Dijkstra algorithm (using Greedy):

Here: d - distance, c- cost,u - initial dist, v- final dist

Algorithm:

1. Initialize Data Structures:

- Create an empty set `sptSet` to keep track of vertices included in the shortest path tree.
- Initialize an array `dist[]` to store distance values from the source vertex.
- Set all distances to INFINITE initially, except for the source vertex, which is set to 0.

2. Main Loop:

- While the `sptSet` does not include all vertices:
- Select a vertex `u` not in `sptSet` with the minimum distance value.
- Include `u` in `sptSet`.
- Update the distance values of all adjacent vertices of `u`.

3. Update Distance Values:

- For each adjacent vertex `v` of `u`:
- If the sum of the distance value of `u` from the source and the weight of edge `u+v` is less than the current distance value of `v`, update the distance value of `v` to this sum.

Code:

#include <stdio.h>

#include <float.h>

#define MAX_SIZE 100

void ShortestPaths(int n, float cost[MAX_SIZE][MAX_SIZE], int src, float dist[MAX_SIZE]) { int visited[MAX_SIZE]; int i, num, u, v;

```
for (i = 1; i <= n; ++i)
{
    dist[i] = FLT_MAX; // Initially setting all distances to infinite
    visited[i] = 0;
}
dist[src] = 0.0; // Distance from source to itself is 0
visited[src] = 1;

for (num = 2; num <= n; num++)
{
    float min_dist = FLT_MAX;
    for (i = 1; i <= n; i++) {
        if (!visited[i] && dist[i] < min_dist)
            {
            min_dist = dist[i];
            u = i;
        }
    }
    visited[u] = 1;
    for (v = 1; v <= n; v++) {
        if (!visited[v] && (dist[u] + cost[u][v]) < dist[v])
            {
            dist[v] = dist[u] + cost[u][v];
        }
    }
}
```

}

int main()

{ int n, i, j, src; float cost[MAX_SIZE][MAX_SIZE], dist[MAX_SIZE];

```
printf("Enter number of vertices: ");
scanf("%d", &n);

printf("Enter adjacency cost matrix (1-based index):\n");
for (i = 1; i <= n; i++)
{
    for (j = 1; j <= n; j++)
      {
        scanf("%f", & cost[i][j]);
    }
}

printf("Enter source vertex: ");
scanf("%d", &src);

ShortestPaths(n, cost, src, dist);

printf("Shortest distances from vertex %d:\n", src);
```

```c
for (i = 1; i <= n; i++)
{
    printf("Vertex %d: %.2f\n", i, dist[i]);
}

return 0;

}
```