BUS ROUTING SYSTEM USING DIJKSTRA'S ALGORITHM

TABLE OF CONTENTS

INTRODUCTION

This 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.
- This project showcases the real world application of Dijkstra 's Algorithm.

OBJECTIVE OF THE PROJECT

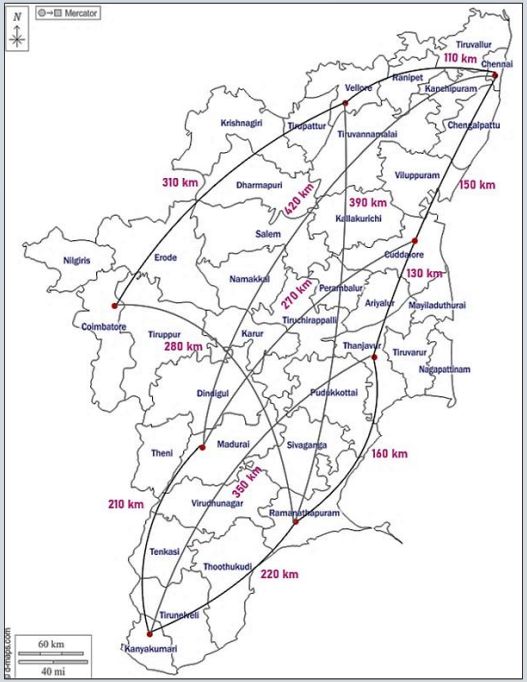
- The main objective of this project is to show how Dijkstra's works in a real world application use case.
- The main specifications of this project are:
    - To find the route based on least cost
    - To get real world usage of theory concepts
    - To use stack concept to check the bus in that region.

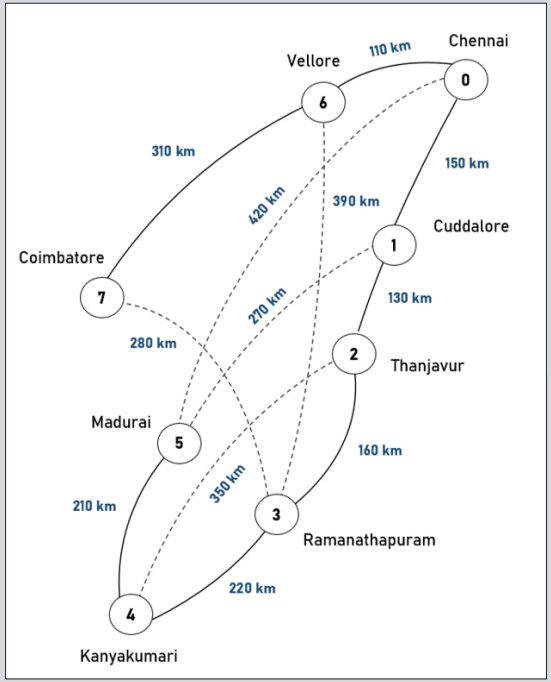
ALGORITHM DESIGN

- 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 neighbours 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.

PROJECT ARCHITECTURE





DIJKSTRA'S ALGORITHM

- Single-source shortest path problem:
  - No negative-weight edges: $w(u, v) > 0 \ \forall \ (u, v) \in E$
- Maintains two sets of vertices:
  - S = vertices whose final shortest-path weights have already been determined
  - Q = vertices in V – S: min-priority queue
    - Keys in Q are estimates of shortest-path weights (d[v])
- Repeatedly select a vertex $u \in V - S$, with the minimum shortest-path estimate d[v]

STEPS IN DIJKSTRA'S ALGORITHM

STEP 1:

Mark all nodes tentative , set distances from source to 0(Zero) for source and ∞(infinity) for all other nodes

STEP 2:

While tentative nodes remain :

a) Extract N , a node with lowest distance

b) Add link to N to the shortest path tree

c) Relax the distances of neighbors of N by lowering any better distance estimates

IMPLEMENTAION

```
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

```
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;
    }
  }
```

- Saves the data of a bus station in the class as object of this class
- Has members Bus name and Bus Number and a constructor

