Proof of Correctness for Bellman-Ford Algorithm

What we need to show:

1. When $(G, w)$ has no directed cycles (reachable from $s$), Algorithm returns TRUE and $d[v] = \delta(s, v) \neq v$.

2. When $(G, w)$ has a negative cycle that can be reached from $s$, algorithm returns FALSE (d[v] value have no meaning).

---

Case 1: Suppose $\exists$ no negative cycles in $(G, w)$.

[This means no cycle with $\Sigma w(u, v) < 0$.]

Claim 1: If a path has a cycle (see figure below) from $s$ to $v$

Since $\Sigma w(u, v) \geq 0$, removing it $(u, v) \in C$ gives us a path from $s$ to $v$ which at least as good as the one shown above. Hence shortest paths do not have cycles. This gives us a way to imagine such paths as simple paths.
Since # of simple paths from $s$ to $v$ is finite (as discussed in Dijkstra's algorithm proof), the shortest simple path from $s$ to $v$ is the shortest path from $s$ to $v$ overall. And such path exists if $(G, w)$ has no negative cycle. Let this path look like:

\[ \begin{array}{cccc}
P & \vdots & \vdots & \vdots \\
9-1 & \cdots & k-1 & k \\
0-1 & \cdots & k-1 & k \\
9 & \cdots & k & k \\
\end{array} \]

Hence, using convergence property (L24.14),
\[ d[v_i] \text{ is correct after Phase I of RELAX} \]
(which includes RELAX $(s, v_i)$) since before any RELAX, $d[T, J] = 0 = \delta(s, s)$ (no neg. cycle). Similarly, $d[v_{i+1}]$ is correct in Phase II, and so on. Hence by Phase $(|V| - 1)$,
\[ d[v] \text{ is correct for all } v \in V. \]

Hence in Phase IV, no value can decrease by Lemma 24.11. Hence Flag is TRUE for the algorithm.

Now we move to case 2.
Case 2: Suppose \( C \) is a directed cycle in \((G, \omega)\) such that \( \sum \omega(u, v) < 0 \) for all \((u, v) \in E\).

We want to show that algorithm returns \( \text{FALSE} \).

Recall algorithm returns \( \text{TRUE} \) if in Phase IV, no \( d[v] \) value changes. \( d[v] \) value does not change implies that when we RELAX \((u, v, w)\) in Phase IV, \( d[v] = d[v] \) before.

Thus, \( \forall (u, v) \in E: \)

\[
\begin{align*}
\text{Add these} & \quad \text{if they are } < \infty \\
\text{BLUE = GREEN} & \quad \Rightarrow \begin{cases}
\end{cases}
\end{align*}
\]
So far, we have completed proof of Correctness for DAG alg. and B.F alg. Now we turn to proof for Dijkstra algorithm. When we do this proof, it will be clear that this is a dyn. prog. based alg. not a greedy alg.

There are several proofs; we give one.

Recall: For using Dijkstra algorithm we need the assumption that \( W(u,v) \geq 0 \) for \( (u,v) \in E \).

You will see repeated use of this in the proof below.

Our proof of Correctness for this algorithm is on the basis of an "invariant" property. This is Stated in the form of a claim which is sufficient for completing the proof.

Claim: Let \( S \) be the set of nodes (at any step) that have been processed so far.

We keep track of the order in which nodes are included in the set. First node to enter \( S \) is the origin \( s \). [Thereafter, nodes enter in increasing order of \( \delta(s,v) \)].

Now we are ready to state and prove the claim (on the next page).
Claim: For each vertex \( u \in V \), at the time when
\( u \) is included in \( S \), we have
\[
d[u] = \delta(u,v)
\]
(1)
And this equality is maintained thereafter.

\[
\text{Pf (See also Lect. Note #12)}
\]
Suppose claim is not true; let \( u \) be the first vertex
(in the order of inclusion in \( S \)) for which \( d[u] \neq \delta(u,v) \)
at the time it is included in \( S \).

1. \( u \neq \emptyset \): When \( \emptyset \) is included in \( S \),
\( d[\emptyset] = 0 = \delta(\emptyset,v) \)
\( \delta(\emptyset,v) = 0 \) since \( w(u,v) \geq 0 \quad \forall (u,v) \in E \).

2. Just prior to the time of inclusion of \( u \) in \( S \),
\( S \neq \emptyset \quad [\emptyset \in S] \)

3. If \( \delta(\emptyset,v) = \infty \), by Lemma 24.11,
\( d[\emptyset] = \infty \).
In this case, (1) is assumed to hold by default.
Hence, since \( d[\emptyset] \neq \delta(\emptyset,v) \) at the time of
inclusion of \( u \) into \( S \), \( \delta(\emptyset,v) < \infty \). This means
that there exists a path from \( \emptyset \) to \( u \).

4. Since there are no negative cycles in \( (G,w) \) [due to
the fact that \( w(u,v) \geq 0 \quad \forall (u,v) \in E \)], in looking
for a shortest path, we can restrict our
attention to simple paths. But the number
of simple paths from \( \emptyset \) to \( v \) is finite for
all \( v \in V \).

5. Hence \( \exists \) a shortest path from \( \emptyset \) to \( u \). Since
\( \exists \) a path.
6. Let this shortest path from \( S \) to \( u \) look like:

\[
\begin{align*}
\delta(x, y) \quad \delta(x, y) & \quad \delta(x, u) \quad \delta(x, u)
\end{align*}
\]

Just prior to including \( u \) in \( S \), we have \( x \in S \), \( u \notin S \). Hence, in the above path, there is a first point when we move from a node in \( S \) to a node in not in \( S \).

Let this be from \( x \) to \( y \): \( x \in S \), \( y \notin S \)

[It is possible that \( x = S \) and/or \( y = u \)]

Since \( x \) was included in \( S \) sometime before this point, edge \((x, y)\) was RELAXed at that time. Hence by convergence property (Lemma 24.11), \( d[xU] = \delta(x, u) \) at that time.

\[
\begin{align*}
d[yU] = \delta(x, y) & \leq \delta(x, u) \leq d[xU] \quad \text{([II])}
\end{align*}
\]

All of this just before including \( u \) in \( S \).

7. Both \( y \) and \( u \notin S \). But the alg. is about to include \( u \in S \) not \( y \). This can only be because \( d[u] \leq d[y] \) → ([III])

Recall Dijkstra alg includes the next node with \( \min d[x] \) among those not in \( S \).
9. Hence $d_{yj} = d_{uj}
   
   Combining (II) & (IV) we have $d_{yj} = d_{uj}$
   
   at this point in time (just before including $u$ in $S$).

   This means
   
   $d_{yj} = d(8, y) = d(8, u) = d_{uj}$

   This part is a contradiction to the assumption that $d_{uj} \neq d(8, u)$
   
   at this point in time.

10. Hence the claim which in turn proves that Dijkstra algorithm correctly computes shortest paths from $s$ to $v$

   for all $v \in V$

Recall complexity:

DAG: $\Theta(1E + 1V)$

Dijkstra: $\Theta(1E + \log 1V)$

Bellman Ford: $\Theta(1E \cdot 1V)$

Now we turn to all pair shortest paths