

**Name: Vanessa D'mello**

**Roll No: 8863**

**Branch: SE Computers A (Batch A)**

**Experiment: Dijkstra's SSSP Algorithm**

```
#include <stdio.h>
void dijkstra(int v, int a[][v], int s, int dist[], int pred[], int visited[]) {
    for(int i = 0; i < v - 1; i++) {
        int min = 9999;
        int min_vertex;
        for(int j = 0; j < v; j++) {
            if(visited[j] == 0 && dist[j] < min) {
                min = dist[j];
                min_vertex = j;
            }
        }
        visited[min_vertex] = 1;
        for(int j = 0; j < v; j++) {
            if(j == s)
                continue;
            if(a[min_vertex][j] != 0 && dist[j] > dist[min_vertex] + a[min_vertex][j])
            {
                dist[j] = dist[min_vertex] + a[min_vertex][j];
                pred[j] = min_vertex + 1;
            }
        }
    }
    shortestPath(dist, pred, v, s);
}
void shortestPath(int dist[], int pred[], int v, int s) {
    printf("\nDistance\tPath");
    for(int i = 0; i < v; i++) {
        if(i == s)
            continue;
        printf("\n %d\t\t", dist[i]);
        printf("->%d", getPath(i + 1, s, pred));
    }
}
int getPath(int n, int s, int pred[]) {
    if(n == s + 1)
        return n;
    else if(pred[n - 1] == s + 1)
        printf("%d", getPath(pred[n - 1], s, pred));
    else
        printf("->%d", getPath(pred[n - 1], s, pred));
    return n;
}
```

```

int main() {
    int v, source, dest, w;
    printf("Enter no. of vertices: ");
    scanf("%d", &v);
    int a[v][v];
    for(int i = 0; i < v; i++) {
        for(int j = 0; j < v; j++)
            a[i][j] = 0;
    }
    int e;
    printf("Enter no. of edges: ");
    scanf("%d", &e);
    for(int i = 0; i < e; i++) {
        printf("Enter source and destination of edge %d: ", i + 1);
        scanf("%d %d", &source, &dest);
        printf("Enter weight of edge %d: ", i + 1);
        scanf("%d", &w);
        a[source - 1][dest - 1] = w;
    }
    int s = 1;
    printf("Enter source vertex: ");
    scanf("%d", &s);
    int visited[v];
    int dist[v];
    int pred[v];
    for(int i = 0; i < v; i++) {
        if(i == s - 1) {
            visited[i] = 1;
            pred[i] = i + 1;
        } else {
            visited[i] = 0;
            pred[i] = s;
        }
        if(a[s - 1][i] || i == s - 1)
            dist[i] = a[s - 1][i];
        else
            dist[i] = 9999;
    }
    dijkstra(v, a, s - 1, dist, pred, visited);
    printf("\nDist: ");
    for(int i = 0; i < v; i++)
        printf("%d ", dist[i]);
    printf("\nPred: ");
    for(int i = 0; i < v; i++)
        printf("%d ", pred[i]);
    printf("\nVisited: ");
    for(int i = 0; i < v; i++)

```

```

        printf("%d ", visited[i]);
    return 0;
}

```

### Postlab Output:

Select C:\Users\dmell\OneDrive\Desktop\Subjects\AOA\DijkstraAlgo.exe

```

Enter no. of vertices: 6
Enter no. of edges: 11
Enter source and destination of edge 1: 1      2
Enter weight of edge 1: 2
Enter source and destination of edge 2: 1      3
Enter weight of edge 2: 8
Enter source and destination of edge 3: 2      3
Enter weight of edge 3: 5
Enter source and destination of edge 4: 2      4
Enter weight of edge 4: 3
Enter source and destination of edge 5: 3      2
Enter weight of edge 5: 6
Enter source and destination of edge 6: 3      5
Enter weight of edge 6: 6
Enter source and destination of edge 7: 4      3
Enter weight of edge 7: 1
Enter source and destination of edge 8: 4      5
Enter weight of edge 8: 7
Enter source and destination of edge 9: 4      6
Enter weight of edge 9: 6
Enter source and destination of edge 10: 5     4
Enter weight of edge 10: 4
Enter source and destination of edge 11: 6     5
Enter weight of edge 11: 2
Enter source vertex: 1

Distance      Path
2             1->2
6             1->2->4->3
5             1->2->4
12            1->2->4->5
11            1->2->4->6
Dist: 0 2 6 5 12 11
Pred: 1 1 4 2 4 4
Visited: 1 1 1 1 1 1
Process returned 0 (0x0)   execution time : 97.798 s
Press any key to continue.

```