Dijkstra's Algorithm is helpful to find the shortest path between the nodes in a graph. The graph we have, should be weighted, i.e., the edges in the graph should represent a connection between the node, like distance, time, etc. This Algorithm is generally used to find the shortest distance from the source node to the target node, thereby producing a shortest-path tree. It differs from the minimum spanning tree as the shortest distance between two vertices may not be included in all the vertices of the graph.

The Algorithm needs a source node, the node with respect to which we want to analyse the shortest distance. During the execution, the Algorithm keeps track of the minimum distance from each node to the source node. The value is updated if it finds a shorter path.The Algorithm then finds the minimum distance between the current node and the other neighbouring nodes, marks them as 'visited' and adds it to the path. This process is recursively repeated till we reach the final cell giving us the shortest possible distance between targeted cells.

For the execution, the source node is designated the value '0', whereas the rest of the node initially has the value ∞ (infinity). At this point, all nodes are considered 'unvisited'. The neighbouring nodes are then analysed to get the distance between two nodes. The adjacent node with minimum distance is hence considered for the path. The process is continued till we have no 'unvisited' nodes left.

We need to keep in mind that Dijkstra's Algorithm only works effectively with graphs that have positive weights. As we have seen, for a 'visited' node, the current path to that node is marked as the shortest path to reach that node. Negative weight can alter this, if the total weight gets a lower value after this step has occurred.

Dijkstra's Algorithm can be used in GPS devices to find the shortest path between two locations. It is often also used as a routing protocol required by the routers to update their forwarding table.

Sakshi Gupta
2020A8PS2036G