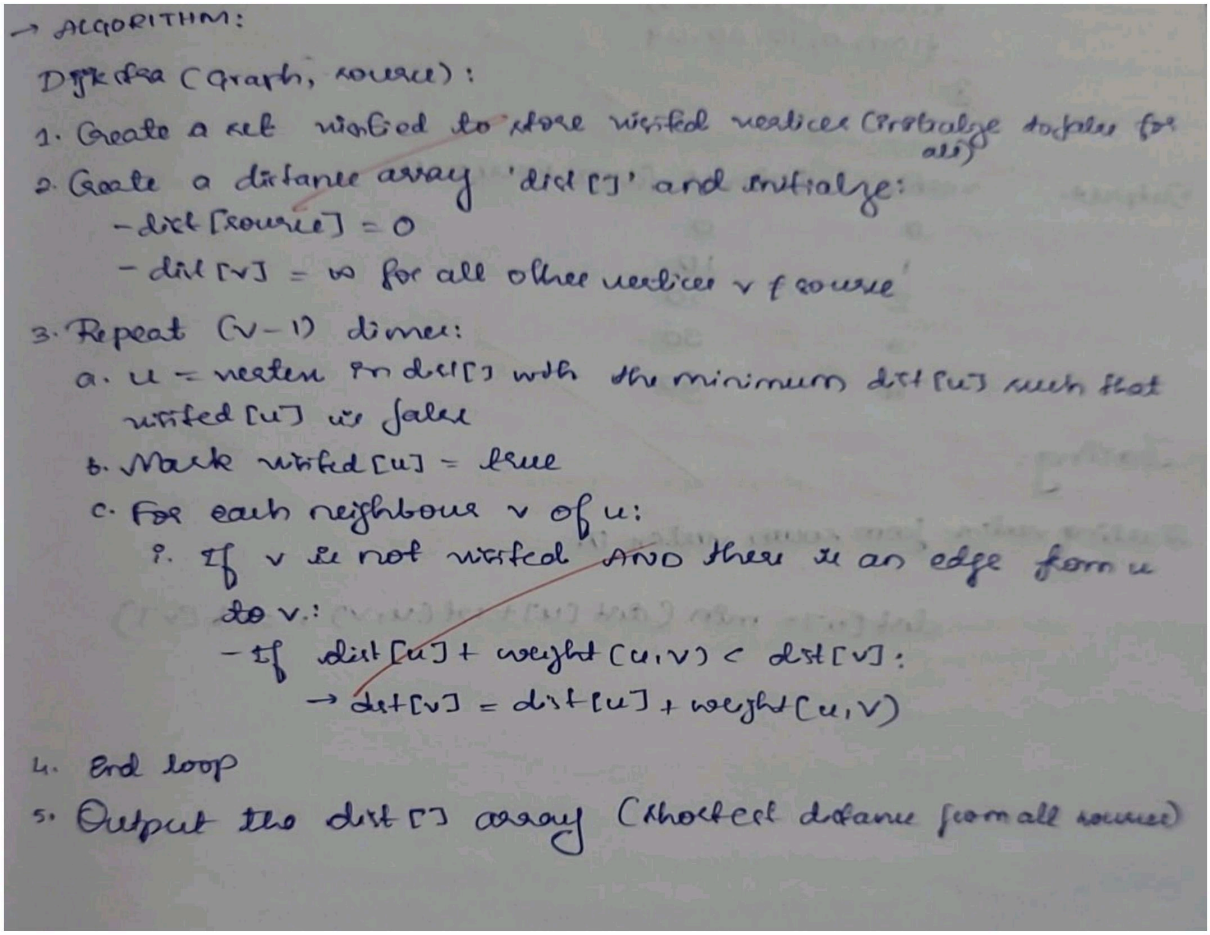
PROGRAM 8

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

ALGORITHM



CODE

```
#include <stdio.h>
#include <limits.h>
#define V 100 // max number of vertices

// Find vertex with minimum distance not yet processed
int minDistance(int dist[], int visited[], int n) {
    int min = INT_MAX, min_index = -1;
    for (int v = 0; v < n; v++) {
        if (!visited[v] && dist[v] <= min) {
```

```
            min = dist[v], min_index = v;
        }
    }
    return min_index;
}

void dijkstra(int graph[V][V], int src, int n) {
    int dist[V];      // Shortest distances
    int visited[V];   // True if vertex is included in shortest path tree

    // Initialize distances and visited
    for (int i = 0; i < n; i++) {
        dist[i] = INT_MAX;
        visited[i] = 0;
    }

    dist[src] = 0; // Distance to itself is 0

    // Find shortest path for all vertices
    for (int count = 0; count < n - 1; count++) {
        int u = minDistance(dist, visited, n);
        visited[u] = 1;

        // Update distances of adjacent vertices
        for (int v = 0; v < n; v++) {
            if (!visited[v] && graph[u][v] && dist[u] != INT_MAX &&
                dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
```

```
    }

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

// Driver code
int main() {
    int n;
    printf("Enter number of vertices: ");
    scanf("%d", &n);

    int graph[V][V];

    printf("Enter the adjacency matrix (use 0 if no edge):\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);

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

    dijkstra(graph, src, n);

    return 0;
}
```

OUTPUT

```
Enter number of vertices: 5
Enter the adjacency matrix (use 0 if no edge
    ):
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0
Enter source vertex: 0
Vertex    Distance from Source
0         0
1         10
2         50
3         30
4         60
```

TRACING