Optimal Route Analyzer

A

Project Report

Submitted in partial fulfillment of the

requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

Akshat Srivastava	Shubham Verma	Manish Kumar
500070091	500067942	500070096
Roll no.: R103218178	Roll no.: R10218152	Roll no.: R103218182
Branch: CSE-BAO	Branch: CSE-BAO	Branch: CSE-BAO

Under the guidance of

Mr. Sudhanshu Srivastava
Assistant Professor
Dept. of Informatics

School of Computer Science

Department of Systemics

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Bidholi, Via Prem Nagar, Dehradun, Uttarakhand

2019-20

CANDIDATE'S DECLARATION

I/We hereby certify that the project work entitled **"*Optimal Route Analyzer*"** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in ***Computer Science and Engineering with specialization in Business Analytics & Optimization*** and submitted to the School of Computer Science Department of Systemics, University of Petroleum & Energy Studies, Dehradun, is an authentic record of my/ our work carried out during a period from ***Aug, 2019*** *to* ***Dec,2019*** under the supervision of ***Mr. Sudhanshu Srivastava, Assistant Professor, Dept. of Informatics.***

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

Akshat Srivastava	Shubham Verma	Manish Kumar
R103218178	R103218152	R103218182

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 1st December 2020

Mr. Sudhanshu Srivastava
Project Guide

Dr. Thipendra Pal Sing
Head Department of Informatics
School of Computer Science
University of Petroleum & Energy Studies
Dehradun – 248 001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide ***Mr. Sudhanshu Srivastava***, for all advice, encouragement and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our respected Program Head of the Department, ***Dr. Thipendra Pal Singh*** , for his guidance and support as when required.

We are also grateful to ***Dr. Manish Prateek,*** Dean- SOCS, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our friends for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our parents who have shown us this world and for every support they have given us.

Name:	Akshat Srivastava	Shubham Verma	Manish Kumar
Roll No:	R103218178	R103218152	R103218182

ABSTRACT

Route planning is the process of computing the most cost-effective route involving several nodes or locations by minimizing the distance traveled and/or time taken. The basis for route planning and optimization is the use of models to describe the transport network that needs to be planned. This project involves the use of the Kruskal's algorithm along with Travelling Salesman Problem in route planning for a case of the Labels logistics company. The work in this project is largely based on the implementation of the Kruskal's and Travelling Salesman together with a mapping API to computer optimized

Our goal is to find the shortest path between N different cities that the salesman takes is called the TOUR. In other words, the problem deals with finding a route covering all cities so that the total distance traveled is minimal but do not return back. This paper gives a solution to find an optimum route for traveling salesman problem using Kruskal algorithm technique, in which cities are selected randomly as initial population.

Keywords: Route Planning/Optimisation; Kruskal's Algorithm; Travelling Salesman Problem; Optimal Path.

TABLE OF CONTENTS

==========

LIST OF FIGURES

==========

LIST OF TABLES

==========

INTRODUCTION

The Economic **impact** of the 2020 **coronavirus** pandemic in **India** has been largely disruptive. **India's** growth in the fourth quarter of the fiscal year 2020 went down to 3.1% according to the Ministry of Statistics.

The revised **Gross Domestic Product (GDP)** estimates for India downwards by 0.2 % points for the fiscal year 2020 to 4.8 % and by 0.5 % for the fiscal year 2021 to 6 %.

The **Indian economy** was expected to lose over ₹32,000 crore (US$4.5 billion) every day during the first 21-days of complete Lockdown.

While the country is facing an economic slowdown, India's travel and tourism industry is showing hardly any signs of retreat.

Travel and Tourism Sector Likely to Lose 5 Rs lakh Crore Due to Covid – 19 Crisis.

The market is set to touch a staggering $9 billion market size by 2025, bolstered by an influx of new-age travel start-ups, and changing trends driven by Travel Agencies

==========

Effects & Overcome

After all this is Over the People Must be Very Eager to Come Outside Since a Very Long Time of Stay.

They just wanted to escape their mundane lives and give them a different beat.

They will be Interested in services and products to help them stay safe and healthy even at extra money

What's Gonna Effect ?

Since Due to Pandemic major of the Industries or Companies are in Deprivation .
From healthcare to education, Every market segment is witnessing the impacts of technological Disruption.
Then why not the Travel and Tourism sector, which is plagued by the ghosts of budgeting and time constraints ? As they see steps towards them they try to profitize themselves , by swindling the customers.

==========

Problem Statement

These are hard times for travellers. They are also pretty tough for those just beginning with a view to getting started in the travel industry.

Why go to your local agency to ask questions about some faraway destination when Google can answer those for you ?

Why rely on a third party agency when we can, through the internet, find the accommodation or service that you're looking for and book directly ?

Any why remain loyal to a particular agent when there are thousands at your fingertips ?

Figuring out and deciding on which path to follow when making deliveries can be time consuming and can result in high operational costs being incurred. Hence, the need for a route optimisation algorithm which will compute an optimal path and enhance improved

decision making.

