Bellman Ford algorithm for SSSP

\( d(u,v) \) is the shortest path distance between vertices \( u \) and \( v \)

\( u \rightarrow v \) is the notation for a path from \( u \) to \( v \)

\( d(s,v) \) is the final output of the algorithm for all \( v \)

\( D(v) \) is an upper bound on \( d(s,v) \)

Initially \( D(v) = \infty \), \( D(s) = 0 \)

Finally \( D(v) = d(s,v) \)

Relax \( (u,v) \quad (u,v) \in E \)

If \( D(v) > D(u) + w(u,v) \), then \( D(v) \leftarrow D(u) + w(u,v) \)

Repeat \( (V-1) \) times

- for all edges \( (u,v) \in E \)
  - Relax \( (u,v) \)

Output the final \( D(v) \) for all \( v \in V \)
Claim: At the end of \( i \) (VI-1 iterations)

\[ D(v) = \delta(s,v) \]

Note: Running time for BF is \( O(|V| \cdot |E|) \)

For every iteration, cost \( \delta \) relaxation for all edges.

Proof by induction on the set of edges in the shortest path from \( s \) to \( v \)

For \( \ell = 0 \)

\[ D(s) = 0 = \delta(s,s) \]

So correctly initialized.

Suppose it is correct for \( \ell < i \) iterations, i.e., all vertices whose shortest path from \( s \) consists of \( \ell < i \) edge have \( \delta(s,x) = D(x) \) after \( i-1 \) iterations.

During the \( i \)th iteration, all edges will undergo relax operation.
Consider a vertex $y$ s.t. the shortest path has $i$ edges.

\[ D(y') = S(x, y') \]

So when we relax edge $(y', y)$

\[ S(x, y) \leq D(y) = D(y') + w(y', y) = S(x, y') + w(y', y) = S(x, y) \]

\[ \Rightarrow \text{All vertices have their correct distances} \]

**Remark:** If there is a -ve cycle in the graph, then BF algorithm can detect it if we let it run for $|V| + |E|$ iterations.

If there is any change in the $D$ values of a vertex then there must be a negative cycle (shortest cycle must have length $\leq n$).
For non-negative weights, we can do better by using Dijkstra

In Dijkstra’s algorithm, the relax operation are carefully scheduled
so that all edges is relaxed at most once.

Partition

\[ V \]

\[ S \]\n
the set of vertices \n\n\[ u : D(u) = \delta(s, u) \]

\[ D(v) \geq \delta(s, v) \]

the correct distances are yet to be determined

Initially \[ S = \{ s \} \]

\[ D(s) = \delta(s, s) = 0 \]

Repeat until \[ U = \emptyset \]

* Iterations \[ v-1 \]

\[ [1. \text{ Choose the vertex with smallest label in } U, \text{ say } x, \text{ and give } O(\log v) \text{ properly que}]

\[ 2. \text{ Relax all outgoing edges for } x \]

\[ \text{Move } x \text{ from } U \text{ to } S \text{ outdegree } (x) \]
Total time: \[ \leq \sum_{v} \log |V| + \text{outdegree}(v) \leq O(|V| \log V + |E|) \]

Why does Dijkstra's algorithm output the correct distances?

\textbf{Claim:} The algorithm discovers the shortest path distances in order of their actual distance from \( s \).

\[ d(s, v) = 2 \]

\[ d(s, v_2) = 10 \]

\( v \), will move into \( S \) before \( v_2 \) in Dijkstra (and the reverse for B.F.)

When a vertex \( x \) has the smallest \( D(x) \) in \( U \) then \( D(z) = d(s, x) \)

\textbf{Proof by contradiction:} Suppose not, i.e.

\[ D(x) > d(s, x) \]
Consider the shortest path from \( s \rightarrow x \)

let \( x' \) be the most recent predecessor of \( x \) in the path. Let \( D(x') = \delta(s, x') \), and \( x' \in S \)

Then \((x', x'')\) must have been relaxed when \( x' \) moved to \( S \)

\[
D(x'') < D(x)
\]

because of non-negative weights

So \( x'' \) should have been picked