Shortest Paths

The problem:

- Let $G = (V, E)$ be a directed graph
- Edge weights $w : E \to \mathbb{R}$
- Convention: $w(u, v) := \infty$ if $(u, v) \notin E$
- The weight of a path $p = \langle v_0, v_1, \ldots, v_k \rangle$:
  \[
  w(p) := \sum_{i=1}^{k} w(v_{i-1}, v_i)
  \]
- The shortest path length from $u$ to $v$:
  \[
  \delta(u, v) := \min \{ w(p) : p \text{ is a path from } u \text{ to } v \} \]
Some extremes:

- If there is no path from $u$ to $v$, then $\delta(u, v) := \infty$
- If there is a path from $u$ to $v$ that contains a negative weight cycle, then $\delta(u, v) := -\infty$

Cycles:

- A shortest path cannot contain either:
  - a negative weight cycle, or
  - a positive weight cycle
  but may contain a zero-weight cycle
- There is always a shortest path with no cycles
- There is always a shortest path with $\leq |V| - 1$ edges
Variations:

- Single source
- Single destination
- Single pair
- All pairs
Single source shortest paths

A unified approach: “relaxation”

- Goal: compute shortest paths from \( s \) to all other nodes

- Initialization
  \[
  \begin{align*}
  \text{for each } v \in V: & \quad d[v] \leftarrow \infty \\
  \pi[v] & \leftarrow \text{Nil} \\
  d[s] & \leftarrow 0
  \end{align*}
  \]

- \text{Relax}(u, v) ((u, v) \in E)
  \[
  \begin{align*}
  \text{if } d[u] + w(u, v) < d[v] \text{ then} & \quad d[v] \leftarrow d[u] + w(u, v) \\
  \pi[v] & \leftarrow u
  \end{align*}
  \]

- Structure of algorithm:
  Initialize, perform sequence of relaxations
Lemma (Triangle inequality)

for all \(x, u, v \in V\), if \((u, v) \in E\), then
\[
\delta(x, v) \leq \delta(x, u) + w(u, v)
\]

Lemma (Relaxation properties)

1) the values \(d[v]\) only decrease over time

2) \(d[v] \geq \delta(s, v) \quad [T]\)

3) if there is a path \(p = \langle v_0, v_1, \ldots, v_k \rangle\) from \(s\) to \(v\), and edges \((v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\) are relaxed in order (interleaved with other relaxations), then \(d[v] \leq w(p)\)

4) if \(\pi[v] = u\), then \(d[u] + w(u, v) \leq d[v] \quad [R1]\)
Bellman-Ford Algorithm

- Detects if negative weight cycles (reachable from s) exist
- If not, computes shortest paths
- Logic:

  Initialize
  repeat |V| times
  for each edge \((u, v) \in E\) do
    Relax\((u, v)\)
  if \{\text{any entries of } d \text{ changed in the last iteration}\} then
    output “negative cycles”
Running Time

- $O(|V| \cdot |E|)$ (which is $O(|V|^3)$)

Correctness:

- Let $p = \langle v_0, v_1, \ldots, v_k \rangle$ be a path from $s$ to a vertex $v$
- In loop iteration $i$, the edge $(v_{i-1}, v_i)$ is relaxed
- By [R1–3], at the end of loop iteration $k$, and all subsequent loop iterations, $\delta(s, v) \leq d[v] \leq w(p)$
- If $p$ is a shortest path, and $k \leq |V| - 1$, then $d[v] = \delta(s, v)$ at the end of loop iterations $|V| - 1$ and $|V|$
Correctness (cont’d):

- **Case 1:** assume a negative cycle
  - Need to show algorithm does not stabilize in last iteration
  - Let $m(k)$ be the minimum entry in $d$ after $k$ loop iterations, where we let $k \to \infty$
  - For every $x > 0$, there exists a path $p$ from $s$ with $w(p) \leq -x$, so $\lim_{k \to \infty} m(k) = -\infty$
  - However, if algorithm stabilized in loop iteration $|V|$, it would remain stable in all subsequent iterations as well, and $\lim_{k \to \infty} m(k) > -\infty \implies \Leftarrow$
Correctness (cont’d):

- Case 2: assume no negative cycles
  - All path weights are correctly computed by the end of loop iteration $|V| - 1$, and remain stable in iteration $|V|$
  - Also, we want to show that $\pi$ gives shortest paths
  - First, we show that the graph defined by $\pi$ is acyclic
  - Suppose not, and consider the first relaxation, say $(\nu_0, \nu_1)$, that creates a cycle, say $\langle \nu_0, \nu_1, \ldots, \nu_{k-1}, \nu_k \rangle$, where $\nu_k = \nu_0$
Correctness (cont’d):

• Case 2 (cont’d):
  – Consider the array $d$ just before we relax $(\nu_0, \nu_1)$
  – We have $d[\nu_0] + w(\nu_0, \nu_1) < d[\nu_1]$
  – By [R4], we also have
    $$d[\nu_i] + w(\nu_i, \nu_{i+1}) \leq d[\nu_{i+1}] \text{ for } i = 1 \ldots k - 1$$
    $$\sum_{i=0}^{k-1} d[\nu_i] + \sum_{i=0}^{k-1} w(\nu_i, \nu_{i+1}) < \sum_{i=0}^{k-1} d[\nu_{i+1}]$$

    $$\sum_{i=0}^{k-1} w(\nu_i, \nu_{i+1}) < 0 \quad \Rightarrow \Leftarrow$$
Correctness (cont’d):

