

<b>Name</b>	<b>Aryaman Agarwal</b>
<b>UID no.</b>	<b>2021700002</b>
<b>Experiment No.</b>	<b>6</b>

<b>AIM:</b>	<b>Implementing dijkstra's algorithm in c.</b>
<b>Program 1</b>	
<b>ALGORITHM/ THEORY:</b>	<p>Create a set <b>sptSet</b> (shortest path tree set) that keeps track of vertices included in the shortest path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.</p> <ul style="list-style-type: none"> <li>Assign a distance value to all vertices in the input graph. Initialize all distance values as <b>INFINITE</b>. Assign the distance value as 0 for the source vertex so that it is picked first.</li> <li>While <b>sptSet</b> doesn't include all vertices <ul style="list-style-type: none"> <li>Pick a vertex <b>u</b> that is not there in <b>sptSet</b> and has a minimum distance value.</li> <li>Include u to <b>sptSet</b>.</li> <li>Then update the distance value of all adjacent vertices of u. <ul style="list-style-type: none"> <li>To update the distance values, iterate through all adjacent vertices.</li> <li>For every adjacent vertex v, if the sum of the distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.</li> </ul> </li> </ul> </li> </ul>

**PROGRAM:**

```
#include <stdio.h>

#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX], int n, int startnode)
{
    int cost[MAX][MAX], distance[MAX], pred[MAX];
    int visited[MAX], count, mindistance, nextnode, i, j;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            if (G[i][j] == 0)
                cost[i][j] = INFINITY;
            else
                cost[i][j] = G[i][j];

    for (i = 0; i < n; i++)
    {
        distance[i] = cost[startnode][i];
        pred[i] = startnode;
        visited[i] = 0;
    }
    distance[startnode] = 0;
    visited[startnode] = 1;
    count = 1;
    while (count < n - 1)
    {
        mindistance = INFINITY;

        for (i = 0; i < n; i++)
            if (distance[i] < mindistance && !visited[i])
            {
                mindistance = distance[i];
                nextnode = i;
            }

        visited[nextnode] = 1;
        for (i = 0; i < n; i++)
            if (!visited[i])
```

```

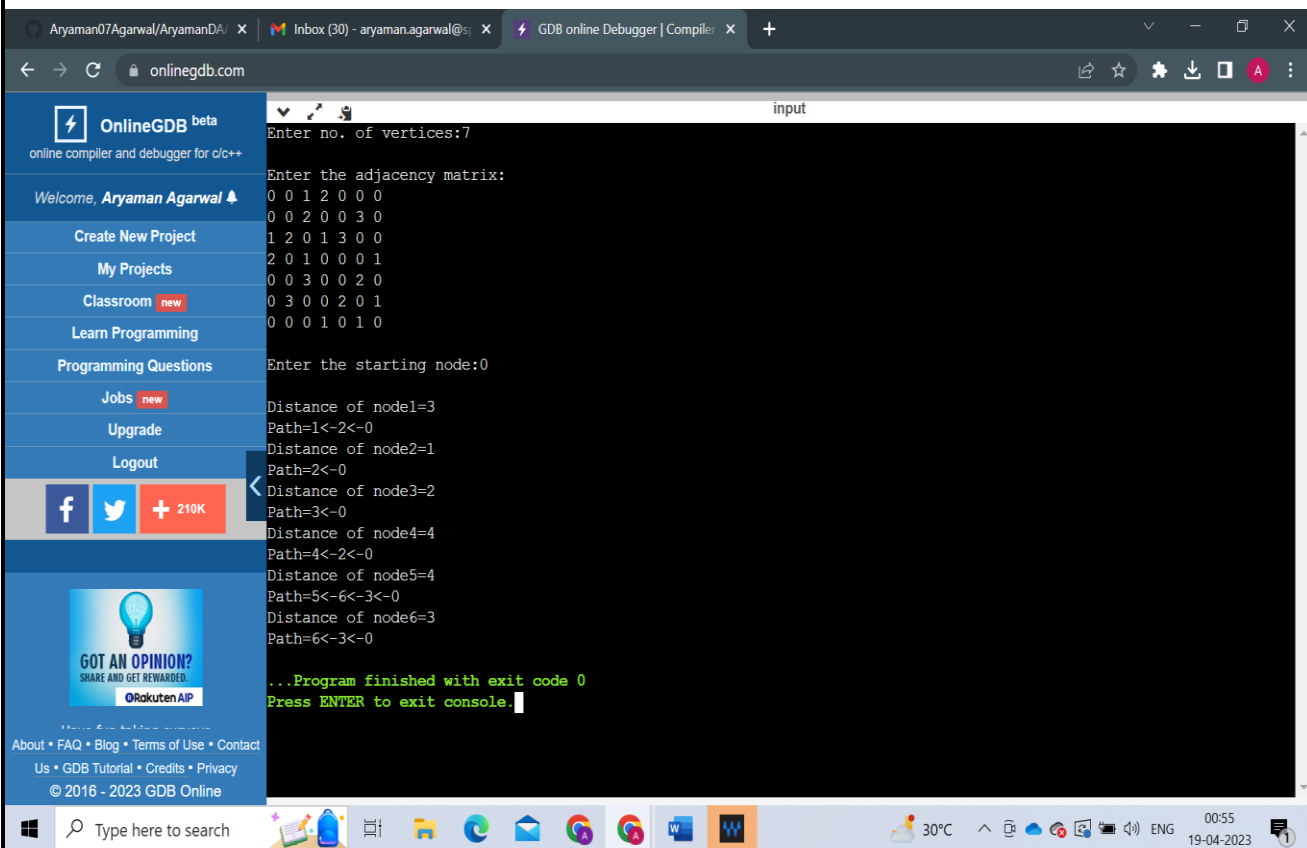
        if (mindistance + cost[nextnode][i] < distance[i])
        {
            distance[i] = mindistance + cost[nextnode][i];
            pred[i] = nextnode;
        }
        count++;
    }

    for (i = 0; i < n; i++)
        if (i != startnode)
        {
            printf("\nDistance of node%d=%d", i, distance[i]);
            printf("\nPath=%d", i);
            j = i;
            do
            {
                j = pred[j];
                printf("<-%d", j);
            } while (j != startnode);
        }
    }

int main()
{
    int G[MAX][MAX], i, j, n, u;
    printf("Enter no. of vertices:");
    scanf("%d", &n);
    printf("\nEnter the adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &G[i][j]);
    printf("\nEnter the starting node:");
    scanf("%d", &u);
    dijkstra(G, n, u);
    return 0;
}

