ESTD : 2001
An Institute with a Difference

RNS Institute of Technology
Department of Information Science and Engineering

DESIGN AND ANALYSIS OF ALGORITHMS LABORATORY (18CSL47)

Bus Route system

Staff in Charge: Mrs. Kusuma S
Designation: Assistant Professor

Carried out by

Phanish SN(1RN19IS100)
Nidish G (1RN19IS094)
Neeraj Skanda BR (1RN19IS092)

ESTD : 2001
An Institute with a Difference

CONTENTS

ESTD : 2001
An Institute with a Difference

Abstract

- It represents simulation of real time Bus travel system with multiple routes and analyzing their results in different aspects.
- Our project uses basic functions to simulate real life Bus travel system by calculating the cost by using Dijkstra's shortest path algorithm and then does the processing based on appropriate conditions and then it presents the results out in a neat and understandable manner.

ESTD : 2001
An Institute with a Difference

Introduction

- **Our project gives user a real time feel of the bus travel system.**
- **It allows user to choose between multiple locations and then receives a route based on the best possible scenario.**
- **Our project showcases the real world application of Dijkstra's Algorithm.**

ESTD : 2001
An Institute with a Difference

Objective of the project

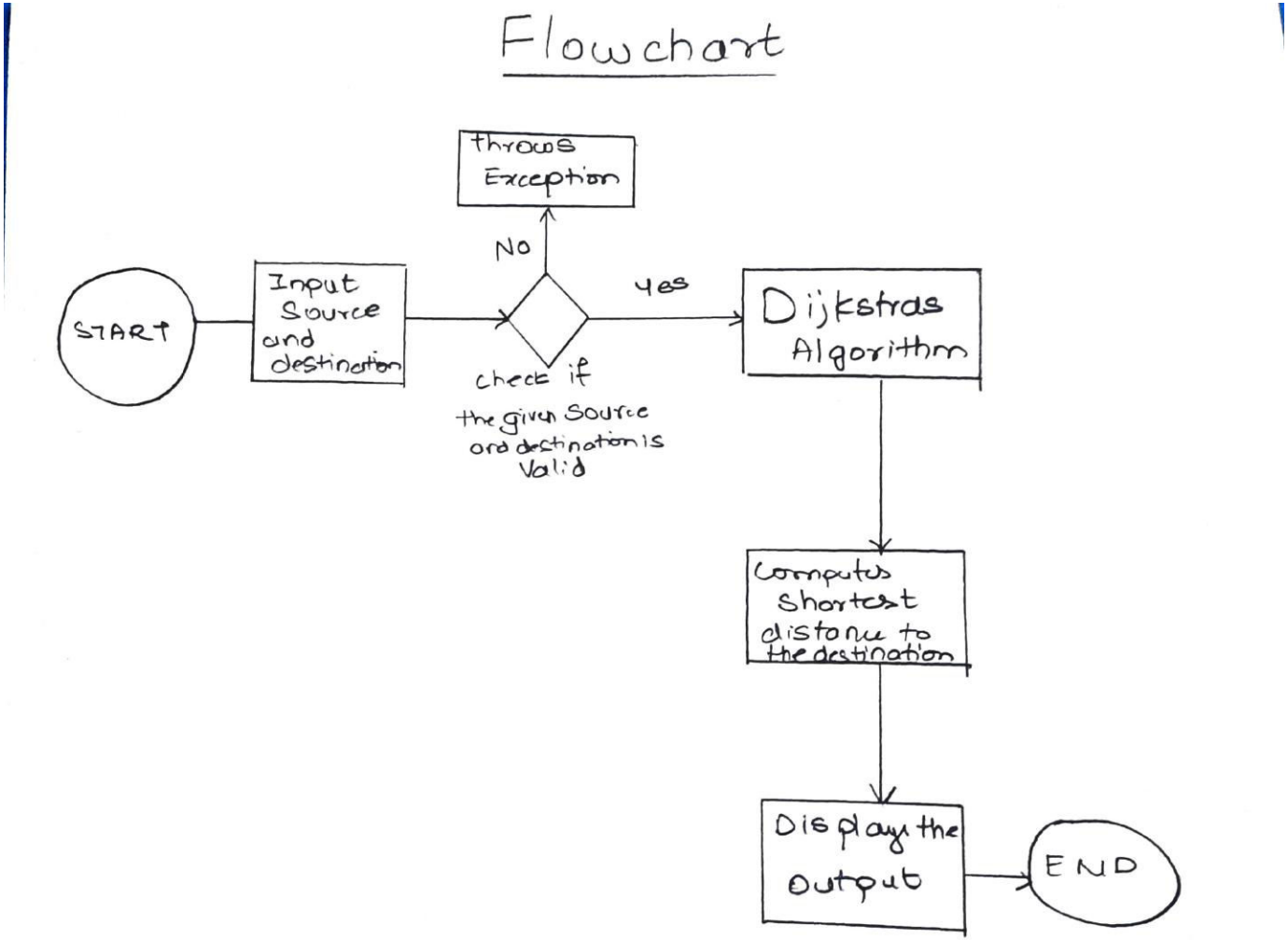
- The main objective of our project is to show how **Dijkstra's** works in a real world application use case.
- The main specifications of our project are:
  - To find the route based on least cost
  - To get real world usage of theory concepts
  - To use stack concept to check the bus in that region.

