23. Dijkstra's Algorithm (Shortest Path)

Aim:
To find shortest paths from a source vertex to all other vertices using Dijkstra's algorithm.

Algorithm:

1. Initialize distances of all vertices as infinity except source (0).
2. Mark all vertices unvisited.
3. Pick the unvisited vertex with the smallest distance.
4. Update the distance to its adjacent vertices.
5. Repeat until all vertices are visited.

CODE:

```
#include <stdio.h>
#define INF 9999
#define MAX 20

void dijkstra(int G[MAX][MAX], int n, int start) {
   int cost[MAX][MAX], dist[MAX], visited[MAX], count, minDist, nextNode;

   for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
         cost[i][j] = (G[i][j] == 0) ? INF : G[i][j];

   for (int i = 0; i < n; i++) {
      dist[i] = cost[start][i];
      visited[i] = 0;
   }
   dist[start] = 0;
   visited[start] = 1;
   count = 1;

   while (count < n - 1) {
```

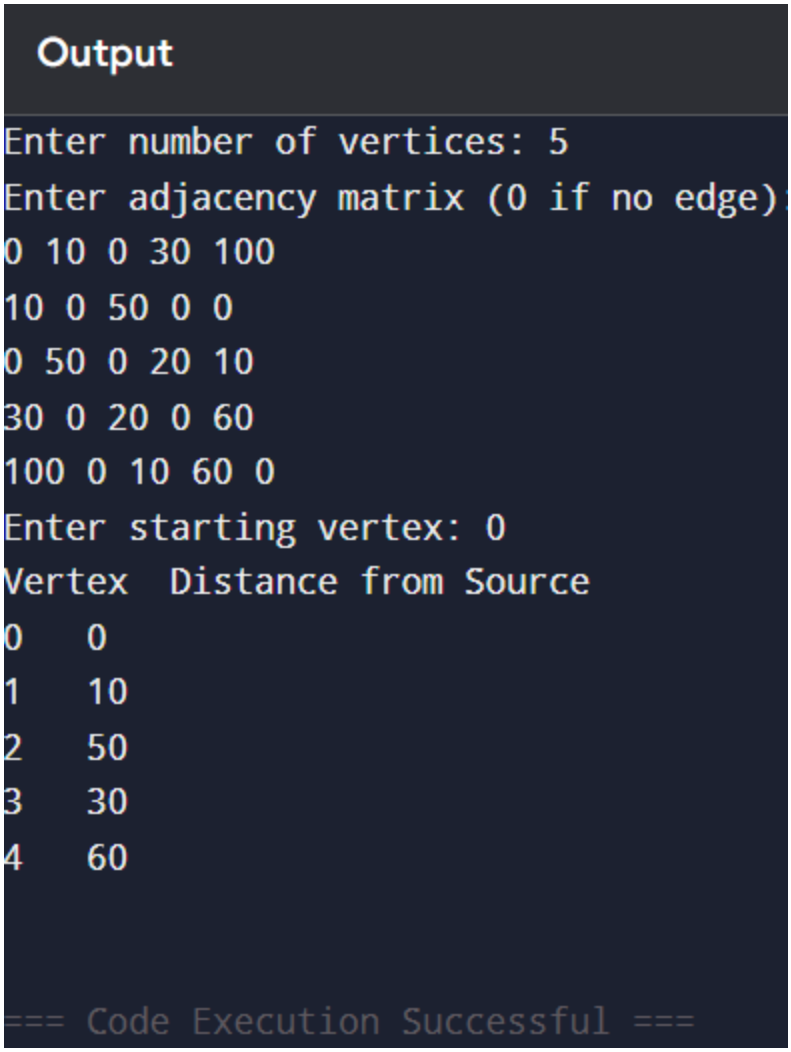
```
        minDist = INF;
        for (int i = 0; i < n; i++)
            if (dist[i] < minDist && !visited[i]) {
                minDist = dist[i];
                nextNode = i;
            }

        visited[nextNode] = 1;
        for (int i = 0; i < n; i++)
            if (!visited[i] && minDist + cost[nextNode][i] < dist[i])
                dist[i] = minDist + cost[nextNode][i];
        count++;
    }

    printf("Vertex\tDistance from Source\n");
    for (int i = 0; i < n; i++)
        printf("%d\t%d\n", i, dist[i]);
}

int main() {
    int n, G[MAX][MAX], start;
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix (0 if no edge):\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &G[i][j]);
    printf("Enter starting vertex: ");
    scanf("%d", &start);

    dijkstra(G, n, start);
    return 0;
}
```



RESULT:

The program successfully executed and displayed the Dijkstra's Algorithm.