The **shortest path** between two vertices in a weighted graph is the path of least weight. (If the graph is not weighted, it’s the path with the fewest edges.)

Let \( G = (V, E) \) be a graph with weight function \( w \) and particular vertices \( s, t \in V \).

**Dijkstra’s Shortest Path Algorithm**

**Step 1:** Assign \( s \) the label \((-, 0)\); \( s \rightarrow s : (-, 0) \).

**Step 2:** Until \( t \) is labeled or no further labels can be assigned, do the following:

1. **Step 2a:** For each labeled vertex \( u : (x, d) \) and each unlabeled vertex \( v \) adjacent to \( u \), compute \( d + w(e) \) for \( e = uv \).
2. **Step 2b:** Find the minimum value \( d' \) over all values \( d + w(e) \) from Step 2a.
3. **Step 2c:** For each \( u \) and \( v \) with \( d + w(e) = d' \), assign the label \((u, d')\) to \( v \). If \( v \) has more than one possible label from different \( u \), pick any \((u, d')\).

Complexity is \( \mathcal{O}(n^3) \) where \( n = |V| \), but can be modified to an \( \mathcal{O}(n^2) \) algorithm.

**Dijkstra’s Shortest Path Algorithm (IMPROVED)**

**Step 1:** Assign \( s \) the permanent label 0. Assign every other vertex a temporary label of \( \infty \).

**Step 2:** Until \( t \) has a permanent label or no temporary labels are changed, do the following:

1. **Step 2a:** Let \( v \) be the vertex with the most recent permanent label \( d \). For each vertex \( u \) adjacent to \( v \) with temporary label \( t \), if \( d + w(vu) < t \), then update temporary label of \( u \) to \( d + w(vu) \).
2. **Step 2b:** Find the vertex \( w \) with the smallest temporary label \( t \), and make its label \( t \) a permanent one. (Break ties arbitrarily).

All shortest paths between pairs of vertices in a graph can be found in time \( \mathcal{O}(n^4) \) (resp. \( \mathcal{O}(n^3) \)) by running Dijkstra’s (improved) algorithm until all vertices have a (permanent) label for each possible start vertex \( s \).

**Floyd–Warshall Algorithm ALL PAIRS SHORTEST PATH**

**Step 0:** For vertices \( v_i \in V \) with \( 1 \leq i \leq n \), consider an \( n \times n \) matrix with entries \( d(i, j) \).

**Step 1:** For \( 1 \leq i \leq n \), set \( d(i,i) = 0 \). For \( i \neq j \), let \( d(i, j) = w(e) \) if \( e = v_i v_j \) is an edge in \( G \). Otherwise, set \( d(i, j) = \infty \).

**Step 2:** For \( k = 1 \) to \( n \), for \( i, j = 1 \) to \( n \), let \( d(i, j) = \min\{d(i, j), d(i, k) + d(k, k)\} \).

The final value of \( d(i, j) \) is the shortest distance from \( v_i \) to \( v_j \), and complexity is \( \mathcal{O}(n^3) \).