**Aim:** To implement Dijkstra’s Shortest Path Routing Algorithm

**Theory :**

The main idea in shortest path routing is to build a graph of the subnet with each node of the graph representing a router and each edge representing a communication line (often called a link). To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

One-way of measuring path length is the number of hops. Another metric is the geographic distance in kilometers. With this graph labeling, the shortest path is the fastest path rather than the path with the fewest arcs or kilometers.

In the general case, the labels on the arcs could be average computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, and other factors. By changing the weighting function, the algorithm would then compute the “shortest” path measured according to any one of a number of criteria or to a combination of criteria.

Each node is labeled with its distance from the source node along the best-known path. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either tentative or permanent. Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

**Algorithm:**

Given a graph, G, with edges E of the form (v1, v2) and vertices V, and a source vertex, s

dist : array of distances from the source to each vertex prev : array of pointers to preceding vertices

i : loop index

F : list of finished vertices

U : list or heap unfinished vertices

/\* Initialization: set every distance to INFINITY until we discover a path \*/

for i = 0 to |V| - 1

dist[i] = INFINITY prev[i] = NULL

end

/\* The distance from the source to the source is defined to be zero \*/

dist[s] = 0

/\* This loop corresponds to sending out the explorers walking the paths, where the step of picking "the vertex, v, with the shortest path to s" corresponds to an explorer arriving at an unexplored vertex \*/

while(F is missing a vertex)

pick the vertex, v, in U with the shortest path to s

add v to F

for each edge of v, (v1, v2)

/\* The next step is sometimes given the confusing name "relaxation"\*/

if(dist[v1] + length(v1, v2) < dist[v2])

dist[v2] = dist[v1] + length(v1, v2)

prev[v2] = v1

possibly update U, depending on implementation

end if

end for

end while

|  |
| --- |
| **Conclusion:** |

|  |
| --- |
| **Post Lab:**  1. What is routing in network? What are the different types of routing algorithms?  2. What is static routing? What are advantages of dynamic routing?  3. Differentiate between static and dynamic routing or adaptive and non-adaptive routing. |

Signature of Faculty Date of Completion