NOTE TO USERS

This reproduction is the best copy available.

UMI
Application Of The Chinese Postman Problem Model To The Toronto Transportation Network Within GIS-Based Software

BY

SLAVENKO CUGALJ

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science
Graduate Department of Civil Engineering
University of Toronto

© Copyright by Slavenko Cugalj 2001
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
Abstract

Application Of The Chinese Postman Problem Model And Its Application To The Toronto Transportation Network Within GIS-Based Software

Master of Applied Science, 2001
Slavenko Cugalj
Graduate Department of Civil Engineering, University of Toronto

The theory of transportation networks as a discipline has been developing for decades, always evoking interest in researchers around the world.

Depending on the transportation network’s design and on the organization of the transportation on the network, infrastructural costs, and the costs related to providing transportation can be greatly reduced on the one hand and the level of service increased on the other. Therefore, the application of existing theoretical models, their modification depending on the characteristics of the network, and the creation of new models can not only improve the level of service but also considerably increase economic productivity of the entire transportation system.

In order to make a contribution to the application of the variety of theoretical ideas developed with the respect to a transportation network, this project presents the author’s development of a model for solving the Chinese Postman Problem utilizing a Geographic Information System tools and then of its application to a transportation network.
Acknowledgment

First and foremost, I would like to send special thanks to my supervisor professor Eric Miller for his interest, guidance, and support throughout this project.

Thanks also to professor Amer Shalaby as the second reader of my thesis, and to Byron Moldofsky from the Geography Department for his support and help in getting the data set necessary for the application of the model.

A special thanks to my friend Ljubomir Ostojic for his early enthusiasm, extraordinary dedication, help, and encouragement through this project.

My gratitude goes to my friends Toni, Anya, Tina, and Maria for being so much fun to work with, and to Boba, Lana, and Natasa for sharing ideas with me, and for helping me to be a happy person.

And finally, a sincere thanks to my friend Vala Mehdi-Nejad for bringing hundreds of memories alive, and to my mom for bringing so much joy and laughter to my life.
Table of Contents

Abstract........................................................................................................... i
Acknowledgment.......................................................................................... ii
Table of Contents .......................................................................................... iii
List of Figures ............................................................................................... v
List of Tables ................................................................................................. vi

Chapter 1. Introduction................................................................................... 1

Chapter 2. Background on the Chinese Postman Problem and Transportation Networks......................................................... 3
  2.1 Basic Definitions in Transportation Network Theory ......................... 3
  2.2 Oriented, Nonoriented, and Mixed Networks.................................... 4
  2.3 The Chinese Postman Problem and Euler’s Theorem ....................... 6
  2.4 The Chinese Postman Problem on an Undirected Graph ............... 7
  2.5 Euler’s Theorem .............................................................................. 8
  2.6 Proof of the Euler Theorem ............................................................... 9
  2.7 Application of Euler Tour to Solve the Chinese Postman Problem ... 10
  2.8 The Chinese Postman Problem for an Oriented Network .......... 11
  2.9 Chinese Postman Problem Algorithm for an Oriented Network .... 11
  2.10 Application of the Chinese Postman Problem Algorithm for an Oriented Network......................................................... 13
  2.11 The Chinese Postman Algorithm for an Nonoriented Network ..... 18
  2.12 Chinese Postman Problem Algorithm for a Nonoriented Network... 18
  2.13 Manual Minimum Length Matching.............................................. 21
  2.14 Drawing an Euler Tour ................................................................. 23
  2.15 The Mixed Postman Problem......................................................... 24
  2.16 The Rural Postman Problem ......................................................... 26
  2.17 Windy Postman Problem .............................................................. 27
  2.18 Postman Problem with Branch Priorities.................................... 27
  2.19 Maximum Benefit Postman Problem........................................... 27
### Chapter 3. Development Of The Model .............................................................. 29
- 3.1 Difference Between Arc and Node Routing ............................................. 30
- 3.2 Information System Design ...................................................................... 31
- 3.3 Routing and Geographic Information Systems ......................................... 32
- 3.4 Raster and Vector Format of the Map ....................................................... 35
- 3.5 Classification of Arc Routing and Scheduling Problems .......................... 36
- 3.6 Development of PKPSsoft Model .............................................................. 39
- 3.7 Model Description ..................................................................................... 42
  - 3.7.1 Form Area .......................................................................................... 43
  - 3.7.2 Form Region ....................................................................................... 44
  - 3.7.3 Form Open Region ............................................................................ 46
  - 3.7.4 Form Map .......................................................................................... 47
  - 3.7.5 Menu Options ..................................................................................... 49
  - 3.7.6 Creation of a Transportation Network in the Form Map .................... 52
  - 3.7.7 Presentation of the Solution to the Chinese Postman Network Problem ................................................................................................................. 55

### Chapter 4. Results and Discussion ................................................................. 58
- 4.1 Postal Delivery Routing Problem In Canada ............................................. 58
- 4.2 Application of the Model to the Toronto Transportation Network .......... 60
  - 4.2.1 Two Way Streets ............................................................................... 69
  - 4.2.2 Streets for which There is no Delivery ............................................... 70

### Chapter 5. Summary and Conclusion ........................................................... 77

### References ...................................................................................................... 80

### Appendix ......................................................................................................... 82
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>An Oriented Network</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>A nonoriented network</td>
<td>4</td>
</tr>
<tr>
<td>2.3</td>
<td>A mixed network</td>
<td>4</td>
</tr>
<tr>
<td>2.4</td>
<td>Graph $G = (N, A)$</td>
<td>6</td>
</tr>
<tr>
<td>2.5</td>
<td>Seven-bridge problem at Konigsberg</td>
<td>7</td>
</tr>
<tr>
<td>2.6</td>
<td>Graphs that illustrate Euler's Theorem</td>
<td>8</td>
</tr>
<tr>
<td>2.7</td>
<td>An oriented network for solving the Chinese Postman Problem</td>
<td>13</td>
</tr>
<tr>
<td>2.8</td>
<td>Chinese Postman Network with Zero Polarity</td>
<td>17</td>
</tr>
<tr>
<td>2.9</td>
<td>The Network for Minimum Length Pairwise Matching</td>
<td>19</td>
</tr>
<tr>
<td>2.10</td>
<td>Matching of Odd Degree Nodes</td>
<td>21</td>
</tr>
<tr>
<td>2.11</td>
<td>Possible Matching of Odd Degree Nodes</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>Transaction-based Information System Design</td>
<td>31</td>
</tr>
<tr>
<td>3.2</td>
<td>Routing and Geographic Information System</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>Area and Region Forms of the PKPSof3 Program</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>Form Open Region</td>
<td>47</td>
</tr>
<tr>
<td>3.5</td>
<td>Form Map as a GIS Feature</td>
<td>48</td>
</tr>
<tr>
<td>3.6</td>
<td>Network’s Pop Down Menu in the Form Map</td>
<td>49</td>
</tr>
<tr>
<td>3.7</td>
<td>Street’s Pop Down Menu in the Form Map</td>
<td>50</td>
</tr>
<tr>
<td>3.8</td>
<td>Nodes’ Pop Down Menu in the Form Map</td>
<td>51</td>
</tr>
<tr>
<td>3.9</td>
<td>The Image of the Created Network</td>
<td>54</td>
</tr>
<tr>
<td>3.10</td>
<td>Solution to Chinese Postman Problem</td>
<td>56</td>
</tr>
<tr>
<td>4.1</td>
<td>The Map of the MST Postal Code in the Downtown Toronto</td>
<td>63</td>
</tr>
<tr>
<td>4.2</td>
<td>The Creation of the Network</td>
<td>64</td>
</tr>
<tr>
<td>4.3</td>
<td>Solution to the Network Problem</td>
<td>65</td>
</tr>
<tr>
<td>4.4</td>
<td>Euler’s Path Solution to the Original Network Problem</td>
<td>67</td>
</tr>
<tr>
<td>4.5</td>
<td>Euler’s Tour for the Original Network</td>
<td>68</td>
</tr>
<tr>
<td>4.6</td>
<td>Increase of Permutations with a Number of No-Delivery Streets</td>
<td>71</td>
</tr>
<tr>
<td>4.7</td>
<td>The Euler’s Path for the Modified Network</td>
<td>74</td>
</tr>
</tbody>
</table>
List of Tables

Table 2.1 Polarity of nodes ...........................................................................................................13
Table 2.2 Optimum Solution Summaries .......................................................................................15
Table 4.1 The Dataset of the Network M5T ..................................................................................62
Table 4.2 Number of Permutations for k Streets with no Delivery ..............................................71
Table 4.3 Optimal Tour for Different Variations ..........................................................................72
Table 4.4 The Summary of Results ...............................................................................................75
Chapter 1. Introduction

Transportation problems related to logistics and distribution management have encouraged the development of transportation modelling procedures. Routing problems are common in several different areas of transportation logistics. A subset of routing problems is the Arc Routing Problem where arcs of a transportation network are to be served.

Strategic transportation modelling in vehicle and crew scheduling has played an important role in transportation planning for a variety of network problems. The solutions obtained for different problems related to transportation networks have made an essential contribution to the improvement of the transportation activities and operations of an entire transportation system.

Following the theoretical background related to the theory of transportation networks and the Chinese Postman Problem and its algorithms, a detailed description of the model development process is presented in this study. The performance of the model is carried out through a transportation network and solution obtained are analysed from the perspective of further research in this area.

This thesis consists of five chapters that are organized as follows:

Chapter 2: Explains a transportation network’s basic definitions and the Chinese Postman Problem (CPP) and provides a theoretical background, including solutions for a variety of problem modifications in the form of algorithms.

Chapter 3: Focuses on Arc and node routing differences and classification, Geographic Information System features, and a detailed description of the model development.
Chapter 4: Contains a description of a database used in the study, and the application of the model to a transportation network

Chapter 5: Provides a summary and conclusion
Chapter 2. Background on the Chinese Postman Problem and Transportation Networks

The solution to the Chinese Postman Problem (CPP) and its different modifications can be obtained efficiently by using algorithms and procedures outlined in this chapter. The material on Euler theorem, and CPP algorithms are covered in details in Edmonds and Johnson [6], Teodorovic [19], Bodin and Golden [4], and Golden and Assad [7], whereas the formulations of the different extension of the CPP are presented in details in Malandraki [11], Pearn [14], and Win [20].

2.1 Basic Definitions in Transportation Network Theory

A transportation network can be defined as a set of nodes and a set of arcs on which traffic operations occur. Depending on the transportation area, nodes in a transportation network can be interpreted as cities, intersections, postal terminals, airports, and railway and bus terminals. The connections between nodes can be streets, airways, railway tracks etc. The connections between nodes are called arcs. A transportation network denoted by \( G (N, A) \) represents a graph \( G \) consisting of a set of nodes, denoted by \( N \), and a set of arcs between these nodes, denoted by \( A \). The notation \((i, j)\) denotes the arc that connects node \( i \in N \) with node \( j \in N \).

Transportation networks and graphs are denoted in the same manner and the only difference between the two is the relationship between nodes and arcs. A graph denotes the structural relationship between arcs and nodes whereas a network is a graph which has quantitative relationships between arcs and nodes.
2.2 Oriented, Nonoriented, and Mixed Networks

If all arcs \((i, j)\) of the network lead from node \(i\) to node \(j\), the network is called an oriented network. A nonoriented network is the network whose branches are not oriented. If some of the branches in the network are oriented and some nonoriented, the network is called a mixed network. All three types are shown below in Figure 2.1, Figure 2.2, and Figure 2.3 respectively.

![Figure 2.1 An Oriented Network](image1)

![Figure 2.2 A Nonoriented Network](image2)

![Figure 2.3 A Mixed Network](image3)

The in-degree of a node in an oriented network represents the number of arcs which enter that particular node. The number of arcs that leave the particular node in an oriented network is called the out-degree of a node. In a nonoriented network, however, the degree of a node represents the number of the arcs that link that node to the other nodes in the network.
In a nonoriented network, on the one hand, the path from node i to node j includes all arcs and nodes it has to go over when passing from node i to node j, and therefore it can be defined by counting either nodes or arcs passed. On the other hand, in an oriented network the path is called a chain. For example, the path (2, 8, 4) denotes the path which leads from node 2 to node 4 through node 8. If we are concerned with arcs, this path can be denoted by ((2, 8), (8, 4)) which means that the path goes from node 2 to node 4 passing through arcs (2, 8) and (8, 4).

A tour that starts and ends at the same node is called a cycle. The starting node on the path is called the source node whereas the finishing node on the path is called the destination node. A simple path is one in which individual arcs appear only once. An elementary path is a path on which each node appears only once.

If there is a path leading from node i to node j, the nodes i and j are said to be connected. A nonoriented network is said to be connected if there is a corresponding path among all pairs of nodes. Since an oriented network has no paths among all nodes due to its oriented character, it is said to be connected if a corresponding nonoriented network (obtained from the oriented network by removing the orientations) is connected. An oriented network that happens to have paths between all pairs of nodes is called a strongly connected oriented network.

Complexity of a transportation network as a system can be addressed by the use of operations research methods, and graph theory. Depending on the problem of the network, the optimal solution can be the shortest path, the longest path, the path with the highest capacity, the cheapest path, the most reliable path, etc., and the length of the connection can be a street length, travel time, travel cost, reliability, etc.
2.3 The Chinese Postman Problem and Euler’s Theorem

Consider the graph $G$ consisting of the set of nodes $N$ and the set of arcs $A$,

$G = (N, A)$ shown in Figure 2.4.

![Figure 2.4 Graph $G = (N, A)$](image)

Assume we are asked to design the routes for a postman, snow removal, or garbage truck on a given map of the city. In addition, every road in the city has to be traversed at least once to ensure that all deliveries or pick-ups are made. The objective is to minimize either the total distance or the total drive time. The problem can be described as the children’s challenge to draw completely the given object without lifting the pen off the paper and without repeating any arc.

The Chinese Postman Problem has an interesting history. There was a parade in the Russian city of Konigsberg, located on the river Pregel. The parade was supposed to cross all seven bridges, which connected two islands with the river banks as shown in Figure 2.5.
The Swiss mathematician Euler had a closer look at this problem, and he tried to find the way the parade could cross all seven bridges shown in Figure 2.5 only once. In 1736 he proved that no solution could be found for this particular routing problem.

At this time, however, algorithms for the Chinese Postman Problem exist for both an oriented graph and for a nonoriented graph.

2.4 The Chinese Postman Problem on an Undirected Graph

Assume that an undirected graph is given with known arc lengths $l(i, j) > 0$ for all arcs $(i, j) \in A$. Then the Chinese Postman Problem can be stated as follows:

Find a route on a given graph $G$ on which every arc is traversed at least once and the sum

$$\sum_{all (i,j) \in A} n(i,j) \cdot l(i,j) \rightarrow \min$$  \hspace{1cm} (2.1)

where $n(i,j)$ represents the number of times arc $(i, j)$ is traversed.

In order to describe the process of getting a solution to the Chinese Postman Problem, we first need to define what is meant by an Euler tour and Euler path.
An Euler tour is a tour which traverses each arc on the graph exactly once starting and finishing at the same node. An Euler path is a path which traverses each arc on the graph exactly once starting and finishing at the specified nodes.

**2.5 Euler's Theorem**

Theorem: A connected graph $G$ has an Euler tour (Euler path) if and only if $G$ contains exactly zero (exactly two) nodes of odd degree.

To illustrate Euler's Theorem consider Figure 2.6.

![Figure 2.6 Graphs that Illustrate Euler's Theorem](image)
Figure 2.6a shows that all nodes are with even degree and, therefore, this graph has an Euler tour. Such a tour can be A, B, C, D, B, C, A. The graph in figure 2.6b has exactly two nodes with odd degree and, therefore, according to the theorem, it possesses an Euler path. Such a path can be D, C, A, B, C, E, B, D, E. Finally, the graph in figure 2.6c has all four nodes with odd degree (more than two) and, therefore, it doesn’t possess either Euler’s tour or Euler’s path. Notice that graph in Figure 2.6c is a network model of the Konigsberg seven-bridges problem.

2.6 Proof of the Euler Theorem

It can be shown that a graph G that has zero nodes of odd degree is a sufficient and necessary condition for an Euler tour to exist. It is necessary because an even number of arcs is required for every node on the graph, since an Euler tour drawn on the graph must always enter a node through one arc and exit through another, and all arcs on the graph must be used exactly once.

Sufficiency will be shown through the construction of a tour on the graph.

On a given graph G = (N, A), we start to draw the circuit C₀ from some initial node j₀ ∈ N through all nodes and arcs and eventually return to j₀. If such a circuit happens to be a Euler tour, we stop.

If a circuit C₀ does not contain all arcs from the graph G, then we form the other graph G₁ consisting of arcs not included in C₀. Since the assumption was that a graph G is connected and all its nodes are of even degree, then at least two arcs not traversed by C₀ have to be connected to the node j₁ that has been passed by C₀ (the C₀ has used up only an even number of arcs connected to j₁). By eliminating arcs of C₀ from G we get G₁
on which we draw another circuit $C_1$. The circuit $C_1$ starts and terminates in node $j_1$. If $C_1$ has used up all arcs of $G_1$, we stop. In that case, an Euler tour is the initial portion of circuit $C_0$ from node $j_0$ to node $j_1$, then a circuit $C_1$ beginning and returning to the node $j_1$, and finally the second portion of circuit $C_0$ from node $j_1$ to node $j_0$.

If we still have uncovered arcs on the graph, this procedure can be repeated until the $k^{th}$ step which results in all arcs being covered. The Euler tour, at that time, would be a combination of circuits $C_0, C_1, C_2, \ldots, C_k$.

### 2.7 Application of Euler Tour to Solve the Chinese Postman Problem

If a given graph $G$ has an Euler tour that is, simply, the solution to the Chinese Postman Problem. However, Euler's theorem provides guidelines to solve the Chinese Postman Problem on a given graph where an Euler tour does not exist. The idea for solving this problem is to modify an existing graph in order to transform odd-degree nodes to even-degree nodes by adding artificial arcs parallel to existing ones in the graph. In that case, a graph $G = (N, A)$ transforms to $G^1 = (N, A^1)$ on which an Euler tour can be drawn.

Now the solution for the Chinese Postman Problem in both an oriented and nonoriented network will be presented.
2.8 The Chinese Postman Problem for an Oriented Network

This problem was solved by Beltrami and Bodin in 1974. They proved that the Chinese Postman Problem can be solved and an Euler tour can be found if and only if the in-degree of each node is equal to the out-degree of the same node. They also defined the node polarity as the difference between the in-degree and out-degree of the same node.

Depending on the polarity we can define supply nodes and demand nodes. Nodes $n_j$ whose in-degree is greater than out-degree are called supply nodes, whereas nodes whose out-degree is greater than in-degree are called demand nodes, $m_k$. The difference between the in-degree and out-degree of the supply nodes is denoted by $s_j$. The polarity of the demand nodes is denoted by $d_k$.

2.9 Chinese Postman Problem Algorithm for an Oriented Network

The following algorithm finds an Euler tour and therefore represents the solution to Chinese Postman Problem for the network concerned.

STEP1: Find all supply nodes $n_j$ and demand nodes $m_k$.

STEP 2: Find the shortest paths $d_{jk}$ from all supply nodes $n_j$ to all demand nodes $m_k$.

STEP 3: In order to find the optimal pairwise matching of supply nodes with demand nodes solve the linear programming problem that reads as follows:
\[
F = \sum_{j} \sum_{k} d_{jk} x_{jk} \rightarrow \min
\]

(2.2)

\[
\sum_{k} x_{jk} = s_j, \ \forall \ j
\]

(2.2a)

\[
\sum_{j} x_{kj} = d_k, \ \forall \ k
\]

(2.2b)

\[
x_{jk} \geq 1
\]

(2.2c)

STEP 4: For every \( x_{jk} \geq 1 \) obtained as a solution to the linear programming problem, we add \( x_{jk} \) artificial arcs parallel to the existing shortest branch from supply node \( n_j \) to demand node \( m_k \). The new network \( G^1 \) obtained this way has a polarity of all its nodes equal zero.

STEP 5: Find an Euler tour for the network \( G^1 \). An Euler tour represents the solution to the Chinese Postman Problem on an oriented network.
2.10 Application of the Chinese Postman Problem Algorithm for an Oriented Network

Let's apply this algorithm and solve Chinese Postman Problem on the transportation network shown in Figure 2.7. The solution will be the tour starting and ending in node A.

![Figure 2.7 An Oriented Network for Solving the Chinese Postman Problem](image)

In order to find the supply and demand nodes, we need to determine the polarity of nodes. In-degree, out-degree, and subsequently the polarity of nodes shown in Figure 2.7 are presented in Table 2.1.

<table>
<thead>
<tr>
<th>Node i</th>
<th>In-degree</th>
<th>Out-degree</th>
<th>Polarity</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>2</td>
<td>-1</td>
<td>D</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>3</td>
<td>-1</td>
<td>D</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>S</td>
</tr>
<tr>
<td>f</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>S</td>
</tr>
</tbody>
</table>

Table 2.1 Polarity of nodes
As indicated in Table 2.1, we can define two sets of nodes: a set of supply nodes $S$ and a set of demand nodes $D$.

$$S = [e, f]$$

$$D = [a, b]$$

The shortest paths from the nodes of set $S$ to the nodes of set $D$ are:

$$d_{E, A} = 7$$

$$d_{E, B} = d_{E, C} + d_{C, B} = 4 + 5 = 9$$

$$d_{F, A} = d_{F, E} + d_{E, A} = 8 + 7 = 15$$

$$d_{F, B} = d_{F, E} + d_{E, C} + d_{C, B} = 8 + 4 + 5 = 17$$

According to Step 3 of the algorithm, the objective function and set of constraints for this transportation problem now read:

$$F = 7 \ X_{E, A} + 9 \ X_{E, B} + 15 \ X_{F, A} + 17 \ X_{F, B} \rightarrow \text{min} \quad (2.3)$$

$$X_{E, A} + X_{E, B} = 1 \quad (2.3a)$$

$$X_{F, A} + X_{F, B} = 1 \quad (2.3b)$$

$$X_{E, A} + X_{F, A} = 1 \quad (2.3c)$$

$$X_{E, B} + X_{F, B} = 1 \quad (2.3d)$$

Let's denote $X_{E, A}$ by $X_1$, $X_{E, B}$ by $X_2$, etc.
Now we obtain:

\[ F = 7X_1 + 9X_2 + 15X_3 + 17X_4 \rightarrow \min \]

\[ X_1 + X_2 = 1 \]
\[ X_3 + X_4 = 1 \]
\[ X_1 + X_3 = 1 \]
\[ X_2 + X_4 = 1 \]

The optimal solution to this problem is obtained by using the Operations Research software called TORA (2.H. A. Taha), and the optimal solution summary is presented in Table 2.2.

<table>
<thead>
<tr>
<th>Title: CPPl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final iteration No: 5</td>
</tr>
<tr>
<td>Objective value (min) = 24.0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Obj Coeff</th>
<th>Obj Val Contrib</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0.0000</td>
<td>7.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>x2</td>
<td>1.0000</td>
<td>9.0000</td>
<td>9.0000</td>
</tr>
<tr>
<td>x3</td>
<td>1.0000</td>
<td>15.0000</td>
<td>15.0000</td>
</tr>
<tr>
<td>x4</td>
<td>0.0000</td>
<td>17.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 2.2 Optimum Solution Summaries
According to Table 2.2, the optimal solution is

\[
\begin{align*}
X_1 &= 0 \\
X_2 &= 1 \\
X_3 &= 1 \\
X_4 &= 0
\end{align*}
\]

and the objective function value is 24.

