Dijsktra’s Function Algorithm

1. START

2. Check whether the visitedIndex is equal to size, if so return the control to the main function after printing the destinationDistance.

3. Initialize a variable isRowFullZero(Flag Variable) as 1 and create a for-lop which loops from 1 to size-1.

4. Check whether the current cell value is not 0, and graph[source][i] is less than graph[source][shortestIndex] or graph[source][shortestIndex] is 0, if so set isRowFullZero to 0 and iterate through that row to find the shortest distance to another edge.

   a. Initialize a flag variable alreadyVisisted as 0 and check whether that vertex is already visited by comparing the i values with the visited array if visited update the flag to 1

   b. If alreadyVisisted is 0, update shortestDistance to i

5. If isRowFullZero is 1 then initialize a variable oldSource to store the old source and subtract the distance last added to reach the previous vertex and update shortestIndex to size - 1;

6. Initialize a variable tempCurrentDistance and calculate the new distance by adding the cell values graph[source][shortestIndex]

7. If the tempCurrentValue is not equal to currentDistance, then check whether the tempCurrentDistance is greater than graph[0][shortestIndex] and the value is not 0, if so update currentDistance to graph[0][shortestIndex]

   a. If the tempCurrentValue equals currentDisnace, add graph[source][shortestIndex] to currentDistance and print the distance.

8. Else if the shortestIndex is not equal to 0, then initialize a variable shortestValue to graph[0][shortestIndex]

   a. Initialize a for-loop which loops from size-1 to > 0 and check whether tthe shortestValue equal to 0 or shortestValue greater than graph[j][shortestIndex] and graph[j][shortestIndex] != 0

   b. If so, set shortestValue to graph[j][shortestIndex]

   c. Finally, add the shortestValue to the currentDistance and print it.

9. If the shortestIndex + 1 is equal to the destination update destinationDistance as currentDistance
10. Push the vertex to the visitedArray and update the visitedIndex by 1
11. Invoke the dijsktra’s function by passing in the
    a. shortestIndex as the new source
    b. graph[shortestIndex[0] as the the shortestIndex
    c. And pass along with other updated values such as currentDistance, size, graph, visited, visitedIndex, destination, and destinationDistance.
12. STOP

Program

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

void dijsktra(int source, int shortestIndex, int currentDistance, int size, int
graph[size][size], int visited[size], int vistedIndex, int destination, int
destinationDistance)
{
    if (vistedIndex == size)
    {
        printf("\nDistance to Destination: %d\n", destinationDistance);
        return;
    }

    int isRowFullZero = 1 ;
    for (int i = 1; i <= size - 1; i++)
    {

        if (graph[source][i] != 0 && (graph[source][i] <= graph[source][shortestIndex] ||
graph[source][shortestIndex] == 0))
        {
            isRowFullZero = 0;
            int alreadyVisited = 0;
            for (int j = 0; j < vistedIndex; j++)
            {
                if (visited[j] == i)
                    alreadyVisited = 1;
            }
            if (alreadyVisited == 0)
                shortestIndex = i;
        }
    }

    if (isRowFullZero == 1)
    {
        int oldSource = visited[vistedIndex - 1];
        currentDistance -= graph[oldSource][source];
        shortestIndex = size - 1;
    }

    int tempCurrentDistance = currentDistance + graph[source][shortestIndex];
    if (tempCurrentDistance != currentDistance)
    {

        if (tempCurrentDistance > graph[0][shortestIndex] && graph[0][shortestIndex] !=
0 )
        {
            currentDistance = graph[0][shortestIndex];
        }
        else
            currentDistance += graph[source][shortestIndex];
        printf("%d \t\t %d\n", (shortestIndex + 1), currentDistance);
    }
    else if (shortestIndex != 0)
    {
        printf("Dey\n");
        int shortestValue = graph[0][shortestIndex];
        for (int j = size - 1; j > 0; j--)
        {
            if (shortestValue == 0 || (shortestValue > graph[j][shortestIndex] &&
graph[j][shortestIndex] != 0))
            {
                shortestValue = graph[j][shortestIndex];
            }
        }

        currentDistance += shortestValue;
        printf("%d \t\t %d\n", (shortestIndex + 1), currentDistance);
    }

    if (shortestIndex + 1 == destination)
        destinationDistance = currentDistance;

    visited[vistedIndex] = source;
    vistedIndex++;
    dijsktra(shortestIndex, graph[shortestIndex][0], currentDistance, size, graph, visited,
vistedIndex, destination, destinationDistance);
}

void main()
{
    int n = 6;

    printf("\nEnter the Number of Vertices: ");
    scanf("%d", &n);

    int graph[n][n];

    for (int i = 0; i < n; i++)
    {
        printf("\nVertex %d: Enter the Distances to Other Vertices(0 for no edge)\n", (i +
1));
        for (int j = 0; j < n; j++)
        {
            printf("%d --> %d: ", (i + 1), (j + 1));
            scanf("%d", &graph[i][j]);
        }
    }

    int destination = 1;
    int destinationDistance = 0;

    printf("\nEnter the destination vertex: ");
    scanf("%d", &destination);

    int visited[n];
    visited[0] = 0;
    int vistedIndex = 1;
    printf("\n\nVertex \t Distance from Source\n");
    dijsktra(0, graph[0][0], 0, n, graph, visited, vistedIndex, destination,
destinationDistance);
}
```

Input Test Cases

Input 1

```
int graph[6][6] = {{0, 2, 4, 0, 0, 0},
                   {0, 0, 1, 7, 0, 0},
                   {0, 0, 0, 0, 3, 0},
                   {0, 0, 0, 0, 0, 1},
                   {0, 0, 0, 2, 0, 5},
                   {0, 0, 0, 0, 0, 0}};
```

Input 2

```
int graph[3][3] = { {0, 1, 3},
                    {3, 0, 1},
                    {3, 2, 0} };
```

Input 3

```
int graph[5][5] = { {0, 4, 2, 0, 0},
                    {0, 0, 3, 2, 3},
                    {0, 1, 0, 4, 5},
                    {0, 0, 0, 0, 0},
                    {0, 0, 0, 1, 0} };
```

Input 4

```
int graph[6][6] = { {0, 50, 45, 10, 0, 0},
                    {0, 0, 10, 15, 0, 0},
                    {0, 0, 0, 0, 30, 0},
                    {10, 0, 0, 0, 15, 0},
                    {0, 20, 35, 0, 0, 0},
                    {0, 0, 0, 0, 3, 0} };
```

Output

Input 1

```
Vertex   Distance from Source
2          2
3          3
5          6
4          8
6          9

Distance to Destination: 2
```

Input 2

```
Vertex   Distance from Source
2          1
3          2

Distance to Destination: 1
```

Input 3

```
Vertex   Distance from Source
3          2
2          3
4          5
5          6

Distance to Destination: 3
```

Input 4

```
Vertex   Distance from Source
4          10
5          25
2          45
3          45

Distance to Destination: 45
```