```
class VertexNames{

  String VertexNames[]=new String[8];
  VertexNames()
  {
      //LOCATIONS
      VertexNames[0]="CHN";//CHENNAI
      VertexNames[1]="CUD";//CUDDALORE
      VertexNames[2]="TNJ";//THANJAVUR
      VertexNames[3]="RNP";//RAMANATHAPURAM
      VertexNames[4]="KAN";//KANYAKUMARI
      VertexNames[5]="MDU";//MADURAI
      VertexNames[6]="VEL";//VELLORE
      VertexNames[7]="CBE";//COIMBATORE
  }
```

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

```
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 array

```
String getBusDepoName(String DepBust)
{
    switch(DepBust)
    {
      case "CHN":
          return "CHENNAI";
      case "CUD":
          return "CUDDALORE";
      case "TNJ":
          return "THANJAVUR";
      case "RNP":
          return "RAMANATHAPURAM";
      case "KAN":
          return "KANYAKUMARI";
      case "MDU":
          return "MADURAI";
      case "VEL":
          return "VELLORE";
      case "CBE":
          return "COIMBATORE";
      default: return "Not Found";
    }
}
}
```

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

```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;
```

- 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.

```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 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 -----TAMIL NADU 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\t ------------------------------------------------------------------
    System.out.println("\t\t\t\t ----------[-o--o] [-o--o] [-o--o] [-o--o] [-o--o] [-o--o] [-o--o] [-o--o] [
    create(cost);
    System.out.println("------------------------------------------------");

    System.out.print("Enter the Departure Bus Terminal code: ");
    DepartureBusTerminal=s.next();
    i=BUST.getBusDepoasIndex(DepartureBusTerminal);

    System.out.print("Enter the Destination Bus Terminal 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
    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);
}
```

- Create the cost matrix
- It also creates the data for the Bus Locations telling which bus company is present at which specific location.
- The bus data is mainly just for visual purposes.
- In real world this will be based on real time data
- Actual creation of object above is done here and assigned to Schedule.

```
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][5]=cost[5][0]=420;
    cost[0][6]=cost[6][0]=110;
    cost[1][2]=cost[2][1]=130;
    cost[1][5]=cost[5][1]=270;
    cost[2][3]=cost[3][2]=160;
    cost[2][4]=cost[4][2]=350;
    cost[3][4]=cost[4][3]=220;
    cost[3][6]=cost[6][3]=390;
    cost[3][7]=cost[7][3]=280;
    cost[4][5]=cost[5][4]=210;
    cost[6][7]=cost[7][6]=310;

   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};
```

- This is the main runner code of this program.
- Here the algorithm of Dijkstra algorithm is done.
- Initially put distance(time) from source to l.
- Initialize minimum distance to max
  - if(src[j]==0) //unvisited
  - dist[v_2] = dist[v_1] + cost[v_1][v_2]

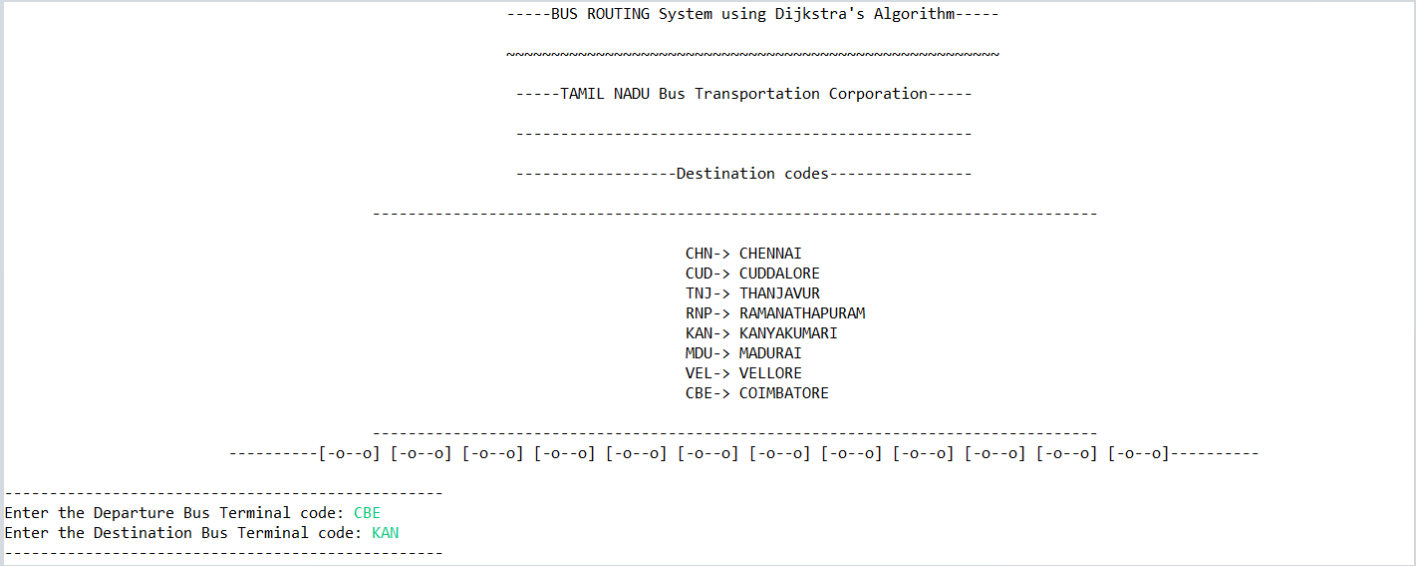
    //path is from source to v_1 to v_2
  - path[v2] = v1 //path is via v_1

```
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;
                }
            }
        }
    }
}
```