According to Step 4 of the algorithm, for every \( x_{jk} \geq 0 \) obtained as a solution to the linear programming problem, we add \( X_{j,k} \) artificial paths parallel to the existing shortest path from \( n_i \) to \( m_k \).

Since \( X_2 = X_{E,B} \), and \( X_3 = X_{F,A} \) and

\[
\begin{align*}
d_{E,B} &= d_{E,C} + d_{C,B} = 4 + 5 = 9 \\
d_{F,A} &= d_{F,E} + d_{E,A} = 8 + 7 = 15
\end{align*}
\]

the optimal solution to optimal pairwise matching of supply nodes to demand nodes is

\[
\begin{align*}
X_2 &= X_{E,B} = 1 \\
X_3 &= X_{F,A} = 1
\end{align*}
\]
This simply means that the artificial arcs have to be added between nodes E-B and nodes F-A. The following figure shows the Chinese Postman network where zero node polarity is obtained by adding artificial arcs (dotted lines).

Figure 2.8 Chinese Postman Network with Zero Polarity

The total length of the artificial arcs is 24 units (which is the objective function value obtained as an optimal solution to the linear programming problem). The new network presented in Figure 2.8 has an Euler tour, which starts and finishes in node A:

A, E, C, B, D, F, E, A, B, F, E, C, B, C, E, A
2.11 The Chinese Postman Algorithm for an Nonoriented Network

The objective of the Chinese Postman Problem is to find the tour on a given graph $G$ where all arcs are to be covered at least once, and the overall length of such a tour is minimized. The only two constraints on this objective are that the graph has to be connected and undirected.

If a given graph has an Euler tour, such a tour is the solution to the Chinese Postman Problem. Otherwise, the solution procedure consists of adding artificial arcs parallel to existing ones in order to transform a graph $G = (N, A)$ to a graph $G' = (N, A')$ on which an Euler tour can be drawn. This transformation means that odd degree nodes on a graph $G$ are to be transformed to even degree nodes on a graph $G'$, which in turn means that arcs on which new artificial arcs are added will be traversed twice in the final tour. The adding of artificial arcs to the set of existing ones should be accomplished by choosing the arcs with the minimum length tour at the end.

2.12 Chinese Postman Problem Algorithm for a Nonoriented Network

The algorithm itself consists of four steps:

STEP 1: Identify all nodes of odd degree in $G(N, A)$. There are $m$ of them where $m$ is an even number. (The number of odd-degree nodes in an undirected graph $G$ is always even, and therefore the sum of the degrees of all nodes in $G$ is an even number, since each arc is connected on two nodes.)
STEP 2: Find a minimum length pairwise matching of the m odd-degree nodes and identify the m/2 shortest paths between the two nodes, each consisting of the m/2 pairs.

STEP 3: For each of the pairs of odd degree nodes in the minimum-length pairwise matching found in step 2, add to the graph G(N, A), the arcs of the shortest path between the two nodes in the pair. Thus, the resulting graph G^1(N, A^1) has no nodes of odd degree.

STEP 4: Find an Euler tour on G^1(N, A^1). This Euler tour is the solution to Chinese Postman problem on the original graph G(N, A). The length of the optimal tour is equal to the total length of the arcs in G(N, A) plus the total length of the arcs in the minimum-length matching.

By using this algorithm, we are going to solve the Chinese Postman Problem for the transportation network shown in Figure 2.9. The starting and ending point of a tour will be node A.

![Network for Minimum Length Pairwise Matching](image)

*Figure 2.9 The Network for Minimum Length Pairwise Matching*
The steps 1, 3, and 4 are simple, but the problem lies in step 2. The number of possible matching combinations $Q$ to achieve an even number of nodes on the network is:

$$Q = \prod_{i=1}^{m/2} (2i - 1)$$

where $m$ is a number of nodes with odd degree (always even number). As we can see the number of possible matching combinations increases rapidly with the number of odd degree nodes. For instance, in the case of $m = 10$ odd degree nodes, the number of possible pairwise matching combinations is 945.

Figure 2.9 shows a nonoriented network consisting of five nodes, four of which are with odd degree (A, B, D, and E). The values on the arcs indicate the length of the arcs. Total length of the network shown in Figure 2.9 is 50 units. In order to make odd degree nodes even, we need to add artificial arcs parallel to existing ones. Since we have four nodes with an odd degree there are three possible pairwise matchings of the odd degree nodes:

1. A-B and D-E (70 units)
2. A-D and B-E (64 units)

The Chinese Postman algorithm concentrates exclusively on the shortest path. Therefore, the shortest total length of the artificial branches is 64 units. This comes from A-D and B-E matching. By adding the artificial arcs (dotted lines) we have completed the steps two and three of the algorithm. The new graph $G'(N, A')$ consists of two new arcs and has no nodes with an odd degree (Figure 2.10). Thus, an Euler tour can be drawn on it, beginning, for instance, at node A and ending at the same node.
An Euler tour in the network, which starts at node A is as follows:

\[ \text{A, D, C, B, E, C, A, B, E, D, A} \]

### 2.13 Manual Minimum Length Matching

The manual search for minimum-length matching of odd degree nodes has been greatly facilitated by the observation [Larson, Odoni, 1981]: “No two shortest paths in the minimum-length matching of odd degree nodes can have any arcs in common.”

To illustrate this, consider Figure 2.11 which contains four odd degree nodes A, B, C, and D. In this figure the shortest paths between the nodes A-C and B-D are shown.
These two paths have an arc $k(i, j)$ in common. If the minimum-length pairwise matching contained the arc $k(i, j)$ then the total length of the path would be longer by at least $2 \times k(i, j)$ units than in the case of matching nodes A and B, and nodes C and D. The latter matching reduces the total length of the path by at least $2 \times k(i, j)$ units.

Experience has shown that a manual approach to matching by closely studying a geographical map is close to an optimum solution. It has two advantages over using Edmond's minimum-length matching algorithm [Lovasz, Plummer, 1986]

1. Keeping in mind the fact that the number of possible pairwise matching of odd degree nodes increases rapidly with the number of nodes in the network, this manual approach eliminates from consideration a large number of possible sets of matching; and

Figure 2.11 Possible Matching of Odd Degree Nodes
2. It indicates that in minimum-length matching an odd degree node has to be matched with an odd degree node in its immediate neighbourhood.

2.14 Drawing an Euler Tour

Assume we have finished all the steps of the algorithm necessary to obtain the graph G(N, A) which does not consist of odd degree nodes. The following rule is used in drawing an Euler tour.

First, we need to determine which is going to be our starting node. Let’s denote the starting node as \( n_0 \in N \) and begin. Starting from node \( n_0 \), we traverse all arcs on the graph keeping track of the route followed, and deleting every arc from the graph G as it is traversed.

In the next step, we are going to use the term isthmus. An isthmus is an arc that divides an undeleted portion of the graph G into two separate components. During the procedure of drawing the Euler tour, you are not allowed to traverse and delete the arc that is the isthmus. Since this is the requirement for the successful drawing of the Euler tour, the procedure is continued until all arcs of the graph G have been deleted, and we have returned to the starting node. At this point, the traversed route is the Euler tour for that particular graph.
2.15 The Mixed Postman Problem

Consider a mixed graph $G(N, A)$ that contains both directed and undirected arcs as shown in Figure 2.3. To obtain an Euler tour, the graph $G$ needs to be connected and even. The graph $G$ is even if the total number of arcs meeting each node, regardless of direction, is even.

There are two necessary and sufficient conditions for an Euler tour to exist:

1. Graph $G$ is connected and even; and

2. For any subset $S$ of the nodes

$$|N_d(S, S) - N_d(\bar{S}, S)| \leq N_u(S, \bar{S})$$ \hfill (2.5)

where

$N_d(S, S)$ is the number of directed arcs originating in a node of $S$ and ending at a node in $S$.

$N_u(S, \bar{S})$ is the number of undirected arcs with exactly one endpoint in each of the sets $S$ and $\bar{S}$.

If the graph $G(N, A)$ is even and balanced, the optimal postman tour coincides with an Euler tour. Condition (a) shows that imbalance between directed and undirected for a node $i \in N$ cannot exceed the number of undirected arcs that meet node $i$.

The mixed postman algorithm requires the original graph $G$ to be even. If the graph $G$ is not even, one can apply the matching algorithm in order to add artificial arcs to achieve an even degree at all nodes. The cost of artificial arcs in the matching algorithm process is to be minimized. According to Edmonds & Johnson [1973] the optimum way to add
artificial arcs to existing ones to obtain even degree nodes may not be the best way, since one might end up adding more arcs to symmetrize the graph. Therefore, the mixed problem is much harder than the classical Chinese Postman Problem on fully undirected or directed network. For more details on the mixed postman algorithm see Minieka [1978].

The Chinese Postman Problem itself represents a very interesting starting point for the explanation of network theory and different kinds of network problems related to arc visiting. An understanding of Euler’s theorem and heuristic algorithms, which solve arc routing problems on the network close to optimality, is a basic foundation for the development and application of a transportation model. Extensions of the basic Chinese Postman Problem are introduced in the next section of the chapter.
2.16 The Rural Postman Problem

Consider a mixed graph $G(N, A)$, and let $R$ denote a subset $S$ of arcs $B$ that require the service $G(R \subset A \cup B)$. Assume that the cost $c(i, j)$ of traversing each arc $(i, j)$ is specified. This formulation means that only the subset of all available arcs in the network requires service, whereas the classic Chinese Problem assumes that all arcs of the graph $G$ must be traversed.

The objective of the Rural Postman Problem (RPP) is to find a minimum cost cycle in $G$ which traverses all arcs in $R$. As one can see, RPP is a practical extension of classic CPP. Classic CPP requires all arcs of the graph $G(N, A)$ to be connected, while RPP may produce the subset $R_1$ which is not connected and therefore may include a traversal over deadheading arcs that serve as connection arcs. Deadheading arcs are designed to generate only a minimum cost. Algorithms, and a detailed description of this problem can be found in the paper by Pearn & Wu [1995].

Beside these there are variants of other Postman Problems such as:

1. Windy postman problem
2. Postman problem with branch priorities
2.17 Windy Postman Problem

Consider the network G that consists of a set of nodes N and a set of arcs A. The Windy Postman Problem can be formulated as traversal over each arc in either direction but with different costs depending on whether the traversal is “with or against the wind”. The objective of WPP is to find a postman tour with a minimum cost where the cost of an arc b can be denoted by $c_{i,j}$ if traversed from i to j or $c_{j,i}$ otherwise. For further reference to this problem see Minieka [1979], and also Win [1989], who provides the Windy Postman algorithm for connected even graphs.

2.18 Postman Problem with Branch Priorities

In a classic Chinese Postman Problem, the order in which the arcs R are traversed does not matter. In some applications of the Chinese Postman Problem, one might face another challenge in solving an arc routing problem by considering a subset of arcs with higher priority since the service of these arcs has preference over others. For analysis of the Chinese Postman Problem with priority relations between arcs, see Dror, Stern & Trudeau [1987].

2.19 Maximum Benefit Postman Problem

This kind of problem also utilizes the priority relationship between arcs and the order in which arcs of the network are traversed. It uses cost to differentiate the utility of visiting specific arcs of the network. In the Maximum Benefit Postman Problem, a benefit is determined every time an arc is traversed, and the objective is to find a tour of maximum net benefits. An Optimal Maximum Benefit Problem does not mean that all
arcs of the network are serviced. The arcs with high cost or with a costly access can be left out. Also MBPP allows multiple traversal of the same arc while providing the service. For a more comprehensive formulation of the problem, see Malandraki and Daskin [1993].

In order to understand fully the variety of arc routing problems related to CPP, a couple of variations are described in this chapter and eventual future user is referred to the theory. This theoretical background on different types of CPP covers the definitions and the in-depth description of a situation one can encounter in solving real time problems which are logical extensions of the existing model.

In the next chapter, the focus will be on development of the CPP model for a nonoriented network.
Chapter 3. Development Of The Model

The field of planning and handling of any complex operation in transportation requires models which support real time decision making. By using recent advances in information technologies, the routing and scheduling of drivers and vehicles can be undertaken in a real time environment with information that is constantly changing. This means that one has to make a decision before the information relevant to it is received and then modify these decisions as new information becomes available. Dynamic and stochastic models driven by operation research methods increasingly dominate the solution of these kinds of problems.

In order to understand this, one needs to distinguish between a problem, a model, and the application of a model. A problem is dynamic in its nature if one or more of its parameters is a function of time, whereas a model is dynamic if it incorporates the interaction of activities over time.

The dynamic application of a problem is finding a solution repeatedly as new information becomes available.

As opposed to static models, in dynamic applications one needs to carry out a new run of an algorithm and optimize the results every time new information is received. Application of this concept comes to play in routing problems on a congested transportation network. Since traffic conditions are constantly changing over time, the problem becomes dynamic. To obtain the solution to this type of problem, one can combine static and dynamic methods in one of these possibilities:

1. Determine an optimal route by using the static model only;
2. As new information becomes available, the static model is solved repeatedly. In this way a dynamic application of a static model is achieved;

3. Solve the problem by considering anticipated changes over time. In other words, according to initial assumptions made, one sticks to the solution despite the change of information over time. This is a static application of a dynamic model.

3.1 Difference Between Arc and Node Routing

The key activity of an arc routing problem is to cover all arcs of a transportation network, as opposed to node routing where the service activity occurs at the nodes.

On a given underlying network $G = (N, A)$, the general arc routing problem involves servicing a set of required arcs $R$. As previously mentioned, the graph $G$ can consist of oriented arcs, nonoriented arcs or a mixture of both. The cost of traversing the arc can be interpreted as a distance or a time. The main objective of an arc routing problem is to minimize the sum of the cost of traversing non-required arcs. Generally speaking, one seeks the cycle that covers all arcs in $R$ with a minimum associated cost.

On the other hand, a node routing problem can be obtained from an arc routing problem simply by replacing the required service in $R$ from arcs to nodes. In this case, the underlying network matters only to provide the shortest path between the nodes requiring the service.
3.2 Information System Design

In order to change the classic solving of real time transportation network problems, the building of a new information system to automate existing procedures will be introduced in this software which is primarily focused on matching an appropriate Euler tour to the current demand on the transportation network. The idea is to apply the existing traditional information system, called transaction-based system.

A transaction-based system is based on the following principle. It takes the input data on record and processes it through the series of programs which manipulate the data. It retrieves additional data from tables and files stored in the database, writes new information to other files in a different form as shown in Figure 3.1, and then prints an output document.

Figure 3.1 Transaction-based Information System Design
3.3 Routing and Geographic Information Systems

In order to optimize the route problem and get accurate information on the demand associated with the routing applications, one has to rely on the underlying transportation network. For an intercity network, the distances are computed from \((x, y)\) coordinates of locations of interest. For urban street networks, however, a much better representation of geographic details is often required.

Acquiring more accurate routing information requires close integration with a geographic database which contains a detailed street network. Such a database has the following characteristics:

1. The display of routes on the actual street networks is an important visualization aid that provides greater acceptance and flexibility;
2. There is a great potential for linking routing information with a variety of other sources such as demographics and zoning.

Both of these can be found and manipulated by a geographic information system which can be defined as a computerized database system for capture, storage, retrieval, analysis, and display of spatial data [Huxhold, 1991]. Therefore, the information in a GIS is organized and related spatially. The following are the tools of a GIS:

1. An automated mapping technology for manipulation of map information;
2. Database management tools for managing attribute data;
3. Geographic information with specific references and locations such as street segments, blocks, buildings, etc.;
4. Topological data structure capturing the spatial relations among points, lines, and polygons;

5. Spatial analysis capabilities for retrieving and displaying map data.

Land record information on the map contains the coordinate system, cartographic data, and location identifiers. One of the applications of spatial analysis capability used for retrieving and displaying map data is the attachment of geographical identifiers to points (the process called geocoding). The generation of the shortest path through the street network and its display is another application of spatial analysis.

Spatial relations among points, lines, and polygons represent a street network. Streets are presented as lines between points, whereas blocks are defined by a boundary composed of lines. In addition, one of the most attractive features of a GIS is the possibility of attaching the addresses of streets so that the street can be traversed in the direction desired. This is a basic component in the application of a GIS to solving different routing problems on the transportation network.

Blocks are situated on the left and right sides of the streets, and therefore each street address is associated with an exact location on the appropriate side of the street.
Consider the points A, B, C, and D in Figure 3.2 which are to be visited. Our ultimate objective is to find the tour which will visit all four points (demand points) through the street network. Assume that the rectangles are images of buildings, and spaces between them represent streets. Also note that all four demand points have an address associated with their position on the map. In the case of vehicle tour design, one has to take not only the direction of the street into account but also the side of the street. In this example, all streets are two-way. If the starting and ending point is the point A, one of the possible solutions to visiting all demand points, considering the travel direction is the tour presented by the arrowed line. If one is asked to design the route for a postman who will walk the distance to visit all four demand nodes, the direction of travel does not matter, as the postman can choose both sides of the street regardless of the street address. Such a route is presented by the dashed line in Figure 3.2.
3.4 Raster and Vector Format of the Map

Spatial features in a GIS database are stored in either vector or raster form. GIS data structures presented in a vector format, store the position of map features as sequences of x, y coordinates. A vector format represents precisely the location and shape of forms and boundaries. Precision during the use of this format depends on the following factors:

- The accuracy and scale of the map
- The resolution of input devices, and
- The skill of the operator inputting data

In contrast, the raster or grid-based format presents features of the map in a grid form. The size of the cells in the grid matrix will determine the level of detail with which the map features are represented. There are advantages to the raster format for storing and processing some types of data in GIS. Four principal cases where the raster format is very useful are listed below:

- Map scanning as a step in map automation;
- Storage and manipulation of images;
- Analysis of grid-based maps;
- Map display and plotting.
3.5 Classification of Arc Routing and Scheduling Problems

Arc routing and scheduling problems can be found in all modes of transportation. L. Bodin and B. Golden [1981] made the following classification:

1. Time to carry out the service in a specific node or on a specific arc
   a) The time to carry out the service which is specified in advance (scheduling problem);
   b) Service in certain nodes must be carried out within a specific time interval (combined routing and scheduling problem);
   c) There are no specific demands regarding service in each node (vehicle routing problem).

2. Number of vehicle depots in the network
   a) There is only one depot in the network;
   b) The network contain several depots.

3. Size of vehicle fleet available
   a) The fleet contains only one vehicle;
   b) The fleet contains several vehicles.

4. Type of vehicles in the fleet
   a) All vehicles in the fleet are the same;
   b) The fleet contains different types of vehicles.

5. Nature of service demands
   a) Deterministic demand appears in the network;
   b) Stochastic demand for service appears in the network.
6. Location of service demands
   a) Service demand appears to be in the network’s nodes;
   b) Service demand appears to be in the network’s arcs;
   c) Service demand appears in both nodes and arcs.

7. Type of transportation network
   a) Oriented transportation network;
   b) Nonoriented transportation network;
   c) Mixed transportation network.

8. Vehicle capacity constraints
   a) All vehicles have the same regulated capacity constraints;
   b) There are differences between vehicles regarding regulated capacity constraints;
   c) There are no constraints regarding vehicle capacity.

9. Maximum allowed vehicle route length
   a) All vehicles in the fleet have the same maximum allowed route length;
   b) Some vehicles have different maximum allowed route length;
   c) There are no constraints regarding the maximum allowed vehicle route length.

10. Costs
    a) Fixed
    b) Variable.
11. Operation carried out

   a) Picking up;
   b) Delivering;
   c) Picking up and delivering (loading and unloading goods or picking up and dropping off the passengers).

12. Objective function on which optimization is based

   a) Minimizing route costs;
   b) Minimizing total fixed or variable costs;
   c) Minimizing the number of vehicles needed to carry out transportation operations.

13. Other constraints (depend on specific problem)
3.6 Development of PKPSof3 Model

PKPsoft is the software developed for solving the Chinese Postman Problem on an unoriented transportation network.

The idea and initial steps of the PKPsoft model development started in 1996 as a result of the research for an undergraduate thesis on transportation network problems. The modelling of a network transformation necessary for an Euler's tour to exist is done through the collaborative work with student colleagues and the following development is an extension of the initial model. The model interface is created in object oriented programming language Visual Basic 5.0.

The program itself is one of the components of transaction-based information system design (Figure 3.1) and represents the link between the input and Access database on the one hand and output on the other hand. The link between input and the model and between the model and output is accomplished through the Data Base Management System (DBMS) which saves or calls the data of concern. Furthermore, every time when user's activity requires the execution of the appropriate source code of the program, DBMS connects necessary forms, searches, and creates indexes according to the desired instruction.

This program is supported by a Microsoft Access database which is created at the same time as the transportation network. PKPSoft is designed using a Geographic Information System (GIS) specifically intended to store, display, manage, and analyse transportation network data. It combines GIS and transportation modelling capabilities for different modes of transportation.
PKPsoft provides a powerful GIS engine with special extensions to a transportation network, as well as mapping and visualization tools designed for transportation network applications. It also includes application modules for Chinese Postman routing problems, including its application development tools. PKPSoft contains a route system that indicates paths taken by pick up or delivery operations from place to place. It includes tools to create, display, edit, and manipulate routes as well as unique display technology for mapping routes in a clear and compelling fashion. PKPSoft is organized in couple of modules whose functions are to prepare, visualize, analyse, and present work before solving routing problem on a transportation network. PKPSoft provides a comprehensive solution to two types of transportation applications:

- Network analysis; and
- Vehicle routing and logistics.

A network analysis model is used to solve many types of transportation applications, but this application is specifically focused on the shortest route between the origin and destination nodes in the network.

Vehicle routing and logistics can be applied to all modes of transportation for both private and public sector through the following applications:

- Pickup and delivery operations;
- Street sweeping and snow removal;
- Solid waste and recyclable collection;
- Distribution planning;
- Facility maintenance;
- Door-to-door delivery and pickup;
- Meter reading.

PKPsoft consists of several different modules, each of which is presented in a different form. In order to run the program, one does not need to be familiar with either the Visual Basic programming or other transportation planning related computer software. This application of the Chinese Postman Model is self explanatory and easy to use.

The Chinese Postman Problem is an arc and routing problem which can be defined according to Bodin and Golden’s classification as following:

1. The time to carry out the service is specified in advance;
2. There is only one depot in the network;
3. The fleet contain only one vehicle (postman) to do the task;
4. All vehicles (postmen) in the fleet are the same in terms of workload;
5. There is stochastic demand for service;
6. Service demand is in the network branches;
7. There is nonoriented transportation network;
8. All vehicles (postmen) have the same regulated capacity constraints;
9. All vehicles (postmen) in the fleet have the same maximum allowed route length;
10. There are fixed costs;
11. There is both delivering and pickup;
12. Route costs are minimized by minimizing the distance.
3.7 Model Description

PKPSof is the program for solving the Chinese Postman Problem. The program itself consists of several different procedures:

- Scanning a map of the city;
- Vectoring the scanned picture of a transportation network region;
- Creating a database of a network;
- Finding the shortest route (Euler's tour and/or Euler's path);
- Printing the report.