However, Nowadays Travellers aren't naïve when it comes to pricing their trip. They're sensible enough to shop around and find the best deal, and they certainly won't be fooled into paying over the odds when they know how many potential options there are available for them.
With more holiday bargains on the internet than you can imagine, along with a host of small agencies and big hitters buying the attention of potential travellers, loyalty is a rare thing in today's market

==========

Aim :

To determine the capabilities of the Kruskal's algorithm in route planning and optimization.

OBJECTIVES

The primary objective of this project is to ***find the shortest possible route that visits the city in a limited amount of time*** in C programming language.

Sub objectives:

The System should be able to:

- Identify a locations of clients on a map.
- Designing algorithm for Optimal Route Analyzer.
- Implementation of algorithm.
- Compute an optimal path connecting all the clients.
- Output the least Time Visit.

==========

METHODOLOGY

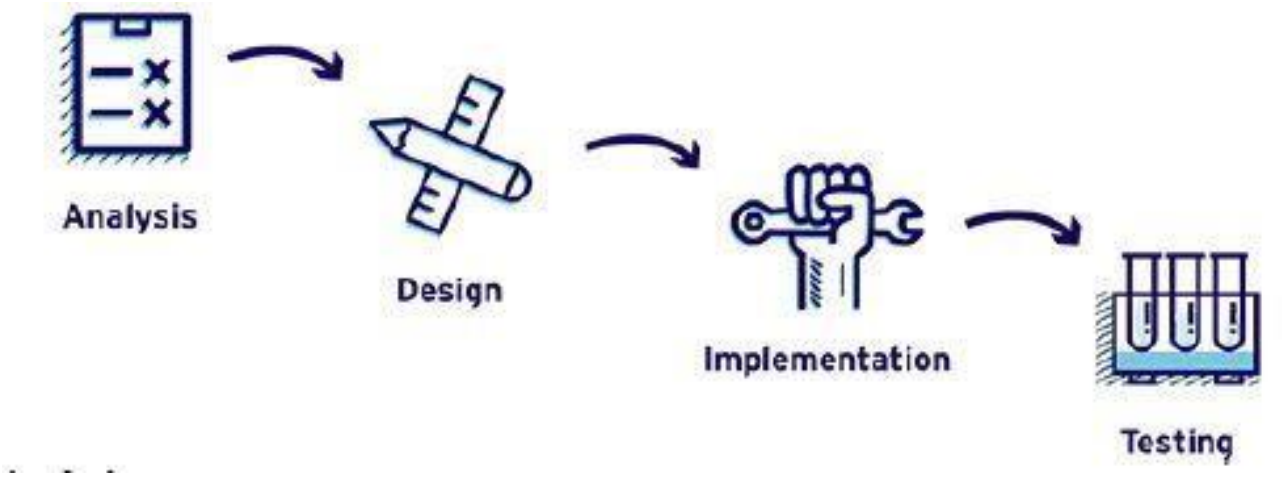

Figure 1: Waterfall Model

1. Analysis:

i) For this project we have searched various research papers to find the focus on finding the most effective route, Our Research is often concerned with finding the cheapest solution

3. Design:

This phase includes designing of DFD, Use case diagram.

Let's Say a trip to Venice , plan to visit all the important Heritage sites but short in time. To make our itinerary work, we can decide to use our Algorithm .

Cannaregio	Ponte Scalzi	Santa Corce	Dell 'Orto	Ferrovia	Piazzale Roma	San Polo	Dorso Duro	San Marco	St. Mark Basilica	Castello	Arsenale
A	B	C	D	E	F	G	H	I	J	K	L

Step 1- Remove all the parallel edges.

~ So for the given map, we have a parallel edge running between Madonna Dell' Orto (D) to St. Mark Basilica (J), which is of length say 2.4kms(2400mts).
~ We will remove the parallel road and keep the 1.8km (1800m) length for representation.
~ Here , the Weights on the Edge represent the Distance Between the Cities.

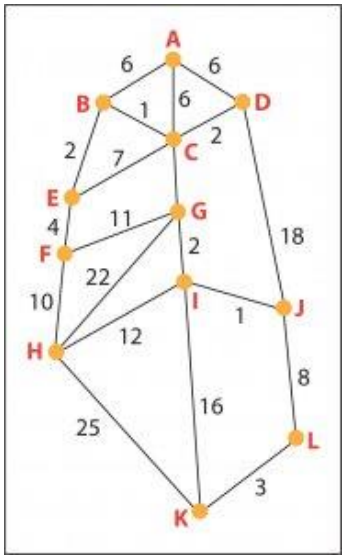

Figure 2: Map to Graph

Step 2 – Arrange all the edges on the graph in ascending order. Algorithm considers each group as a tree and applies disjoint sets to check how many of the vertices are part of other trees.

Step 3 – Add edges with least weight; we begin with the edges with least weight/cost. Hence, B, C is connected first considering their edge cost only 1. Similarly, we connect all the nodes one by one mountingly acc. to their weights.

