0. **Recursive structure.** Consider all possible shortest paths from u to v. Each path contains a last edge (t,v) for some $t \in V - \{v\}$, such that the part of the path from u to t is a shortest path (including the degenerate case when $t = u$). This means that the number of distinct shortest paths from u to v is equal to the number of distinct shortest paths from u to t, summed over all of the last vertices t on a shortest u–v path.

1. **Array definition.** We use two separate, parallel arrays.
   - $L[k,u,v]$ = minimum weight of any path from u to v with length at most k (where "length" = number of edges), for $k,u,v \in \{1,\dots,n\}$. (This is the same as the array used to solve the problem in week 4's tutorial.)
   - $N[k,u,v]$ = number of distinct paths from u to v with length at most k and total weight equal to $L[k,u,v]$, for $k,u,v \in \{1,\dots,n\}$.

2. **Recurrence relation.** Based on the recursive structure of the problem, as we saw in week 4's tutorial,

$$L[1,u,v] = \begin{cases} 0 & \text{if } u = v, \\ w(u,v) & \text{if } u \neq v \wedge (u,v) \in E, \\ \infty & \text{if } u \neq v \wedge (u,v) \notin E. \end{cases}$$

$$L[k,u,v] = \min\left\{L[k-1,u,v], L[k-1,u,t] + w(t,v) : (t,v) \in E\right\}, \quad \text{for } k = 2,\dots,n.$$

$$N[1,u,v] = \begin{cases} 1 & \text{if } u = v, \\ 1 & \text{if } u \neq v \wedge (u,v) \in E, \\ 0 & \text{if } u \neq v \wedge (u,v) \notin E. \end{cases}$$

$$N[k,u,v] = \sum_{\substack{(t,v)\in E \text{ such that} \\ L[k-1,u,t]+w(t,v)=L[k,u,v]}} N[k-1,u,t], \quad \text{for } k = 2,\dots,n.$$

   $N[k,u,v]$ is the sum of $N[k-1,u,t]$ over every t such that $L[k,u,v] = L[k-1,u,t] + w(t,v)$. *This is the case whether or not $L[k,u,v] = L[k-1,u,v]$ because paths of length strictly less than $k-1$ are already counted in $N[k-1,u,t]$ (over the possible values of t).*

3. **Iterative algorithm.**

```
# Initialization.
for u = 1,...,n:
    for v = 1,...,n:
        if u = v:
            L[1,u,v] ← 0
            N[1,u,v] ← 1
        elif (u,v) ∈ E:
            L[1,u,v] ← w(u,v)
            N[1,u,v] ← 1
        else:
            L[1,u,v] ← ∞
            N[1,u,v] ← 0
```

```
# Main loop.
for k = 2,...,n:
    for u = 1,...,n:
        for v = 1,...,n:
            L[k,u,v] ← L[k−1,u,v]
            N[k,u,v] ← 0
            for (t,v) ∈ E:
                if L[k−1,u,t] + w(t,v) < L[k,u,v]:
                    L[k,u,v] ← L[k−1,u,t] + w(t,v)
                    N[k,u,v] ← N[k−1,u,t]
                elif L[k−1,u,t] + w(t,v) = L[k,u,v]:
                    N[k,u,v] ← N[k,u,v] + N[k−1,u,t]
```

4. **Optimum solution.** $N[n,u,v]$ contains the total number of shortest paths from u to v, for every $u,v \in V$—there is no addition information to compute. Worst-case runtime is $\Theta(n^4)$.