Project Report

on

Dijkstra's Algorithm to Find the Shortest Rail Route Between Cities

Submitted by:

Shwetha Iyer (1032211195)

Ayushi Sachan (1032211768)

Aastha Sisodia (1032211929)

in

Data Structures and Algorithms

SY B.Tech ECE AI-ML

Under the Guidance of

Dr. Nilam Pradhan

MIT WORLD PEACE UNIVERSITY

School of Electronics & Communication Engineering

2022 - 23

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Table of Contents

Acknowledgement

We would like to thank our subject teacher, Dr. Nilam Pradhan, for providing us the wonderful opportunity to work on this project. It has been a motivating and insightful learning experience.

Chapter 1: Introduction

A graph is a mathematical data structure defined by two non-empty sets: a set of nodes and a set of edges. Graphs can be used to represent a network of cities, wherein the nodes indicate the cities and the edges indicate the distances by train between any two cities. Dijkstra's algorithm is a greedy algorithm for finding the shortest path between two nodes in a graph. In this project, Dijkstra's algorithm has been implemented to compute the shortest rail route between two cities.

Chapter 2: Problem Statement

Given a set of cities and the distances between them by direct train, find the shortest route to be taken from a specified starting city to all the other cities.

Chapter 3: Data Structure Used – *Graph*

A graph is a non-linear data structure that consists of the following two components:

- A finite set of vertices also called as nodes.
- A finite set of edges that connect any two nodes in the graph.

Graphs can be directed or undirected. In an undirected graph, edges are not associated with the directions with them. Hence, if an edge exists between vertex A and B then the vertices can be traversed from B to A as well as A to B. Graphs can also have weighted edges. In a weighted graph, each edge is assigned with some data such as length or cost.

The terminologies usually associated with graphs are:

- ***Adjacency:*** A vertex is said to be adjacent to another vertex if there is an edge connecting them.

- ***Path:*** A sequence of edges that allows you to go from vertex A to vertex B is called a path.

Graphs are commonly represented in two ways:

- ***Adjacency Matrix:***
  An adjacency matrix is a 2-D array where the rows and columns represent vertices. If the value of an element a[i][j] is 1, it represents that there is an edge connecting vertex i and vertex j. In case of a weighted graph, If adj[i][j] = w, then there is an edge with weight w from vertex i to vertex j.
- ***Adjacency List:***
  An adjacency list represents a graph as an array of linked lists. The index of the array represents a vertex and each element in its linked list represents the other vertices that form an edge with the vertex.

Chapter 4: Dijkstra's Shortest Path Algorithm

Step 1. Start with a weighted graph.
Step 2. Choose a starting node and set its distance to zero. Assign infinity path values to all other nodes.
Step 3. Set the starting node as current.
Step 4. Consider all of its unvisited neighbors and calculate their distances from the current node.
Step 5. Pick the unvisited node with the least path length from the current node.
Step 6. Mark the current node as visited and set the next node as the new current node.
Step 7. Repeat steps 4, 5, 6 until all the nodes have been visited.

In this program, the algorithm has been implemented as follows:

Chapter 5: Program Code

```
#include<stdio.h>
#include<string.h>
#define INFINITY 99999
#define MAX 10
```

```
void Dijkstra(char [][20],int [][MAX],int,int);
int findIndex(char [][20],char [], int);
void create(char [][20],int [][MAX],int);
void display(int [][MAX], int);

//DIJKSTRA'S SHORTEST PATH ALGORITHM

void Dijkstra(char cities[][20],int G[][MAX],int n,int startnode)
{

    int cost[MAX][MAX],distance[MAX],pred[MAX];
    int visited[MAX],count,mindistance,nextnode;
    int i,j;

    /*
    cost[i][j] is the distance from city i to city j
    cost is INFINITY is there is no forward path from city i to city 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];
        }
    }

    //pred[] stores the previous of each node
    //Initialize pred[],distance[] and visited[]

    for(i=0;i<n;i++)
    {
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
```

