Dijkstra's Shortest Path Algo

<u>Version 1:</u>

Uses:	visitation table	-	can be local to function
	distance table	-	table entries each contain vertex key, cumulative distance, path distance table is returned to caller

```
void shortestpath (graph, vertex, distance table)

build visitation table for all vertices in graph
initialize all visitation table entries to not visited
position at starting vertex
create entry in distance table for starting vertex
    vertex key
    distance from start (0)
mark starting vertex as visited

for each neighbor, create entry in distance chart:
    vertex key
    distance from start
    key of starting vertex (path to this neighbor)

while (there are unvisited vertexes)
    find distance table entry with lowest distance for an unvisited vertex
    call that the current vertex
    mark current vertex as visited
    position at first neighbor
    while (more neighbors to do)
        if table entry does not exist for that neighbor
             create entry in distance table:  1-key, 2-distance from start – this is the distance from
             the current vertex chart entry plus the distance to this neighbor, 3-path - this is the key
             of the neighbor
        else
            if distance to this neighbor along this path < distance in table
            replace old distance with this one
            replace old path with this one
        endif
        position at next neighbor
    end while
end while

return to caller:  distance table.  Entries now contain:
            distance to that vertex from start
            path back to start
```

Version 2: Malik p 705:

uses:	weight table	-	2D table of vertices and edges. This is an adjacency matrix. Each cell holds an edge weight. No edge indicated by a weight of 0.
	weight found table	-	visitation table. Indexed using vertex key.
	smallest weight table	-	distance table. Indexed using vertex key. Returned to caller.
	"infinity"	-	a really large number, a number so large that it will never occur naturally during processing (DBL_MAX)

```
void shortestpath (vertex)

for ( j iterates from first to last vertex )
    smallest weight [j] = weight [vertex] [j]
end for

initialize weight found table to all false
mark weight found [vertex] = true
set smallest weight [vertex] = 0

for ( i iterates from first to last vertex )
    min weight = infinity
    for ( j iterates from first to last vertex )          find distance table entry with lowest distance
        if (vertex not visited)
            if ( smallest weight [j] < min weight )
                v = j
                min weight = smallest weight [v]
            endif
        endif
    end for
    weight found [v] = true                               lowest distance is the vth entry

    for ( j iterates from first to last vertex )          go thru distance table
        if ( vertex not visited )
            if ( min weight + weight [v] [j] < smallest weight [j] )      a new shorter distance?
                smallest weight [j] = min weight + weight [v] [j]         yes, update
            endif                                                         else do nothing
        endif
    end for
end for
```