This gives us the following graph,

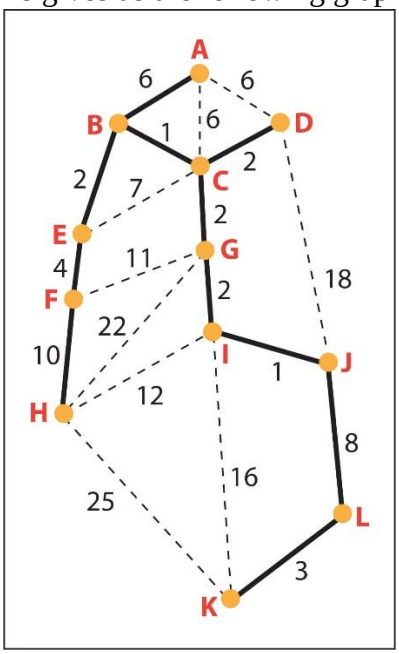

Figure 3: Final Visiting Path

==========

ALGORITHM

- Start
- Create an Array of Size MAX (Defined)
- Initializing (n) as the number of user inputs to store no. of cities (nodes)
- Initializing (c) and (c1) as a counter variable.
- Enter the number of cities along with the name.

- Displaying all the total possible routes covering all cities.

- Filtering of routes is done for the Ease

- Displaying the total possible combinations that formed.

- Now, Taking the input from the user to enter the distance between the cities which displayed as weights and store it into the array data.

- Now , The Process Starts for finding the Minimum route Comparing each Nodes and Select it.

- Displaying Node-pair along with unique id and data to give the Sorted view of the Procedure Followed.

- Display Which Route to Select as a Shortest Path.

- End

==========

PSEUDOCODE

Step 1: Initializing an Array of Size MAX (Defined).
int data[MAX];

Step 2: Initializing essential variables such as (n) to store number of nodes (c) & (c1) as the counter variables other variables for the condition support.

Step 3: Entering (n) from user side as number of nodes.
("%d",&n);

Step 4: Displaying the total possible routes as well as the routes which are filtered out

```
Step 4.1:  for ( j=0;j<n;j++)
                for ( i=0;i<n;i++)
                        {
                                if(i!=j)
                                {
                                        Printf("%c ----- %c", city[j],city[i]);
                                }
                        }

Step 4.2:  Printf ("\n          --- Filtering the routes ---       \n");
                for(int j = 0;j<n;j++)
                        for(int i = 0;i<n;i++)
                                {
                                        if(i!=j)
                                        {
                                                printf("%c ---- %c\n",city[j],city[i]);
                                                c1++;
                                        }
                                }
```

Step 5: Storing the weights as Distance between the cities through input by user in array.

```
Step 5.1: for(i=0;i<c1;i++)
                {
                        scanf("%d",&data[i]);
                }
```

Step 6: Finding the Minimum route for each row

```
Step 6.1: for(j=0;j<n;j++)
                {
                        mn = data[l];
                        l = l + (n-1);
Step 6.2:               for(int i=g;i<n1;i++)
                                if(data[i] < mn)
                                {
                                        mn=data[i];
                                }
                                g = g + (n-1);
                                n!= n1 + (n-1);
```

}

Step 7: Displaying the Node-pair along with unique id and data to give the Sorted view

Step 8: Finally Showing the Optimal Route to Visit.

==========

DESIGNING PHASE

Data Flow Diagram

DFD Level 0

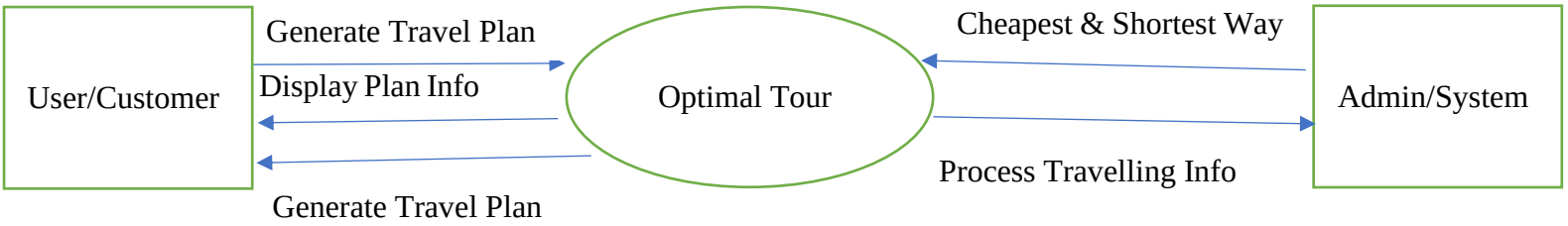

Figure 4: DFD Level 0

Figure 4 represents the DFD level 0 which shows the basics steps involved in our project.

