1. Find a shortest weighted path from vertex D to vertex F.

2. How many steps will our naive algorithm use to a shortest weighted path from vertex D to vertex F.
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: $D \leftarrow [\infty, \infty, \ldots, \infty]$ \hspace{1cm} ⊢ n copies of $\infty$
2: $D[s] \leftarrow 0$
3: $P \leftarrow []$
4: $Q \leftarrow V$ \hspace{1cm} ⊢ list of nodes to visit
5: while length($Q$) > 0 do
6: \hspace{1cm} find $v \in Q$ such that $D[v]$ is minimal
7: \hspace{1cm} $Q \leftarrow$ remove($Q, v$)
8: \hspace{1cm} for each $u \in \text{adj}(v) \cap Q$ do
9: \hspace{1cm} \hspace{1cm} if $D[u] > D[v] + w(vu)$ then
10: \hspace{1cm} \hspace{1cm} $D[u] \leftarrow D[v] + w(vu)$
11: \hspace{1cm} \hspace{1cm} $P[u] \leftarrow v$
12: return $(D, P)$

3. Use Dijkstra’s Algorithm to find a shortest weighted path from vertex C to vertex H.