Dijkstra's Algorithm

\[ \text{label} = \text{distance} \]
\[ \text{label}(a) = 0 \]

Start with a:
\[ \text{label}(a) + w(a, b) = 10 < \text{label}(b) = \infty \]
\[ \text{label}(a) + w(a, f) = 9 < \text{label}(f) = \infty \]
\[ \text{label}(a) + w(a, e) = 7 < \text{label}(e) = \infty \]

permanent, as now \( \text{label}(e) \) is the smallest.
Dijkstra's Algorithm

Continue:

\[ \text{label}(e) + w(e, d) = 7 + 15 = 22 \neq \text{label}(d) = \infty \]

\[ \text{label}(e) + w(e, f) = 17 > \text{label}(f) = 9 \]

Do not update

Next, label(f) is the smallest:

\[ \text{label}(f) + w(f, b) = 9 + 2 = 11 < \text{label}(b) = 20 \]

Next, label(b) is the minimum:

\[ \text{label}(b) + w(b, c) = 19 + 9 = 28 < \text{label}(c) = \infty \]

Next, label(d) is the minimum:

\[ \text{label}(d) + w(d, c) = 20 + 6 = 26 \geq \text{label}(c) = 20 \]

Do not update
Dijkstra's Algorithm

distance from A to E is 24.

Here we put a slash '/' to denote that the distance was updated.
Dijkstra's correctness

Use induction:

$u$ is "discovered" by $v$. That is assume $\text{label}(v) + w(v, u)$ is the smallest among all $v$.

The fact that $\text{label}(u)$ is the shortest path length from $s$ to $u$ is because if we "break" the path at any intermediate vertex $w$

We must have that both these paths $sww, wvu$ are of shortest length by induction.
Finding the shortest path

Here, ties are chosen arbitrarily.
According to our choice, the shortest path from A to E is \text{AHGIFE}.
More possibilities are: \text{AHGICOE},
\text{AHGIJDE}.

I recall that we put an arrow when a vertex becomes permanent. The arrow points to the "previous" vertex that discovers it.
Find shorter paths, etc.

shortest path from LAX to JFK is:

LAX - ORF - ORD - JFK.

Length: 2777.

shortest path from LAX to BOS:

LAX - ORF - ORD - BOS.

Length: 2904
Floyd-Warshall Algorithm

This is a recursive algorithm, where we recurse on the set of intermediate vertices on the shortest path.

This in fact uses dynamic programming:

A form of recursion, where we can store values for the solutions of subproblems in a matrix $M_{k}[i,j] = $ shortest path length from $i$ to $j$ using only $V_i, V_2, \ldots, V_k$ as intermediate vertices.

**Observation**: When all weights are positive, each $V_i$, $1 \leq i \leq k$, appears at most once!
Otherwise, if some $v_i$ appears at least twice, then the "path" from $v_i$ to $v_j$ contains a cycle, but then this is not the shortest path. Cycle only increases the total weight on a path.}

**Recursion:**

- Using only $v_i, \ldots, v_{k-1}$
- Using only $v_i, \ldots, v_k$
The key observation is that we can make local improvements with:

\[ M_k(i,j) = \min \left\{ M_{k-1}(i,k), \right\} \]

\[ \{ M_{k-1}(i,k) + M_{k-1}(k,j) \} \]

path relaxation.

But we can accelerate the computation of \( M_k(i,j) \) by observing that

\[ M_k(i,j) = \min \left\{ M_{k-1}(i,j), \right\} \]

\[ \{ M_{k-1}(i,k) + M_{k-1}(k,j) \} \]

This is \( O(1) \) to compute each entry \( M_k(i,j) \), while \( \oplus \) takes \( O(n) \).
Floyd-Warshall Algorithm

\[ M_0 \]

\[
\begin{array}{cccc}
    & v_1 & v_2 & v_3 & v_4 \\
v_1 & 0 & 2 & 1 & 4 \\
v_2 & 0 & 3 & 3 & 0 \\
v_3 & 0 & 0 & 1 & 0 \\
v_4 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[ M_1 \]

\[
\begin{array}{cccc}
    & v_1 & v_2 & v_3 & v_4 \\
v_1 & 0 & 2 & 1 & 4 \\
v_2 & 0 & 2 & 3 & 0 \\
v_3 & 0 & 0 & 1 & 0 \\
v_4 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[ M_2 \]

\[
\begin{array}{cccc}
    & v_1 & v_2 & v_3 & v_4 \\
v_1 & 0 & 2 & 1 & 4 \\
v_2 & 0 & 3 & 3 & 0 \\
v_3 & 0 & 0 & 1 & 0 \\
v_4 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[ M_3 \]

\[
\begin{array}{cccc}
    & v_1 & v_2 & v_3 & v_4 \\
v_1 & 0 & 2 & 1 & 2 \\
v_2 & 0 & 3 & 3 & 0 \\
v_3 & 0 & 0 & 1 & 0 \\
v_4 & 0 & 0 & 0 & 0 \\
\end{array}
\]