**Experiment No: 4**

**SINGLE SOURCE SHORTEST PATH**

**Aim:** To implement single source shortest path using greedy approach.

**Theory:**

Greedy algorithms build a solution part by part, choosing the next part in such a way, that it gives an immediate benefit. This approach never reconsiders the choices taken previously. This approach is mainly used to solve optimization problems.

Greedy method is easy to implement and quite efficient in most of the cases. Hence, we can say that Greedy algorithm is an algorithmic paradigm based on heuristic that follows local optimal choice at

**Single source shortest path:**

It is a greedy algorithm that solves the single-source shortest path problem for a directed graph G = (V, E) with nonnegative edge weights, i.e., w (u, v) ≥ 0 for each edge (u, v) ∈ E.

Dijkstra's Algorithm maintains a set S of vertices whose final shortest - path weights from the source s have already been determined. That's for all vertices v ∈ S; we have d [v] = δ (s, v). The algorithm repeatedly selects the vertex u ∈ V - S with the minimum shortest - path estimate, insert u into S and relaxes all edges leaving u.

Because it always chooses the "lightest" or "closest" vertex in V - S to insert into set S, it is called as the greedy strategy.

**Algorithm-**

**Input Data-**

* Cost Adjacency Matrix for Graph G, say cost
* Source vertex, say s

**Output Data-**

* Spanning tree having shortest path from s to all other vertices in G

### **Following are the steps used for finding the solution-**

**Step 1**; Set dist[s]=0, S=ϕ // s is the source vertex and S is a 1-D array having all the visited vertices

**Step 2**: For all nodes v except s, set dist[v]= ∞

**Step 3**: Find q not in S such that dist[q] is minimum // vertex q should not be visited

**Step 4**: Add q to S // add vertex q to S since it has now been visited

**Step 5**: update dist[r] for all r adjacent to q such that r is not in S //vertex r should not be visited dist[r]=min(dist[r], dist[q]+cost[q][r])

**Step 6**: Repeat Steps 3 to 5 until all the nodes are in S // repeat till all the vertices have been visited

**Step 7**: Print array dist having shortest path from the source vertex u to all other vertices

**Step 8**: Exit

**Program and Output:**

**Conclusion:**

Single source shortest path using greedy approach was studied and implemented successfully.