- Case 2 (cont’d):
  - It follows that $\pi$ defines a tree rooted at $s$
  - Let $(\nu_0, \ldots, \nu_k)$ be a tree path from $s$ to $\nu$
  - Claim: This is a shortest path to $\nu$
  - Prove by induction on $k$
  - We have
    \[
    \delta(s, \nu_k) \leq \delta(s, \nu_{k-1}) + w(\nu_{k-1}, \nu_k) \quad [T]
    = d[\nu_{k-1}] + w(\nu_{k-1}, \nu_k) \quad [d = \delta]
    \leq d[\nu_k] \quad [R4]
    = \delta(s, \nu_k) \quad [d = \delta]
    \]
    \[
    \therefore \delta(s, \nu_k) = \delta(s, \nu_{k-1}) + w(\nu_{k-1}, \nu_k), \text{ and the claim follows by induction}
    \]
Shortest paths in a DAG

Algorithm:

let \( u_1, \ldots, u_n \) be a topological sort of \( G \)
for \( i \leftarrow 1 \) to \( n \) do
    for each \( v \in \text{Succ}(u_i) \) do
        Relax\((u_i, v)\)

Running time: \( O(|V| + |E|) \)

Correctness:

- Consider a shortest path \( p \) from \( s \) to a vertex \( v \)
- By Top Sort Property, \( p = \langle u_{i_1}, u_{i_2}, \ldots, u_{i_k} \rangle \),
  where \( i_j < i_{j+1} \) for \( j = 1, \ldots, k - 1 \)
- Since the edges \((u_{i_j}, u_{i_{j+1}})\) are relaxed in order,
  by \([R3]\), we have \( d[v] = \delta(s, v) \) on termination
Dijsktra’s Algorithm

Assumption: no negative edges

Definition:

- A subset $S \subseteq V$ is called a cluster about $s$ if $s \in S$ and $\delta(s, u) \leq \delta(s, v)$ for all $u \in S$ and $v \in V \setminus S$

- For a cluster $S$ and $v \in V \setminus S$, define

$$D_S(v) := \min \left\{ \delta(s, u) + w(u, v) : u \in S \right\}$$
Lemma (Cluster properties)

1) \{s\} is a cluster

2) If \(S\) is a cluster, and \(v \in V \setminus S\) with minimal \(D_S(v)\), then \(\delta(s, v) = D_S(v)\) and \(S \cup \{v\}\) is also a cluster

Proof:

• (1) is clear (no negative edges)
• For (2), we first prove the Claim:

\[D_S(v) \leq \delta(s, v') \text{ for all } v' \in S \setminus V\]
Proof (cont’d):

- Let \( \langle \nu_0, \ldots, \nu_k \rangle \) be a shortest path from \( s \) to \( \nu' \)
- Let \( i \) be the \textit{smallest} index such that \( \nu_i \notin S \) (we have \( 1 \leq i \leq k \))
- We have

\[
\delta(s, \nu') = w(\langle \nu_0, \ldots, \nu_k \rangle)
\geq w(\langle \nu_0, \ldots, \nu_{i-1}, \nu_i \rangle) \quad [w \geq 0]
= w(\langle \nu_0, \ldots, \nu_{i-1} \rangle) + w(\nu_{i-1}, \nu_i)
\geq \delta(s, \nu_{i-1}) + w(\nu_{i-1}, \nu_i)
\geq D_S(\nu_i)
\geq D_S(\nu)
\]
Proof (cont’d):

- That proves the claim:
  \[ D_S(\nu) \leq \delta(s, \nu') \quad \text{for all} \quad \nu' \in S \setminus V \]

- Applying claim with \( \nu' := \nu \), \( D_S(\nu) \leq \delta(s, \nu) \)

- By \([T]\), \( D_S(\nu) \geq \delta(s, \nu) \)

- \( \therefore D_S(\nu) = \delta(s, \nu) \)

- Applying claim with arbitrary \( \nu' \in V \setminus S \):
  \[ \delta(s, \nu') \geq D_S(\nu) = \delta(s, \nu), \]
  which implies that \( S \cup \{\nu\} \) is a cluster
Algorithm:

Initialize
\[ Q \leftarrow V \]

while \( Q \neq \emptyset \)

\begin{align*}
\text{select } u \in Q \text{ with minimal } d[u] \\
Q \leftarrow Q \setminus \{u\} \\
\text{for each } v \in \text{Succ}(u) \cap Q \text{ do} \\
\text{Relax}(u, v)
\end{align*}

Implementation: priority queue

- \(|V|\) \textit{ExtractMin’s, and }\(|E|\) \textit{Decrease’s}

Correctness:

- Follows from Cluster Properties: \( S = V \setminus Q; \)
  loop invariant: \( d[v] = D_S(v) \) for each \( v \in Q \)
Running Time:

- unsorted list: $O(|V|^2)$
- binary heap: $O(|E| \log |V|)$
- Fibonacci heap: $O(|V| \log |V| + |E|)$
- “bucket heap”: $O(B|V| + |E|)$, if edge weights are bounded by $B$

By the Cluster Properties Lemma, the quantity

$$\min\{d[v] : v \in Q\}$$

never decreases, and

$$\max\{d[v] : v \in Q, d[v] < \infty\} \leq B|V|$$