If one executes the PKPSof model, two forms, Area and Region, pop up on the screen. They are shown in Figure 3.3. These two forms are designed for input of area and region information. Upon entering data for a certain region, it is necessary to confirm the process of entering by pressing the enter key. In this way the program automatically checks whether there is another duplicated set of data. If so, the program generates the message that requires the change of certain values since there cannot be two regions with the same name within the same area of the city.

A primary key is designed to make sure that every text box of a form uniquely identifies every cell of the database table. Unique keys are very important for the correct link of tables in the program, different areas, and regions. These relations are established by the 1-∞ rule. This relation means that one value of the ID text box in the parent-table (form Area) is associated with couple of different cells in child-tables (form Regions) which have the same value of the primary key in the ID-Region.
3.7.1 Form Area

Region button is used to call the form Region to allow a user to enter data for a new region, to change existing data, or to display a map with a previously scanned network of the region.

Euler tour and Euler path options are the solution to the Chinese Postman Problem.

Search is used only if data about the region already exists. This option allows the data to be changed or the previously scanned map to be modified. In the case of more than one region is entered with the same name, additional information including the name of the area is required so that the desired region can be precisely identified. The outcome
of such a search is either data on the desired region or a statement in the message box stating that such a region does not exist.

**Add** in the form Area enables users to enter a new area and a region. By pressing this button, the text box value of the primary key ID increases by one.

**Refresh** button allows a change to be applied to the entire database since, all changes of a parent-table are carried over automatically on a child-table

**Delete** command has to be used carefully, since it deletes all data of the region in both a parent table and a child table. When the button is pressed, there is a warning message to make sure that user is executing the appropriate action.

**Save** action is self-explanatory. It saves all data in the database and checks whether there are any duplicates of the regions within the area.

**Exit** option means that the program is terminated and all databases are closed.

In addition, the form Area contains information related to the region (postal code), area, and city name. All this information is entered manually.

### 3.7.2 Form Region

The Region form contains the data about elements of the network. These data are obtained as a result of a vectoring process of the scanned map of the region. After vectoring the scanned map, some data are entered automatically into the database. There are two different kinds of data related to this form. Let’s simply call them visible data and invisible data. Visible data define a transportation network, such as a set of nodes and a set of branches. Text boxes within the Region form containing data on street names and distances are considered to be invisible: invisible since coordinates of nodes cannot be
changed and they are shaded. All data of this form are entered automatically as a result of
the creation of a network on the appropriate map of a region.

Since accuracy depends on the user, there are two options that can be changed manually:
distance (if differs from the actual value) and the name of the street.

Like the previous form, Region has options for the use and manipulation of the
database with additional commands for searching different forms and connecting them.

These options are:

- Pointer at left
- Pointer at right
- Cancel (branches)
- "Twoo"
- Map
- Add
- Delete
- Refresh
- Save
- Street search
**Pointer at left** shows the previous action that belongs to the region of concern.

**Pointer at right** displays the following action in the form which belongs to the region.

**Cancel** is a very useful option that allows the temporary deletion of the branch in the network. If there is no delivery on the certain branch (no demand on a street), this option lets us calculate the optimum route on the network by not taking this branch into consideration. Therefore, Euler tour consists only of the branches on which there are deliveries to be made. The data about cancelled streets remain in the database and they are avoided during determination of the optimal route.

**Map** key calls up the form Map, which contains the scanned image of the map of the region.

### 3.7.3 Form Open Region

In a standard windows application, the manipulation of files is executed by different commands such as Open, Save, Save As, Exit etc. These commands are located in the pop down menu. The form Open Region has a similar interpretation with a different look for the form (Figure 3.4). The form lets us choose the disk where the scanned maps are stored, the region, and the file type (extension of the image).
After choosing the map of the city we can either open the map (next form) and automatically close the existing form (Open Region) or cancel the entire action.

3.7.4 Form Map

Elements of a Geographic Information System (GIS) are presented in the form Map where the actual network is created. This form consists of the menu for converting the maps from raster to vector form as well as tools for creating a transportation network on such a map (Figure 3.5)

There are a few advantages of using GIS as opposed to making a manual database of network characteristics. One can:

- Graphically display the information on the screen;
- Edit, change, and transform the information;
- Measure distances of the streets; and
Combine maps of the same area together.

For the purpose of this project, creating the GIS to solve the Chinese Postman Problem is done through the following phases:

- Scanning the maps
- Removing the abnormalities and deformations from maps in raster form
- Storing and archiving scanned maps in a file
- Converting the graphical data of the maps from raster to vector form
- Linking graphical information with corresponding database
- Creating and maintaining the database

Figure 3.5 Form Map as a GIS Feature
There are two levels in the process of making a transportation network on the GIS map stored in the database and used as a background in the form Map. The first level is the process of defining and creating a net of nodes and a net of streets. At that level the access to all options in the pop down menu is granted. At the second level during the display of an optimal route, only specific options are accessible.

3.7.5 Menu Options

The form Map consists of different menus with a number of options designed to facilitate the use of this program. The main menu is made up of three categories which are the crucial components of any transportation network and which are used to manipulate and adapt the network to the real time problem (Figure 3.6, 3.7, and 3.8 respectively).

![Network's Pop Down Menu in the Form Map](image)

Figure 3.6 Network’s Pop Down Menu in the Form Map
A network is created on the background of the form Map, and its menu options are the following:

**Open** calls up the form that opens the map with the desired image of the region.

**Save** option saves the created network in the database. By executing this option the process of vectoring the static image in the form Map is finished.

**Network** option displays the entire network in case other options were previously used to show different components of the network (e.g. streets only or nodes only).

**Delete** is necessary when a better view of the raster map is needed to extract the name of the street or other relevant information directly from the map or to display only nodes or only streets.

**Region** provides the access to the form Region, should any street name or distance data requires correction.

**Exit** is accompanied by the message prompt asking whether the network data has been saved before exiting the form.

Streets represent the branches between the nodes in the network that need to be traversed.

(Figure 3.7)

![Figure 3.7 Street's Pop Down Menu in the Form Map](image)
The process of drawing branches in the network is done after placing nodes on the network, by clicking the mouse on the desired locations. The direction of streets during the process of connecting nodes needs to correspond to the direction of streets indicated on the background map. The distance of the street is calculated according to the scale of the map entered in the form Map. If the street distance obtained this way shows discrepancy from the real values it can be modified later in the form Region.

The Street Menu in the form Map, shown in Figure 3.7 contains:

**Find Street** option searches the desired street and shows its position on the network (blinking of the street).

**Display Street** is an option used to show the network without nodes as opposed to "show nodes" option which will be discussed later.

Nodes correspond to the position of the intersections on the transportation network. Placing nodes on the background of the map is necessary to create the network and to determine coordinates of the streets on the network. Let us have a look at the form Map with the options contained in the node’s pop down menu (Figure 3.8):

![Figure 3.8 Nodes’ Pop Down Menu in the Form Map](image-url)
Starting Node is used at the beginning of the network creation. The process of creating a network starts with a node number chosen from 0 to 100 (default = 100 nodes). All variables are predefined to have maximum of 100 elements since the number of nodes (intersections) of the real transportation network within the region (postal code) is approximately 30, rarely exceeding 50 *

Skip Node is useful tool if there is a demand on only one portion of the branch which is closely connected to the street being traversed. Since the number of nodes increases by one every time the node is created by the mouse click on the form Map, this option helps to skip the node and leave it for later modification of the network.

Add Node is a response to the Skip Node option and allows the placement of a node on the intersection previously skipped. If more than one node is to be added the counter goes automatically to the next number in the sequence.

Display Nodes option shows the nodes of the network only.

3.7.6 Creation of a Transportation Network in the Form Map

A transportation network can be created and manipulated in the form Map where graphical spatial features are connected with the descriptive attributes in databases through the unique identifiers. In this way these data can be searched and modified, and the change of one value means the change of all belonging positions related to that change. The actual creation of the network is carried out in two phases:

(* Author's observation)
1. Placing nodes; and

2. Creating streets and making a network.

Placing nodes is done by moving a cursor on the desired location and clicking the left mouse button. This action creates a little circle on the clicked location with the number in its centre. A user has to click as closely as possible to the actual street intersection since these points are nodes in the network. This number increases automatically by one every time the click is made. Each time the cursor is placed over the node, the additional sign of the node number is displayed. This is useful if the node number cannot be read because of either similar colours in the background or a small scale in the scanned map. In case the node is wrongly positioned, the click on right mouse button deletes it.

Before we create the street grid between nodes on the scanned network picture, the option Connect has to be checked off (see the form Map). Every street is connected with the two nodes (intersections). Selecting the desired nodes and entering the street name connects the nodes and creates the street's grid on the network. The streets are drawn in red, and positioning the cursor on the street displays the name previously entered. The distance between the two nodes is calculated according to the scale of the map input in the form Map, and it is automatically entered into the database and into the Distance text box of the form Region. The right mouse button deletes the selected street and all data that were stored in the database when the street was first created.
After all nodes have been connected the network is made, and it has to be saved and exited before the Chinese Postman program is executed. At this stage of the model, all options in the main menus are available. In the event that one tries to exit the network without saving, the prompt message gives a warning.

Figure 3.9 shows the network created on the background map of the region previously scanned. (The map is obtained from the Internet for experimental purposes only.)

The scale of the map is entered arbitrarily and the Connect option is checked off which allows the process of connecting nodes to be accomplished. The right side of the Figure 3.9 shows the form Region with the information on streets (name, distance,
starting and ending nodes that it connects). Exiting the form Map (with the network created and saved) brings the form Area up.

### 3.7.7 Presentation of the Solution to the Chinese Postman Network Problem

To solve the Chinese Postman Problem on a nonoriented network, as this one, on the form Area, one has two options to choose from depending on the desired outcome. By clicking on the Euler tour, the route sequence is going to start and end at the same node, whereas the Euler path is going to start and end its optimal route at the specified nodes. If an Euler tour is selected, the starting/ending node (the same one) can be chosen in advance, before the program is executed.

On the other hand, the form Region offers two options to choose from in order to make the network and the solution to the problem feasible for different scenarios that might arise in the real problem. The cancel option deletes/saves the street(s) of the network for which there is no demand, whereas “twoo” represents the demand for two-way streets.

The solution to the network problem drawn in this example is shown in the Figure 3.10. The form Map shows the network with the visual interpretation of the streets traversed, and the meaning of different colours is shown in legend of the form List of Streets.
The colour indicates that all streets have been covered at least once (red) with an exception of white streets that are actually the result of connecting odd nodes in the network.

The form List of Streets represents the solution to the problem as it contains the list of streets along with the nodes that connect them. This sequence is the optimal solution to the network problem, and total distance is also calculated. In addition, the report with the street sequence for the region is available. This report is in the Word Document format and can be obtained simply by clicking the Report icon on the form List of Streets.

After the final solution to the problem is presented in the list of streets that need to be traversed, the process of obtaining the optimal solution to the Chinese Postman Problem can be summed up in the following steps:

1. Determines the set of odd nodes;
2. Determines the set of even nodes;
3. Determines the optimal matching of odd nodes;
4. Creates the connection between odd nodes;
5. Asks the user for the input of starting node (and ending node in case of an Euler path);
6. Shows the optimal tour in the form of node pairs and street sequence;
7. Calculates the length of the optimal route; and
8. Prints the report with the street sequence.
Chapter 4. Results and Discussion

There are different areas in which a PKPSoft model can be successfully used:

- Distribution of mail
- Distribution of newspapers
- Distribution of goods
- School children pick up
- Cleaning and washing of city streets
- Snow removal
- Garbage pick up

This project primarily focuses on the distribution of mail and the routing problems that a postman can encounter because of the mail delivery.

4.1 Postal Delivery Routing Problem In Canada

The scale of postal delivery operations in the USA and Canada, which all together account for about 40% of the volume of the mail worldwide, requires constant improvements through automation and the use of new technology. In recent years, many new technologies have been applied in the area of sorting and processing mail. One of these improvements is an automated sorting based on optical character reading. Furthermore, developments are focused on field operations that depend on geographical database and information systems.
Considering the volume of mail, deliveries in the USA and Canada are almost the same with a slight difference related to internal regulations and working hours. There are three elements involved in the postal delivery procedure which are rather characteristic of the USA post:

1. Initial collection of mail and its sorting at postal facility;
2. Driving from the postal facility to delivery area and from one delivery area to another; and
3. Delivery of the mail by vehicles or on foot.

One of the problems facing the postal service in Canada is the scheduling of their postal carriers. Postal carriers deliver mail to residences and industries. One portion of that service is carried out in the vehicle or public transport, while another is carried out by the carriers walking the route.

The postal delivery procedure in Canada is slightly different from that of the USA and can be defined in the following manner. The workday of a carrier at the Canada Post Corporation consists of morning and afternoon portions. The postal carrier starts his day at the postal station, goes to the area, does deliveries, returns to the postal station for the lunch break, goes for deliveries again during the afternoon portion of his workday, and eventually returns to the station at the end of the workday.

The duration of the morning portion is approximately five hours, whereas only three hours are available for the afternoon route. Those hours include the time a postman needs to get to the area where he starts delivery and the time required to return to the station. In most cases, the postman uses public transit to get to the starting point and to return to the postal station. Therefore, she/he does not need to drive a postal vehicle. Another delivery
method is to pick up the mail from relay boxes which are located along the route and fed before the carrier’s workday starts.

4.2 Application of the Model to the Toronto Transportation Network

The data set used for this project is the portion of the Toronto Transportation Network which belongs to the Postal Code MST. It has been obtained from the ArcView Street Network File. The dataset contains the information on starting nodes, ending nodes, length of the streets, arc identifier, the street names, street types, and street direction. All necessary data to apply this model on have been imported from the Street Network Files into the new data file. The reason for this is that the Chinese Postman model has been developed in the way that it assigns the appropriate ID code to the data set at the same time as the user creates the network.

Nevertheless, because of the creation of the network and the storage of network information in the database, the nodes are chosen arbitrarily, and therefore the information on starting and ending nodes from the street network files corresponding to the street names is not used. Street identifiers are used to identify segments of the same street among the nodes drawn at the street intersections. Every street consists of a number of branches which belong to the same street and which are divided by nodes drawn at intersections. A street identifier and a street length are assigned to each branch and are accompanied by an ID street map of the region.

The ID street map of the region is part of the appendix section, and the dataset is shown in the Table 4.1.
<table>
<thead>
<tr>
<th>Starting Node</th>
<th>Ending Node</th>
<th>Street</th>
<th>Street Arc's Identifiers</th>
<th>Length (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>College St.</td>
<td>789</td>
<td>156</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>College St.</td>
<td>149</td>
<td>93</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>College St.</td>
<td>144+139</td>
<td>94</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>College St.</td>
<td>138</td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>College St.</td>
<td>136+130</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>College St.</td>
<td>125+118</td>
<td>238</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>Cecil St.</td>
<td>207</td>
<td>86</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>Cecil St.</td>
<td>201</td>
<td>74</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>Cecil St.</td>
<td>187</td>
<td>92</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>Cecil St.</td>
<td>179</td>
<td>99</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>Cecil St.</td>
<td>174</td>
<td>99</td>
</tr>
<tr>
<td>27</td>
<td>26</td>
<td>Baldwin St.</td>
<td>253</td>
<td>155</td>
</tr>
<tr>
<td>26</td>
<td>25</td>
<td>Baldwin St.</td>
<td>237</td>
<td>195</td>
</tr>
<tr>
<td>25</td>
<td>24</td>
<td>Baldwin St.</td>
<td>220</td>
<td>98</td>
</tr>
<tr>
<td>24</td>
<td>23</td>
<td>Baldwin St.</td>
<td>209</td>
<td>106</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>D'Arcy St.</td>
<td>289</td>
<td>162</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td>D'Arcy St.</td>
<td>277</td>
<td>193</td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>D'Arcy St.</td>
<td>257</td>
<td>201</td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>Dundas St.W.</td>
<td>318+312</td>
<td>158</td>
</tr>
<tr>
<td>33</td>
<td>34</td>
<td>Dundas St.W.</td>
<td>300+292</td>
<td>200</td>
</tr>
<tr>
<td>34</td>
<td>35</td>
<td>Dundas St.W.</td>
<td>287</td>
<td>204</td>
</tr>
<tr>
<td>35</td>
<td>36</td>
<td>Dundas St.W.</td>
<td>270</td>
<td>88</td>
</tr>
<tr>
<td>36</td>
<td>37</td>
<td>Dundas St.W.</td>
<td>264</td>
<td>91</td>
</tr>
<tr>
<td>37</td>
<td>38</td>
<td>Dundas St.W.</td>
<td>259</td>
<td>68</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>Spadina Ave.</td>
<td>183+208</td>
<td>212</td>
</tr>
<tr>
<td>8</td>
<td>27</td>
<td>Spadina Ave.</td>
<td>219+248+254</td>
<td>137</td>
</tr>
<tr>
<td>27</td>
<td>28</td>
<td>Spadina Ave.</td>
<td>276+290</td>
<td>119</td>
</tr>
<tr>
<td>28</td>
<td>32</td>
<td>Spadina Ave.</td>
<td>319</td>
<td>110</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>Huron St.</td>
<td>188</td>
<td>209</td>
</tr>
<tr>
<td>11</td>
<td>26</td>
<td>Huron St.</td>
<td>238</td>
<td>137</td>
</tr>
<tr>
<td>26</td>
<td>29</td>
<td>Huron St.</td>
<td>278</td>
<td>115</td>
</tr>
<tr>
<td>29</td>
<td>33</td>
<td>Huron St.</td>
<td>301</td>
<td>114</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>Beverley St.</td>
<td>175</td>
<td>209</td>
</tr>
<tr>
<td>13</td>
<td>25</td>
<td>Beverley St.</td>
<td>221</td>
<td>139</td>
</tr>
<tr>
<td>25</td>
<td>30</td>
<td>Beverley St.</td>
<td>258</td>
<td>113</td>
</tr>
<tr>
<td>30</td>
<td>34</td>
<td>Beverley St.</td>
<td>288</td>
<td>111</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>Henry St.</td>
<td>168</td>
<td>206</td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>Henry St.</td>
<td>210</td>
<td>140</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>McCaul St.</td>
<td>153</td>
<td>138</td>
</tr>
<tr>
<td>15</td>
<td>23</td>
<td>McCaul St.</td>
<td>195</td>
<td>209</td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>McCaul St.</td>
<td>215</td>
<td>51</td>
</tr>
<tr>
<td>22</td>
<td>31</td>
<td>McCaul St.</td>
<td>236</td>
<td>51</td>
</tr>
<tr>
<td>31</td>
<td>35</td>
<td>McCaul St.</td>
<td>271</td>
<td>111</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>University Ave.</td>
<td>786+194</td>
<td>412</td>
</tr>
<tr>
<td>20</td>
<td>38</td>
<td>University Ave.</td>
<td>231+252</td>
<td>174</td>
</tr>
</tbody>
</table>
Table 4.1 The Dataset of the Network M5T

Note that the street arc’s identifiers which are shown as a sum of two identifiers (e.g. College St. 144+139) are obtained from the ID map of the region. These identifiers belong to the same street but are divided by the intersection made up of the street belonging to the region of concern and the street outside our area of interest.

The network on which the Chinese Postman model will be applied is shown in Figure 4.1. The map of the M5T postal code has been scanned and all necessary data describing the network obtained in the dataset are presented in Table 4.1 above.

The scale of the map scanned in the form Map is 1:14300.
Figure 4.1 The Map of the M5T Postal Code in the Downtown Toronto

The next steps in the procedure of solving the Chinese Postman Problem described in Section 3.7, are vectoring the scanned map of the transportation network region and creating the database. Figure 4.2 shows the network which is created on the given map in the form Map. The network itself consists of 38 nodes, 28 of which are odd degree nodes, and 57 branches, yielding a graph $G(38, 57)$. The total distance of links in the network is 7887m.
According to the algorithm for solving the Chinese Postman problem on a given network, the minimum length pairwise matching is based on the number of odd degree nodes and the fact that the best solution is found among $213458 \times 10^9$ combinations. The number of possible matching combinations $Q$ required to achieve the network consisting of even nodes and consequently an Euler's tour on the network, is obtained from the equation 1.4 (below)

$$Q = \prod_{i=1}^{m/2} (2i - 1)$$

(1.4)

where $m$ is the number of odd degree nodes.
In addition, computational time also plays important role in applying this model. For instance, 19.34 seconds was necessary for 400MHz processor to obtain the Euler path in case of 7-38 starting/ending node.

The Figure 4.3 shows the solution to the network previously drawn on the map of the desired region. The red colour indicates the streets which have been visited once. The white colour on the network solution shows the set of streets that have been visited twice. This set of white coloured streets not only shows the number of times the particular streets are covered but also the set of artificial arcs added to the network in order to make the odd degree nodes even and to make the whole network ready for finding a feasible solution to the problem.
After adding artificial arcs to the existing set of streets, one can determine what the distance cost is because of changing the network from $G$ to $G'$. The final solution expressed in meters of distance is 10222m, whereas the original network’s length was 7887m. The difference of 2335m is the distance cost of adding artificial arcs and finding the Euler’s path on the network.

Since the postmen use TTC first to get to the point where the delivery starts and then to return to the postal distribution station, the nodes 7 and 38 are assigned to the Queen’s Park and St. Patrick St. subway stations respectively. Therefore, a postman can start and finish his/her route at the actual TTC stops.

The final step in the process of finding the Euler’s path is the list of order in which the streets have to be visited. This report is attached below (Figure 4.4).