Algorithm design technique

- Mark all nodes unvisited. Create a set of all the unvisited nodes called the *unvisited set*.
- Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
- For the current node, consider all of its unvisited Neighbours and calculate their *tentative* distances through the current node. Compare the newly calculated *tentative* distance to the current assigned value and assign the smaller one. For example, if the current node *A* is marked with a distance of 6, and the edge connecting it with a neighbour *B* has length 2, then the distance to *B* through *A* will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, the current value will be kept.
- When we are done considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again.
- If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
- Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

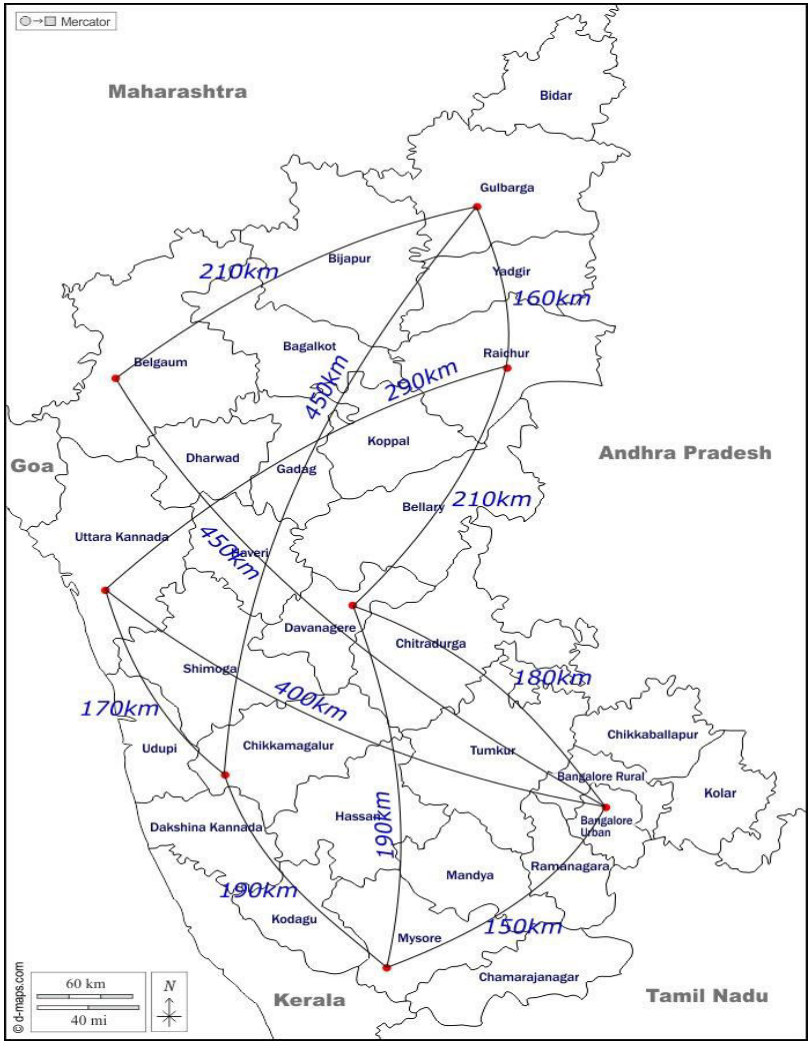
ESTD : 2001
An Institute with a Difference