- 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: ");
       System.out.println("_____________________________________________________________________________");
       showData(Destination);
       System.out.println("_____________________________________________________________________________");
   }

public static void showData(int dest)
{
    int i=Buffer.pop();
    while(i!=dest)
    {
        // System.out.println(i);
        System.out.println("\n From BUS TERMINAL ----> "+BUST.VertexNames[i]+"\n\n     BUS TERMINAL\t TRAVEL COST   DESTINATION C
        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]+"\t   Rs "+Schedule[i].BusCost[j]+"/-\t
        }

        i=Buffer.pop();
        System.out.println(" _ _____________________________________________________________ _");
    }

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

RESULT



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

- 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

CBE -> RNP -> KAN

```
Buses departing from CBE BusTerminal to KAN are:

The route from CBE to KAN is:

KAN <--> RNP <--> CBE

The Bus Details on your route are:
```

From BUS TERMINAL ----> CBE

BUS TERMINAL		TRAVEL COST	DESTINATION CODE	DESTINATION NAME
bRed Busways	433	Rs 800/-	-RNP-	RAMANATHAPURAM
bRed Busways	223	Rs 650/-	-RNP-	RAMANATHAPURAM
bRed Busways	213	Rs 700/-	-RNP-	RAMANATHAPURAM
VolvoLines	197	Rs 500/-	-RNP-	RAMANATHAPURAM

From BUS TERMINAL ----> RNP

BUS TERMINAL		TRAVEL COST	DESTINATION CODE	DESTINATION NAME
bRed Busways	986	Rs 450/-	-KAN-	KANYAKUMARI
bRed Busways	45	Rs 650/-	-KAN-	KANYAKUMARI
WeRL Buslines	965	Rs 500/-	-KAN-	KANYAKUMARI
VolvoLines	102	Rs 1300/-	-KAN-	KANYAKUMARI
VolvoLines	202	Rs 1000/-	-KAN-	KANYAKUMARI
VolvoLines	333	Rs 500/-	-KAN-	KANYAKUMARI

- This is the output for a single route with no stops in between as seen.
- It goes from

  VEL -> CHN

```
-------------------------------------------------
Enter the Departure Bus Terminal code: VEL
Enter the Destination Bus Terminal code: CHN
-------------------------------------------------

Buses departing from VEL BusTerminal to CHN are:

The route from VEL to CHN is:

CHN <--> VEL

The Bus Details on your route are:
```

From BUS TERMINAL ----> VEL

BUS TERMINAL		TRAVEL COST	DESTINATION CODE	DESTINATION NAME
VolvoLines	784	Rs 450/-	-CHN-	CHENNAI
bRed Busways	486	Rs 650/-	-CHN-	CHENNAI
VolvoLines	777	Rs 500/-	-CHN-	CHENNAI

FUTURE ENHANCEMENTS

- 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.
- In our upcoming days, we plan to 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.

THANK YOU