City: Toronto
Area: Downtown
Date: 10-28-2000

REPORT
The order of streets in which the postman needs to traverse the network: College

1. College St. (125+118)
2. College St. (136+130)
3. College St. (138)
4. College St. (144+139)
5. College St. (144+139)
6. Beverley St. (175)
7. Cecil St. (179)
8. Cecil St. (187)
9. Cecil St. (201)
10. Cecil St. (207)
11. Cecil St. (207)
12. Glasgow St. (202)
13. Glasgow St. (202)
14. Cecil St. (201)
15. Huron St. (188)
16. College St. (789)
17. Spadina Ave. (183+208)
18. Spadina Ave. (219+248+254)
19. Spadina Ave. (276+290)
20. D'Arcy St. (289)
21. D'Arcy St. (277)
22. Beverley St. (288)
23. Dundas St. W. (287)
24. Dundas St. W. (270)
25. Dundas St. W. (264)
26. Simcoe St. (260)
<table>
<thead>
<tr>
<th>Location</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>27. Elm St.</td>
<td>203</td>
</tr>
<tr>
<td>28. Elm St.</td>
<td>203</td>
</tr>
<tr>
<td>31. Elm St.</td>
<td>203</td>
</tr>
<tr>
<td>32. Elm St.</td>
<td>204</td>
</tr>
<tr>
<td>33. Elm St.</td>
<td>794</td>
</tr>
<tr>
<td>34. McCaul St.</td>
<td>236</td>
</tr>
<tr>
<td>35. D'Arcy St.</td>
<td>257</td>
</tr>
<tr>
<td>36. Beverley St.</td>
<td>258</td>
</tr>
<tr>
<td>37. Baldwin St.</td>
<td>237</td>
</tr>
<tr>
<td>38. Baldwin St.</td>
<td>253</td>
</tr>
<tr>
<td>39. Spadina Ave.</td>
<td>276+290</td>
</tr>
<tr>
<td>40. Spadina Ave.</td>
<td>319</td>
</tr>
<tr>
<td>41. Dundas St. W.</td>
<td>318+312</td>
</tr>
<tr>
<td>42. Dundas St. W.</td>
<td>300+292</td>
</tr>
<tr>
<td>43. Dundas St. W.</td>
<td>300+292</td>
</tr>
<tr>
<td>44. Huron St.</td>
<td>301</td>
</tr>
<tr>
<td>45. Huron St.</td>
<td>278</td>
</tr>
<tr>
<td>46. Huron St.</td>
<td>238</td>
</tr>
<tr>
<td>47. Huron St.</td>
<td>188</td>
</tr>
<tr>
<td>48. College St.</td>
<td>149</td>
</tr>
<tr>
<td>49. Ross St.</td>
<td>180</td>
</tr>
<tr>
<td>50. Cecil St.</td>
<td>179</td>
</tr>
<tr>
<td>51. Cecil St.</td>
<td>174</td>
</tr>
<tr>
<td>52. Cecil St.</td>
<td>174</td>
</tr>
<tr>
<td>53. Beverley St.</td>
<td>221</td>
</tr>
<tr>
<td>54. Baldwin St.</td>
<td>220</td>
</tr>
<tr>
<td>55. Baldwin St.</td>
<td>209</td>
</tr>
<tr>
<td>56. Baldwin St.</td>
<td>209</td>
</tr>
<tr>
<td>29. Elm St.</td>
<td>196</td>
</tr>
<tr>
<td>30. Elm St.</td>
<td>196</td>
</tr>
<tr>
<td>57. Henry St.</td>
<td>210</td>
</tr>
<tr>
<td>58. Henry St.</td>
<td>168</td>
</tr>
<tr>
<td>59. College St.</td>
<td>136+130</td>
</tr>
<tr>
<td>60. McCaul St.</td>
<td>153</td>
</tr>
<tr>
<td>61. Orde St.</td>
<td>794+204</td>
</tr>
<tr>
<td>62. Orde St.</td>
<td>240</td>
</tr>
<tr>
<td>63. Orde St.</td>
<td>240</td>
</tr>
<tr>
<td>64. Murray St.</td>
<td>788</td>
</tr>
<tr>
<td>65. Elm St.</td>
<td>204</td>
</tr>
<tr>
<td>66. St. Patrick St</td>
<td>795</td>
</tr>
<tr>
<td>67. Dundas St. W.</td>
<td>270</td>
</tr>
<tr>
<td>68. McCaul St.</td>
<td>271</td>
</tr>
<tr>
<td>69. McCaul St.</td>
<td>236</td>
</tr>
<tr>
<td>70. Elm St.</td>
<td>794</td>
</tr>
<tr>
<td>71. Elm St.</td>
<td>794</td>
</tr>
<tr>
<td>72. McCaul St.</td>
<td>215</td>
</tr>
<tr>
<td>73. McCaul St.</td>
<td>215</td>
</tr>
<tr>
<td>74. McCaul St.</td>
<td>215</td>
</tr>
<tr>
<td>75. McCaul St.</td>
<td>195</td>
</tr>
<tr>
<td>76. McCaul St.</td>
<td>153</td>
</tr>
<tr>
<td>77. College St.</td>
<td>125+118</td>
</tr>
<tr>
<td>78. University Ave.</td>
<td>786+194</td>
</tr>
<tr>
<td>79. University Ave.</td>
<td>231+252</td>
</tr>
<tr>
<td>80. Dundas St. W.</td>
<td>259</td>
</tr>
<tr>
<td>81. Dundas St. W.</td>
<td>259</td>
</tr>
</tbody>
</table>

Figure 4.4 Euler's Path Solution to the Original Network Problem
<table>
<thead>
<tr>
<th></th>
<th>University Ave. (786+194)</th>
<th>42. Elm St. (794)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Elm St. (196)</td>
<td>43. St. Patrick St. (795)</td>
</tr>
<tr>
<td>2</td>
<td>Elm St. (196)</td>
<td>44. Dundas St. W. (270)</td>
</tr>
<tr>
<td>3</td>
<td>University Ave. (231+252)</td>
<td>45. Dundas St. W. (287)</td>
</tr>
<tr>
<td>4</td>
<td>Dundas St. W. (259)</td>
<td>46. Dundas St. W. (300+292)</td>
</tr>
<tr>
<td>5</td>
<td>Simcoe St. (260)</td>
<td>47. Dundas St. W. (318+312)</td>
</tr>
<tr>
<td>6</td>
<td>Elm St. (203)</td>
<td>48. Spadina Ave. (319)</td>
</tr>
<tr>
<td>7</td>
<td>Elm St. (203)</td>
<td>49. Spadina Ave. (276+290)</td>
</tr>
<tr>
<td>8</td>
<td>Elm St. (203)</td>
<td>50. Baldwin St. (253)</td>
</tr>
<tr>
<td>9</td>
<td>Elm St. (204)</td>
<td>51. Huron St. (238)</td>
</tr>
<tr>
<td>10</td>
<td>Elm St. (204)</td>
<td>52. Cecil St. (201)</td>
</tr>
<tr>
<td>11</td>
<td>Murray St. (788)</td>
<td>53. Glasgow St. (202)</td>
</tr>
<tr>
<td>12</td>
<td>Orde St. (240)</td>
<td>54. Glasgow St. (202)</td>
</tr>
<tr>
<td>13</td>
<td>Orde St. (240)</td>
<td>55. Cecil St. (207)</td>
</tr>
<tr>
<td>14</td>
<td>Orde St. (794+204)</td>
<td>56. Cecil St. (207)</td>
</tr>
<tr>
<td>15</td>
<td>McCaul St. (195)</td>
<td>57. Cecil St. (201)</td>
</tr>
<tr>
<td>16</td>
<td>McCaul St. (215)</td>
<td>58. Cecil St. (187)</td>
</tr>
<tr>
<td>17</td>
<td>McCaul St. (215)</td>
<td>59. Ross St. (180)</td>
</tr>
<tr>
<td>18</td>
<td>McCaul St. (215)</td>
<td>60. College St. (149)</td>
</tr>
<tr>
<td>19</td>
<td>Baldwin St. (209)</td>
<td>61. College St. (789)</td>
</tr>
<tr>
<td>20</td>
<td>Henry St. (210)</td>
<td>62. Spadina Ave. (183+208)</td>
</tr>
<tr>
<td>21</td>
<td>Henry St. (168)</td>
<td>63. Spadina Ave. (219+248+254)</td>
</tr>
<tr>
<td>22</td>
<td>College St. (138)</td>
<td>64. Spadina Ave. (276+290)</td>
</tr>
<tr>
<td>23</td>
<td>College St. (144+139)</td>
<td>65. D'Arcy St. (289)</td>
</tr>
<tr>
<td>24</td>
<td>College St. (149)</td>
<td>66. Huron St. (301)</td>
</tr>
<tr>
<td>25</td>
<td>Huron St. (188)</td>
<td>67. Dundas St. W. (300+292)</td>
</tr>
<tr>
<td>26</td>
<td>Cecil St. (187)</td>
<td>68. Beverley St. (288)</td>
</tr>
<tr>
<td>27</td>
<td>Cecil St. (179)</td>
<td>69. Beverley St. (258)</td>
</tr>
<tr>
<td>28</td>
<td>Beverley St. (221)</td>
<td>70. Baldwin St. (220)</td>
</tr>
<tr>
<td>29</td>
<td>Baldwin St. (237)</td>
<td>71. Henry St. (210)</td>
</tr>
<tr>
<td>30</td>
<td>Huron St. (278)</td>
<td>72. Cecil St. (174)</td>
</tr>
<tr>
<td>31</td>
<td>D'Arcy St. (277)</td>
<td>73. Beverley St. (175)</td>
</tr>
<tr>
<td>32</td>
<td>D'Arcy St. (257)</td>
<td>74. College St. (138)</td>
</tr>
<tr>
<td>33</td>
<td>McCaul St. (236)</td>
<td>75. College St. (136+130)</td>
</tr>
<tr>
<td>34</td>
<td>Elm St. (794)</td>
<td>76. McCaul St. (153)</td>
</tr>
<tr>
<td>35</td>
<td>Elm St. (204)</td>
<td>77. McCaul St. (153)</td>
</tr>
<tr>
<td>36</td>
<td>Elm St. (203)</td>
<td>78. College St. (125+118)</td>
</tr>
<tr>
<td>37</td>
<td>Simcoe St. (260)</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Dundas St. W. (264)</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Dundas St. W. (270)</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>McCaul St. (271)</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>McCaul St. (236)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.5 Euler's Tour for the Original Network
4.2.1 Two Way Streets

According to the theoretical approach to solving the Chinese Postman Problem, each street is to be traversed at least once before the postman finally returns to the starting point or to the point specified in advance in the case of Euler’s path. However, there are certain situations in which some streets must be visited twice for example when deliveries to both sides of the street cannot be accomplished in one visit because of one of the following:

- Wide streets
- Intensive traffic on the streets
- Ongoing construction
- Crossing only at crosswalks

In order to take these into consideration, the “Twoo” option in the form Area enhances the model, allowing the user to manipulate the network more efficiently and responding more accurately to real time problems. Simply by checking off this option, the user indicates that the particular street has to be visited twice. If these streets have to be traversed twice as a result of adding artificial branches to make odd degree nodes even, the model will ignore two-way options and treat the network accordingly.

Note that the need of visiting the same street twice cannot include the specification of the direction in which the street has to be traversed. In other words, the direction does not matter as long as the street is visited twice. The specification of the direction is impossible because the network becomes mixed when the direction is
requested. Therefore, the Chinese Postman Problem would change from NP-complete to NP-hard for which there is no known algorithm for computing the optimal solution in less than an exponential time in the size of input.

Since every street starts and ends in the nodes (intersections) which are accessible by crosswalks, there should not be any problems switching from one side of the street to the other, even though our starting point ends up being at the same end of the street for each of two visits. Therefore, each side of the street can be visited once, even though the direction is not specified.

4.2.2 Streets for which There is no Delivery

In order to address this question, let us consider the graph G (N, A) consisting of the set of nodes N and set of branches A. The solution to the Chinese Postman Problem is dynamic in its nature since the demand on the network changes from day to day. Therefore, we cannot use static models to find an Euler’s tour because network demand changes.

If no deliveries have to be made on certain streets of the network one can construct the network in the way these streets will (will not) be included in the final solution. It simply means that every subset of the branch set A may or may not be present at every time of the optimization problem. Depending on how many no delivery streets n we have on any given instance, there are $2^n$ instances of the problem. These instances are obviously possible subsets of A.
Suppose that an instance $C_i$ has a length of $l(C_i)$. A method $V$ for updating an a priori solution $g$ to the "full scale" optimization problem on the original graph $G (N, A)$ will produce a feasible solution to the $S_g(C)$ with a value $L_g(C)$ as to minimize the total length travelled by a postman in the final solution $S_f$

$$S_f = \min_{\mathcal{C} \in \text{A}} \left\{ L_k(C_1), L_k(C_2), \ldots, L_k(C_{z^*}) \right\}$$

(5.1)

<table>
<thead>
<tr>
<th>$k$</th>
<th>$V^2_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
</tr>
</tbody>
</table>

Table 4.2 Number of Permutations for $k$ Streets with no Delivery

![Permutations vs. Streets](image)

Figure 4.6 Increase of Permutations with the Number of No-Delivery Streets
As one can see in Table 4.2 and Figure 4.6, the number of permutations exponentially increases with the number of no delivery streets. Hence, in order to save computational time in the model, the number of streets that can be deleted is limited to nine \((2^9 = 512\) permutations\) since it would take a considerable amount of time to determine the optimal solution to each of these permutations.

This limitation is applied to the model in the way that the permutation with the minimum number of odd nodes is found, and then the optimal tour is determined for that instance of the problem. The original network is modified so that a couple of subsets of the problem are created. The model then treats every subset separately and evaluates it individually. All variations are applied to the network, and the number of odd degree nodes is determined for each variation. The variation giving the minimum number of odd degree nodes on the network is chosen. The Table 4.3 shows the process of defining the network and getting the final solution by adding or deleting the number of streets with no demands on them.

<table>
<thead>
<tr>
<th>Number</th>
<th>Street 1</th>
<th>Street 2</th>
<th>Street 3</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3560m</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4010m</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3300m</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3990m</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3734m</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4230m</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3420m</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3520m</td>
</tr>
</tbody>
</table>

Table 4.3 Optimal Tour for Different Variations
If the street is cancelled, the binary value 1 is attached to it in the Table 4.3, whereas the binary value 0 represents the no delivery street that is being kept as a part of the network in the process of determining the final solution with the minimum length tour.

Assume we are given a network with three no demand streets. The streets cancelled correspond to different variations shown in the first column of the Table 4.3 followed by the actual streets, denoted as street 1, street 2, and street 3 respectively. The elements of each variation are binary numbers (0, 1), whereas the number of the kth class of the variation is the total number of cancelled streets. The total length presents the final solution to the problem taking into account different variations of cancelled or kept streets on which there is no delivery. In other words, no delivery streets can be kept if the final solution is better than the solution, in the case that the streets were not taken into account.

Both double visit and no delivery streets have been identified and tested on the existing network. There are two streets that have been chosen arbitrarily for this test.

Assume Ross St. (180) has to be traversed twice because of the inaccessibility of both sides of the street due to ongoing construction, and McCaul St. (215) has no delivery on a given day. The network is modified accordingly by checking off the appropriate boxes in the form Region. The rest of the network remains the same. In the case of cancelled and double visit streets, an Euler path for the network is shown in the Figure 4.7. The total distance of the Euler path for the modified network is 10218m.
<table>
<thead>
<tr>
<th></th>
<th>Street</th>
<th>Numbers</th>
<th></th>
<th>Street</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>College St.</td>
<td>(125+118)</td>
<td>40</td>
<td>Henry St.</td>
<td>(210)</td>
</tr>
<tr>
<td>2</td>
<td>McCaul St.</td>
<td>(153)</td>
<td>41</td>
<td>Baldwin St.</td>
<td>(209)</td>
</tr>
<tr>
<td>3</td>
<td>McCaul St.</td>
<td>(195)</td>
<td>42</td>
<td>McCaul St.</td>
<td>(215)</td>
</tr>
<tr>
<td>4</td>
<td>Baldwin St.</td>
<td>(209)</td>
<td>43</td>
<td>Elm St.</td>
<td>(794)</td>
</tr>
<tr>
<td>5</td>
<td>Baldwin St.</td>
<td>(220)</td>
<td>44</td>
<td>St. Patrick St.</td>
<td>(795)</td>
</tr>
<tr>
<td>6</td>
<td>Baldwin St.</td>
<td>(237)</td>
<td>45</td>
<td>Dundas St. W.</td>
<td>(264)</td>
</tr>
<tr>
<td>7</td>
<td>Baldwin St.</td>
<td>(253)</td>
<td>46</td>
<td>Dundas St. W.</td>
<td>(264)</td>
</tr>
<tr>
<td>8</td>
<td>Spadina Ave.</td>
<td>(219+248+254)</td>
<td>47</td>
<td>Dundas St. W.</td>
<td>(270)</td>
</tr>
<tr>
<td>9</td>
<td>Cecil St.</td>
<td>(207)</td>
<td>48</td>
<td>McCaul St.</td>
<td>(271)</td>
</tr>
<tr>
<td>10</td>
<td>Glasgow St.</td>
<td>(202)</td>
<td>49</td>
<td>McCaul St.</td>
<td>(271)</td>
</tr>
<tr>
<td>11</td>
<td>Glasgow St.</td>
<td>(202)</td>
<td>50</td>
<td>Dundas St. W.</td>
<td>(287)</td>
</tr>
<tr>
<td>12</td>
<td>Cecil St.</td>
<td>(201)</td>
<td>51</td>
<td>Dundas St. W.</td>
<td>(300+292)</td>
</tr>
<tr>
<td>13</td>
<td>Cecil St.</td>
<td>(201)</td>
<td>52</td>
<td>Dundas St. W.</td>
<td>(318+312)</td>
</tr>
<tr>
<td>14</td>
<td>Cecil St.</td>
<td>(207)</td>
<td>53</td>
<td>Spadina Ave.</td>
<td>(319)</td>
</tr>
<tr>
<td>15</td>
<td>Spadina Ave.</td>
<td>(183+208)</td>
<td>54</td>
<td>Spadina Ave.</td>
<td>(276+290)</td>
</tr>
<tr>
<td>16</td>
<td>College St.</td>
<td>(789)</td>
<td>55</td>
<td>Spadina Ave.</td>
<td>(276+290)</td>
</tr>
<tr>
<td>17</td>
<td>College St.</td>
<td>(149)</td>
<td>56</td>
<td>D'Arcy St.</td>
<td>(289)</td>
</tr>
<tr>
<td>18</td>
<td>College St.</td>
<td>(144+139)</td>
<td>57</td>
<td>Huron St.</td>
<td>(301)</td>
</tr>
<tr>
<td>19</td>
<td>College St.</td>
<td>(144+139)</td>
<td>58</td>
<td>Dundas St. W.</td>
<td>(300+292)</td>
</tr>
<tr>
<td>20</td>
<td>Ross St.</td>
<td>(180)</td>
<td>59</td>
<td>Beverley St.</td>
<td>(288)</td>
</tr>
<tr>
<td>21</td>
<td>Cecil St.</td>
<td>(179)</td>
<td>60</td>
<td>D'Arcy St.</td>
<td>(257)</td>
</tr>
<tr>
<td>22</td>
<td>Cecil St.</td>
<td>(174)</td>
<td>61</td>
<td>McCaul St.</td>
<td>(236)</td>
</tr>
<tr>
<td>23</td>
<td>Henry St.</td>
<td>(168)</td>
<td>62</td>
<td>Elm St.</td>
<td>(794)</td>
</tr>
<tr>
<td>24</td>
<td>College St.</td>
<td>(136+130)</td>
<td>63</td>
<td>Elm St.</td>
<td>(204)</td>
</tr>
<tr>
<td>25</td>
<td>College St.</td>
<td>(136+130)</td>
<td>64</td>
<td>Elm St.</td>
<td>(203)</td>
</tr>
<tr>
<td>26</td>
<td>College St.</td>
<td>(138)</td>
<td>65</td>
<td>Elm St.</td>
<td>(203)</td>
</tr>
<tr>
<td>27</td>
<td>Beverley St.</td>
<td>(175)</td>
<td>66</td>
<td>Murray St.</td>
<td>(788)</td>
</tr>
<tr>
<td>28</td>
<td>Cecil St.</td>
<td>(179)</td>
<td>67</td>
<td>Orde St.</td>
<td>(240)</td>
</tr>
<tr>
<td>29</td>
<td>Cecil St.</td>
<td>(187)</td>
<td>68</td>
<td>Orde St.</td>
<td>(240)</td>
</tr>
<tr>
<td>30</td>
<td>Cecil St.</td>
<td>(187)</td>
<td>69</td>
<td>Orde St.</td>
<td>(794+204)</td>
</tr>
<tr>
<td>31</td>
<td>Cecil St.</td>
<td>(187)</td>
<td>70</td>
<td>McCaul St.</td>
<td>(153)</td>
</tr>
<tr>
<td>32</td>
<td>College St.</td>
<td>(149)</td>
<td>71</td>
<td>College St.</td>
<td>(125+118)</td>
</tr>
<tr>
<td>33</td>
<td>Huron St.</td>
<td>(188)</td>
<td>72</td>
<td>University Ave.</td>
<td>(786+194)</td>
</tr>
<tr>
<td>34</td>
<td>Huron St.</td>
<td>(238)</td>
<td>73</td>
<td>University Ave.</td>
<td>(231+252)</td>
</tr>
<tr>
<td>35</td>
<td>Huron St.</td>
<td>(278)</td>
<td>74</td>
<td>Dundas St. W.</td>
<td>(259)</td>
</tr>
<tr>
<td>36</td>
<td>D'Arcy St.</td>
<td>(277)</td>
<td>75</td>
<td>Simcoe St.</td>
<td>(260)</td>
</tr>
<tr>
<td>37</td>
<td>Beverley St.</td>
<td>(258)</td>
<td>76</td>
<td>Elm St.</td>
<td>(196)</td>
</tr>
<tr>
<td>38</td>
<td>Beverley St.</td>
<td>(221)</td>
<td>77</td>
<td>University Ave.</td>
<td>(231+252)</td>
</tr>
<tr>
<td>39</td>
<td>Cecil St.</td>
<td>(174)</td>
<td>78</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4.7 The Euler’s Path for the Modified Network**
In conclusion, this chapter has discussed the application of the Chinese Postman model to a particular section of the Toronto transportation network, beginning with the identification of the relationship between the efficiency of a postman delivery on the network and arc routing, and ending with the brief description of the real time problem that might occur as well as the flexibility of the model to cover and efficiently solve such problems.

<table>
<thead>
<tr>
<th>Type of Network</th>
<th>Starting-Ending Node</th>
<th>Total Length</th>
<th>&quot;Deadhead&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Network</td>
<td></td>
<td>7887 m</td>
<td>0</td>
</tr>
<tr>
<td>Euler's Tour</td>
<td>7- 7</td>
<td>9762 m</td>
<td>1875</td>
</tr>
<tr>
<td>Euler's Path #1</td>
<td>7- 38</td>
<td>10222 m</td>
<td>2335</td>
</tr>
<tr>
<td>Euler's Path #1 (modified)</td>
<td>7- 38</td>
<td>10218 m</td>
<td>2331</td>
</tr>
<tr>
<td>Euler's Path #2</td>
<td>3- 26</td>
<td>10094 m</td>
<td>2207</td>
</tr>
</tbody>
</table>

Table 4.4 The Summary of Results

In the case of a modified network to obtain an Euler's Path, McCaul St. (215) is chosen only because it has been visited three times in the final solution for the original network. By cancelling McCaul St. (215), the network’s number of odd degree nodes has decreased to 36, whereas the number of streets remains the same, since the next requirement on the network is to visit Ross St. (180) twice. If one simply calculates the distances of the two additional requirements imposed on the original network, one concludes that the modified path should yield to the solution that is 359 meters longer [2 x Ross St. (180) – McCaul St. (215)] than the original path.
The solution is obtained for both the original and modified network with the same starting and ending nodes. The model generates the feasible final solution for both cases with the length difference of only 4 meters.

In the case of Euler’s Path, the solution to the network problem differs for the different pairs of starting and ending nodes. As mentioned, the objective of CPP is to find the least costly path through the network covering every arc at least once by minimizing deadhead miles. Deadhead miles are segments of arcs visited by a postman who is not servicing those segments but who is using them exclusively as an access to serviceable segments.

When comparing the total number of nodes presented, our example shows an extremely high number of odd degree nodes. It further produces billions of combinations from which the best solution is to be obtained.

The model yields different street order and total distance in the final solution for the different pairs of starting/ending nodes because once the optimal matching is completed, the tour is obtained randomly.