Project architecture



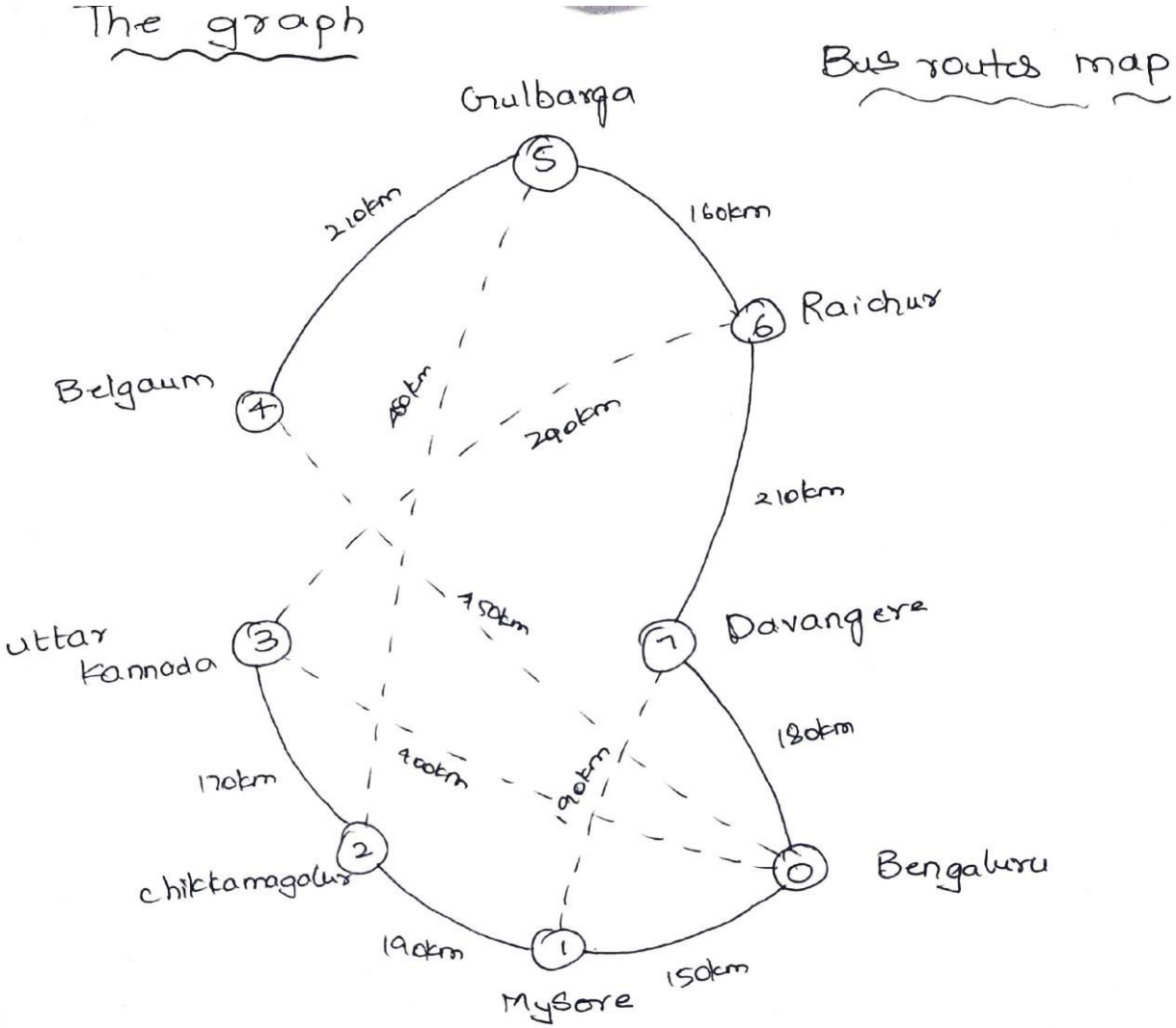
ESTD : 2001
An Institute with a Difference

Project architecture



ESTD : 2001
An Institute with a Difference

Project architecture



ESTD : 2001
An Institute with a Difference

Project architecture

Adjacency matrix

	0	1	2	3	4	5	6	7
0	0	150	0	400	450	0	0	180
1	150	0	190	0	0	0	0	190
2	0	190	0	170	0	450	0	0
3	400	0	170	0	0	0	290	0
4	450	0	0	0	0	210	0	0
5	0	0	450	0	210	0	160	0
6	0	0	0	290	0	160	0	210
7	180	190	0	0	0	0	210	0

ESTD : 2001
An Institute with a Difference

Implementaion modules

```
class Stack{
    int top=-1;
    int stackArray[]=new int[8];
    void push(int x)
    {
        stackArray[++top]=x;
    }
    int pop()
    {
        if(top==-1)
            return 0;
        return stackArray[top--];
    }
}
```

- Basic implement of stack data structure
- Class Stack does basic Push-Pop operation
- Maximum Stack array Value is 8

Implementaion modules

```
class ArrDepData{
    String Busname[]=new String[8];
    int BusNumber[]=new int[8];
    ArrDepData(String A[],int flno[])
    {
        Busname=A;
        BusNumber=flno;
    }
}
```

- Saves the data of a bus station in the class as object of this class
- Has members Busname and BusNumber and a constructor

ESTD : 2001
An Institute with a Difference

Implementaion modules

```
class VertexNames{
    String VertexNames[]=new String[8];
    VertexNames()
    {
        VertexNames[0]="BLR";//Bangalore
        VertexNames[1]="MYS";//Mysore
        VertexNames[2]="CML";//CHIKMANGALURU
        VertexNames[3]="UKA";//uTTAR KARNATAKA
        VertexNames[4]="BEL";//BELGAUM
        VertexNames[5]="GUL";//GULBURGA
        VertexNames[6]="RAI";//RAICHUR
        VertexNames[7]="DVG";//DAVANGERE
    }
```

- Class has VertexNames I.e. station names stored in "3 character" easily identifiable codes
- Has constructor and member functions
- getBusDepoasIndex()
- getBusDepoName()

Implementaion modules

```
int getBusDepoasIndex(String DepBuspt)
{
    int i=0;
    try {
        while(VertexNames[i].equalsIgnoreCase(DepBuspt)==false)
        {
            i++;
        }
        return i;
    }catch (Exception e)
    {
        System.out.println("Location not in the specific array or list of locations we have selected");
        System.exit(0);
    }
    return i;
}
```

- Takes input as string "3 character code" and converts it into int key value stored in the above array

ESTD : 2001
An Institute with a Difference

Implementaion modules

```
String getBusDepoName(String DepBust)
{
    switch(DepBust)
    {
    case "BLR":
        return "Bangalore";

    case "MYS":
        return "Mysore";

    case "CML":
        return "Chikkamagalur";

    case "UKA":
        return "Uttar Kannada";

    case "BEL":
        return "Belgaum";

    case "GUL":
        return "Gulbarga";

    case "RAI":
        return "Raichur";

    case "DVG":
        return "Davangere";

    default: return "Not Found";
    }
}
```

- It takes input as string of and then returns the full name of the specific bus station.