```

## RESULT:



The screenshot shows the OnlineGDB website interface. The left sidebar contains navigation links: Create New Project, My Projects, Classroom (new), Learn Programming, Programming Questions, Jobs (new), Upgrade, and Logout. Below these are social media icons for Facebook and Twitter, and a button for '210K'. The main area displays the execution of a C++ program. The program prompts for the number of vertices (7) and the adjacency matrix. The matrix is a 7x7 grid of integers. The program then prompts for the starting node (0) and outputs the shortest path and distance for each node from 1 to 6. The output shows that the shortest path from node 0 to node 1 is 3, to node 2 is 1, to node 3 is 2, to node 4 is 4, to node 5 is 4, and to node 6 is 3. The program finishes with exit code 0.

```
Enter no. of vertices:7
Enter the adjacency matrix:
0 0 1 2 0 0 0
0 0 2 0 0 3 0
1 2 0 1 3 0 0
2 0 1 0 0 0 1
0 0 3 0 0 2 0
0 3 0 0 2 0 1
0 0 0 1 0 1 0

Enter the starting node:0

Distance of node1=3
Path=1<-2<-0
Distance of node2=1
Path=2<-0
Distance of node3=2
Path=3<-0
Distance of node4=4
Path=4<-2<-0
Distance of node5=4
Path=5<-6<-3<-0
Distance of node6=3
Path=6<-3<-0

...Program finished with exit code 0
Press ENTER to exit console.
```

The adjacency matrix:

```
0 0 1 2 0 0 0
0 0 2 0 0 3 0
1 2 0 1 3 0 0
2 0 1 0 0 0 1
0 0 3 0 0 2 0
0 3 0 0 2 0 1
0 0 0 1 0 1 0
```

<b>CONCLUSION:</b>	From this experiment I understood dijkstra's algorithm to find shortest path.
--------------------	---