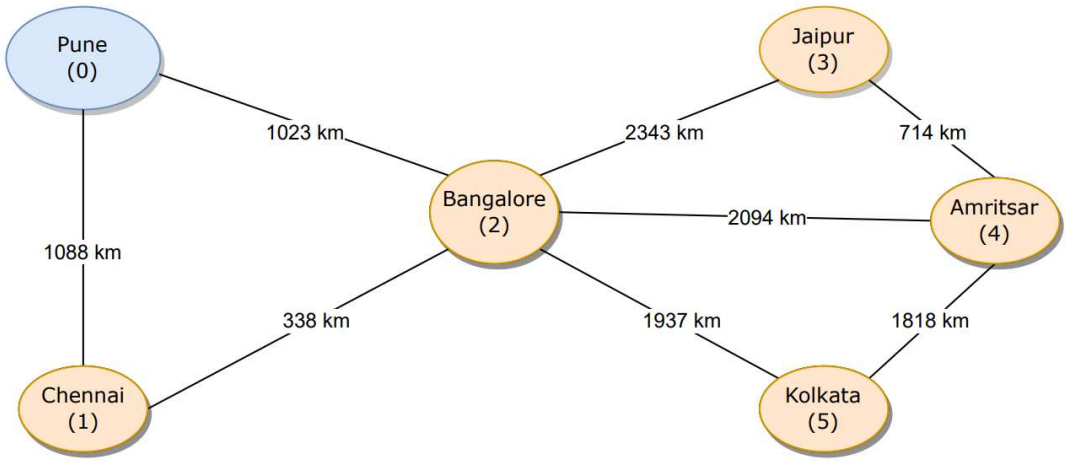
```
}

//count is the number of nodes visited so far

/*
Step 1: Mark starting node as visited
        Set distance path value of all other nodes to INFINITY
*/

distance[startnode]=0;
visited[startnode]=1;
count=1;
while(count<n-1)
{
    mindistance=INFINITY;
    for(i=0;i<n;i++)
    {
        //Choosing unvisited vertex with minimum distance as next node
        if(distance[i]<mindistance&&!visited[i])
        {
            mindistance=distance[i];
            nextnode=i;
        }
    }

    //Marking the next node as visited
    visited[nextnode]=1;

    //To check if a better path exists through nextnode

    for(i=0;i<n;i++)
    if(!visited[i])
    {
        //Calculate distance of remaining unvisited nodes from source
        if(mindistance+cost[nextnode][i]<distance[i])
        {
            distance[i]=mindistance+cost[nextnode][i];
            pred[i]=nextnode;
        }
```

```
        }
        count++;
    }

    //To print the route and distance of each node
    for(i=0;i<n;i++)
    if(i!=startnode)
    {
        printf("\n\nDistance from %s to %s = %d
km",cities[startnode],cities[i],distance[i]);
        if(distance[i]==INFINITY)
        printf("\nRoute: Viable route does not exist.\n");
        else
        printf("\nRoute: %s",cities[i]);
        j=i;
        do
        {
            j=pred[j];
            printf(" <- %s",cities[j]);
        }while(j!=startnode);
    }
}


//To map each city to a node number in the graph
int findIndex(char arr[][20],char st[],int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        if(strcmp(arr[i],st)==0)
        return i;
    }
    return -1;
}

//To create a directed weighted graph using user input
void create(char cities[][20],int G[][MAX],int n)
{
```

```
    int i,j;
    char c;
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(i==j)
            G[i][j]=0;
            else
            {
                printf("Is there a direct train from %s to %s? y/n
",cities[i],cities[j]);
                scanf(" %c",&c);
                if(c=='y'||c=='Y')
                {
                    printf("Enter the distance covered from %s to %s:
",cities[i],cities[j]);
                    scanf("%d",&G[i][j]);
                }
                else
                G[i][j]=0;
                printf("\n");
            }
        }
    }
}

//To display adjacency matrix of graph
void display(int G[][MAX], int n)
{
    int i,j;
    printf("Adjacency Matrix:\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("%d\t",G[i][j]);
        }
        printf("\n");
```

```
    }
}

int main()
{
    int G[MAX][MAX],i,j,n,src;
    char cities[MAX][20];
    char start[20];
    printf("Enter the no. of cities: ");
    scanf("%d",&n);
    printf("Enter the names of cities: \n");
    for(i=0;i<n;i++)
    scanf("%s",cities[i]);
    create(cities,G,n);
    display(G,n);
    printf("\nEnter the starting city: ");
    scanf("%s",start);
    src=findIndex(cities,start,n);

    //Loop until user enters a valid city
    while(1)
    {
        if(src==-1)
        {
            printf("City not present in graph.");
            printf("\nEnter the starting city: ");
            scanf("%s",start);
            src=findIndex(cities,start,n);
        }
        else
        break;
    }
    Dijkstra(cities,G,n,src);
    return 0;
}
```