==========

DFD Level 1

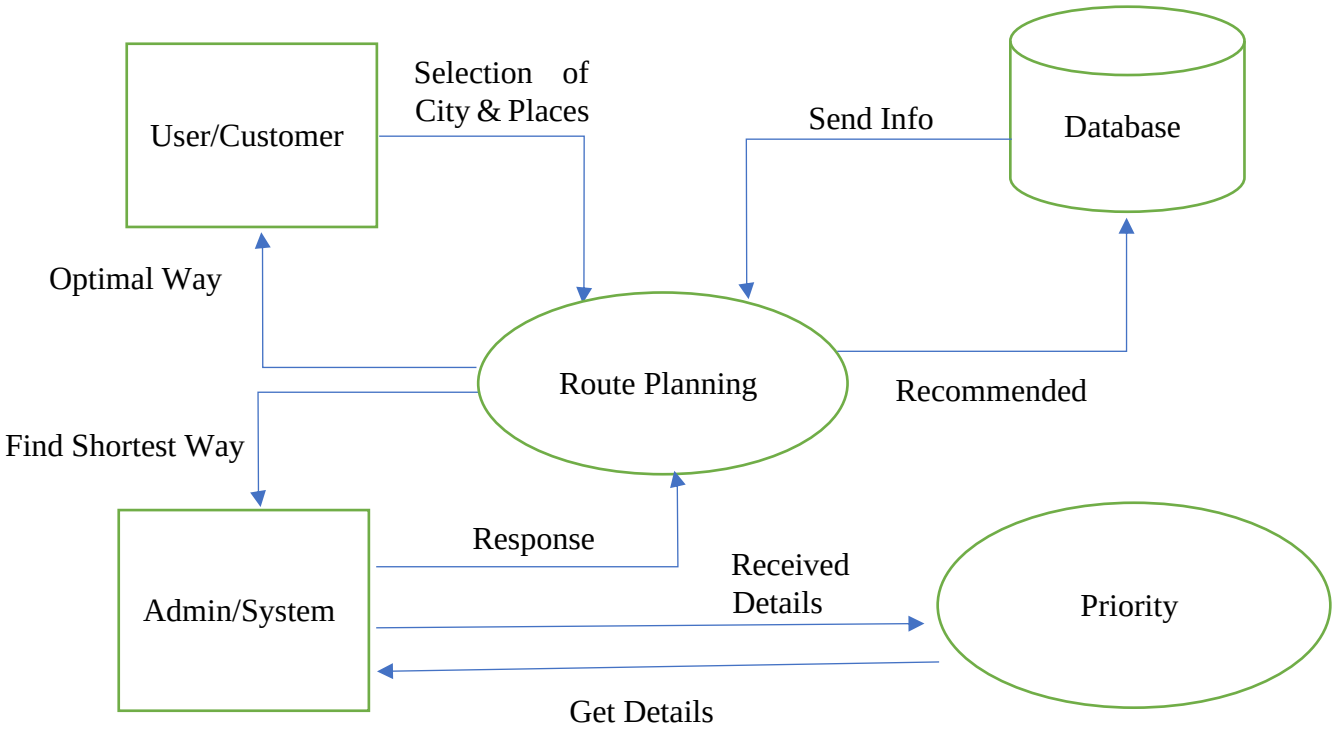

Figure 5: DFD Level 1

Figure 5 represents the DFD level 1, which shows the steps involved in the backend of the system.

==========

FLOWCHART

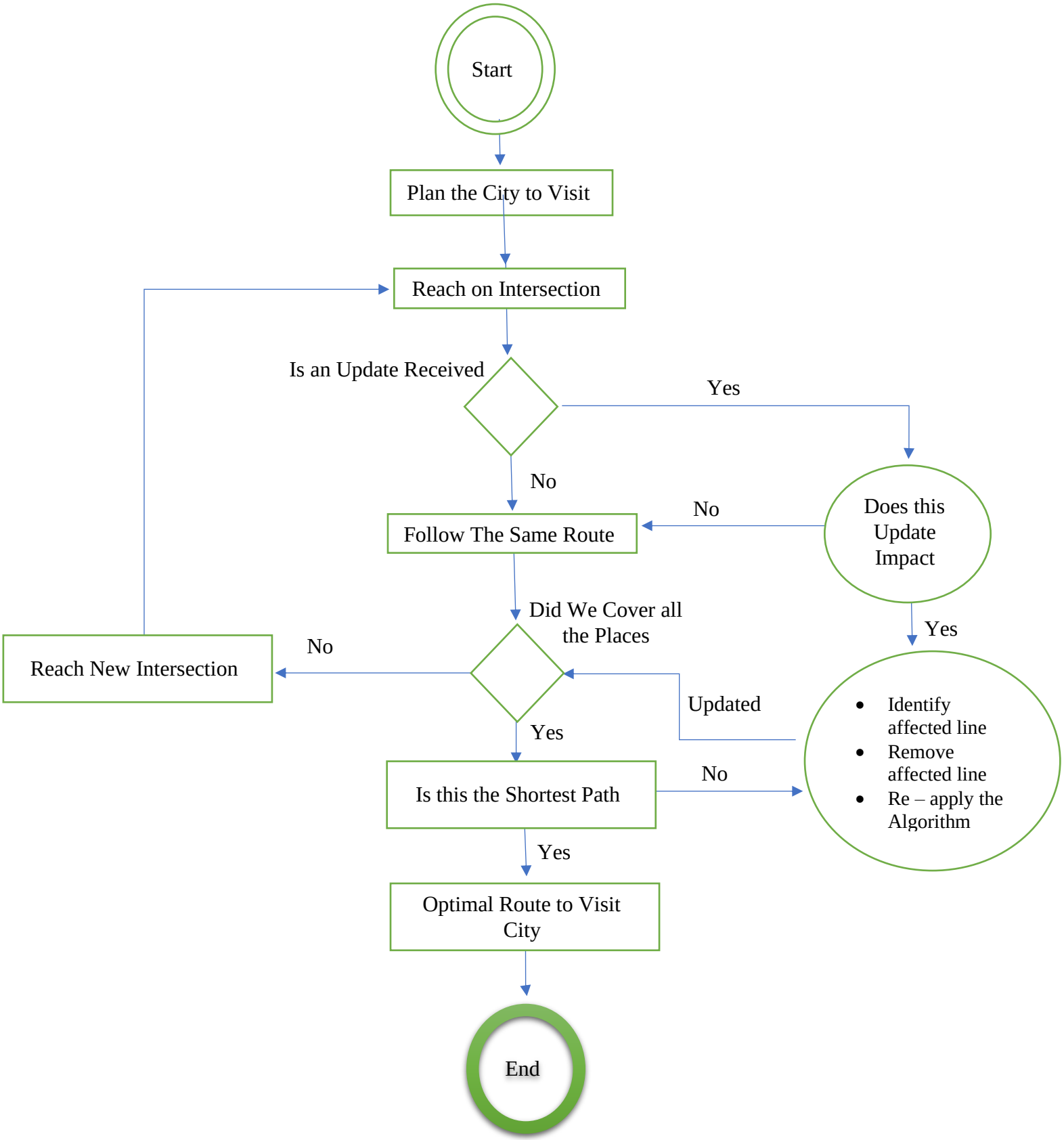
Start
Plan the City to Visit
Reach on Intersection
Is an Update Received
Yes
No
Does this Update Impact
No
Follow The Same Route
Yes
Did We Cover all the Places
No
Reach New Intersection
Updated
Identify affected line
Remove affected line
Re – apply the Algorithm
Yes
Is this the Shortest Path
No
Yes
Optimal Route to Visit City
End

