Shortest Paths with BFS

Here is my own solution to "Quiz 7". The problem was to modify BFS so that it finds *shortest paths* to all vertices from a given starting vertex (i.e. the SSSP problem).

The key changes to the algorithm are:

1. Change the function signature to include the starting vertex. (I also renamed the algorithm.)
2. Maintain a value for "distance" of every vertex, initialized to "infinity" for all vertices.
3. Eliminate the loop in the main part, instead calling helper just one time. (NOTE: At this point, since the helper function is not recursive, there's almost no need to have a "helper" function at all. This code could all be put together in a single function. But I left it this way to preserve the underlying similarity to BFS.)
4. In the helper function:
    a. Set distance of v to be 0 (this is the starting vertex)
    b. Each time you visit a vertex (w), its distance is one step farther than the vertex you are "coming from" (Q.head)

Here is the algorithm with the locations of those changes marked:

```
------------------------------------------------------
   Algorithm Shortest_Paths_From_S(Graph G, vertex S)    [1]
   // Graph G = {V,E}
   // vertex S = starting vertex
       initialize visited to false for all vertices
       initialize distance to "infinity" for all vertices   [2]
       // call the helper function just once, with S
       bfs_helper(S)                                       [3]
   END

   function bfs_helper(Vertex v)
       visit node v
       set distance(v) = 0                                   [4a]
       initialize a queue Q
       add v to Q
       while Q is not empty
           for each w adjacent to Q.head
              if w has not been visited
                  visit node w
                  // w is one step farther away than the
                  // vertex we are currently processing
                  set distance(w) = distance(Q.head) + 1     [4b]
                  add w to Q
               endif
           endfor
           Q.dequeue()
       endwhile
   END
------------------------------------------------------
```