Chapter 6: Results



Let Dijkstra's algorithm be applied on the above graph, when considering the starting vertex as Pune (0).

```
Enter the no. of cities: 6
Enter the names of cities:
Pune
Chennai
Bangalore
Jaipur
Amritsar
Kolkata
Is there a direct train from Pune to Chennai? y/n y
Enter the distance covered from Pune to Chennai: 1088

Is there a direct train from Pune to Bangalore? y/n y
Enter the distance covered from Pune to Bangalore: 1023

Is there a direct train from Pune to Jaipur? y/n n

Is there a direct train from Pune to Amritsar? y/n n

Is there a direct train from Pune to Kolkata? y/n n

Is there a direct train from Chennai to Pune? y/n y
Enter the distance covered from Chennai to Pune: 1088

Is there a direct train from Chennai to Bangalore? y/n y
Enter the distance covered from Chennai to Bangalore: 338

Is there a direct train from Chennai to Jaipur? y/n n

Is there a direct train from Chennai to Amritsar? y/n n
```

```
Is there a direct train from Chennai to Kolkata? y/n n

Is there a direct train from Bangalore to Pune? y/n y
Enter the distance covered from Bangalore to Pune: 1023

Is there a direct train from Bangalore to Chennai? y/n y
Enter the distance covered from Bangalore to Chennai: 338

Is there a direct train from Bangalore to Jaipur? y/n y
Enter the distance covered from Bangalore to Jaipur: 2343

Is there a direct train from Bangalore to Amritsar? y/n y
Enter the distance covered from Bangalore to Amritsar: 2094

Is there a direct train from Bangalore to Kolkata? y/n y
Enter the distance covered from Bangalore to Kolkata: 1937

Is there a direct train from Jaipur to Pune? y/n n

Is there a direct train from Jaipur to Chennai? y/n n

Is there a direct train from Jaipur to Bangalore? y/n y
Enter the distance covered from Jaipur to Bangalore: 2343
```

```
Is there a direct train from Jaipur to Amritsar? y/n y
Enter the distance covered from Jaipur to Amritsar: 714

Is there a direct train from Jaipur to Kolkata? y/n n

Is there a direct train from Amritsar to Pune? y/n n

Is there a direct train from Amritsar to Chennai? y/n n

Is there a direct train from Amritsar to Bangalore? y/n y
Enter the distance covered from Amritsar to Bangalore: 2094

Is there a direct train from Amritsar to Jaipur? y/n y
Enter the distance covered from Amritsar to Jaipur: 714

Is there a direct train from Amritsar to Kolkata? y/n y
Enter the distance covered from Amritsar to Kolkata: 1818

Is there a direct train from Kolkata to Pune? y/n n

Is there a direct train from Kolkata to Chennai? y/n n

Is there a direct train from Kolkata to Bangalore? y/n y
Enter the distance covered from Kolkata to Bangalore: 1937
```

```
Is there a direct train from Kolkata to Jaipur? y/n n

Is there a direct train from Kolkata to Amritsar? y/n y
Enter the distance covered from Kolkata to Amritsar: 1818

Adjacency Matrix:
0       1088    1023    0       0       0
1088    0       338     0       0       0
1023    338     0       2343    2094    1937
0       0       2343    0       714     0
0       0       2094    714     0       1818
0       0       1937    0       1818    0

Enter the starting city: Mumbai
City not present in graph.
Enter the starting city: Pune


Distance from Pune to Chennai = 1088 km
Route: Chennai <- Pune

Distance from Pune to Bangalore = 1023 km
Route: Bangalore <- Pune

Distance from Pune to Jaipur = 3366 km
Route: Jaipur <- Bangalore <- Pune
```

```
Distance from Pune to Amritsar = 3117 km
Route: Amritsar <- Bangalore <- Pune

Distance from Pune to Kolkata = 2960 km
Route: Kolkata <- Bangalore <- Pune

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

Chapter 7: Conclusion

Through this project, we have successfully implemented Dijkstra's algorithm for a railway network, using a graph with nodes corresponding to cities and edges corresponding to the distances between them. The result is a series of shortest routes from a chosen start city to all the other cities.

References

1. https://www.programiz.com/dsa/graph
2. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
3. https://www.thecrazyprogrammer.com/2014/03/dijkstra-algorithm-for-finding-shortest-path-of-a-graph.html
4. https://www.programiz.com/dsa/dijkstra-algorithm