Experiment No.6

Shortest Paths using Dijkstra's Algorithm

Reg.No.24141028

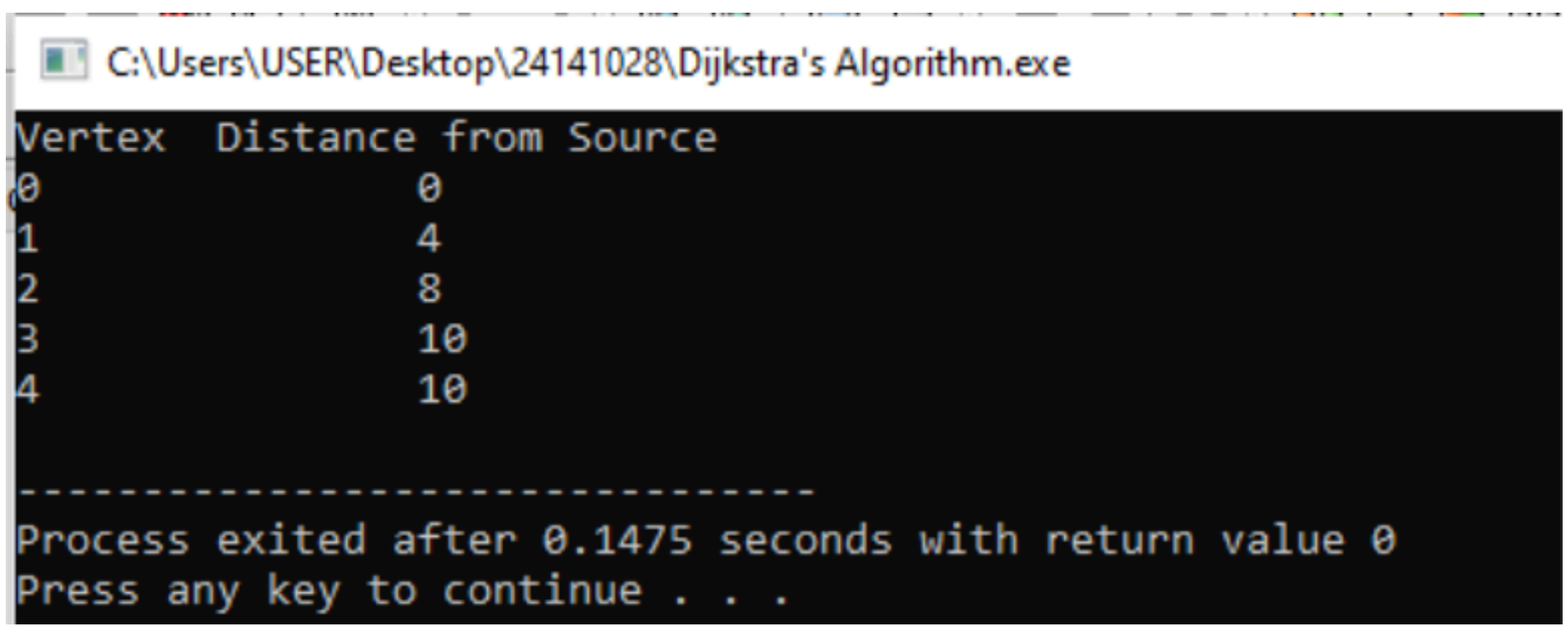
Program :

```
#include <stdio.h>

#define V 5

#define MAX 9999

// Function to find the vertex with minimum distance value

int minDistance(int dist[], int visited[]) {

        int min = MAX, min_index = -1;

        int v;

        for (v = 0; v < V; v++) {

                if (!visited[v] && dist[v] < min) {

                        min = dist[v];

                        min_index = v;

                }

        }

        return min_index;

}

// Dijkstra's algorithm implementation

void dijkstra(int graph[V][V], int src) {
```

```
int dist[V];

int visited[V];

int i, count, u, v;

// Initialize distances and visited[] array

 for (i = 0; i < V; i++) {

        dist[i] = MAX;

        visited[i] = 0;

        }

         dist[src] = 0;

        // Find shortest path for all vertices

        for (count = 0; count < V - 1; count++) {

                u = minDistance(dist, visited);

                visited[u] = 1;

                  for (v = 0; v < V; v++) {

                        if (!visited[v] && graph[u][v] && dist[u] + graph[u][v] <
dist[v]) {

                                dist[v] = dist[u] + graph[u][v];

                        }

                }

        }

// Print the result

printf("Vertex\tDistance from Source\n");
```

```
        for (i = 0; i < V; i++)
                printf("%d\t\t%d\n", i, dist[i]);
}
int main() {
        int graph[V][V] = {
                {0, 4, 8, 0, 0},
                {4, 0, 0, 0, 6},
                {8, 0, 0, 2, 0},
                {0, 0, 2, 0, 10},
                {0, 6, 0, 10, 0}
        };
         dijkstra(graph, 0);
        return 0;
}
```

Output :