Figure 6: Flow Diagram

==========

Use Case Diagram

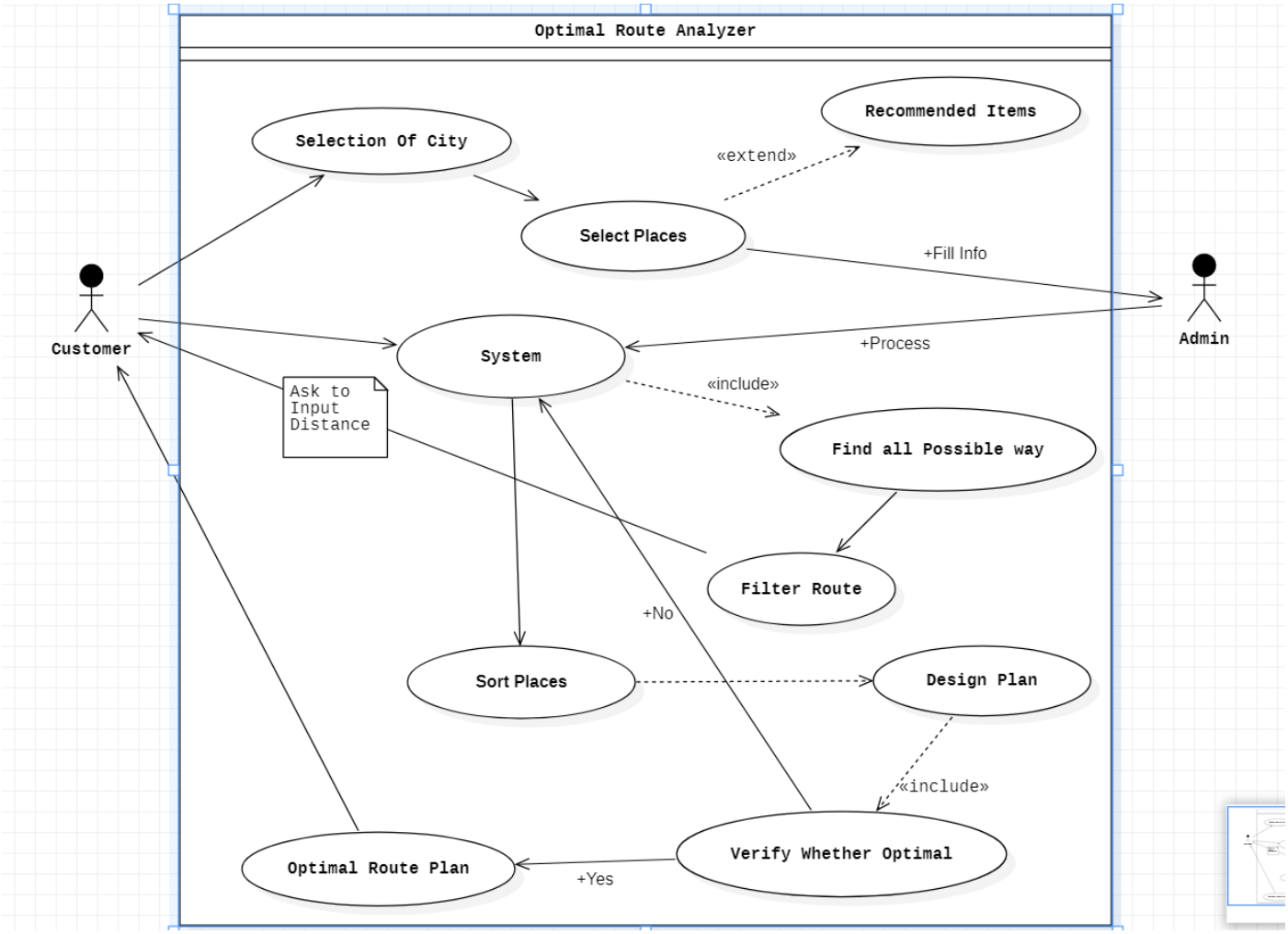

Figure 7: Use case Diagram

==========

OUTPUTS

```
Enter the number of cities to visit : -- 4
Enter the 1th city : -- Manchester
Enter the 2th city : -- Oxford
Enter the 3th city : -- London
Enter the 4th city : -- Norwich
```

