

# DATA STRUCTURE

## PROGRAMS:

### 1.Shortest Path Algorithm

```
#include <stdio.h>
```

```
#include <limits.h>
```

```
#include <stdbool.h>
```

```
#define V 9
```

```
int minDistance(int dist[], bool sptSet[]) {
```

```
    int min = INT_MAX;
```

```
    int min_index;
```

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

```
        if (!sptSet[v] && dist[v] <= min) {
```

```
            min = dist[v];
```

```
            min_index = v;
```

```
        }
```

```
    }
```

```
    return min_index;
```

```
}
```

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

*int dist[V]; // The output array dist[i] holds the shortest distance from src to j*

*bool sptSet[V]; // sptSet[i] will be true if vertex i is included in the shortest path tree*

```
for (int i = 0; i < V; i++) {  
    dist[i] = INT_MAX;  
    sptSet[i] = false;  
}
```

*dist[src] = 0;*

```
for (int count = 0; count < V - 1; count++) {  
    int u = minDistance(dist, sptSet);
```

*sptSet[u] = true;*

```
for (int v = 0; v < V; v++) {  
    if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX &&  
dist[u] + graph[u][v] < dist[v]) {  
        dist[v] = dist[u] + graph[u][v];  
    }  
}
```

}

*printf("Vertex Distance from Source\n");*

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

*printf("%d \t\t %d\n", i, dist[i]);*

*}*

*}*

*int main() {*

*int graph[V][V] = {*

*{0, 4, 0, 0, 0, 0, 0, 8, 0},*

*{4, 0, 8, 0, 0, 0, 0, 11, 0},*

*{0, 8, 0, 7, 0, 4, 0, 0, 2},*

*{0, 0, 7, 0, 9, 14, 0, 0, 0},*

*{0, 0, 0, 9, 0, 10, 0, 0, 0},*

*{0, 0, 4, 14, 10, 0, 2, 0, 0},*

*{0, 0, 0, 0, 0, 2, 0, 1, 6},*

*{8, 11, 0, 0, 0, 0, 1, 0, 7},*

*{0, 0, 2, 0, 0, 0, 6, 7, 0}*

*};*

*dijkstra(graph, 0);*

*return 0;*

*}*

## OUTPUT:

*Vertex    Distance from Source*

<i>0</i>	<i>0</i>
<i>1</i>	<i>4</i>
<i>2</i>	<i>12</i>
<i>3</i>	<i>19</i>
<i>4</i>	<i>21</i>
<i>5</i>	<i>11</i>
<i>6</i>	<i>9</i>
<i>7</i>	<i>8</i>
<i>8</i>	<i>14</i>

## 2.Dijkstra's Algorithm

```
#include <stdio.h>
```

```
#include <limits.h>
```

```
#include <stdbool.h>
```

```
#define V 9
```

```
int minDistance(int dist[], bool sptSet[]) {
```

```
    int min = INT_MAX;
```

```
    int min_index;
```

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

```
        if (!sptSet[v] && dist[v] <= min) {
```

```

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

```

```

void dijkstra(int graph[V][V], int src) {
    int dist[V];
    bool sptSet[V];

    for (int i = 0; i < V; i++) {
        dist[i] = INT_MAX;
        sptSet[i] = false;
    }
}

```

```

dist[src] = 0;

```

```

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

```

```

    int u = minDistance(dist, sptSet);

```

```

    sptSet[u] = true;

```

```

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

        if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX &&
            dist[u] + graph[u][v] < dist[v]) {
            dist[v] = dist[u] + graph[u][v];
        }
    }
}

```

```

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

```

```

int main() {
    int graph[V][V] = {
        {0, 4, 0, 0, 0, 0, 0, 8, 0},
        {4, 0, 8, 0, 0, 0, 0, 11, 0},
        {0, 8, 0, 7, 0, 4, 0, 0, 2},
        {0, 0, 7, 0, 9, 14, 0, 0, 0},
        {0, 0, 0, 9, 0, 10, 0, 0, 0},
        {0, 0, 4, 14, 10, 0, 2, 0, 0},
    };
}

```

```

        {0, 0, 0, 0, 0, 2, 0, 1, 6},
        {8, 11, 0, 0, 0, 0, 1, 0, 7},
        {0, 0, 2, 0, 0, 0, 6, 7, 0}
    };

    dijkstra(graph, 0);

    return 0;
}

```

## OUTPUT:

<i>Vertex</i>	<i>Distance from Source</i>
0	0
1	4
2	12
3	19
4	21
5	11
6	9
7	8
8	14