ESTD : 2001
An Institute with a Difference

Implementaion modules

```java
public class Buses {


    public static int tot_nodes=8;
    public static int tot_edges=12;
    public static int path[]=new int[8];
    static Scanner s=new Scanner(System.in);
    static VertexNames BUST=new VertexNames();
    static ArrDepData Schedule[]=new ArrDepData[8];
    static Stack Buffer=new Stack();
    static long MinimumTime;
```

```java
public static void main(String[] args){
    int i,j;
    long cost[][]=new long[8][8];
    long dist[]=new long[8];
    String DepartureBusTerminal;
    String ArrivalBusTerminal;
    System.out.print("\t\t\t\t\t\t\t-----BUS ROUTING AGENDA using Dijkstra's Algorithm----- \n\n");
    System.out.println("\t\t\t\t\t\t\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
    System.out.println("\t\t\t\t\t\t\t    -----------------------------------------\n");
    System.out.println("\t\t\t\t\t\t\t    ---Karnataka Bus Transportation Corporation---\n");
    System.out.println("\t\t\t\t\t\t\t    -----------------------------------------");
    create(cost);
    System.out.print("Enter the Departure Bus Stand code: ");
    DepartureBusTerminal=s.next();
    i=BUST.getBusDepoasIndex(DepartureBusTerminal);
    System.out.print("Enter the Destination BusTerminal code: ");
    ArrivalBusTerminal=s.next();
    int A=BUST.getBusDepoasIndex(ArrivalBusTerminal);
    System.out.println("\nBuses departing from  "+(BUST.VertexNames[i])+" BusTerminal to "+(BUST.VertexNames[A])+" are: \n");
    j=A;
    Dijkstra(cost,i,dist);
    if(dist[i]==1441)
        System.out.println("\nNo Path from "+BUST.VertexNames[i]+" to "+BUST.VertexNames[j]);
    else
        display(i,j,dist);
}
```

- It is the main class of the program that consists of the main function and all the other code.
- It has the printing of the display input, calling of the main Dijkstra algorithm and also calling of the display function

Implementation modules

- Create the cost matrix
- It also creates the data for the the Bus Locations telling which bus company is present at which specific location.
- The bus data is mainly just for decoration/ visual purposes.
- In real world this will be based on real time data

```
public static void create(long cost[][])
{
    int i,j;
    String Busname[];
    int BusNumber[];

    for(i=0;i<tot_nodes;i++)
    {
        for(j=0;j<tot_nodes;j++)
        {
            if(i==j)
                cost[i][j]=0;
            else
                cost[i][j]=1441;
        }
    }

    cost[0][1]=cost[1][0]=150;
    cost[0][3]=cost[3][0]=400;
    cost[0][4]=cost[4][0]=450;
    cost[0][7]=cost[7][0]=180;
    cost[1][2]=cost[2][1]=190;
    cost[1][7]=cost[7][1]=190;
    cost[2][3]=cost[3][2]=170;
    cost[2][5]=cost[5][2]=450;
    cost[3][6]=cost[6][3]=290;
    cost[4][5]=cost[5][4]=210;
    cost[5][6]=cost[6][5]=160;
    cost[6][7]=cost[7][6]=210;

    Busname=new String[] {"Airavat Bus    ","bRed Busways","Airavat Bus    "};
    BusNumber=new int[] {784,486,777,-1};
    Schedule[6]=new ArrDepData(Busname,BusNumber);

    Busname=new String[] {"bRed Busways","bRed Busways","bRed Busways","Airavat Bus    "};
    BusNumber=new int[] {433,223,213,197,-1};
    Schedule[7]=new ArrDepData(Busname,BusNumber);

    Busname=new String[] {"WeRL Buslines", "bRed Busways","Airavat Bus    ", "bRed Busways"};
    BusNumber=new int[] {566,311,259,448,-1};
    Schedule[4]=new ArrDepData(Busname,BusNumber);
```

- Actual creation of object above is done here and assigned to Schedule.

