the shortest path to \( v \) that goes only through settled nodes (called a special path).
  - If \( v \) is settled, then \( \text{dist}(v) \) is the correct shortest distance to \( v \).

Basis: Initially, only \( s \) is settled.

- \( \text{dist}(s) = 0 \), and \( \text{dist}(v) \) for other nodes \( v \) is either the length of an arc \( s \to v \) or \( \infty \) if there is no such arc.

Induction: Find the least \( \text{dist}(v) \) for any \( v \) that is not settled.

1. Make \( v \) settled.

2. For every unsettled node \( u \), see if there is now a shorter special path that goes through \( v \), the newly settled node.
   - Compare \( \text{dist}(u) \) with \( \text{dist}(v) + \) the length of arc \( v \to u \).
   - Replace \( \text{dist}(u) \) with the latter, if the latter is smaller.

Why Does It Work? (FCS, pp. 504ff)

Intuition: if there were a shorter path from \( s \) to \( v \), then it would first leave the settled region to some other node \( w \).
Thus, $\text{dist}(w) < \text{dist}(v)$.

Note needed assumption that lengths are $\geq 0$.

$O(n^2)$ Implementation

There are $n-1$ “rounds” in which a node is settled. In each round:

- $O(n)$ time to pick the smallest $\text{dist}$ among unsettled nodes.
- $O(n)$ time to consider if other $\text{dist}$ values need to be lowered.

$O(m \log n)$ Implementation (FCS, pp, 506ff)

Better if $m < n^2$ (i.e., the graph is sparse) and adjacency lists are used. Key ideas:

1. Keep $\text{dist}$ in a priority queue, so we can find and delete the least distance of an unsettled node in $O(\log n)$ time.
   - Actually, “priority” is lowest-first here, not greatest-first.
   - When we lower $\text{dist}(u)$, the position of $u$ in the PQ may change, so it will take $O(\log n)$ time to “bubbleup.”

2. Count the work of updating successors $u$ of the settled node $v$ more carefully.
   - If $v$ has $m_v$ successors, then work is $O(m_v \log n)$ ($\log n$ for bubbling up for each of $m_v$ nodes).
   - Thus, total update work $= \sum_v m_v \log n = O(m \log n)$.
   - That is also the dominant term of the whole algorithm.

Class Problem

Suppose we have already computed $\text{dist}(v)$ for all nodes $v$. Now, we add another arc $y \to z$ with some length. Do we have to recompute all the distances, or can we take advantage of the old distances?