Dijkstra's Algorithm

Dijkstra's algorithm is an algorithm we can use to find shortest distances or minimum costs depending on what is represented in a graph. The steps to this algorithm are as follows:

Algorithm rundown :

Here’s how the algorithm is implemented:

1. Mark all nodes as *unvisited*.
2. Mark the selected **initial** node with a *current* distance of **0** and the rest with **infinity**.
3. Set the **initial** *node* as **current** *node*.
4. For the **current** *node*, consider all of its **unvisited** neighbors and calculate their distances by adding the *current* distance of **current** *node* to the *weight* of the *edge* connecting **neighbor** *node* and **current** node.
5. Compare the newly calculated distance to the current distance assigned to the **neighboring** *node* and set is as the **new** *current* distance of **neighboring** *node*.
6. When done considering all of the **unvisited** *neighbors* of the **current** *node*, mark the **current** node as **visited**.
7. If the **destination** *node* has been marked **visited** then stop. The algorithm has finished.
8. Otherwise, select the **unvisited** node that is marked with the **smallest** distance, set it as the new **current node**, and go back to **step 4**.

Pseudo Code :

```
function Dijkstra(Graph, source):

  for each vertex v in Graph: // Initialization
    dist[v] := infinity // initial distance from source to vertex v is set to infinite
    previous[v] := undefined  // Previous node in optimal path from source

  dist[source] := 0 // Distance from source to source

  Q := the set of all nodes in Graph
```

```
  while Q is not empty: // main loop

    u := node in Q with smallest dist[ ]
    remove u from Q

    for each neighbor v of u: // where v has not yet been removed from Q.
      alt := dist[u] + dist_between(u, v)
      if alt < dist[v]  // Relax (u,v)
        dist[v] := alt
        previous[v] := u

return previous[ ]
```