ESTD : 2001
An Institute with a Difference

Implementaion modules

- This is the main runner code of this program.
- Here the algorithm of Dijkstra algorithm is done.
- Initially put distance(time) from source to I.
- initialize minimum distance to max

- if(src[j]==0)//unvisited

- dist[v2]=dist[v1]+cost[v1][v2];//path is from source to v1 to v2

- path[v2]=v1;//path is via v1

```
public static void Dijkstra(long[][] cost, int source, long[] dist)
{
    int i,j,v1,v2;
    long minD;
    int src[]=new int[10];
    for(i=0;i<tot_nodes;i++)
    {
        dist[i]=cost[source][i];
        src[i]=0;
        path[i]=source;
    }
    src[source]=1;
    for(i=1;i<tot_nodes;i++)
    {
        minD=9999;
        v1=-1;
        for(j=0;j<tot_nodes;j++)
        {
            if(src[j]==0)
            {
                if(dist[j]<minD)
                {
                    minD=dist[j];
                    v1=j;
                }
            }
        }
        src[v1]=1;
        for(v2=0;v2<tot_nodes;v2++)
        {
            if(src[v2]==0)
            {
                if((dist[v1]+cost[v1][v2])<dist[v2])
                {
                    dist[v2]=dist[v1]+cost[v1][v2];
                    path[v2]=v1;
                }
            }
        }
    }
}
```

ESTD : 2001
An Institute with a Difference

Implementaion modules

- It is the main Display function which calls the show data function.
- The show data function shows the buses stationed in that specific station and then shows the destination that it goes to.

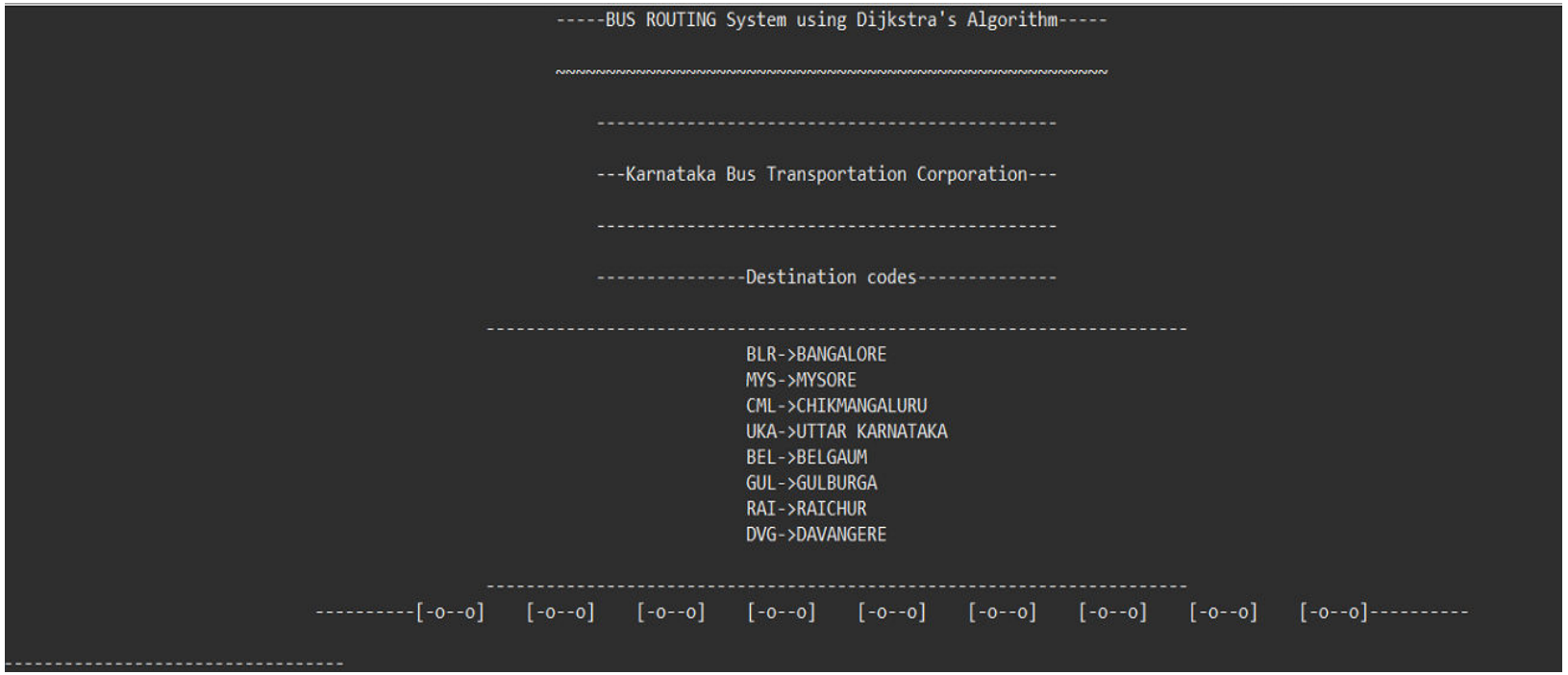
```
public static void display(int Source,int Destination,long dist[])
{
    int i;
    System.out.println("The route from "+BUST.VertexNames[Source]+" to "+BUST.VertexNames[Destination]+" is: \n");
    for(i=Destination;i!=Source;i=path[i])
    {
        System.out.print(BUST.VertexNames[i]+" <-- ");
        Buffer.push(i);
    }
    System.out.println(" "+BUST.VertexNames[i]);
    Buffer.push(i);
    System.out.println("\nThe Bus Details on your route are: \n");
    showData(Destination);
}

public static void showData(int dest)
{
    int i=Buffer.pop();
    while(i!=dest)
    {
        System.out.println(i);
        System.out.println("From BusTerminal "+BUST.VertexNames[i]+"\n\nBUS TERMINAL\t\t   DESTINATION\n_____");
        System.out.println();
        for(int j=0;Schedule[i].BusNumber[j]!=-1;j++)
        {
            int k=Buffer.pop();
            Buffer.push(k);
            System.out.println(Schedule[i].Busname[j]+" "+Schedule[i].BusNumber[j]+"\t    "+BUST.VertexNames[k] + "" +  BUST.getBusDepoName(BUST.VertexNames[k]));
        }
        i=Buffer.pop();
    }

    System.out.println();
    Buffer.pop();
}
```

