Dijkstra's Algorithm

Dijkstra's algorithm is a greedy algorithm that finds shortest paths from a starting node to all other nodes in a weighted and/or directed network. It has a lot in common with BFS and DFS. It starts searching from the starting node, repeatedly exploring from the shortest known paths to discover more shortest paths.
We start with a network and a designated starting node, say node 0. The weights on the edges indicate length, so a shortest path is one of minimal total edge weight. The algorithm keeps track of two pieces of information for each node w: the currently known length of a shortest path, and the previous node on a path from 0 to w. The latter is all the information we need in order to reconstruct a shortest path from 0 to any node. (Why?)
At the start of our algorithm, we initialize these two lists. We start by setting the distance of 0 to 0 (since any node is distance 0 from itself) and the distance of all the other nodes to ∞, since that is the shortest known paths (having not explored anything yet). In practice, it makes more sense to set this distance to a number that is bigger than any path will be, such as the sum of all edge weights plus one. No path would ever use an edge more than once. (Why?) We just need to be careful not to output this number as a real distance, since it is only a placeholder.
The second list of information holds the previous nodes. We can initialize this to be -1, which is clearly a placeholder if we assume that nodes are labelled by indices $0, 1, \ldots, n-1$. At the end of the algorithm, if any node still has a previous node -1, we know that there is no path to it. This is of course a possibility, since the network may not be connected.
The algorithm then does a search, always exploring from an unvisited node of minimal distance. Since we always start from an unvisited node of minimal distance, we know that we have found this node's true distance, since exploring from other nodes could never reveal a shorter path. This is not a formal proof, but rather an intuition. This is a very famous algorithm and you can find many resources on more theoretical proofs. Whenever you write a function, you should always take the time to check its correctness, whether you do it yourself or use other sources. Make it a habit to try to come up with bad cases.
Dijkstra in pseudo code:

```
DIJKSTRA (N,v)
set d[v]=0 and d[i]= a large enough number everywhere else
set prev[i]=-1 for all nodes
label all nodes as ''unvisited''
while there are unvisited nodes
   let x be an unvisited node of minimal d[x]
   for all neighbors y of x
      if d[y]>d[x]+N[x,y]
         let d[y]>d[x]+N[x,y] and set prev[y]=x
   mark x as ''visited''
return d, prev
```

Exercises

- Suppose that we do not always explore from a node of minimal distance. Find an example of a small network where this leads to a failure to find shortest paths.
- Check what happens in the algorithm if the network is not connected.
- Do an analysis of the run-time of Dijkstra's algorithm, either as a function of the number of nodes or as a function of the number of edges.
- Is this a greedy algorithm?