Figure 8: Entering the path

In figure 8, Here user is entering the no. of cities user wants to visit.

```
The cities are : --
M       O       L       N
 The possible routes through all cities : --
M ---- M
M ---- O
M ---- L
M ---- N
O ---- M
O ---- O
O ---- L
O ---- N
L ---- M
L ---- O
L ---- L
L ---- N
N ---- M
N ---- O
N ---- L
N ---- N
```

Figure 9: Possible Path

In figure 9, Displaying all the possible routes that one can visit.
For example, in the above screenshot, user wants to visit 4 places so the possible routes are = 2^n => 2^4 => 16 Here.

```
---> There are total 16 combinations possible <---
             --- Filtering the routes ---
M ---- O
M ---- L
M ---- N
O ---- M
O ---- L
O ---- N
L ---- M
L ---- O
L ---- N
N ---- M
N ---- O
N ---- L
```

Figure 10: Route Filtering

In figure 10, It display the combinations after filtering of routes.
This means that removing all the self loops Ex. O---O , M---M ,So Final Possible combination routes = (2^n) – n => 16 – 4 => 12.

```
--->Once filtered it comes out to be total 12 combinations possible <---
               Enter the weight or distance for each pair of nodes : --
Enter the data for node pair :- [M - O]
16
Enter the data for node pair :- [M - L]
25
Enter the data for node pair :- [M - N]
9
Enter the data for node pair :- [O - M]
18
Enter the data for node pair :- [O - L]
29
Enter the data for node pair :- [O - N]
31
Enter the data for node pair :- [L - M]
11
Enter the data for node pair :- [L - O]
37
Enter the data for node pair :- [L - N]
40
Enter the data for node pair :- [N - M]
19
Enter the data for node pair :- [N - O]
18
Enter the data for node pair :- [N - L]
21
```

Figure 11: Distance Measuring

In figure 11, Taking distance between cities as weights from user.

In the above screenshot, the user has been asked to input the distance between cities which are display after final filtering.

