Dijkstra’s Shortest Path Algorithm.

1. Find a shortest weighted path from vertex A to vertex H.

2. Find a shortest weighted path from vertex C to vertex H.

3. Describe a naive algorithm for finding a shortest weighted path between any pair of vertices.
4. Use Dijkstra’s Algorithm to find a shortest weighted path from vertex C to vertex H.

**Algorithm 3.5** A general template for Dijkstra’s algorithm.

**Input** An undirected or directed graph $G = (V, E)$ that is weighted and has no self-loops. The order of $G$ is $n > 0$. A vertex $s \in V$ from which to start the search. Vertices are numbered from 1 to $n$, i.e., $V = \{1, 2, \ldots, n\}$.

**Output** A list $D$ of distances such that $D[v]$ is the distance of a shortest path from $s$ to $v$. A list $P$ of vertex parents such that $P[v]$ is the parent of $v$, i.e., $v$ is adjacent from $P[v]$.

1. $D \leftarrow [\infty, \infty, \ldots, \infty]$ $\quad \triangleright$ $n$ copies of $\infty$
2. $D[s] \leftarrow 0$
3. $P \leftarrow []$
4. $Q \leftarrow V$ $\quad \triangleright$ list of nodes to visit
5. while length($Q$) > 0 do
6. \hspace{1em} find $v \in Q$ such that $D[v]$ is minimal
7. \hspace{1em} $Q \leftarrow \text{remove}(Q, v)$
8. \hspace{2em} for each $u \in \text{adj}(v) \cap Q$ do
9. \hspace{3em} if $D[u] > D[v] + w(vu)$ then
10. \hspace{4em} $D[u] \leftarrow D[v] + w(vu)$
11. \hspace{4em} $P[u] \leftarrow v$
12. return $(D, P)$