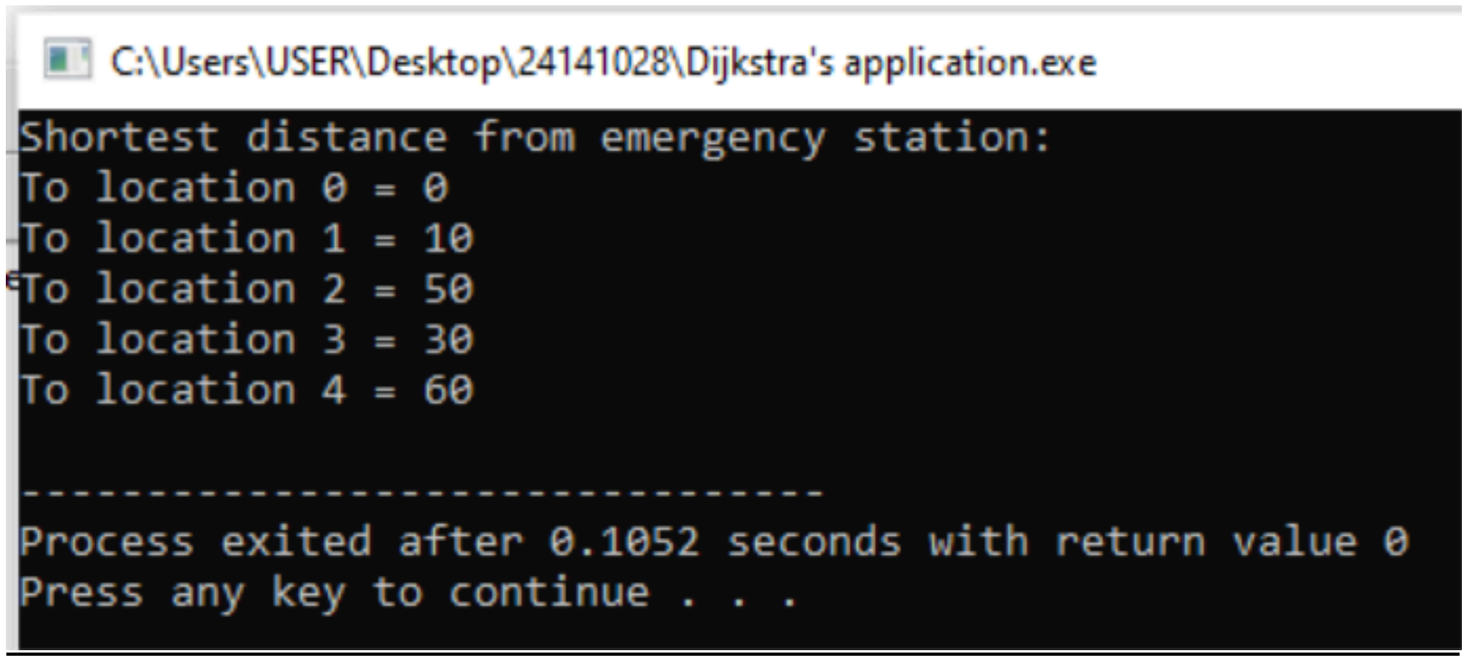


Application : Emergency Vehicle Route Optimization

```
#include <stdio.h>
#define V 5
#define INF 9999
int minDistance(int dist[], int visited[]) {
        int min = INF, min_index = -1;
        for (int v = 0; v < V; 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 dist[V], visited[V];
        for (int i = 0; i < V; i++) {
                dist[i] = INF;
                visited[i] = 0;
}
dist[src] = 0;
for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, visited);
        visited[u] = 1;
        for (int v = 0; v < V; v++) {
```

```
                if (!visited[v] && graph[u][v] && dist[u] + graph[u][v] <
dist[v])
                    dist[v] = dist[u] + graph[u][v];
            }
        }
        printf("Shortest distance from emergency station:\n");
        for (int i = 0; i < V; i++)
            printf("To location %d = %d\n", i, dist[i]);
}
int main() {
        int graph[V][V] = {
            {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}
        };
        int source = 0; // Emergency station location
        dijkstra(graph, source);
        return 0;
}
```

Output :



Time Complexity :

(a) Using Adjacency Matrix

Each vertex is selected once → O(V)

For each vertex, we check all other vertices → O(V)

Total Time Complexity = $O(V^2)$

b) Using Adjacency List + Min-Heap / Priority Queue

Extracting minimum distance node: O(log V)

Relaxing all edges: O(E log V)

Total Time Complexity = O(E log V)

Space Complexity :

graph[V][V] → takes $O(V^2)$ space

dist[], visited[], parent[] → each takes O(V)

Total Space Complexity = $O(V^2)$ for adjacency matrix

If using adjacency list → $O(V + E)$

Real - Time Applications :

1. Disaster Management: Finds safest and fastest rescue routes during floods or earthquakes.
2. Drone Delivery: Helps drones choose shortest and obstacle-free

flight paths.

3. EV Route Planning: Selects optimal routes considering battery range and charging stations.
4. Game Development: Used for smart movement and navigation of game characters.
5. Hospital Ambulance System: Finds nearest ambulance to reach patients quickly.