On the other hand, the Euler’s Tour that starts and ends at the same node (seven) gives a better solution than the Euler’s Path with deadhead travel of 1875 m comparing to 2335 m for the Euler’s Path. By choosing the Euler’s Tour solution over the Euler’s Path, a savings in distance of almost 20% is achieved.
Chapter 5. Summary and Conclusion

The PKPSoft model has been developed to facilitate the application of the existing Chinese Postman Problem algorithm in a real time environment. The attempts to solve the problems faced on a network that is constantly changing will be significantly facilitated and more easily understood by applying this model.

The practical usage of the model lies in the fact that every step is visually presented and is designed to help a user to create a network and manipulate it as the new demand information becomes available. This allows the creation of the network, including or excluding all data necessary to describe the current demand and the network itself. This approach reduces the time necessary to find the solution to the network problem manually or by an arbitrary route choice based on previous experience.

The model developed in this project and based on a theoretical algorithm provides competitive results, and its application to a real time network problem leads to considerable improvements in operational performance. Initially the model was developed in consistence with the theoretical algorithm and later was improved by adjusting the current model for real time demands.

The model itself is a very useful tool for understanding different routing problems and for identifying and presenting the heuristic approaches to solving them. Besides, the computational results are presented to assess the merits of the problem.

The arc routing problems presented here consists of routing postmen when only the set of streets with demands is to be satisfied. The mail is collected and distributed to depots along the routes, which in turn are emptied by a postman who does the actual
delivery. Hence, it is necessary to schedule the delivery routes from the depot to the customers along the route. On the one hand, since the load of mail is much higher than a single postman’s capacity, a route for each postman is to be determined independently. On the other hand, crew routing problem is a hard combinatorial problem, and only relatively small instances can be solved to optimality. Therefore, the number of nodes has to be limited to the number manageable by the model. The aim of the model is to design the optimal delivery routes from a depot to a number of customers. The classic operation research problem is to find the minimum cost cycle that visits every arc on a given network at least once.

Overall results presented in the previous chapter show that the actual rule for the diversity of problem-solution situations cannot be developed since a slight difference in network definition generates a feasible solution in the random fashion. The model offers results in the form of the street order which a postman needs to travel in order to achieve the shortest tour and to satisfy the demand on the network. Since the model is capable of predicting the actual street order for two different instances of the problem on the network (a tour and a path), it is up to a user to choose which result gives a better solution. Simply by applying the model twice to the same network problem, one can get two sets of results, and by comparing them come up with the better solution.

There are several limitations of the model. For one thing it is solely based on the dataset obtained from the street files, and it cannot take into account the distances necessary to travel from the actual street to the place (box) where the delivery is to be made. Another limitation is the number of nodes and arcs the model can efficiently handle. Still, the model is suitable for small scale transportation modelling systems at a
considerable level of accuracy. Despite these disadvantages, the model reduces the amount of data necessary to compute the route of a postman on an actual network.

Through the revision of the routes of a postman, including the times/speeds needed to cover certain areas and satisfy their demand, the model can be substantially improved by optimization over time, where not only distance but also time limitations play a very important role in delivery activity.

In the future, it is expected the current model will be improved with the addition of new optimization criteria, where not only length but also time is taken into account. It would be worthwhile to compare the optimization results for both length and time to complete the picture of the delivery process from beginning to end.

Even though the formulation of this model is not a completely realistic representation of the actual problem (for example, it overlooks congestion on the network), it does provide a feasible lower bound solution and a good starting point for more complex models. In fact, the integration of the PKPSoft model into existing transportation models would be useful for the prediction of the route choices relevant to either vehicle or crew scheduling. In that case, the entire network could be taken into account, and even larger scales of the problem would be feasible under time-varying traffic conditions.
References


Appendix

Appendix I The Source Code of PKPSoft Model

Appendix II The ID Street Network Map
Appendix I

The Source Code of PKPSoft Model

The source code can be found in the accompanied 3.5” diskette in Microsoft Word (Version 2000), in read-only format. The code is in the file *Forms.Doc*. 
Private Sub Form_Load()
    frmPTT.Rejoni.Show
    On Error GoTo Greska

    Picture1.AutoRedraw = True
    Picture1.DrawWidth = 2
    Snimiti = False
    Combo = False
    Zoom = 890
    Dim j As Integer
    For j = 1 To brovoc
        X(j) = cvorX(Kraj(j))
        Y(j) = cvorY(Kraj(j))
        Xul(j) = CvorX(Poc(j))
        Yul(j) = CvorY(Poc(j))
        Ulica(j) = Ulica(Poc(j), Kraj(j))
    Next j

    If design = False Or frmPTTjedinica.txtMapa(7).Text <> "" Then
        brevora = brevor: Imax = brevor: I = brcvora: brcvora = brcvora
        If Val(frmPTTjedinica.txtKlizaH(5).Text) = 0 Then frmPTTjedinica.txtKlizaH(5).Text = 1
        If Val(frmPTTjedinica.txtKlizaV(6).Text) = 0 Then frmPTTjedinica.txtKlizaV(6).Text = 1
        Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
        txtRazmera.Text = frmPTTjedinica.txtKlizaH(5).Text
        CheckLuk.Enabled = False
        CheckPovezivanje.Enabled = False
        Mapa = frmPTTjedinica.txtMapa(7).Text
    End If

    frmCrtanje.Picture1.Picture = LoadPicture(Mapa)

    If Povezivanje = False Then
        Povezivanje = True
        Pocetni = False
    End If

    If brovoc = False Then
        lukovi = 2
        X = CvorX(lukPoc): Y = CvorY(lukPoc)
    End If
End Sub
Picture1.Line (X, Y)-(x1, y1)
X = x1; Y = y1
lukovi = lukovi + 1
Wend
lukovi = 0
Picture1.Line (X, Y)-(CvorX(lukKraj), CvorY(lukKraj))
lukPoc = 0: lukKraj = 0
End If
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub CheckPovezivanje_Click()
On Error GoTo Greska
Povezivanje = False
If CheckPovezivanje Then
Pocetni = False
Povezivanje = True
End If
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu1Cls_Click()
On Error GoTo Greska
If design = True Then
If MsgBox("ARE YOU SURE YOU WANT TO DELETE THE NETWORK?", vbYesNo, "Question") = vbYes Then
Brevora = 0: Imax = 0
If brevor <> 0 Then
Brevora = brevor
Imax = brevor
Erase CvorX, CvorY, cvorX1, cvorY1, Ulica, Poc, Kraj, Ulica1, X1ul, Y1ul, Xul, Yul
Call Pretraga(frmPTTrejoni, "DefMreze")
For j = 1 To brevor
X1ul(j) = cvorX1(Kraj(j)): Y1ul(j) = cvorY1(Kraj(j))
Xul(j) = CvorX(Poc(j)): Yul(j) = CvorY(Poc(j))
Ulica1(j) = Ulica(Poc(j), Kraj(j))
Next j
Else
Brevora = 0: Imax = 0
Erase CvorX, CvorY, cvorX1, cvorY1, Ulica, Poc, Kraj, Ulica1, X1ul, Y1ul, Xul, Yul
End If
End If
Else
frmUlice.Show
End If
Picture1.CLS
mnu3PrikazCvorova.Checked = False
mnu2PrikazUlica.Checked = False
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu1PrikazMreze_Click()
On Error GoTo Greska
Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
HScroll2.Value = Zoom
frmCrtanje.mnu1PrikazMreze.Checked = True
frmCrtanje.mnu2PrikazUlica.Checked = False
frmCrtanje.mnu3PrikazCvorova.Checked = False
pro = 2999
frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKlizaclH(5).Text)
frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKlizaclV(6).Text)
Private Sub mmu1FTëonei_Click()
    frmPTTrejoni.Show
End Sub

Private Sub mmu1Snimi_Click()
    On Error GoTo Greska
    If design = False Then Exit Sub
    frmPTTrejoni.PKP1.UpdateRecord
    Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
    HScroll2.Value = Zoom
    frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKlizacH(5).Text)
    frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKlizacV(6).Text)
    ' frmPTTrejoni.PKP1.Recordset.Bookmark = PKP1.Recordset.LastModified
    frmPTTjedinica.txtZoom(4).Value = Zoom
    If frmCrtanje.HScroll1.Value = 0 Then frmCrtanje.HScroll1.Value = 1
    If Val(frmPTTjedinica.txtKlizacH(5).Text) = 0 Then frmCrtanje.HScroll1.Value = 1
    If Val(frmPTTjedinica.txtKlizacV(6).Text) = 0 Then frmCrtanje.VScroll1.Value = 1
    frmPTTjedinica.txtKlizacV(6).Text = frmCrtanje.VScroll1.Value
    If frmCrtanje.VScroll1.Value = 0 Then frmCrtanje.VScroll1.Value = 1
    Dim X, Y, x1, y1 As Integer
    Dim foundUlica As String
    Dim pronadjeno As Boolean
    foundUlica = InputBox("Please enter the street name")
    Call PretragaMreze(frmPTTrejoni, "D", "Ulica", "Ulica", frmPTTjedinica.txtID(0).Text, foundUlica, foundUlica, pronadjeno)
    If frmPTTrejoni.txtUlica(4).Text = foundUlica Then
        frmPTTrejoni.Show
        X = frmPTTrejoni.txtX(5)
        Y = frmPTTrejoni.txtY(6)
        x1 = frmPTTrejoni.txtX(7)
        y1 = frmPTTrejoni.txtY(8)
        For j = 1 To 6
            Picture1.ForeColor = RGB(0, 0, 0)
            Picture1.Line(X, Y)-(x1, y1)
            PauseTime = 1 ' Set duration.
            Start = Timer ' Set start time.
            Do While Timer < Start + PauseTime
                DoEvents ' Yield to other processes.
            Loop
        Next j
    Else
        MsgBox "SUCH STREET DOES NOT EXIST: " & foundUlica, vbInformation, "WARNING!"
    End If
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu2Napred_Click()
    On Error GoTo Greska
    If design = False Then Exit Sub
    Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
    HScroll2.Value = Zoom

    frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKlizacH(5).Text)
    frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKlizacV(6).Text)

    Picture1.Refresh
    If Imax <> 1 Then I = I + 1

    Picture1.Circle (Xul(1), Yul(1)), 120, 225
    For k = 1 To I
        If I < 10 Then
            odu = 90
        Else
            odu = 140
        End If
        Picture1.Circle (Xul(k), Yul(k)), 120, 225
        Picture1.CurrentX = Xul(k) - odu
        Picture1.CurrentY = Yul(k) - 90
    Next k

    Picture1.Print k + 1
    Picture1.DrawWidth = 2
    Picture1.CurrentX = CvorX(k)
    Picture1.CurrentY = CvorY(k)
    Picture1.Circle (Xlul(k), Ylul(k)), 120, 225
    Picture1.CurrentX = Xlul(k) - odu
    Picture1.CurrentY = Ylul(k) - 90

    Picture1.Print k
    Picture1.Line (Xul(k), Yul(k))-(Xlul(k), Ylul(k))
    'If k > 1 Then Picture1.Line (CvorX(k - 1), CvorY(k - 1))-(CvorX(k), CvorY(k))

    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu1Exit_Click()
    On Error GoTo Greska
    If design = True Then
        frmPTTjedinica.txtRazmer(8).Text = frmCrtanje.txtRazmera.Text
        If Shnimi = False Then
            If MsgBox("HAVE YOU SAVED THE CHANGES MADE ON THE NETWORK?", vbYesNo, "Question") = vbNo
                Then Exit Sub
            End If
        End If
    End If
End If

frmUlize.Hide
Erase CvorX, CvorY, cvorXl, cvorYl, Ulica, Poc, Kraj, Tura, Cvor, D1, D2, Ulica1, Xlul, Ylul, Xul, Yul
'frmCrtanje.Hide
Brcvora = 0: brGrana = 0: UKUP = 0: brcvor = 0
Unload Me
frmPozadin.Show
frmPTTjedinica.Show
frmPTTrezont.Show
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu1Open_Click()
    On Error GoTo Greska
If design = False Then Exit Sub
frmOpen.Show
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu2PrikazUlica_Click()
On Error GoTo Greska
Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
HScroll2.Value = Zoom

Call Zoming(Zoom)
frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKlizacH(5).Text)
frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKlizacV(6).Text)
mnu2PrikazUlica.Checked = True
frmCrtanje.mnu1PrikazMreze.Checked = False

Dim j As Integer
frmCrtanje.Picture1.ForeColor = RGB(0, 0, 0)
If design = False Then
For j = 2 To t
If Tura(j - 1) = 99 Or Tura(j - 1) = 98 Or Tura(j) = 99 Or Tura(j) = 98 Then GoTo 20
frmCrtanje.Picture1.ForeColor = RGB(0, 0, 0)
frmCrtanje.Picture1.DrawWidth = 7
frmCrtanje.Picture1.Line (CvorX(Tura(j - 1)), CvorY(Tura(j - 1)))-(CvorX(Tura(j)), CvorY(Tura(j)))

frmCrtanje.Picture1.DrawWidth = 2
frmCrtanje.Picture1.DrawMode = 4
frmCrtanje.Picture1.ForeColor = RGB(255, 0, 0)
frmCrtanje.Picture1.Line (CvorX(Tura(j - 1)), CvorY(Tura(j - 1)))-(CvorX(Tura(j)), CvorY(Tura(j)))

frmCrtanje.Picture1.Line (CvorX(Tura(j - 1)), CvorY(Tura(j - 1)))-(CvorX(Tura(j)), CvorY(Tura(j)))
20 Next j
Else
For j = brGrana To Imax
frmCrtanje.Picture1.Line (CvorX(Poc(j)), CvorY(Poc(j)))-(cvorX1(Kraj(j)), cvorY1(Kraj(j)))
Next j
End If
If design = False Then frmUlice.Show
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu3Dodajcvor_Click()
On Error GoTo Greska
If design = False Then Exit Sub
Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
HScroll2.Value = Zoom

frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKlizacH(5).Text)
frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKlizacV(6).Text)
10 Brcvora = Val(InputBox("Please enter the number of the node(<=100)", "Node input"))
If Brcvora > 100 Then GoTo 10
If Brcvora = 0 Then
Brcvora = Imax
Exit Sub
End If
Brcvora = Brcvora - 1
If CvorX(Brcvora + 1) <> 0 Then
MsgBox "SORRY, THERE IS A NODE WITH THE SAME NUMBER!", vbCritical, "Warning!"
GoTo 10
End If
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu3Pocetni_Click()  ' Restart the program
On Error GoTo Greska
If design = False Then Exit Sub
If brcvora = 0 Then
If MsgBox("ARE YOU SURE YOU WANT TO RESTART A CREATION OF THE NETWORK?", vbYesNo, "Warning!") = vbYes Then
    Picture1.CLS
    brcvora = 0
Else
    brcvora = Val(InputBox("PLEASE ENTER THE NUMBER OF THE STARTING NODE (<100)")) - 1
    If brcvora > 100 Then GoTo 10
End If
End If
End Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu3Preskok_Click()  ' Skip nodes
On Error GoTo Greska
If design = False Then Exit Sub
Zoom = Val(frmJedinica.txtZoom(Text))
HScroll2.Value = Zoom

frmCrtanje.HScroll1.Value = Val(frmJedinica.txtKlizacH(Text))
frmCrtanje.VScroll1.Value = Val(frmJedinica.txtKlizacV(Text))
Brcvora = Brcvora + 1
MsgBox "A NODE SKIPPED: " & Brcvora, vbInformation, "WARNING!"
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu3Prikazcvorova_Click()  ' Show nodes
On Error GoTo Greska
Zoom = Val(frmJedinica.txtZoom(Text))
HScroll2.Value = Zoom

frmCrtanje.HScroll1.Value = Val(frmJedinica.txtKlizacH(Text))
frmCrtanje.VScroll1.Value = Val(frmJedinica.txtKlizacV(Text))
Brcvora = Brcvora + 1
MsgBox "A NODE SKIPPED: " & Brcvora, vbInformation, "WARNING!"
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Dim j As Integer
j = 1

frmCrtanje.Picture1.ForeColor = RGB(225, 0, 0)
Picture1.CLS
While Cvor(j) <> 0
    If Cvor(j) < 10 Then
        odu = 60
    Else
        odu = 100
    End If
    frmCrtanje.Picture1.ForeColor = RGB(225, 0, 0)
    If CvorX(Cvor(j)) <> 0 Then
        If CvorX(Cvor(j)) = CvorX(Cvor(j)) + 120, 255
        frmCrtanje.Picture1.CurrentX = CvorX(Cvor(j)) - odu
        frmCrtanje.Picture1.CurrentY = CvorY(Cvor(j)) - 90
        frmCrtanje.Picture1.ForeColor = RGB(0, 0, 0)
        frmCrtanje.Picture1.Print Cvor(j)
    Else
        frmCrtanje.Picture1.Circle (cvorX(Cvor(j)), cvorY(Cvor(j)))
        120, 225
        frmCrtanje.Picture1.CurrentX = cvorX(Cvor(j)) - odu
        frmCrtanje.Picture1.CurrentY = cvorY(Cvor(j)) - 90
        frmCrtanje.Picture1.ForeColor = RGB(0, 0, 0)
        frmCrtanje.Picture1.Print Cvor(j)
        CvorX(Cvor(j)) = cvorX1(Cvor(j))
CvovY(Cvov(j)) = cvovY1(Cvov(j))

End If
j = j + 1
Wend
If design = True Then
For j = brcvor + 1 To Brcvora
If CvovX(j) <> 0 And CvovY(j) <> 0 Then
    frmCrtanje.Picture1.ForeColor = RGB(225, 0, 0)
    Picture1.Circle (CvovX(j), CvovY(j)), 120, 225
    If j < 10 Then
        odu = 100
    Else
        odu = 160
    End If
    Picture1.CurrentX = CvovX(j) - odu
    Picture1.CurrentY = CvovY(j) - 90
    frmCrtanje.Picture1.ForeColor = RGB(0, 0, 0)
    Picture1.Print j
End If
Next j
End If
If design = False Then frmUlicce.Show
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub
Private Sub mnuHelp_Click()
MsgBox "PLEASE SEE THE USER'S MANUAL", vbInformation, "WARNING!"
End Sub
Private Sub Picture1_Mousemove(Button As Integer, Shift As Integer, X2 As Single, Y2 As Single)
On Error GoTo Greska
Dim duz01, duz02, duz12, duzXY, j As Integer

'Picture1.DrawWidth = 2
j = 0
If UKUP = 0 Then
    Do While j <> 1
        j = j + 1
        duz01 = (Sqr((Abs(X1(j) - X1(j))) ^ 2 + (Abs(Y1(j) - Y1(j))) ^ 2))
        duz02 = (Sqr((Abs(X2 - X1(j))) ^ 2 + (Abs(Y2 - Y1(j))) ^ 2))
        duz12 = (Sqr((Abs(X2 - X1(j))) ^ 2 + (Abs(Y2 - Y1(j))) ^ 2))
        If (duz01 + duz12) > 1 Then
            Picture1.ToolTipText = Ulica1(j)
            trenCvor = j
        Exit Do
    End If
    If (duz01 + duz12) > 1 And trenCvor = j Then
        Picture1.ToolTipText = ""
        trenCvor = 0
    End If
Loop
Else
    j = 0;
    Do While j <> brcgrana
        j = j + 1
        duz01 = (Sqr((Abs(cvorX1(Kraj(j)) - CvorX(Poc(j)))) ^ 2 + (Abs(cvorY1(Kraj(j)) - CvorY(Poc(j)))) ^ 2))
        duz02 = (Sqr((Abs(X2 - CvorX(Poc(j)))) ^ 2 + (Abs(Y2 - CvorY(Poc(j)))) ^ 2))
        duz12 = (Sqr((Abs(X2 - cvorX1(Kraj(j)))) ^ 2 + (Abs(Y2 - cvorY1(Kraj(j)))) ^ 2))
        If (duz01 + duz12) < 3 Then
            frmCrtanje.AutoRedraw = True
            frmCrtanje.Picture1.ToolTipText = Ulica(Poc(j), Kraj(j))
            trenCvor = j
            Exit Do
        End If
    End Do
End If
Do
End If

' Prolaz = Prolaz + 1
If (duz02 + duz12) - duz01 > 10 And trenCvor = j Then
    frmCrtanje.Picture1.ToolTipText = ""
    trenCvor = 0
End If
Loop
End If

j = 0
If design = False Then Brcvora = brcvor
If Brcvora > BrcvorMax Then BrcvorMax = Brcvora
Do While BrcvorMax > j
    j = j + 1
    duzXY = (Sqr((Abs(CvorX(j) - X2)) ^ 2 + (Abs(CvorY(j) - Y2)) ^ 2))
    If duzXY < 150 Then
        Picture1.ToolTipText = j
        trenCvor = j
    End If
End If

If I = 0 Then duz02 = 20 ' potreban uslov ako nije povucena nijedna ulica
If duzXY > 150 And trenCvor = j And (duz02 + duz12) - duz01 > 10 Then
    Picture1.ToolTipText = ""
    trenCvor = 0
End If

Loop
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub mnu2Nazad_Click()
On Error GoTo Greska
If design = False Then Exit Sub
Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
HScroll2.Value = Zoom

    frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKlizaH(5).Text)
    frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKlizaV(6).Text)
    Picture1.Cls

