Bellman-Ford Algorithm (Lecture 3 - 31-2020) (Contd from 3-26-2020)

Since this algorithm is applicable to cases of graphs with cycles, and possible negative edges, we have two possible termination cases of existence of:

1. Algorithm indicates the existence of a directed cycle C such that \( \sum w((u, v)) < 0 \) for \( (u, v) \in C \) (called a negative cycle).

   In this case, there exists an algorithm that is polynomially bounded that solves the problem exactly.

   Very likely such algorithms cannot exist either (unless P = NP).

2. Algorithm produces a directed tree rooted at \( s \) (origin) which gives SP for node s in the tree with remaining nodes not having a path.

The algorithm terminates in Case (1) with a flag set at FALSE and in Case (2) with flag set at TRUE and SP. (This is the best we can do.)
**Bellman-Ford Algorithm**

After **initialize**, we `relax` each edge in each phase; there are $|V|$ phases.

If in the last phase, any $d[vj]$ change, flag is set to `FALSE`; else to `TRUE`.

And $d[vj]$ in this case represent SP and $T[vj]$ predecessor node to $v$ in a SP from $s$.

**Complexity** $\Theta (|E|.|V|)$

Undirected graphs with positive weights can be converted

Undir. graphs with possible negative weights but no undir. negative cycles

$\rightarrow$ Done in CS 6381

Not based on DP
Proof of Correctness of Single Origin SP Algorithms

Ch 24 of CLRS: Let 3-26-30 (cont'd) Let 3-31-30

Lemma 24.1 (page 645)
Subpaths of a SP are SP.

Let a shortest path from \( s \) to \( v \) look like

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

Then, \( (x,y) \) subpath is a SP from \( x \) to \( y \).

[This is Optimal Substructure & DP for this problem]

Proof: Suppose lemma is not true. Path \( x\rightarrow m\rightarrow y \)

in green is better

Then path \( x\rightarrow m\rightarrow x\rightarrow m\rightarrow y \) is a better path for \( s \) to \( v \) than the original black path.

But this is a contradiction to black path from \( s \) to \( v \) being optimal.

Now we prove a bunch of results regarding RELAX operation.
Recall $\text{RELAX}(u,v,w)$ // $\text{RELAX}(u,v,w)$

\[ \exists (u, v, w) \]

\[ w(u,v) \]

\[ \text{if } d[v] > d[u] + w(u,v) \]

\[ \text{then } d[v] \leftarrow d[u] + w(u,v) \]

\[ \Pi[v] \leftarrow u \]

Hence, after RELAXing edge $(u,v)$, the only changes that can occur are to $d[v]$ and $\Pi[v]$. (one node)

Moreover, after

before

$d[v]$ is either $d[v]$ or $d[u] + w(u,v)$ whichever is smaller.

Hence $d[v]$ values can NEVER increase.

Now we move on to results in page 670 & 671-673.

Lemma 24.10 (Triangle Inequality) // Recall Triangle Inequality from Geometry

\[ \delta(s,v) \leq \delta(s,u) + w(u,v) \] $\forall (u,v) \in E$

\[ Pf: \] One way to reach $v$ from $s$ is first to go from $s$ to $u$ (using a shortest path) and then going to $v$ using edge $(u,v)$. The length of this is on the right; left is best way to $u$. 


Lemma 24.11 (Upper bound property).

Let the graph be \textsc{initialized} with \textsc{original}.

Then, let $d[v]$ be obtained after some number of \textsc{relax} operations carried out in some order. Then $d[v] \geq \delta(v, u)$ at all times, and if equality holds at some point, it continues to hold thereafter.

\textbf{Pf:} By induction on the number of \textsc{relax} operations.

\underline{Basis Case:} When 0 RELAXes have been done:

\begin{align*}
    d[v] &= \infty \quad \forall v \in V - \{v\} \\
    \geq \delta(v, u) \quad \forall v \in V - \{v\}
\end{align*}

[This is true even if $\delta(v, u) = \infty$ by assumption.]

\begin{align*}
    d[v] &= 0 \quad \text{and this is one way of going from } v \text{ to itself} \\
    \delta(v, v) \quad \text{is the best way. (May be } \leq 0 \text{)}
\end{align*}

$\therefore d[v] \geq \delta(v, v)$.

Hence, statement in Lemma holds initially.

Now suppose by I.H., it holds up to some point and we then \textsc{relax} $(u, v, w)$. 
Only $d[v]$ can change. Hence results holds for $x \neq v$

$$d[\hat{u}] \leq d[u]+w(u,v)$$

before

$$d[v] \geq 8/(s,v)$$

I. H

$$d[u]+w(u,v) \geq 8/(s,u)+w(u,v) \geq 8/(s,v)$$

Lemma 24.10
Triangle Ineq.