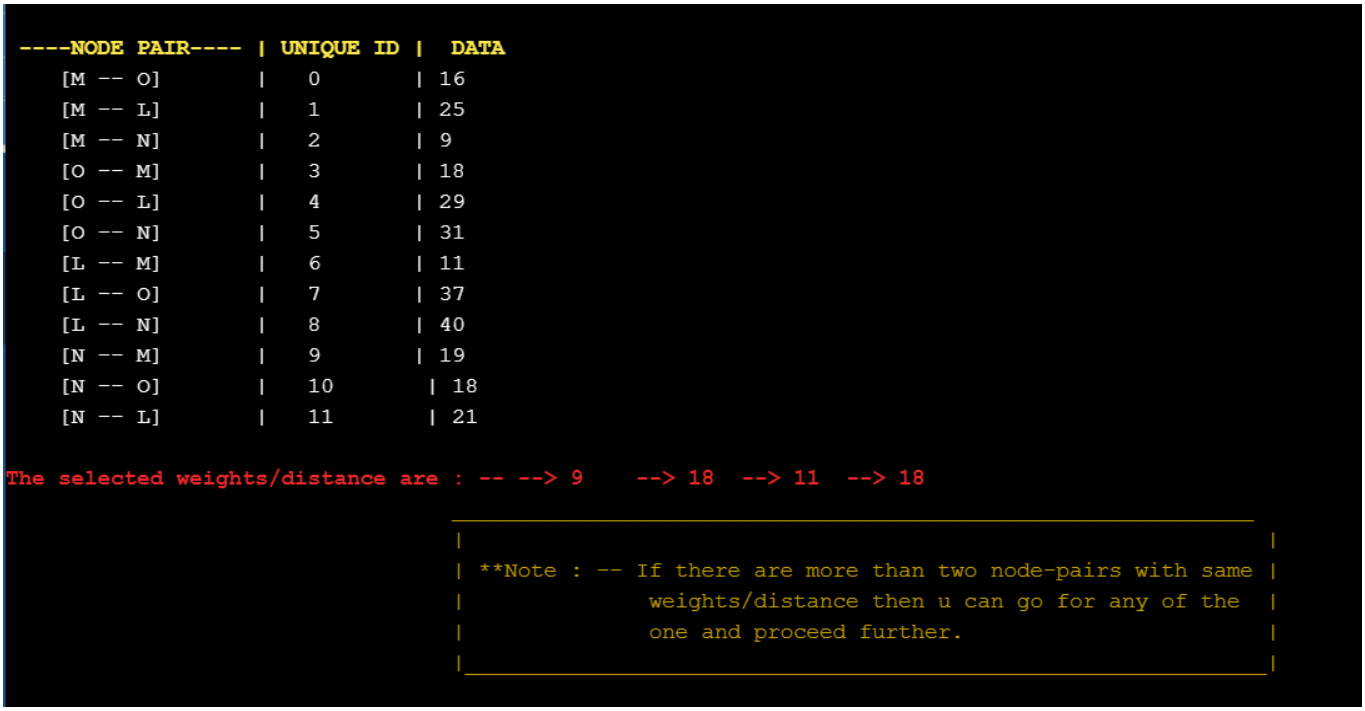

Figure 12: Optimal Route

In the above figure , Finally it display data to the user which shows all the details and shows an optimal route.

==========

Conclusion

This Project focus on the Algorithm of achieving the Shortest Path. It draws the idea of the minimum spanning tree algorithm , and gives the work-flow push system based on TSP algorithm. Through the example analysis, in which cities are selected randomly as initial population, It is concluded that this selection path of algorithm is more accurate, stable, focused on Optimization and has the Advantage of time that gives us better Solution often means a Solution that is Cheaper, Shorter, or Faster

==========

Project Timeline

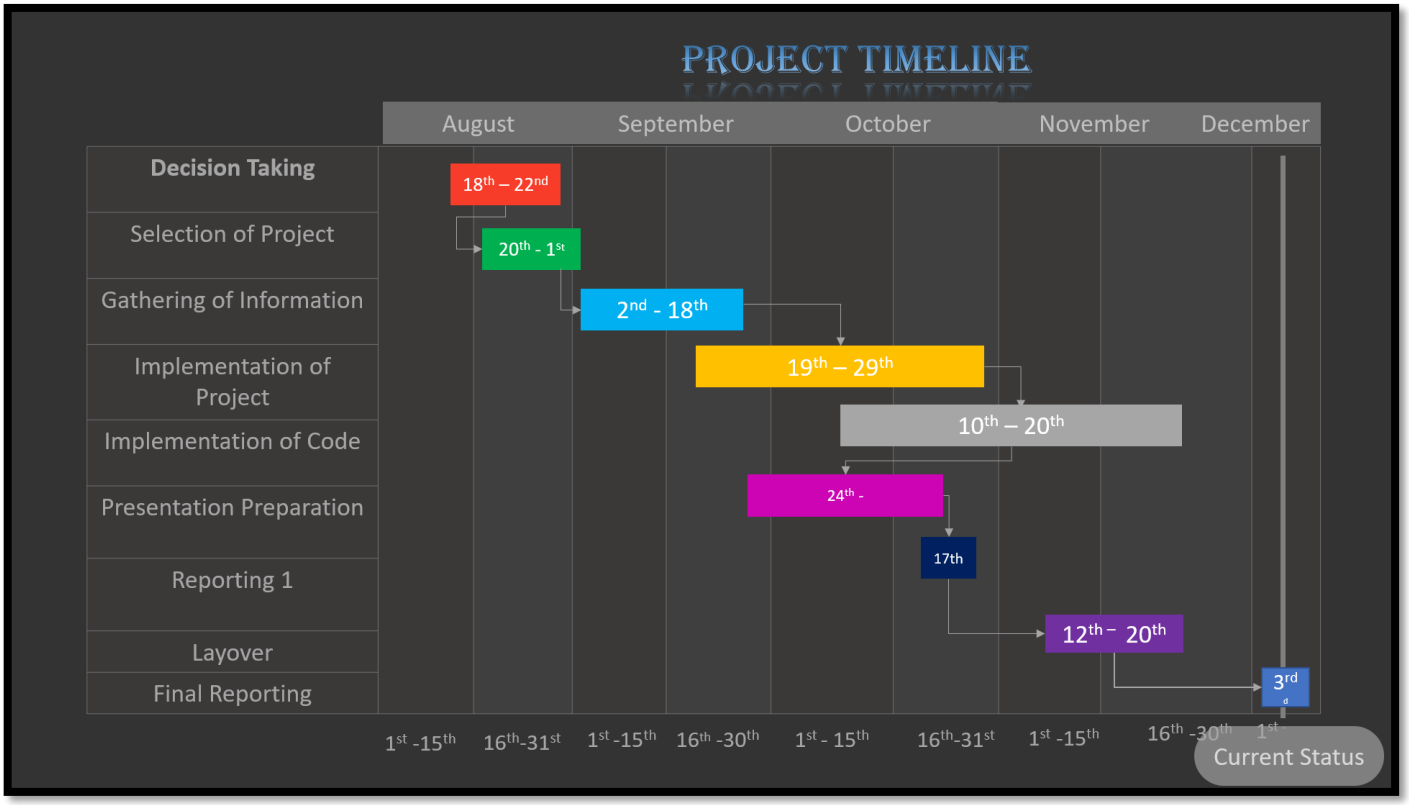

Figure 23:Gantt Chart

==========

Appendix Code

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

int ary[10][10],completed[10],n,cost=0,data[100];
  int mn,temp,temp3,minimum;
  int p = 0,p1 = 0,c = 0,l = 0,count = 0,g = 1,n1=0,location = 0,c1 = 0;
  char city[1000000];