If I <> 0 Then I = I - 1

For k = 1 To I max
    If k < 10 Then
        odu = 90
    Else
        odu = 140
    End If
    Picture1.Circle (Xul(k), Yul(k)), 120, 225
    Picture1.CurrentX = Xul(k) - 90
    Picture1.CurrentY = Yul(k) - 90
    Picture1.Print k
    Picture1.Circle (X1ul(k), Y1ul(k)), 120, 225
    Picture1.CurrentX = X1ul(k) - odu
    Picture1.CurrentY = Y1ul(k) - 90
    Picture1.Print k
Next k
For k = 1 To I
    Picture1.Line (Xul(k), Yul(k))-(X1ul(k), Y1ul(k))
Next k
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, x1 As Single, y1 As Single)
    On Error GoTo Greska
    Dim razmeraZOOM As Single
    If design = False Then Exit Sub
    Snimiti = False
    If Brcvora > 1 Then
        Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
        HScroll1.Value = Zoom
        frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKlizacH(5).Text)
        frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKlizacV(6).Text)
    End If
    CheckLuk.Enabled = True
    CheckPovezivanje.Enabled = True
    Dim pronadjeno As Boolean
    pronadjeno = False
    Dim t, odu As Integer
    If Button = 2 And trenCvor <> 0 Then
        If Picture1.ToolTipText <> Ulica(trenCvor) Then
            CvorX(trenCvor) = 0
            CvorY(trenCvor) = 0
        Else
            Xul(trenCvor) = 0
            Yul(trenCvor) = 0
            X1ul(trenCvor) = 0
            Y1ul(trenCvor) = 0
            Call PretragaMreza(frmPTTrejoni, "ID", "Ulica", "Ulica", frmPTTrejoni.txtID(t), Ulica(trenCvor), Ulica1(trenCvor), pronadjeno)
        End If
        If Ima = True Then
            With frmPTTrejoni.PKP1.Recordset
                .Delete
                .MoveNext
            End With
        End If
        Picture1.ToolTipText = ""
    End If
    Picture1.CLS
    Call Mreza
    Exit Sub
End If
If Povezivanje = False And Luk = False Then
    frmCrtanje.PictureBox1.ForeColor = RGB(225, 0, 0)
    Picture1.Circle (x1, y1), 120, 225
    If t < 10 Then
        odu = 120
    Else
        odu = 160
    End If
    Brcvora = Brcvora + 1
    While CvorX(Brcvora) <> 0
        Brcvora = Brcvora + 1
    Wend
    If Brcvora = 1 Then
        frmPTTjedinica.txtZoom(4).Text = Zoom
        frmPTTjedinica.txtKlizacH(5).Text = frmCrtanje.HScroll1.Value
        frmPTTjedinica.txtKlizacV(6).Text = frmCrtanje.VScroll1.Value
    End If
    Picture1.CurrentX = x1 - odu
    Picture1.CurrentY = y1 - 90
    frmCrtanje.PictureBox1.ForeColor = RGB(0, 0, 0)
Picture1.Print Brcvora
CvorX(Brcvora) = x1
CvorY(Brcvora) = y1
Else
  If Picture1.ToolTipText = "" Then
    If Iukovi = 0 Then MsgBox "THE NODE IS NOT POSITIONED", vbCritical, "Warning!"
    Else
      If Pocetni = False Then
        X = CvorX(trenCvor)
        Y = CvorY(trenCvor)
        pretCvor = trenCvor
        Pocetni = True
      Else
        Pocetni = False
        x1 = CvorX(trenCvor)
        y1 = CvorY(trenCvor)
      If pretCvor = trenCvor Then MsgBox "A BRANCH CANNOT START AND END AT THE SAME NODE!", vbCritical, "Warning!"
      Exit Sub
    End If
  End If
'Povlacenje grane izmedju cvorova
  Razmera = 15
  PaintNow = False ' Turn off painting.
  Picture1.ForeColor = RGB(200, 0, 0) ' Select random color.
  I = I + 1
20 Ulica(pretCvor, trenCvor) = InputBox("Please enter the street name:")
  If Ulica(pretCvor, trenCvor) = "" Then GoTo 20
  Ulica1(I) = Ulica(pretCvor, trenCvor)
  Iukovi = 0

  Razmera = Val(txtRazmera.Text)
  razmeraZOOM = frmCrtnje.PictureClip1.ClipWidth / 489
  If Razmera = 0 Then
    MsgBox "PLEASE ENTER THE MAP SCALE.", vbInformation, "WARNING!"
  Exit Sub
  End If

  Razmera = ( Razmera * 1.18 ) / 10000

  Razmera = Razmera * razmeraZOOM
  If Luk = False Then Picture1.Line (X, Y)-(x1, y1)
  duzina = ((Sqr(Abs(x1 - X) ^ 2 + Abs(y1 - Y) ^ 2)) / 5) * Razmera
  duzina = Int(duzina)

  If Val(frmPTTrejoni.txtPoc(1).Text) <> 0 Then
    frmPTTrejoni.PKP1.Recordset.AddNew
  End If
  frmPTTrejoni.txtID(0).Text = frmPTTrejoni.txtID(0).Text
  frmPTTrejoni.txtPoc(1) = pretCvor
  frmPTTrejoni.txtKraj(2) = trenCvor
  lukPoc = pretCvor. lukKraj = trenCvor
  frmPTTrejoni.txtRast(3) = duzina
  frmPTTrejoni.txtUlica(4) = Ulica(pretCvor, trenCvor)
  frmPTTrejoni.txtX(5) = CvorX(pretCvor)
  frmPTTrejoni.txtY(6) = CvorY(pretCvor)
  frmPTTrejoni.txtX(7) = CvorX(trenCvor)
  frmPTTrejoni.txtY(8) = CvorY(trenCvor)
  ' If i = 1 Then frmPTTrejoni.txtPoc(1).Text = 1
  imax = 1
  Xuv(1) = CvorX(pretCvor): Yuv(1) = CvorY(pretCvor): X1uv(1) = CvorX(trenCvor): Y1uv(1) = CvorY(trenCvor)
'zavrsetak povlacenja grane
    Pocetni = False
    End If
    End If
End If

If Luk = True Then
    lukovi = lukovi + 1
    krivaX(lukPoc, lukKraj, lukovi) = xl
    krivaY(lukPoc, lukKraj, lukovi) = yl
    End If
10 Rem
Picture1.AutoRedraw = True
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub Picture1_MouseUp(Button As Integer, Shift As Integer, x1 As Single, y1 As Single)
On Error GoTo Greska
If design = False Then Exit Sub
If Brevo > 1 Then
    Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
    HScroll2.Value = Zoom
    frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKizucH(5).Text)
    frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKizucV(6).Text)
End If
    If frmCrtanje.CheckPovezivanje = False Then Exit Sub
10 Rem
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub HScroll2_Change()
On Error GoTo Greska
    HScroll2.Min = 70
    If PictureClip1.CellHeight > PictureClip1.CellWidth Then
        Stroosto = PictureClip1.CellWidth / 2
        HScroll2.Max = PictureClip1.CellWidth - 10
    Else
        HScroll2.Max = PictureClip1.CellHeight - 10
        Stroosto = PictureClip1.CellHeight / 2
    End If
    HScroll2.SmallChange = 10
    'MsgBox HScroll2.Max + 10
    'MsgBox PictureClip1.CellHeight
    Zoom = HScroll2.Value
    Call Zoming(Zoom)
    If design = False Then frmUlica.Show
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub HScroll1_Change()
On Error GoTo Greska
If HScroll1.Value = 0 Then HScroll1.Value = 1
    mnu3PrikazCvorova.Checked = False
    mnu2PrikazUlica.Checked = False
    'MsgBox Picture1.Picture.Width
    'MsgBox Picture1.ScaleMode
Dim SaveMode As Integer
    Picture1.ScaleMode = 3
    SaveMode = Picture1.ScaleMode
    HScroll1.Min = 1
    'If Picture1.Picture.Width < 12000 Then Exit Sub
    HScroll1.SmallChange = 10
If VScroll1.Max < 1 And HScroll1.Max < 1 Then
    VScroll1.Max = 1
    HScroll1.Max = 1
    Picture1.ScaleMode = SaveMode
    Picture1.ScaleMode = 1
    Exit Sub
End If
If VScroll1.Max < 1 Or HScroll1.Max < 1 Then
    HScroll2.Value = 300
    HScroll1.Value = 1
    VScroll1.Value = 1
Else
    If HScroll2.Value = HScroll2.Max Then pro = 100
    'Exit Sub
End If
PictureClip1.ClipX = HScroll1.Value
PictureClip1.ClipY = VScroll1.Value
'MsgBox pro
If pro = 100 Then
    frmCrtanje.PictureClip1.ClipHeight = 377
    frmCrtanje.PictureClip1.ClipWidth = 489
    'Picture1.Picture = PictureClip2.Picture
Else
    frmCrtanje.PictureClip1.ClipHeight = Zoom / 1.297
    frmCrtanje.PictureClip1.ClipWidth = Zoom
End If
frmCrtanje.PictureClip1.StretchX = frmCrtanje.Picture1.ScaleWidth * 1
frmCrtanje.PictureClip1.StretchY = frmCrtanje.Picture1.ScaleHeight * 1
If HScroll1.Value > 1 Then Picture1.Picture = PictureClip1.Clip
    Picture1.ScaleMode = SaveMode
    Picture1.ScaleMode = 1
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub VScroll1_change()
    On Error GoTo Greska
    If VScroll1.Value = 0 Then VScroll1.Value = 1
    mnu3PrikazCvorova.Checked = False
    mnu2PrikazUlica.Checked = False
    Dim SaveMode As Integer
    SaveMode = Picture1.ScaleMode
    Picture1.ScaleMode = 3
    VScroll1.Min = 1
    VScroll1.Max = PictureClip1.CellHeight - PictureClip1.ClipHeight
    If VScroll1.Max < 1 And HScroll1.Max < 1 Then
        VScroll1.Max = 1
        HScroll1.Max = 1
        Picture1.ScaleMode = SaveMode
        Picture1.ScaleMode = 1
        Exit Sub
    End If
    If VScroll1.Max < 1 Or HScroll1.Max < 1 Then
        HScroll2.Value = 300
        HScroll1.Value = 1
        VScroll1.Value = 1
        HScroll1.Min = 1
        VScroll1.Min = 1
        VScroll1.Max = PictureClip1.CellHeight - PictureClip1.ClipHeight
    Else
        If HScroll2.Value = HScroll2.Max Then pro = 100
        'exit sub
    End If
    'MsgBox VScroll1.Value
    VScroll1.SmallChange = 10
```vbscript
PictureClip1.ClipX = HScroll1.Value
PictureClip1.ClipY = VScroll1.Value
'MsgBox pro
If pro = 100 Then
    frmCrtanje.PictureClip1.ClipWidth = 377
Else
    frmCrtanje.PictureClip1.ClipHeight = Zoom / 1.297
    frmCrtanje.PictureClip1.ClipWidth = Zoom
End If
End If
End If
End If
End If
End If
End If
End Sub
Public Sub Mreza()
On Error GoTo Greska
Dim j, brevor1 As Integer
Dim dupli As Boolean
Dim Ulica, Poc, Kraj, Xul, Yul, X1ul, Y1ul, Cvor
Erase Ulica, Poc, Kraj, Xul, Yul, X1ul, Y1ul, Cvor
Call Pretraga(frmPTTrezoni, "DefMreze")

For j = brevor + 1 To Brevo
    If CvorY(j) <> 0 And CvorY(j) <> 0 Then
        frmCrtanje.Picture1.ForeColor = RGB(225, 0, 0)
        Picture1.Circle (CvorX(j), CvorY(j)), 120, 225
    End If
    For j = 1 To brcvor1
        If CvorY(j) = Poc(i) Then dupli = True
    Next j
If dupli = True Then
    brevor = brevor + 1
    Cvor(brevor) = Poc(i)
End If
Next j
End If
End Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub
```

End If
brcvor1 = brcvor
I = I + 1
dupli = False
Wend
I = 1
While Kraj(I) <> 0

For j = 1 To brcvor1

If Cvor(j) = Kraj(I) Then dupli = True
Next j
If dupli = False Then
brcvor = brcvor + 1
Cvor(brcvor) = Kraj(I)
End If
brcvor1 = brcvor
I = I + 1
dupli = False
Wend
j = 1
While Cvor(j) <> 0
If Cvor(j) < 10 Then
odu = 60
Else
odu = 100
End If
frmCrtanje.Picture1.ForeColor = RGB(225, 0, 0)
If CvorX(Cvor(j)) <> 0 Then
frmCrtanje.Picture1.Circle (CvorX(Cvor(j)), CvorY(Cvor(j))), 120, 255
frmCrtanje.Picture1.CurrentX = CvorX(Cvor(j)) - odu
frmCrtanje.Picture1.CurrentY = CvorY(Cvor(j)) - 90
frmCrtanje.Picture1.ForeColor = RGB(0, 0, 0)
frmCrtanje.Picture1.Print Cvor(j)
Else
frmCrtanje.Picture1.Circle (cvorX1(Cvor(j)), cvorY1(Cvor(j))), 120, 225
frmCrtanje.Picture1.CurrentX = cvorX1(Cvor(j)) - odu
frmCrtanje.Picture1.CurrentY = cvorY1(Cvor(j)) - 90
frmCrtanje.Picture1.ForeColor = RGB(0, 0, 0)
frmCrtanje.Picture1.Print Cvor(j)
CvorX(Cvor(j)) = cvorX1(Cvor(j))
CvorY(Cvor(j)) = cvorY1(Cvor(j))
End If
If BrcvorMax < Cvor(j) Then BrcvorMax = Cvor(j)

j = j + 1
Wend
For j = 1 To brcGrana
frmCrtanje.Picture1.Line (CvorX(Poc(j)), CvorY(Poc(j)))-(cvorX1(Kraj(j)), cvorY1(Kraj(j)))
Next j
trenCvor = 0
For j = 1 To brcvor
X1ul(j) = cvorX1(Kraj(j)); Y1ul(j) = cvorY1(Kraj(j))
Xul(j) = CvorX(Poc(j)); Yul(j) = CvorY(Poc(j))
Ulica1(j) = Ulica(Poc(j), Kraj(j))
Next j
For j = brcGrana + 1 To imax
If Xul(j) <> 0 And X1ul(j) <> 0 Then Picture1.Line (Xul(j), Yul(j))-(X1ul(j), Y1ul(j))
Next j
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub
Form: Chinese postman problem

Private Sub cmdStart_Click()
    frmSifra.Show
    Unload Me
End Sub

Form: Open

Private Sub cmdCancel_Click()
    On Error GoTo Greska
    Unload Me
    Exit Sub
    Greska:
        MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdOpen_Click()
    On Error GoTo Greska
    frmCrtanje.Show
    Mapa = Dir1.Path & "\" & txtFileName.Text
    'Picture1.Picture = LoadPicture("D:\VisualB\Projekti\Kina\Rejoni\Gornji grad\slide1.JPG")
    frmOpen.Hide
    frmCrtanje.PictureBox1.Picture = LoadPicture(Mapa)
    frmPTTjedinica.txtMapa(7).Text = Mapa

    frmCrtanje.PictureBoxClip1.Picture = frmCrtanje.PictureBox1.Picture
    frmCrtanje.PictureBoxClip2.Picture = frmCrtanje.PictureBox1.Picture
    frmCrtanje.PictureBoxClip1.ClipHeight = frmCrtanje.PictureBox1.ScaleHeight
    frmCrtanje.PictureBoxClip1.ClipWidth = frmCrtanje.PictureBox1.ScaleWidth
    frmCrtanje.HScroll1.Value = 1
    frmCrtanje.VScroll1.Value = 1
    frmCrtanje.HScroll2.Value = frmCrtanje.HScroll2.Max
    Exit Sub
    Greska:
        MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub Combo1_Change()
    On Error GoTo Greska
    txtFileName.Text = ""
    File1.Pattern = Combo1.Text
    Exit Sub
    Greska:
        MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub Dir1_Change()
    On Error GoTo Greska
    If Dir1.Path = Dir1.Path
    lblDirektorjum.Caption = Dir1.Path
    txtFileName.Text = ""
    Exit Sub
    Greska:
        MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub Drive1_Change()
    On Error GoTo diskProblem
    txtFileName.Text = ""
    Dir1.Path = Drive1.Drive
    Exit Sub
    diskProblem:
        Drive1.Drive = Dir1.Path
        MsgBox "THIS DISC UNIT IS UNAVAILABLE", 48, "DISC ERROR"
    Exit Sub
End Sub

Private Sub File1_Click()
On Error GoTo Greska
txtFileName.Text = File1.filename
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub Form_Load()
    On Error GoTo Greska
    cmdOpen.Enabled = False
    lblDirektorijum.Caption = Dir1.Path
    Initialpattern = "*.JPG"
    File1.Pattern = Initialpattern
    combo1.Pattern = Initialpattern
    combo1.AddItem Initialpattern
    combo1.AddItem "*.BMP"
    combo1.AddItem "*.GIF"
    combo1.AddItem "*.*
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub combo1_click()
    File1.Pattern = combo1.Text
    txtFileName.Text = ""
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub txtFileName_Change()
    On Error GoTo Greska
    If txtFileName.Text = "" Then
        cmdOpen.Enabled = False
    Else
        cmdOpen.Enabled = True
    End If
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Form: Area

Private Sub cmdTura_Click()
Put1 = False
Call Start
End Sub

Private Sub Form_Load()
    Minimiziranje = "Put"
    CheckPutMin.Value = True
    End Sub

Private Sub checkPutMin_Click()
    Minimiziranje = "Put"
    End Sub

Private Sub checkVremeMin_Click()
    Minimiziranje = "Vreme"
    End Sub

Private Sub cmdAdd_Click()
    On Error GoTo Greska
    PKP1.Recordset.AddNew
dodato = True
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdDelete_Click()
On Error GoTo Greska
If MsgBox("ARE YOU SURE YOU WANT TO DELETE THIS REGION?", vbYesNo + vbQuestion, "Warning!") <> vbNo Then

    With frmPTTjedinica.PKP1.Recordset
        .Delete
        .MoveNext
        If .EOF Then .MoveLast
    End With
End If

frmPTTrejoni.PKP1.Refresh
Call Pretraga(frmPTTrejoni, "RejonFind")
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdRefresh_Click()
    On Error GoTo Greska
    Dim pronadjeno As Boolean
    Dim Povratak As String
    Povratak = frmPTTjedinica.txtID(0)
    'This is only needed for multi user apps
    Call PretragaMreze(frmPTTjedinica, "Rejon", "PTTjedinica", "RJPTT", txtRejon(1), txtPTTjed(2), txtRJPTT(3), pronadjeno)
    If pronadjeno = False Then

        PKP1.Refresh
        'Dim pttRejoni As Recordset

        Dim strIDPTTjed As String
        Dim strRejoni As Field
        Dim I As Integer
        Dim strFind As String
        Dim Prosli As String
        Dim Prosli1 As String
        Dim strKriterij As String
        Dim pro As Integer

        strKriterij = "ID"
        strFind = Povratak
        strRejon = strFind
        If strFind <> "" Then
            strFind = strKriterij & " Like" & strFind & ""

            frmPTTjedinica.PKP1.Recordset.FindFirst strFind
            If frmPTTjedinica.PKP1.Recordset.NoMatch Then
                MsgBox "THERE IS NO REQUESTED DATA", vbInformation, "SEARCH RESULTS"
                End If
            End If
        End If
    End If
End Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub Start()
    On Error GoTo Greska
    Rem  PROGRAM ZA SOLVING CHINESE POSTMAN PROBLEM!
    design = False

    strRejon = frmPTTjedinica.txtRejon(1).Text

    If strRejon = "" Then
        MsgBox "PLEASE ENTER A NAME WORKING FILE!", vbCritical, "WARNING!"
        GoTo 2333
    End If
End If

UKUP = 0: t = 0: brcvor = 0: brGrana = 0: Zoom = 1: brGranal = 0
Erase Poc, Kraj, Tura, D1, Cvor, CvorX, cvorX1, CvorY, cvorY1, Ulica
Call Pretraga(frmPTTrezioni, "DefMreze")
frmPTTrezioni.Show
Mapa = frmPTTZedinica.txtMapa(7).Text
strRejon = """frmCrtanje.Show
frmPonuka.Show
MsgBox "SOLVING CHINESE POSTMAN PROBLEM!", vbInformation, "WARNING!"

Dim D(100, 100), DD(100, 100): Dim UKID1(100), ukid2(100) As Integer
Dim N1str(100), CVORstr, Nstr(100), NNstr, Putanja As String
'Dim Cvor(100) As Integer
Dim Q(100, 100) As Integer
Dim spoj(100, 100) As Integer
Dim POSTVEZE(100, 100), VEZA(100, 100) As Integer
Dim POVEZ(100), NN1(100), N4(100) As Integer
Dim A(100), Breza(100), ZAM(100), k As Integer
Dim PARCVOR(100), PARAL(100, 30) As Integer
Dim NEPCVOR(100), VEZAPAR(100, 20), N3(100), N As Integer
Dim N(100), N2(100) As Integer
Dim POSTVEZE(100, 100), VEZA(100, 100) As Integer
Dirn Cvor(100) As Integer
Dirn Q(100, 100) As Integer
Dim spoj(I00, 100) As Integer
Dim POSTVEZE(100, 100), VEZA(100, 100) As Integer
Dim PARNI, I2, 13, j1, dupl%, vezani, vezani1, susedNep As Integer
Dim PRVI, ZADNJI, t1, NARSPOJ, odu, UKUPI, NAP, P, NAP1 As Integer
Dim ZBIRMIN, pro As Variant
Dim CeoBroj, t2, Dzukin(10), Bin(10), Povez(100), ZbrMinICvor2, varijacije As Integer
'Određivanje broja cvorova
Dim i, j, brcvor, Cvor99, Cvor98, CvorPoc, MinT(100), brNepCvorMin As Integer
Dim brvezaDD(100) As Integer
Dim dupli, MinNepar As Boolean
Erase Cvor, Tura
brGrana = 0: t1 = 0: brGrana1 = 0: Breza1 = 0: brNepCvorMin = 50
dupli = False: Bmin = 3: ZBIRMIN = 10000000000
j = 1
brcvor = 0
brcvor1 = 1
While Poc(j) <> 0
For j = 1 To brcvor1

If Cvor(j) = Poc(0) Then dupli = True
Next j
If dupli = False Then
brcvor = brcvor + 1
Cvor(brcvor) = Poc(0)
End If
brcvor1 = brcvor
i = i + 1
dupli = False
Wend
i = 1
While Kraj(i) <> 0
For j = 1 To brcvor1

If Cvor(j) = Kraj(i) Then dupli = True
Next j
If dupli = False Then
brcvor = brcvor + 1
Cvor(brcvor) = Kraj(i)
End If
brcvor1 = brcvor
i = i + 1
dupli = False
Wend
If brcvor < 2 Then
MsgBox "THERE IS NO STREETS ON THE MAP!", vbCritical, "Warning!"
frmPonuka.Hide
Exit Sub
End If
' definisanje koord. ulica
' MsgBox Brcvor, vbInformation, "Broj cvorova"

1018 Rem PKPpkp NA NEORIJENTISANIM MREZAMA
brGrana = 0
N = brcvor
plus = 0
tl = 0: Erase D2uzkin
1030 For I = 1 To N
    For j = 1 To N
        D2(Cvor(I), Cvor(j)) = D1(Cvor(I), Cvor(j))
        Qj(Cvor(I), Cvor(j)) = Cvor(I)
        If D2(Cvor(I), Cvor(j)) = 1 Then
            If Cvor(I) < Cvor(j) Then
                tl = tl + 1
                If tl > 9 Then
                    MsgBox "THE NUMBER OF CANCELLED STREETS HAS TO BE LESS THAN TEN !", vbCritical, "Warning!"
                    Exit Sub
                End If
            End If
        End If
    Next j
    brGrana = brGrana / 2
    ' pravljenje Ojlerovog puta ****
    If Put1 = True Then
        CVORstr = InputBox("PLEASE ENTER THE NUMBER OF THE STARTING NODE", "Node input")
        Cvor99 = Val(CVORstr)
        If Cvor99 = 0 Or Cvor99 > 100 Or Val(CVORstr) = 0 Then
            MsgBox "INCORRECT NODE VALUE", vbCritical, "Warning!"
            GoTo 1040
        End If
        dupli = False
        ima = False
        For j = 1 To brcvor
            If D2(Cvor99, Cvor(j)) Then dupli = True
            If D2(Cvor99, Cvor(j)) = 1 And Brveza(Cvor99) = 1 Then
                ima = True
                MsgBox "A BRANCH IS CANCELLED: " & Cvor99 & " - " & Cvor(j), vbInformation, "WARNING!"
            End If
            If Brveza(Cvor99) = 0 Then ima = True
        Next j
        If dupli = False Or ima = True Then
            MsgBox "ENTERED NODE DOES NOT EXIST!", vbInformation, "WARNING!"
            GoTo 1040
        End If
    End If

    D(Cvor(I), Cvor(j)) = D1(Cvor(I), Cvor(j))
    Next j

    brGrana = brGrana / 2
    ' pravljenje koord. ulica
    MsgBox Brveza, vbInformation, "Broj cvorova"

