Dijkstra's Algorithm is an algorithm to find the shortest path from a starting node to all other nodes in a weighted graph.

Here's the algorithm:

Variables:

1. A weighted graph G
2. int D[SIZE] – to store the current shortest distance from the starting node to another node at index i.
3. bool S[SIZE] – to keep track of which node has been included in the shortest path solution.

Initialization:

1. Initialize D[] to max value INT32_MAX.
2. Initialize S[] to false;
3. S[start] = true; // the starting node is in the shortest path solution
4. D[start] = 0; // distance to itself is always 0

Helper functions:

MinimumDistance() – Find the node that is currently not in S (i.e., S[i] == false) and has the minimum distance in D (i.e., smallest value in D[i]). Return the node index.

Relax(u, v) – To relax an edge from node u to v consists of testing whether we can improve the shortest path to v found so far (i.e., D[v]) by going through u (i.e., D[u] + weight(u, v)) and, if so, updating D[v] with the new shortest path.

```
given the weight of an edge from node u to v, weight(u, v)
if (D[v] > D[u] + weight(u, v) ) {
        D[v] = D[u] + weight(u, v);      // update D[v]
}
```

Dijkstra's:

1. For all nodes in G:
    2. u = MinimumDistance()
    3. S[u] = true; // u is in the shortest path solution
    4. For each node v that is adjacent to u, and v is not already in the shortest path solution:
        5. Relax(u, v)

Termination:

The algorithm terminates when all the nodes in the graph have been processed.