void takeInput()
{
        int i,j;

  printf("Enter the number of cities to visit : -- ");

  scanf("%d",&n);
  int q=n-1;
  for(int i = 0;i<n;i++)
  {
     printf("Enter the %dth city : -- ",i+1);
     scanf("%s",&city[i]);
  }

 printf("The cities are : -- \n");


 for(int i = 0;i<n;i++)
  {

     printf("%c \t",city[i]);
```

```
}
printf("\n The possible routes through all cities : -- \n");

  for(int j = 0;j<n;j++)
  {
      for(int i = 0;i<n;i++)
      {
      printf("%c ---- %c\n",city[j],city[i]);
      c++;
      }
  }
printf("\n");

printf("---> There are total %d combinations possible <---",c);


 printf("\n            --- Filtering the routes ---          \n");
  for(int j = 0;j<n;j++)
  {
      for(int i = 0;i<n;i++)
      {
         if(i!=j)
         {
            printf("%c ---- %c\n",city[j],city[i]);
             c1++;
         }
      }
  }
   printf("\n");

   printf("--->Once filtered it comes out to be total %d combinations
possible <---",c1);
```

```
    printf("\n                Enter the weight or distance for each pair of
nodes : --");
    printf("\n");

     for(int j = 0;j<n;j++)
       {
           for(int i = 0;i<n;i++)
           {
              if(i!=j)
              {
                 printf("Enter the data for node pair :- [%c - %c]
\n",city[j],city[i]);
                 scanf("%d",&data[p]);
                 p++;
              }
           }
       }
 printf("\n");
 printf("\nInternally created a cost matrix. . . \n");
        for(i=0;i < n;i++)
        {


                for( j=0;j < n;j++)
                {
                if(i==j)
                {
                        ary[i][j]=0;
                }
                else
                {
                   ary[i][j]=data[n1];
```

```
                n1++;
            }
            }

            completed[i]=0;
        }



        printf("\n");

    printf(" ----NODE PAIR----  |  UNIQUE ID  |   DATA \n");

    for(int j = 0;j<n;j++)
      {
          for(int i = 0;i<n;i++)
          {
              if(i!=j)
              {
                  if(data[p1]!=0)
                  {
                      printf("    [%c -- %c]        |    %d        | %d
\n",city[j],city[i],p1,data[p1]);
                      p1++;
                  }

              }
          }
      }
      printf("\n");

      printf("\n\nThe cost list is:");

       for( i=0;i < n;i++)
```

```
        {
                printf("\n");

                for(j=0;j < n;j++)
                        printf("\t%d",ary[i][j]);
        }
        printf("\n\nThe selected weights/distance through using row wise
minimization are : -- ");


    printf("\n");

   printf("                              _____\n");
  printf("                             |
|\n");
  printf("                             | **Note : -- If there are more than two
node-pairs with same |\n");
  printf("                             |             weights/distance then u can go
for any of the  |\n");
  printf("                             |             one and proceed further.
|\n");
  printf("                             |_____|\n");


}

void mincost(int city)
{
        int i,ncity;

        completed[city]=1;

        printf("%d--->",city+1);
        ncity=least(city);
```

```
        if(ncity==999)
        {
                ncity=0;
                printf("%d",ncity+1);
                cost+=ary[city][ncity];

                return;
        }

        mincost(ncity);
}

int least(int c)
{
        int i,nc=999;
        int min=999,kmin;

        for(i=0;i < n;i++)
        {
                if((ary[c][i]!=0)&&(completed[i]==0))
                        if(ary[c][i]+ary[i][c] < min)
                        {
                                min=ary[i][0]+ary[c][i];
                                kmin=ary[c][i];
                                nc=i;
                        }
        }

        if(min!=999)
                cost+=kmin;

        return nc;
}
```

```
int main()
{
	takeInput();

	printf("\n\nThe Path is:\n");
	mincost(0); //passing 0 because starting vertex

	printf("\n\nMinimum cost is %d\n ",cost);

	return 0;
}
```

==========

REFERENCES

[1] Damitha Bandara - Assistant Professor of Supply Chain Management Albany State University, Albany, GA. USA College of Business ; Operation analyzing and Application on the travelling Salesman Problem. (https://airccse.com/oraj/papers/4417oraj02.pdf)

[2] Margaret Rouse - IT professionals at Whalts.comTechnical Writer and Director, WhatIs.com — TechTarget United States , application of Kruskal algorithm problem. (https://www.techtarget.com/contributor/Margaret-Rouse)

[3] Arpit Mishra - Empowering developers at HackerEarth , Implementation of Kruskal problem and TSP together with real life examples. (https://www.hackerearth.com/blog/developers/kruskals-minimum-spanning-tree-algorithm-example/)

[4] Frederick M. Dandure ,"National University of Science and Technology " (https://www.researchgate.net/publication/333672845_Route_Planning_using_The_Kruskal's_Algorithm_A_Case_of_Lobels_Bulawayo)

[5] Whitsett, D., (2017). Process Flow: Charts , Diagram and Development. Available at: https://study.com/academy/lesson/scrum-process-flow-diagram-development.html