    If D1(Cvor(I), Cvor(j)) = 0 Then
        D1(Cvor(I), Cvor(j)) = 9999 ' iskljuceno 8.12.99
        D2(Cvor(I), Cvor(j)) = D1(Cvor(I), Cvor(j)) ' dodato 8.12.99
    Else
        tt = 1
        If D2(Cvor(I), Cvor(j)) = 2 Or D2(Cvor(j), Cvor(I)) = 2 Then tt = 2
        MsgBox Cvor(I) & " & " & Cvor(j) & " & D1(Cvor(I), Cvor(j))
        Brveza(Cvor(I)) = Brveza(Cvor(j)) + tt
        brGrana = brGrana + tt
    End If

    D(Cvor(I), Cvor(j)) = D1(Cvor(I), Cvor(j))
    Next j

    If D2(Cvor99, Cvor(j)) = 1 And Brveza(Cvor99) = 1 Then
        ima = True
        MsgBox "A BRANCH IS CANCELLED: " & Cvor99 & " - " & Cvor(j), vbInformation, "WARNING!"
    End If
    If Brveza(Cvor99) = 0 Then ima = True
Next j
If dupli = False Or ima = True Then
    MsgBox "ENTERED NODE DOES NOT EXIST!", vbInformation, "WARNING!"
    GoTo 1040
End If

1040 CVORstr = InputBox("PLEASE ENTER THE ENDING NODE OF AN OPTIMAL ROUTE", "Node input")
    Cvor98 = Val(CVORstr)
    If Cvor98 = 0 Or Cvor98 > 100 Or Val(CVORstr) = 0 Or Cvor98 = Cvor99 Then
        MsgBox "INCORRECT NODE VALUE", vbCritical, "Warning!"
dupli = False
Ima = False
For j = 1 To brcvor
  If Cvor98 = Val(Cvor(j)) Then dupli = True
  If D2(Cvor98, Cvor(j)) = 1 And Brveza(Cvor98) = 1 Then
    MsgBox "A BRANCH IS CANCELLED: " & Cvor98 & " - " & Cvor(j), vbInformation, "WARNING!"
  End If
  If Brveza(Cvor99) = 0 Then Ima = True
Next j
If dupli = False Or Ima = True Then
  MsgBox "ENTERED NODE DOES NOT EXIST!", vbInformation, "WARNING!"
  GoTo 1045
End If
Brveza(Cvor99) = Brveza(Cvor99) + 1
Brveza(Cvor98) = Brveza(Cvor98) + 1
End If
' Napravljen put

Erase MinT: h = 0
If t1 > 9 Then
  MsgBox "THE NUMBER OF CANCELLED STREETS HAS TO BE LESS THAN TEN!", vbCritical, "Warning!"
Exit Sub
End If
varijacije = 2 ^ t1
If varijacije = 0 Then varijacije = 2
If t1 = 1 Then varijacije = 2
For t = 0 To (varijacije - 1) ' t1 je broj ukinutih gran
  For I = 1 To brcvor: brvezaDD(Cvor(I)) = Brveza(Cvor(I)): Next I
  Bin(t1) = (t) Mod 2
  CeoBroj = t
  For t2 = t1 To 1 Step -1
    Bin(t2) = CeoBroj Mod 2
    CeoBroj = Int(CeoBroj / 2)
  Next t2
  For t2 = 1 To t1
    If t = 0 Then Bin(t2) = 0
    If Bin(t2) = 1 Then
      Cvor1 = Int(D2ukin(t2) / 100)
      Cvor2 = D2ukin(t2) - Cvor1 * 100
      brvezaDD(Cvor1) = brvezaDD(Cvor1) - 1
      brvezaDD(Cvor2) = brvezaDD(Cvor2) - 1
    End If
  Next t2
  k = 0
  For I = 1 To brcvor
    If brvezaDD(Cvor(I)) = 0 Then GoTo 1046
    If brvezaDD(Cvor(I)) / 2 <= Int(brvezaDD(Cvor(I)) / 2) Then
      k = k + 1
    End If
  Next I
1046 Next I
If k < brNepCvorMin Then
  brNepCvorMin = k
  h = h: Erase MinT
  MinT(h) = t
End If
If k = brNepCvorMin Then
  h = h + 1
  MinT(h) = t
End If
Next t
For I = 1 To brcvor: brvezaDD(Cvor(I)) = Brveza(Cvor(I)): Next I
k = 0
varijacije = 2 ^ t1
If varijacije = 0 Then varijacije = 2
If t1 = 1 Then varijacije = 2
`MsgBox "tl: " & tl
zbihin1 = 50000000
For t = 0 To (varijacije - 1) 'tl je broj ukinutih grana
Brmin = 3
For I = 1 To brcvor: Brveza(Cvor(I)) = brvezaDD(Cvor(I)): Next I
MinNepar = False
For I = 1 To h
If MinT(I) = t Then MinNepar = True
Next I
If MinNepar = False And t <> 0 Then GoTo 1144
MinNepar = False
Bin(tI) = (t) Mod 2
CeoBroj = t
For tI = tl To 1 Step -1
Bh(t2) = CeoBroj Mod 2
CeoBroj = Int(CeoBroj / 2)
Next tI
For tI = 1 To tl
If t = 0 Then Bin(t2) = 0
If Bin(t2) = 1 Then
Cvr1 = Int(D2ukin(t2) / 100)
Cvr2 = D2ukin(t2) - Cvr1 * 100
'MsgBox Cvr1 & " ukinuto " & Cvr2
D2(Cvr1, Cvr2) = 1: D2(Cvr2, Cvr1) = 1
D1(Cvr1, Cvr2) = 9999: D1(Cvr2, Cvr1) = 9999
Brveza(Cvr1) = Brveza(Cvr1) - 1
Brveza(Cvr2) = Brveza(Cvr2) - 1
brGrana = brGrana - 1
End If
Next tI
k = 0
PARNI = 0
Erase NEPCVOR, PARCVOR
1060 For I = 1 To N
'MsgBox Cvor(I) & "=" & Brveza(Cvor(I))
If Brveza(Cvor(I)) = 0 Then GoTo 1080
1070 If Brveza(Cvor(I)) / 2 <= Int(Brveza(Cvor(I)) / 2) Then
   k = k + 1
   NEPCVOR(k) = Cvor(I)
Else
   PARNI = PARNI + 1
   PARCVOR(PARN1) = Cvor(I)
End If
1080 Next I
1085 ' brGrana = brGrana / 2
Debug.Print
Debug.Print
Debug.Print
Debug.Print "SOLUTION TO CHINESE POSTMAN PROBLEM"
Debug.Print
Debug.Print
1086 Debug.Print "UKUPAN BROJ GRANA NA GRAFU: " & brGrana
Debug.Print
Debug.Print "PARNI CVOROVI:"
For I = 1 To PARNI - 1: Debug.Print PARCVOR(I): ":";: Next I: Debug.Print PARCVOR(PARN1)
Debug.Print
Debug.Print "NEPARNI CVOROVI:"
For I = 1 To k - 1: Debug.Print NEPCVOR(I): ":";: Next I: Debug.Print NEPCVOR(k)
brNepCvor = k
Debug.Print
Rem *** UCITANNA RASTOJANJE ** UCITANA NAJKRACA RASTOJANJA ***
1100 GoSub 1700: Rem *** FLOYDOV ALGORITAM ***
1105 Rem ** ODREĐIVANJE VEZA IZMEDJU CVOROVA **
1110 For I = 1 To N:
1 = 0; i2 = 0; i3 = 0: If Brveza(Cvor(I)) = 0 Then GoTo 1221
For j = 1 To N:

If Cvor(I) <> Cvor(j) Then

If D1(Cvor(I), Cvor(j)) < 9999 Then
i3 = i3 + 1

If t = 0 Then POSTVEZE(Cvor(I), i3) = Cvor(j); 'MsgBox Cvor(I) & ":" & Cvor(j) 'dodato if 1.12
If D2(Cvor(I), Cvor(j)) = 2 And t = 0 Then ' Dodato 6.11.1999
i3 = i3 + 1
POSTVEZE(Cvor(I), i3) = Cvor(j)
End If
GoTo 1120 13.13.25
If Brveza(Cvor(1/j))/2 <> Int(Brveza(Cvor(1/j))/2) Then
i = i + 1:
VEZA(Cvor(I), i) = Cvor(j)
Else
i2 = i2 + 1:
VEZAPAR(Cvor(I), i2) = Cvor(j)
End If
If j1 = 1 To N
If D1(Cvor(1/j), Cvor(1/j)) < 9999 And Cvor(1/j) <> Cvor(1/j) And Cvor(1/j) <> Cvor(I) Then
For dupla = 1 To 1
If VEZA(Cvor(I), dupla) = Cvor(1/j) Then DuD = 1
Next dupla
If DuD <> 1 Then
If Brveza(Cvor(1/j))/2 <> Int(Brveza(Cvor(1/j))/2) Then
i = i + 1:
VEZA(Cvor(I), i) = Cvor(j)
End If
End If
End If
DuD = 0
End If
Next j
Rem END IF
1119 End If
End If

1120 Next j:
1121 N1(Cvor(I)) = 1; N2(Cvor(I)) = 0:
N3(Cvor(I)) = i3; ""
"MsgBox Cvor(I) & ":" & Brveza(Cvor(I))
If t = 0 Then N3(Cvor(I)) = Brveza(Cvor(I)) ' dodato 11.12.
N4(Cvor(T)) = i2:
Next I
GoTo 1223
Rem *** DODAVANJE VEZA PARNIH CVOROVA NEPARNIM ***
'If k < 14 Then GoTo 1130

1222 GoTo 1223: For I = 1 To brcvor
If Brveza(Cvor(I))/2 <> Int(Brveza(Cvor(I))/2) Then
i = 0: DuD = 0
For i2 = 1 To N4(Cvor(I))
For II = 1 To N1(VEZAPAR(Cvor(I), i2))
For dupla = 1 To N1(Cvor(I))
If VEZA(Cvor(I), dupla) = VEZA(VEZAPAR(Cvor(I), i2), II) Then
DuD = 1
End If
Next dupla
If DuD <> 1 Then
If i = 0 Then
VEZA(Cvor(I), N1(Cvor(I)) + i) = VEZA(VEZAPAR(Cvor(I), i2), II)
End If
DuD = 0
Next II
Next i2
N1(Cvor(I)) = N1(Cvor(I)) + 1
End If
Next I
1223 Rem
Erase VEZA:
For I = 1 To brcvor
If Brveza(Cvor(I))/2 = Int(Brveza(Cvor(I))/2) Then GoTo 12237
j1 = 0
For \( j = 1 \) To \( Brmin \)
\[
\begin{align*}
  &j_1 = j_1 + 1 \\
  &\text{If } Cvor(I) <> \text{NEPCVOR}(I) \text{ Then} \\
  &\text{VEZA}(Cvor(I), j) = \text{NEPCVOR}(I) \\
  &\text{Else} \\
  &\quad j_1 = j_1 + 1 \\
  &\quad \text{VEZA}(Cvor(I), j) = \text{NEPCVOR}(I) \\
  &\text{End If} \\
  &' \text{Debug.Print Cvor(I); "-"; NEPCVOR(I); } \\
  &\text{Next } j \\
  &\text{For } j = 1 \text{ To } brcvor \\
  &' \text{Debug.Print} \\
  &' \text{If } \text{Brveza}(Cvor(I)) \text{ / 2 } = \text{Int(} \text{Brveza}(Cvor(I)) \text{ / 2) Then GoTo 12235} \\
  &' \text{If } Cvor(I) <> Cvor(j) \text{ Then} \\
  &'j_1 = 1 \\
  &'\text{While } j_1 <> Brmin + 1 \\
  &'\text{If } DD(Cvor(I), Cvor(j)) < DD(Cvor(I), \text{VEZA}(Cvor(I), j)) \text{ Then} \\
  &'\quad \text{Ima} = \text{False} \\
  &'\quad \text{For } I = 1 \text{ To } Brmin \\
  &'\quad \text{If } \text{VEZA}(Cvor(I), I) = Cvor(j) \text{ Then Ima = True} \\
  &'\quad \text{Next } I \\
  &'\quad \text{If Ima = False Then} \\
  &'\quad \text{VEZA}(Cvor(I), j_1) = Cvor(j) \\
  &'\quad j_1 = Brmin \\
  &'\quad \text{End If} \\
  &'\quad \text{End If} \\
  &'\quad \text{End If} \\
  &'\quad j_1 = j_1 + 1 \\
  &'\text{Wend} \\
  &'\text{End If} \\
  &12235 \text{ Next } j \\
  &'\text{Debug.Print Cvor(I); "-"; } \\
  &'\text{For } j = 1 \text{ To } Brmin \\
  &'\text{Debug.Print VEZA(Cvor(I), j);:} \\
  &'\text{Next } j; \text{ Debug.Print} \\
  &'\text{N1(Cvor(I)) = Brmin} \\
  &12237 \text{ Next } I; \\
  &\text{GoTo 1137} \\
\end{align*}
\]

For \( I = 1 \) To \( k; j = 0 \)
\[
\begin{align*}
  &\text{If } \text{Brveza(NEPCVOR(I))} = 1 \text{ Then} \\
  &\quad \text{vezani} = \text{VEZAPAR(NEPCVOR(I), 1)} \\
  &\quad \text{vezani} = \text{NEPCVOR(I)} \\
  &\quad \text{While } \text{Brveza(vezani)} = 2 \\
  &\quad \quad j = j + 1 \\
  &\quad \text{If } \text{vezani} = \text{POSTVEZE(vezani, 1) Then} \\
  &\quad \quad \text{vezani} = \text{vezani} \\
  &\quad \quad \text{vezani} = \text{POSTVEZE(vezani, 2)} \\
  &\quad \quad \text{Else} \\
  &\quad \quad \text{vezani} = \text{vezani} \\
  &\quad \quad \text{vezani} = \text{POSTVEZE(vezani, 1)} \\
  &\quad \quad \text{End If} \\
  &\quad \text{Wend} \\
  &\quad \text{If } j > 0 \text{ Then} \\
  &\quad \quad \text{If } \text{Brveza(vezani)} / 2 \not= \text{Int(} \text{Brveza(vezani)} / 2) \text{ Then} \\
  &\quad \quad \quad \text{N1(NEPCVOR(I))} = \text{N1(NEPCVOR(I))} + 1 \\
  &\quad \quad \quad \text{N1(vezani)} = \text{N1(vezani)} + 1 \\
  &\quad \quad \quad \text{VEZA(vezani, N1(vezani)) = NEPCVOR(I)} \\
  &\quad \quad \quad \text{VEZA(NEPCVOR(I), N1(NEPCVOR(I)))} = \text{vezani} \\
  &\quad \quad \text{Else} \\
  &\quad \quad \quad \text{N4(NEPCVOR(I))} = \text{N4(NEPCVOR(I))} + 1 \\
  &\quad \quad \quad \text{VEZAPAR(NEPCVOR(I), N4(NEPCVOR(I)))} = \text{vezani} \\
  &\quad \quad \text{End If} \\
  &\quad \text{End If} \\
  &\quad \text{End If} \\
  &\quad j = 0 \\
  &\text{Next } I \\
\end{align*}
\]
**PRONADJENE SU VEZE ** * ODREDJIVANJE MOGUĆIH PAROVA ***

'1130 Rem
' CLS
' Print "SKUP GRANA OD JEDNOG CVORA KA OSTALIM:
' 1137 If brNepCvor = 2 Then
  VEZA(NepCvor(1), 1) = NepCvor(2)
  VEZA(NepCvor(2), 1) = NepCvor(1)
  POZEA(1) = NepCvor(1)
  POZEA(2) = NepCvor(2)
  GoTo 11445
End If
1138 If k = 0 Then
  ZBIRMIN = 0
  GoTo 1140
End If
1139 Rem
  MsgBox "gosub"
MsgBox brNepCvorMin
If brNepCvorMin = k Or t1 = 0 Then
  GoSub 1500: 'MsgBox "gosub"; Podprogram za spajanje neparnih cvorova
Else
  End If
  MsgBox "Minimim je": MsgBox ZBIRMIN
  If k < 14 Then plus = 3
  If k < 14 Then plus = 1
  If ZBIRMIN > 500000000 And plus < 3 Then
    If ZBIRMIN > 500000000 And Brmin < 10 Then
      'plus = plus + 1
      Brmin = Brmin + 1
      GoTo 1223
    'GoTo 1222
  End If
  If plus = 3 Then
    Erase VEZA
    For I = 1 To k
      For j = 1 To k
        If I <> j Then
          VEZA(NepCvor(I), 1) = NepCvor(j)
          MsgBox NepCvor(I): MsgBox NepCvor(j)
        End If
      Next j
    Next I
    MsgBox "THE NETWORK IS UNSOLVABLE DUE TO INSUFFICIENT CONNECTION", vbCritical, "Warning!"
    'Exit Sub
    Exit Sub
  End If
  If brNepCvorMin = k Or t1 = 0 Then
    GoSub 1500
  Else
    GoTo 1144
  End If
End If

1140 Rem *** OPTIMALNO SPARIVANJE CVOROVA ***
Debug.Print
CLS: Debug.Print "KOMBINACIJA KOJA PREDSTAVLJA OPTIMALNO SPARIVANJE NA MREZI:
Debug.Print "______________________________"
24.3 ZBIRMIN = 0
  'For I = 2 To k Step 2: MsgBox POZEA(I - 1) & " povezuje " & POZEA(I): ZBIRMIN = ZBIRMIN + D(POZEA(I - 1), POZEA(I)): Debug.Print POZEA(I - 1); ";"; POZEA(I); ";": Next
  Debug.Print
Debug.Print "______________________________"
'MsgBox "Zbirmin = " & ZBIRMIN
For t2 = 1 To t1
If Bin(t2) = 1 Then
  Cvor1 = Int(D2ukin(t2) / 100)
  Cvor2 = D2ukin(t2) - Cvor1 * 100
  Brveza(Cvor1) = Brveza(Cvor1) + 1
  Brveza(Cvor2) = Brveza(Cvor2) + 1

If ZBIRMIN < zbirmin1 Then
  For j = 1 To Brveza(Cvor1) - 1
    If Val(POSTVEZE(Cvor1, j)) = Val(Cvor2) Then
      POSTVEZE(Cvor1, j) = POSTVEZE(Cvor1, Brveza(Cvor1))
      POSTVEZE(Cvor1, Brveza(Cvor1)) = Cvor2
    End If
  Next j
  For j = 1 To Brveza(Cvor2) - 1
    If Val(POSTVEZE(Cvor2, j)) = Val(Cvor1) Then
      POSTVEZE(Cvor2, j) = POSTVEZE(Cvor2, Brveza(Cvor2))
      POSTVEZE(Cvor2, Brveza(Cvor2)) = Cvor1
    End If
  Next j

N3(Cvor1) = Brveza(Cvor1) - 1
N3(Cvor2) = Brveza(Cvor2) - 1
'MsgBox Cvor1 & " = " & N3(Cvor1)
'MsgBox Cvor2 & " = " & N3(Cvor2)
End If

End If
Next t2
Debug.Print MINIMALNI zbir " & ZBIRMIN: 'MsgBox "Zbirmin zadnji " & ZBIRMIN
'MsgBox ZBIRMIN & " = " & zbirmin1 & " = " & ZBIRMIN
If ZBIRMIN < zbirmin1 Then
  zbirmin1 = ZBIRMIN
  Erase Povezl
  For I = 1 To k
    Povezl(I) = POVEZ(I)
  Next
  Debug.Print
End If
For I = 2 To k Step 2:
  Debug.Print POVEZ(I - 1); "; "; POVEZ(I); " ";
Next I:

'MsgBox "Evo sad"
For I = 1 To N: For j = 1 To N3(Cvor(I)): 'MsgBox "(" & Cvor(I) & "," & j & ")" & " = " & POSTVEZE(Cvor(I), j): Next: Next

144 Next I
kraj velike petije
Debug.Print
For I = 2 To k Step 2:
  POVEZ(I - 1) = Povezl(I - 1); POVEZ(I) = Povezl(I): 'MsgBox POVEZ(I - 1) & " sa " & Povezl(I):
Debug.Print POVEZ(I - 1); "; "; POVEZ(I); " ";
Next I: ZBIRMIN = zbirmin1

1445 Debug.Print
For I = 2 To k Step 2:
  If D1(POVEZ(I - 1)) < 9999 Then
    N2(POVEZ(I - 1)) = N2(POVEZ(I - 1)) + 1
    PARAL(POVEZ(I - 1), N2(POVEZ(I - 1))) = POVEZ(I)
    'MsgBox N2(POVEZ(I)):
    'MsgBox POVEZ(I)
    N2(POVEZ(I)) = N2(POVEZ(I)) + 1
    PARAL(POVEZ(I), N2(POVEZ(I))) = POVEZ(I - 1)
Goto 1148
End If
'Debug.Print ","
Debug.Print "VEZA :", POVEZ(I - 1); "; POVEZ(I):
Debug.Print "OSTVARUJE SE PREKO CVOROVA: "; POVEZ(I - 1); POVEZ(I - 1)
ZADNJI = POVEZ(I)
1145 If PRVI = Q(PRVI, ZADNJI) Then
If Put1 = True And D2(PRVI, ZADNJI) = 2 Then Goto 1146
N2(PRVI) = N2(PRVI) + 1
PARAL(PRVI, N2(PRVI)) = ZADNJI
N2(ZADNJI) = N2(ZADNJI) + 1
PARAL(ZADNJI, N2(ZADNJI)) = PRVI
'end if
1146 PRVI = ZADNJI
Debug.Print ",", PRVI
Goto 1148
End If
Else
ZADNJI = Q(PRVI, ZADNJI)
End If
Goto 1145
1148 Next I
Debug.Print
Debug.Print "MINIMALNI ZBIR NEPARNIH CVOROVA: " & ZBIRMIN
Debug.Print "-----------------------------"
' Print "PREGLED PARALELNIH VEZA:"
Brveza = 0
'For I = 1 To N: For j = 1 To N3(Cvor(I)): MsgBox Cvor(I) & ";" & POSTVEZE(Cvor(I), j): Next: Next 1170 For I = 1 To brcvor: 'For j = 1 To N3(Cvor(I)): 'MsgBox Cvor(I) & ";" & POSTVEZE(Cvor(I), j): Next
'N3(Cvor(I)) = brvezaDD(Cvor(I)) '24.3. u 20.50
Brveza(Cvor(I)) = N3(Cvor(I)): 'MsgBox Cvor(I) & ";" & Brveza(Cvor(I))
For j = 1 To N2(Cvor(I))
POSTVEZE(Cvor(I), N3(Cvor(I)) + j) = PARAL(Cvor(I), j)
Next j
'MsgBox Cvor(I) & ";" & N3(Cvor(I)) & ";" & Brveza(Cvor(I)) & ";" & N2(Cvor(I))
N1(Cvor(I)) = N3(Cvor(I)) + N2(Cvor(I))
'MsgBox Cvor(I) & ";" & N1(Cvor(I))
Brveza1 = Brveza1 + N1(Cvor(I))
Next I
briGrana = Brveza1 / 2: 'MsgBox briGrana
briTura = 0
' pravljenje Ojlerovog puta ****
If Put1 = True Then
' Cvorl = Cvor99
N1(Cvorl) = N1(Cvorl) + 1: POSTVEZE(Cvorl, N1(Cvorl)) = 99
N = N + 1: Cvor(N) = 99: N1(99) = 1: POSTVEZE(99, 1) = Cvor1:
Brveza(99) = 1
Cvor2 = Cvor98
N1(Cvor2) = N1(Cvor2) + 1: POSTVEZE(Cvor2, N1(Cvor2)) = 98
N = N + 1: Cvor(N) = 98: N1(98) = 1: POSTVEZE(98, 1) = Cvor2:
Brveza(98) = 1
brcvor = brcvor + 2
Cvor1 = 99: CvorPoc = 99
For I = 1 To brcvor: Debug.Print Cvor(I) & "--" 
' For j = 1 To N1(Cvor(I))
' Debug.Print Cvor(I) & "." & POSTVEZE(Cvor(I), j)
' Debug.Print POSTVEZE(Cvor(I), j);