ESTD : 2001
An Institute with a Difference

Results



- Displays the short 3 character codes for each of the destinations and a small ascii art of the buses

Results

- Shows the route that the bus takes
- It also highlights the bus terminal from and the bus terminal to and also the cost and destination name
- This is the output for a multiple routes with stops in between as seen.
- It goes from
- BLR-> DVG-> RAI -> GUL

```
Buses departing from  BLR BusTerminal to GUL are:

The route from BLR to GUL is:

GUL <-- RAI <-- DVG <--  BLR

The Bus Details on your route are:


From BUS TERMINAL----> BLR

BUS TERMINAL          TRAVEL COST     DESTINATION CODE   DESTINATION NAME


Airavat Bus  648        Rs 550/-          -DVG-              Davangere
Airavat Bus  448        Rs 750/-          -DVG-              Davangere
bRed Busways 742        Rs 600/-          -DVG-              Davangere
bRed Busways 445        Rs 800/-          -DVG-              Davangere
Airavat Bus  287        Rs 450/-          -DVG-              Davangere

From BUS TERMINAL----> DVG

BUS TERMINAL          TRAVEL COST     DESTINATION CODE   DESTINATION NAME


bRed Busways 433        Rs 800/-          -RAI-              Raichur
bRed Busways 223        Rs 650/-          -RAI-              Raichur
bRed Busways 213        Rs 700/-          -RAI-              Raichur
Airavat Bus  197        Rs 500/-          -RAI-              Raichur

From BUS TERMINAL----> RAI

BUS TERMINAL          TRAVEL COST     DESTINATION CODE   DESTINATION NAME
```

ESTD : 2001
An Institute with a Difference

Results

- This is the output for a Single route with no stops in between as seen.
- It goes from BLR-> MYS

```
---------------------------------
---------------------------------
Enter the Departure Bus Stand code: BLR

Enter the Destination BusTerminal code: MYS

_____________________________________

Buses departing from  BLR BusTerminal to MYS are:

The route from BLR to MYS is:

MYS <--  BLR

The Bus Details on your route are:

_____________________________________________________________________

From BUS TERMINAL----> BLR

BUS TERMINAL         TRAVEL COST     DESTINATION CODE   DESTINATION NAME

_____________________________________________________________________

Airavat Bus  648       Rs 550/-         -MYS-            Mysore
Airavat Bus  448       Rs 750/-         -MYS-            Mysore
bRed Busways 742       Rs 600/-         -MYS-            Mysore
bRed Busways 445       Rs 800/-         -MYS-            Mysore
Airavat Bus  287       Rs 450/-         -MYS-            Mysore
```

ESTD : 2001
An Institute with a Difference

Applications

- This project can be further developed into a full-fledged full stack web app with a some more effort and some code rebasing and translation into more web friendly languages.
- We hope to learn more about the algorithm by looking at one of its real world example

ESTD : 2001
An Institute with a Difference

Conclusion and future enhancements

- In our upcoming days, as we learn more about data structures, we plan to implement them and enhance this project further more.
- We are also open to any type of suggestions/advises .
- Permanent data storage and also add more limits to data inputs.
- Adding a web interface for ease of use.

References

- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
- https://github.com
- https://vtu.ac.in