Hence the first part follows.

After equality is achieved, $d[v]$ can not go down any further (RELAX can only make it go down) and hence equality is maintained thereafter.

**Lemma 24.14 (Convergence Property)**

Let a shortest path from $s$ to $v$ look like

\[ \delta(s,u) \xrightarrow{k} (\text{Lemma } 24.1) \]

Suppose at some point $d[u] = \delta(s,u)$

Now we RELAX $(u,v)$. Then $d[u] = \delta(s,v)$

Proof on next page.
Proof Lemma 24.14

\[ d[v] \leq d[u]+w(u,v) \]
\[ = \delta(8,u) + w(u,v) \]
\[ = \delta(8,v) \quad \text{(Look at the picture on previous page)} \]

But Lemma 24.11 says \[ d[v] \geq \delta(8,v) \]

\[ \therefore \text{Combining, we get } d[v] = \delta(8,v) \]

Lemma 24.15 (Path Relaxation Property)

Suppose a SP from \( s \) to \( v \) looks like:

\[ s \longrightarrow \delta(8,v) \longrightarrow v \]

\[ k \leq \delta(8,u) \]

\[ k \leq \delta(8,v) \]

\[ k \leq \delta(8,v_{k+1}) \]

Suppose we perform RELAX operations in the order:

\[ (s,v_1) \longrightarrow (v_1,v_2) \longrightarrow \ldots \longrightarrow (v_{k+1},v_k) \]

Where \( \ldots \) are some other RELAXes.
then after RELAX \((V_{i-1}, V_i)\), \(d_{UV_J} = \delta_{V,J} \) \((6)\)

\(i = 1, \ldots, k-1; \quad |k| \leq |V| - 1\)

Since without negative cycle, \(E\) a SP which
which we assume
has no node, repeat (Simple path)

\[ Pf \]
Assuming no negative cycles (without this
Assumption, we may not have a SP at all)
\[ d_{E,J} = \delta_{E,J} = 0. \]
Hence \(d_{E,J}\) is correct after INITIALIZE
and before any RELAX operation is performed.

Hence by Convergence property (Lemma 24.14)
before \((s, u_1)\) is RELAXed, \(d_{UV_J}\) is
Correct and remains Correct. Now

before after \((V_1, V_2)\) is RELAXed, \(d_{UV_J}\) is cor-
d\((V_1, V_2)\) is correct. Now again
\(\delta_{E,J} \) (E(8,41))

Using Convergence property, \(d_{E,J}\) is
Correct after RELAX \( (V_1, V_2, W(V_1, V_2)) \)
And so on.

Since path is Simple, after \((k-1)\) RELAXes,
of the type \((V_{i-1}, V_i)\), all are Correct.

This is a Proof by Induction.
Now we will use these results to prove Correctness of DAG Algorithm, Bellman-Ford Algorithm, and Dijkstra Algorithm in this order.

Proof of Correctness
Correctness of DAG algorithm

Recall in this algorithm, we first INITIALIZE
And then RELAX edges going out of nodes taken in topological order. The main fact

About topological order in this directed graph:
If there is any edge \((i, j)\) then \(i\) occurs before \(j\) in topological order.

Since the graph is acyclic, all paths are Simple (no node repeats). The number of such paths from \(s\) to \(v\) is finite (consider all possible sets of intermediate nodes in all possible orders)
Hence, the shortest of these simple paths is indeed the overall shortest path
for any pair of nodes. Hence, for all \(v\) that can be reached from \(s\), there exists a SP (which is Simple
Let a SP from $s$ to $v$ look like

$\begin{align*}
 & S \\
 & u_1 \\
 & u_2 \\
 & \vdots \\
 & u_{k-1} \\
 & v \\
\end{align*}$

Hence, in the topological order $\{1, \ldots, \ell\}$ occurs before $v_i$.

In the beginning (before any RELAX) $d[\{s\}]=0$ is correct. By convergence,

when $(S, u_1)$ is RELAXed, $d[\{u_2\}]$ is correct. Hence after $(u_1, u_2)$ is RELAXed $d[\{u_2\}]$ is correct. Recall $u_i$ is taken up after $S$. In general $v_i$ is taken up some time after $v_{i-1}$.

Hence the Alg. correctly computes $d[\{v\}] = \delta(s, v)$ for all $v$.

This completes proof of correctness for Dijkstra algorithm.

Incidentally, complexity of DFS: $\Theta(1E+1V)$

# of RELAX: $\Theta(1E)$

Work/RELAX: $\Theta(E)$

Fastest of 3 algorithms.