Professor Gaedel has written a program that he claims implements Dijkstra’s algorithm. The program produces \( v.d \) and \( v.\pi \) for each vertex \( v \in V \). Give an \( O(V+E) \) time algorithm to check the output of the professor’s program. It should determine whether the \( d \) and \( \pi \) attributes match those of some shortest-paths tree. You may assume that all edge weights are nonnegative.

1. Verify that \( s.d = 0 \) and \( s.\pi = \text{NIL} \)
2. Verify that \( v.d = v.\pi.d + w(v.\pi, v) \) for all \( v \neq s \)
3. Verify that \( v.d = \infty \) if and only if \( v.\pi = \text{NIL} \) for all \( v \neq s \)
4. If any of above verification tests fail, declare the output to be incorrect. Otherwise, run one pass of Bellman-Ford, i.e. relax each edge \((u,v) \in E\) one time. If any values of \( v.d \) changes, then declare the output to be incorrect; otherwise, declare the output to be correct.

Let \( G = (V, E) \) be a weighted, directed graph with nonnegative weight function \( w: E \rightarrow \{0,1,\ldots,W\} \) for some nonnegative integer \( W \). Modify Dijkstra’s algorithm to compute the shortest paths from a given source vertex \( s \) in \( O(WV + E) \) time.

Dijkstra’s algorithm is given as follows:

Dijkstra(G, w, s)
1. INITIALIZE-SINGLE-SOURCE(G, s)
2. \( S \leftarrow \{ \} \)
3. \( Q \leftarrow V[G] \)
4. while \( Q \neq \{ \} \)
   a. do \( u \leftarrow \text{EXTRACT-MIN(Q)} \)
   b. \( S \leftarrow S \cup \{u\} \)
   c. For each vertex \( v \in Adj[u] \)
      i. Do RELAX(\( u, v, w \) )

The running time of Dijkstra’s algorithm depends on the implementation of the min-priority queue. In Dijkstra’s algorithm, we process those vertices closest to the source vertex first. Because each edge has at most weight \( W \), we know the maximum possible value of the longest path in the graph is \((V-1)W\). We can prioritize the vertices based on their \( d[] \) values. Remember, \( d[v] \) is the shortest path from the
source to vertex $v$.

The queue consists of $(V-1)W$ buckets. Vertex $v$ can be found in bucket $d[v]$. Since all other than the source have $d[v]$ value between 1 and $(V-1)W$, so they can be found in buckets $1,\ldots,(V-1)W$. If $s$ is the source vertex, $d[s] = 0$. So, $s$ can be found in bucket 0. INITIALIZE-SINGLE-SOURCE ensures that for all vertices $v$ other than the root, $d[v]$ is initialize to $\infty$. The final bucket holds all vertices whose $d[]$-values are infinity (all undiscovered vertices).

After initializing all of the vertices, we scan the buckets from 0 to $(V – 1)W$. When a non-empty bucket is encountered, the first vertex is removed, and all adjacent vertices are relaxed. This step is repeated until we have reached the end of the queue—in $O(WV)$ time. Since we relax a total of $E$ edges, the total running time for this algorithm is $O(VW + E)$.

24.3-10
Suppose that we are given a weighted, directed graph $G = (V, E)$ in which edges that leave the source vertex $s$ may have negative weights, all other edge weights are nonnegative, and there are no negative-weight cycles. Argue that Dijkstra’s algorithm correctly finds shortest paths from $s$ in this graph.

According to the question, we know that there is no negative weight cycle in the graph $G$ and all negative weight edges are connected to the source vertex $s$. Therefore, we just need to prove that below:

If some vertex $v \neq s$ is connected with some negative weight edge $e$, the shortest path from $s$ to $v$ must cover the negative weight edge $e$.

Next, we could apply the Theorem 24.6 and know that Dijkstra’s algorithm is still correct.

Prove: (By contradiction)

![Diagram](// Both E and E’ are negative weight edges)
Assume that the shortest path from $S$ to $V$ is $S \rightarrow V'$ instead of $S \rightarrow V$. Then we could get the equation $E' + P < E$. 

$E' + P < E$  

$\Rightarrow E + E' + P < 2E < 0 \quad // \text{negative cycle}$

Contradiction occurs.