ESTD : 2001

An Institute with a Difference

THANK YOU

```
package applicationofdijkstras;

import java.util.*;


class Stack{
 int top=-1;
 int stackArray[]=new int[8];
 void push(int x)
 {
  stackArray[++top]=x;
 }
 int pop()
 {
  if(top==-1)
   return 0;
  return stackArray[top--];
 }
}


class ArrDepData{
 String Busname[]=new String[8];
 int BusNumber[]=new int[8];
 int BusCost[]=new int[8];
 ArrDepData(String A[],int flno[],int C[])
 {
  Busname=A;
  BusNumber=flno;
  BusCost=C;

 }
}




class VertexNames{
 String VertexNames[]=new String[8];
 VertexNames()
 {
  //LOCATIONS
  VertexNames[0]="BLR";//BANGALORE
  VertexNames[1]="MYS";//MYSORE
  VertexNames[2]="CML";//CHIKMANGALURU
  VertexNames[3]="UKA";//uTTAR KARNATAKA
  VertexNames[4]="BEL";//BELGAUM
  VertexNames[5]="GUL";//GULBURGA
  VertexNames[6]="RAI";//RAICHUR
  VertexNames[7]="DVG";//DAVANGERE
 }
 int getBusDepoasIndex(String DepBuspt)
 {
  int i=0;
  try {
   while(VertexNames[i].equalsIgnoreCase(DepBuspt)==false)
   {
    i++;
   }
   return i;
  }catch (Exception e)
  {
   System.out.println("Location not in the specific array or list of locations we have selected");
   System.exit(0);
  }
  return i;
 }
```

```java
 String getBusDepoName(String DepBust)
 {
  switch(DepBust)
  {
  case "BLR":
   return "Bangalore";

  case "MYS":
   return "Mysore";

  case "CML":
   return "Chikkamagalur";

  case "UKA":
   return "Uttar Kannada";

  case "BEL":
   return "Belgaum";

  case "GUL":
   return "Gulbarga";

  case "RAI":
   return "Raichur";

  case "DVG":
   return "Davangere";

  default: return "Not Found";
  }
 }

}
public class Buses {


 public static int tot_nodes=8;
 public static int tot_edges=12;
 public static int path[]=new int[8];
 static Scanner s=new Scanner(System.in);
 static VertexNames BUST=new VertexNames();
 static ArrDepData Schedule[]=new ArrDepData[8];
 static Stack Buffer=new Stack();
 static long MinimumTime;


 public static void main(String[] args){
  int i,j;
  //test
  long cost[][]=new long[8][8];
  long dist[]=new long[8];
  String DepartureBusTerminal;
  String ArrivalBusTerminal;
  System.out.print("\t\t\t\t\t\t\t-----BUS ROUTING System using Dijkstra's Algorithm----- \n\n");
  System.out.println("\t\t\t\t\t\t\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
  System.out.println("\t\t\t\t\t\t\t    --------------------------------------------------\n");
  System.out.println("\t\t\t\t\t\t\t    ---Karnataka Bus Transportation Corporation---\n");
  System.out.println("\t\t\t\t\t\t\t    --------------------------------------------------\n");
  System.out.println("\t\t\t\t\t\t\t    ---------------Destination codes--------------\n");
  System.out.println("\t\t\t\t\t        ---------------------------------------------------------------------------");

  System.out.print("\t\t\t\t\t\t\t\t\t   BLR->BANGALORE\n"
    + "\t\t\t\t\t\t\t\t\t   MYS->MYSORE\r\n"
    + "\t\t\t\t\t\t\t\t\t   CML->CHIKMANGALURU\r\n"
    + "\t\t\t\t\t\t\t\t\t   UKA->UTTAR KARNATAKA\r\n"
    + "\t\t\t\t\t\t\t\t\t   BEL->BELGAUM\r\n"
    + "\t\t\t\t\t\t\t\t\t   GUL->GULBURGA\r\n"
```

```
   + "\t\t\t\t\t\t\t\t\t   RAI->RAICHUR\r\n"
   + "\t\t\t\t\t\t\t\t\t   DVG->DAVANGERE\n\n");
 System.out.println("\t\t\t\t\t         ---------------------------------------------------------------------");
 System.out.println("\t\t\t    ----------[-o--o]    [-o--o]    [-o--o]    [-o--o]    [-o--o]    [-o--o]    [-o--o]    [-o--o]----------   \n");
 create(cost);
 System.out.println("-----------------------------------");
 System.out.println("-----------------------------------");
 System.out.print("Enter the Departure Bus Stand code: ");
 DepartureBusTerminal=s.next();
 i=BUST.getBusDepoasIndex(DepartureBusTerminal);
 System.out.print("Enter the Destination BusTerminal code: ");
 ArrivalBusTerminal=s.next();
 int A=BUST.getBusDepoasIndex(ArrivalBusTerminal);
 System.out.println("_____________________________________________________________");
 System.out.println("\nBuses departing from  "+(BUST.VertexNames[i])+" BusTerminal to "+(BUST.VertexNames[A])+" are: \n");
 j=A;
 Dijkstra(cost,i,dist);
 if(dist[i]==1441)
  System.out.println("\nNo Path from "+BUST.VertexNames[i]+" to "+BUST.VertexNames[j]);
 else
  display(i,j,dist);
}


public static void create(long cost[][])
{
 int i,j;
 String Busname[];
 int BusNumber[];
 int BusCost[];

 for(i=0;i<tot_nodes;i++)
 {
  for(j=0;j<tot_nodes;j++)
  {
   if(i==j)
    cost[i][j]=0;
   else
    cost[i][j]=1441;
  }
 }

 cost[0][1]=cost[1][0]=150;
 cost[0][3]=cost[3][0]=400;
 cost[0][4]=cost[4][0]=450;
 cost[0][7]=cost[7][0]=180;
 cost[1][2]=cost[2][1]=190;
 cost[1][7]=cost[7][1]=190;
 cost[2][3]=cost[3][2]=170;
 cost[2][5]=cost[5][2]=450;
 cost[3][6]=cost[6][3]=290;
 cost[4][5]=cost[5][4]=210;
 cost[5][6]=cost[6][5]=160;
 cost[6][7]=cost[7][6]=210;

 Busname=new String[] {"VolvoLines   ","bRed Busways ","VolvoLines   "};
 BusNumber=new int[] {784,486,777,-1};
 BusCost=new int[] {450,650,500,-1};
 Schedule[6]=new ArrDepData(Busname,BusNumber,BusCost);

 Busname=new String[] {"bRed Busways ","bRed Busways ","bRed Busways ","VolvoLines   "};
 BusNumber=new int[] {433,223,213,197,-1};
 BusCost=new int[] {800,650,700,500,-1};
 Schedule[7]=new ArrDepData(Busname,BusNumber,BusCost);

 Busname=new String[] {"WeRL Buslines", "bRed Busways ","VolvoLines   ", "bRed Busways "};
 BusNumber=new int[] {566,311,259,448,-1};
```

```
 BusCost=new int[] {900,350,500,600,-1};
 Schedule[4]=new ArrDepData(Busname,BusNumber,BusCost);

 Busname=new String[] {"VolvoLines   ","VolvoLines   ","bRed Busways ","bRed Busways ","VolvoLines   "};
 BusNumber=new int[] {648,448,742,445,287,-1};
 BusCost=new int[] {550,750,600,800,450,-1};
 Schedule[0]=new ArrDepData(Busname,BusNumber,BusCost);

 Busname=new String[] {"WeRL Buslines","VolvoLines   ","bRed Busways ","bRed Busways ","VolvoLines   "};
 BusNumber=new int[] {124,667,446,824,334,-1};
 BusCost=new int[] {450,650,500,1000,700,-1};
 Schedule[1]=new ArrDepData(Busname,BusNumber,BusCost);

 Busname=new String[] {"WeRL Buslines", "VolvoLines   ","WeRL Buslines","WeRL Buslines","bRed Busways "};
 BusNumber=new int[] {156,187,934,438,555,-1};
 BusCost=new int[] {450,650,500,1200,600,-1};
 Schedule[5]=new ArrDepData(Busname,BusNumber,BusCost);

 Busname=new String[] {"VolvoLines   ","bRed Busways ", "WeRL Buslines", "VolvoLines   ","bRed Busways ","VolvoLines   "};
 BusNumber=new int[] {789,963,846,748,225,499,-1};
 BusCost=new int[] {450,650,500,700,400,900,-1};
 Schedule[2]=new ArrDepData(Busname,BusNumber,BusCost);

 Busname=new String[] {"bRed Busways ","bRed Busways ","WeRL Buslines", "VolvoLines   ","VolvoLines   ","VolvoLines   "};
 BusNumber=new int[] {986,45,965,102,202,333,-1};
 BusCost=new int[] {450,650,500,1300,1000,500,-1};
 Schedule[3]=new ArrDepData(Busname,BusNumber,BusCost);
}

public static void Dijkstra(long[][] cost, int source, long[] dist)
{
 int i,j,v1,v2;
 long minD;
 int src[]=new int[10];
 for(i=0;i<tot_nodes;i++)
 {
  dist[i]=cost[source][i];
  src[i]=0;
  path[i]=source;
 }
 src[source]=1;
 for(i=1;i<tot_nodes;i++)
 {
  minD=9999;
  v1=-1;
  for(j=0;j<tot_nodes;j++)
  {
   if(src[j]==0)
   {
    if(dist[j]<minD)
    {
     minD=dist[j];
     v1=j;
    }
   }
  }
  src[v1]=1;
  for(v2=0;v2<tot_nodes;v2++)
  {
   if(src[v2]==0)
   {
    if((dist[v1]+cost[v1][v2])<dist[v2])
    {
     dist[v2]=dist[v1]+cost[v1][v2];
     path[v2]=v1;
    }
   }
```

```
   }
  }
 }

 public static void display(int Source,int Destination,long dist[])
 {
  int i;
  System.out.println("The route from "+BUST.VertexNames[Source]+" to "+BUST.VertexNames[Destination]+" is: \n");
  for(i=Destination;i!=Source;i=path[i])
  {
   System.out.print(BUST.VertexNames[i]+" <-- ");
   Buffer.push(i);
  }
  System.out.println(" "+BUST.VertexNames[i]);
  Buffer.push(i);
  System.out.println("\nThe Bus Details on your route are: \n");
  showData(Destination);
 }

 public static void showData(int dest)
 {
  int i=Buffer.pop();
  while(i!=dest)
  {
//   System.out.println(i);
   System.out.println("_____________________________________________________________________________\n");
   System.out.println("From BUS TERMINAL----> "+BUST.VertexNames[i]+"\n\nBUS TERMINAL\t      TRAVEL COST
DESTINATION CODE\t DESTINATION NAME\n");
   System.out.println("_____________________________________________________________________________");
   System.out.println();
   for(int j=0;Schedule[i].BusNumber[j]!=-1;j++)
   {
    int k=Buffer.pop();
    Buffer.push(k);
    System.out.println(Schedule[i].Busname[j]+""+Schedule[i].BusNumber[j]+"        Rs "+Schedule[i].BusCost[j]+"/-\t  -
"+BUST.VertexNames[k] + "-\t\t     " +  BUST.getBusDepoName(BUST.VertexNames[k]));
   }
   i=Buffer.pop();
  }

  System.out.println();
  Buffer.pop();
 }
}
```