Next: Debug.Print
Next

1195  frmCranje.Hide
CVORstr = InputBox("PLEASE ENTER THE STARTING NODE OF AN OPTIMAL ROUTE", "Node input")

If CVORstr = "s" Then Exit Sub
If CVORstr = "" Then GoTo 1195
Cvor1 = Val(CVORstr)
If Cvor1 = 0 Or Cvor1 > 100 Then
    MsgBox "NODE ENTERED DOES NOT EXIST!", vbCritical, "Warning!"
    GoTo 1195
End If

dupli = False
Ima = False
For j = 1 To brcvor
    If Val(CVORstr) = Val(Cvor1) Then dupli = True
    If Brveza(Cvor1) = 0 Then Ima = True
Next j
If dupli = False Or Ima = True Then
    MsgBox "ENTERED NODE DOES NOT EXIST!", vbInformation, "WARNING!"
    GoTo 1195
End If

tl = 0: CvorPoc = Cvor1
1195  Tura(1) = Cvor1: brTura = 1:
'Spoj: "postveze": MsgBox POSTVEZE(99, 1)
1196  Erase Tura: Cvor1 = CvorPoc:
Tura(1) = CvorPoc: brGrana = Brveza1 / 2 + 1
'MsgBox Cvor1
'Debug.Print
brTura = brTura + 1
'Debug.Print Tura(1);
'If t = 40 Or t = 41 Or t = 42 Or t = 43 Or t = 44 Then MsgBox t
If brTura = 700 Then
    MsgBox "INAPPROPRIATE INPUT DATA!", vbCritical, "Warning!"
    frmPoruka.Hide
    MsgBox brGrana: MsgBox t
    frmUlice.Hide
    Exit Sub
End If

1197  For I = 1 To N: For j = 1 To N1(Cvor(I)): spoj(Cvor(I), j) = POSTVEZE(Cvor(I), j): Next j
NN1(Cvor(I)) = N1(Cvor(I));
Next I

UKUP = 0
1198  t = 1:
1200  GoSub 2000
1210  t = t + 1: Tura(t) = tl: ' Debug.Print tl;
1214  I = 0
1215  I = I + 1:
    If I = 30 Then
    GoTo 1196
    MsgBox "INAPPROPRIATE INPUT DATA!", vbCritical, "Warning!"
    frmPoruka.Hide
    Exit Sub
End If
'Debug.Print "Spoj cvora: " & Tura(t) & ", sa " & Cvor1 & ", tl=" & I
If Tura(t) = 0 Then GoTo 1196

'MsgBox t & ", br grana: " & brGrana
If Tura(t) = 98 And t = brGrana + 1 Then GoTo 1230
If spoj(Tura(t), I) = Cvor1 Then
NARSPOS = i:
GoSub 2020:
If Tura(i) = 0 Then GoTo 1216
Else
GoTo 1215
End If
1220 Cvorl = Tura(i):
'MsgBox Cvor(I) & " brveza " & NN1(Cvor(I))

If NN1(Cvor(I)) = 0 Then
"If brGrana <= t - 1 Or Tura(I) <> Cvorl Then
"If brGrana + 1 <= t And brGrana <= 30 And brGrana - 1 <= t Then ' dodato 15.3
If brGrana <= t And brGrana <= t - 1 And brGrana <= t + 1 Then
'If brGrana <= t Then
GoTo 1196:
Else
'MsgBox t & " br grana: " & brGrana

'4. If brGrana + 1 <= t And brGrana <= t And brGrana - 1 <= t And Putl = True Then GoTo 1196 ' dodato 15.3
If brGrana <= t And Putl = True Then GoTo 1196
GoTo 1230:
End If
Else
GoTo 1200
End If
1230
'if checkpoint.Value
Debug.Print
Debug.Print "RESENIJE PROBLEMA KINESKOG POSTARA NA MREŽI: " & frmPTTjedinica.txtRejon(l).Text
Debug.Print
UKUP = 0
Debug.Print

Debug.Print "JEDNA OD MOGUCIH OJEROVII TURA KOJA POČINJE U CVORU": Tura(1); ":"
Debug.Print

If Tura(l) < 10 Then
odu = 90
Else
odu = 140
End If
frmPrahKHide
frmCrtanje.Show
frmCrtanje.CheckLuk.Enabled = False
frmCrtanje.CheckPovezivanje.Enabled = False
If CvorX(Tura(l)) <> 0 Then
'frmCrtanje.PictureBox1.Circle (CvorX(Tura(l)), CvorY(Tura(l))), 120, 225
frmCrtanje.PictureBox1.Circle (CvorX(Tura(l)), CvorY(Tura(l))), 120, 225
frmCrtanje.PictureBox1.CurrentX = CvorX(Tura(l)) - odu
frmCrtanje.PictureBox1.CurrentY = CvorY(Tura(l)) + 90
frmCrtanje.PictureBox1.Print Tura(l)
Else
frmCrtanje.PictureBox1.Circle (CvorX(l(Tura(l)), cvoY(Tura(l))), 120, 225
frmCrtanje.PictureBox1.CurrentX = cvoX(l(Tura(l))) - odu
frmCrtanje.PictureBox1.CurrentY = cvoY(l(Tura(l))) + 90
frmCrtanje.PictureBox1.Print Tura(l)
CvorX(Tura(l)) = cvoX(l(Tura(l)))
CvorY(Tura(l)) = cvoY(l(Tura(l)))
End If

Debug.Print Tura(l);
frmUlise.Show
frmUlise.ListUlise.Clear

frmUlise.ListUlise.Text = ""
Tura(l + 1) = Tura(l)
1240 For I = 2 To t: Debug.Print Tura(I);
If Tura(I) < 10 Then
odu = 90
Else
odu = 140
End If
If CvorX(Tura(i)) <> 0 Then
  frmCrtanje.Picture1.Circle (CvorX(Tura(i)), CvorY(Tura(i))), 120, 225
  frmCrtanje.Picture1.CurrentX = CvorX(Tura(i)) - odu
  frmCrtanje.Picture1.CurrentY = CvorY(Tura(i)) - 90
  frmCrtanje.Picture1.Print Tura(i)
Else
  frmCrtanje.Picture1.Circle (cvorX1(Tura(i)), cvorY1(Tura(i))), 120, 225
  frmCrtanje.Picture1.CurrentX = cvorX1(Tura(0)) - odu
  frmCrtanje.Picture1.CurrentY = cvorY1(Tura(0)) - 90
  frmCrtanje.Picture1.Print Tura(i)
CvorX(Tura(i)) = cvorX1(Tura(0))
CvorY(Tura(i)) = cvorY1(Tura(0))
End If
Putanja = (" & Tura(i - 1) & "; & Tura(i) & ")
j = Len(Putanja)
j = 25 - j
If j = 18 Then j = 16
If j = 19 Then j = 18
Putanja = Putanja & Space(j - 3) & Ulica(Tura(i - 1), Tura(i))
frmCrtanje.Picture1.ForeColor = RGB(225, 0, 0)
frmCrtanje.Picture1.DrawMode = 1
frmCrtanje.Picture1.DrawWidth = 7
frmCrtanje.Picture1.AutoRedraw = True
If Tura(i - 1) = 99 Or Tura(i - 1) = 98 Or Tura(i) = 99 Or Tura(i) = 98 Then GoTo 1247
frmCrtanje.Picture1.Line (CvorX(Tura(i - 1)), CvorY(Tura(i - 1)))-(CvorX(Tura(i)), CvorY(Tura(i)))
 FrmCrtanje.Picture1.ForeColor = RGB(225, 0, 0)
frmCrtanje.Picture1.DrawWidth = 2
frmCrtanje.Picture1.DrawMode = 4
frmUlice.ListUlica.AddItem Putanja
frmCrtanje.Picture1.Line (CvorX(Tura(i - 1)), CvorY(Tura(i - 1)))-(CvorX(Tura(i)), CvorY(Tura(i)))
 If Minimiziranje = "Put" Then
  D1(Tura(i - 1), Tura(i)) = DD(Tura(i - 1), Tura(i))
 Else
  D1(Tura(i - 1), Tura(i)) = D3(Tura(i - 1), Tura(i))
 If D2(Tura(i - 1), Tura(i)) = 2 Then
  D2(Tura(i - 1), Tura(i)) = 3
  Else
  D2(Tura(i - 1), Tura(i)) = CInt(DD(Tura(i - 1), Tura(i)) / 100)
End If
End If
1247  UKUP = UKUP + D1(Tura(i - 1), Tura(i)): Next i
Debug.Print
Debug.Print "UKUPNA DUZINA OPTIMALNE RUTE := UKUP"
If Minimiziranje = "Put" Then
  frmUlice.txtTura.Text = UKUP & " m"
Else
  frmUlice.txtTura.Text = UKUP & " min"
End If
UKUP1 = UKUP

Erase D1, D2
Erase D, DD, UKID1, ukid2
Erase N1str: Erase N1str: NNstr = ""
Erase D2
Erase Q
Erase spoj
Erase POSTVEZE, VEZA
Erase POVEZ, NN1, N4
Erase A, Brzeza, ZAM
Erase PARCVOR, PARAL
Erase NEPCVOR, VEZAPAR, N3
Erase N1, N2
Brvezal = 0: DuD = 0: Cvor1 = 0: CVORstr = ""
strRejon = ""
GoTo 2333

1500 Rem *** KOMBINACIJE ***
' MsgBox "gosub"
' MsgBox k
1510 ZBIRM = 100000000000#: NAP1 = 0: pro = 0: NAP = 0: X = 0
1520 For I = 1 To k
   A(I) = NEPCVOR(I)
   ZAM(I) = I
   Next I
   NAP = 0: GoSub 1650
1530 NAP = 2: GoTo 1565

1540 If ZAM(NAP) <> k Then
   GoSub 1650
   GoTo 1550
Else
   NAP = NAP - 2
If NAP = 0 Then
   MsgBox pro
   Return
Else
   For I = NAP + 1 To k: ZAM(I) = I: Next I: GoTo 1540
End If
End If
1550 ZAM(NAP) = ZAM(NAP) + 1
1560 pro = pro + 1
1570 NAP = NAP + 1
If VEZA(A(NAP - 1), II) <> A(NAP1) Then
   If VEZA(A(NAP - 1), II) = 0 Then
      II = 0
      If ZAM(NAP1) <> k Then
         GoTo 1550
      Else
         GoTo 1540
      End If
   Else
      GoTo 1570
   End If
End If
1580 If NAP1 < k + 1 Then
   NAP1 = NAP1 + 2: NAP = NAP1: II = 0:
   GoTo 1570
End If
1590 Zbir = 0: For I = 2 To k Step 2: ' Debug.Print A(I - 1); ":; A(I); " ":; ' MsgBox A(I - 1) & ":; & A(I):
   Zbir = Zbir + D(A(I - 1), A(I)): Next I: ' Debug.Print Zbir: Debug.Print
   If Zbir < ZBIRM Then
      ZBIRM = Zbir: For I = 1 To k: POVEZ(I) = A(I): Next I
   End If
1620 NAP = k - 2
If pro = 100000 Then
   MsgBox "THE NETWORK IS UNSOLVABLE DUE TO INSUFFICIENT CONNECTION", vbInformation,
"Information"
   ' MsgBox plus
   Return
' Exit Sub
End If
1630 GoTo 1540
1650 Rem *** REDJANJE U RASTUCI NIZ ***
1660 For I = NAP + 1 To k: For j = I + 1 To k
   If A(I) * 10 < A(I) * 10 Then
P = A(j): A(j) = A(l): A(l) = P
End If
1670 Next j: Next I
1680 Return

1700 Rem *** ALGORITAM FLOYD-a ***
1710 For k1 = 1 To N
1720 For I = 1 To N
1730 For j = 1 To N
1740 If Cvor(I) = Cvor(j) Then
1750 GoTo 1760
End If
1750 If D(Cvor(I), Cvor(j)) > D(Cvor(I), Cvor(k1)) + D(Cvor(k1), Cvor(j)) Then
1760 Q(Cvor(I), Cvor(j)) = Cvor(k1)
End If
1760 Next j
1770 Next I
1780 Next k1
1790 Return

2000 NARSPOJ = Int(Rnd * NN1(Cvor1)) + 1: tl = spoj(Cvor1, NARSPOJ)
2010 For I = NARSPOJ To NN1(Cvor1): spoj(Cvor1, I) = spoj(Cvor1, I + 1):
2020 Next I: NN1(Cvor1) = NN1(Cvor1) - 1
2030 Return

Exit Sub
Greska:
' MsgBox "Restart the program!", vbCritical, "Warning!"
2333 End Sub

Private Sub cmdUpdate_Click()
On Error GoTo Greska
Dim pronadjeno As Boolean
Call PretragaMreze(frmPTTJedinica, "Rejon", "PTTjedinica", "RJPTT", txtRejon1.Text, txtPTTjed(1).Text, txtRJPTT(3).Text, pronadjeno)
If pronadjeno = False Then
    PKP1.UpdateRecord
    ' MsgBox "THERE IS NO DUPLICATES"
    frmPTTjedinica.PKP1.Recordset.Bookmark = PKP1.Recordset.LastModified
    Call Pretraga(frmPTTRejoni, "RejonFind")
    frmPTTjedinica.Show
Else
    MsgBox "NUMBER OF REGIONS WITH THE SAME NAME:" & duplikat, vbCritical, "Warning!"
End If
Exit Sub
Greska:
' MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdClose_Click()
On Error GoTo Greska
Screen.MousePointer = vbDefault
Unload Me
frmPTTRejoni.Hide
End Sub
Greska:
' MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub
Private Sub cmdPretraga_Click()
On Error GoTo Greska

Dim Povratak As String
If dodato = True Then
    PKP1.UpdateRecord
    PKP1.Recordset.MoveLast
dodato = False
End If
Povratak = frmPTTJedinica.txtID(0).Text

    Call Pretraga(frmPTTJedinica, "RejonFind")
    frmPTTJedinica.Show
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdPretraga_Click()
On Error GoTo Greska
'Dim ptrRejoni As Recordset
'Dim Forma As Recordset

Dim strDPTTjed As String
'Dim strRejoni As Field
Dim I As Integer
Dim strFind, strPTTjed As String
Dim Prosli As String
Dim strKriterij As String
Dim pronadjeno As Boolean
strKriterij = "Rejon"

strFind = InputBox("PLEASE ENTER THE REGION OF A SEARCH:" & strFind, "Search")
strRejoni = strFind
If strFind <> "" Then
    strFind = strKriterij & " Like" & strFind & ""

    frmPTTJedinica.PKP1.Recordset.FindFirst strFind
    If frmPTTJedinica.PKP1.Recordset.NoMatch = True Then
        MsgBox "NO REQUESTED REGION ", vbInformation, "SEARCH RESULTS"
    Else
        frmPTTJedinica.PKP1.Recordset.FindNext strFind
        If frmPTTJedinica.PKP1.Recordset.NoMatch = True Then
            Call Pretraga(frmPTTJedinica, "DelMreze")
        Else
            strPTTjed = InputBox("PLEASE ENTER THE AREA OF A SEARCH:" , "Area search")
            Call PretragaMreze(frmPTTJedinica, "PTTJedinica", "Rejon", "Rejon", strPTTjed, strRejoni, strRejoni, pronadjeno)
        End If
    End If
Else
    strPTTjed = frmPTTJedinica.txtPTTjed2(0).Text
    If strPTTjed <> frmPTTJedinica.txtPTTjed2(2).Text Then MsgBox "There is no this region < & strRejoni & ">", vbInformation, "WARNING!"
End If
frmPTTJedinica.Show
End Sub

Private Sub pkpl_Error(DataErr As Integer, Response As Integer)
On Error GoTo Greska
    MsgBox "Problems with data base! ", vbCritical, "Error"
Response = 0 'Throw away the error
End Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub
Private Sub pkp1_Reposition()
  Screen.MousePointer = vbDefault
  On Error Resume Next
  'This will display the current record position for dynasets and snapshots
  PKP1.Caption = "Record: " & (PKP1.Recordset.AbsolutePosition + 1)
End Sub

Private Sub pkp1_Validate(Action As Integer, Save As Integer)
  On Error GoTo Greska
  Dim pronadjeno As Boolean
  Promena = False
  If Save = -1 Then Promena = True
  'This is where you put validation code
  This event gets called when the following actions occur
  Select Case Action
    Case vbDataActionMoveFirst
    Case vbDataActionMovePrevious
      If Promena = True Then
        PKP1.UpdateRecord
        Call PretragaMreze(frmPTTjedinica, "Rejon", "PTTjedinica", "RJPTT", txtRejon(1), txtPTTjed(2), txtRJPTT(3), pronadjeno)
        If pronadjeno = True Then
          PKP1.Recordset.Delete
        End If
      End If
    End If
    Case vbDataActionMoveNext
      If Promena = True Then
        PKP1.UpdateRecord
        Call PretragaMreze(frmPTTjedinica, "Rejon", "PTTjedinica", "RJPTT", txtRejon(1), txtPTTjed(2), txtRJPTT(3), pronadjeno)
        If pronadjeno = True Then
          PKP1.Recordset.Delete
        End If
      End If
    End If
    Case vbDataActionMoveLast
    Case vbDataActionAddNew
    Case vbDataActionUpdate
    Case vbDataActionDelete
    Case vbDataActionFind
    Case vbDataActionBookmark
    Case vbDataActionClose
      Screen.MousePointer = vbDefault
  End Select
  Screen.MousePointer = vbHourglass
  Exit Sub
Greska:
  MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub Form_Unload(Cancel As Integer)
  On Error GoTo Greska
  Screen.MousePointer = vbDefault
  Exit Sub
Greska:
  MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub txtID_Change(Index As Integer)
  On Error GoTo Greska
  Dim Povratak As String
  If dodato = True Then
    'PKP1.UpdateRecord
    'PKP1.Recordset.MoveLast
    dodato = False
  End If
End Sub
End If
Povratak = frmPTTjedinica.txtID(0).Text

Call Pretraga(frmPTTrejoni, "RejonFind")
frmPTTrejoni.Show
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Form: Region

Private Sub CheckDvosmema_Click()
    If CheckDvosmema = 1 Then
        txtDvosmema(9).Text = 2
    Else
        txtDvosmema(9).Text = 0
    End If
    CheckUkidanje.Value = 0
End Sub

Private Sub CheckUkidanje_Click()
    If CheckUkidanje.Value = 1 Then
        txtDvosmema(9).Text = 1
    Else
        txtDvosmema(9).Text = 0
    End If
    CheckDvosmema.Value = 0
End Sub

Private Sub cmdAdd_Click()
    On Error GoTo Greska
    dpt1.Recordset.AddNew
    txtID(0).Text = frmPTTjedinica.txtID(0).Text
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdDelete_Click()
    On Error GoTo Greska
    With dpt1.Recordset
        .Delete
        .MoveNext
        If .EOF Then .MoveLast
    End With
    Call Chekiranje
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdDesno_Click()
    On Error GoTo Greska
    Call Pretraga(frmPTTrejoni, "Desno")
    Call Chekiranje
    Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdDupla_Click()
    On Error GoTo Greska
    Dim findUlica As String
    Dim pronadjeno As Boolean

    findUlica = InputBox("Please enter the street name")
    Call PretragaMreze(frmPTTrejoni, "ID", "Ulica", "Ulica", frmPTTjedinica.txtID(0).Text, findUlica, findUlica, pronadjeno)
    If frmPTTrejoni.txtUlica(4).Text = findUlica Then
        frmPTTrejoni.Show
    End If
End Sub

formPTTjedinica.Show
Call Chekiranje
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"

End Sub

Private Sub cmdGrana_Click()
On Error GoTo Greska
Dim pronadjeno As Boolean
Dim PocCvor As String
Dim KrajCvor As String
PocCvor = InputBox("PLEASE ENTER THE STARTING NODE", "TEMPORAL BRANCH CANCELATION")
KrajCvor = InputBox("PLEASE ENTER THE ENDING NODE", "TEMPORAL BRANCH CANCELATION")
Ima = False
Call PretragaMreze(frmPTTrejoni, "ID", "Pocetncvor", "Krajnjicvor", frmPTTrejoni.txtID(0), PocCvor, KrajCvor, pronadjeno)
If Ima = False Then
    Call Pretraga(frmPTTrejoni, "RejonFind")
    Call PretragaMreze(frmPTTrejoni, "ID", "Pocetncvor", "Krajnjicvor", frmPTTrejoni.txtID(0), KrajCvor, PocCvor, pronadjeno)
End If
End If
If Ima = True Then
    D2(PocCvor, KrajCvor) = 1
    D2(KrajCvor, PocCvor) = 1
Else
    MsgBox "THERE IS NO SUCH STREET", vbInformation, "Information"
End If
Ima = False
frmPTTJedinica.Show
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdLevo_Click()
On Error GoTo Greska
Call Pretraga(frmPTTrejoni, "Levo")
Call Chekiranje
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdMapa_Click()
On Error GoTo Greska
design = True
Erase CvorX, CvorY, cvorX1, cvorY1, Ulica, Poc, Kraj, Tura
UKUP = 0
If frmPTTJedinica.txtMapa(7).Text <> "" Then
    Call Pretraga(frmPTTrejoni, "DefMreze")
    Dim i, j, brevor1 As Integer
    Dim dupli As Boolean
    Erase Cvor, Tura
dupli = False
    I = 1
    brevor = 0
    brevor1 = 1
    While Poc(I) <> 0
        For j = 1 To brevor1
            If Cvor(j) = Poc(I) Then dupli = True
            Next j
            If dupli = False Then
                brevor = brevor + 1
                Cvor(brevor) = Poc(I)
            End If
            brevor1 = brevor
            I = I + 1
            dupli = False
Wend
  I = 1
  While Kraj(I) <> 0
  For j = 1 To brcvor1
  If Cvor(j) = Kraj(I) Then dupli = True
  Next j
  If dupli = False Then
  brcvor = brcvor + 1
  Cvor(brcvor) = Kraj(I)
  End If
  brcvor1 = brcvor
  I = I + 1
  dupli = False
Wend
  frmCrtanje.Show
Else
  frmPTTjedinica.pkp1.UpdateRecord
  frmPTTjedinica.pkp1.Recordset.MoveLast
  frmOpen.Show
End If
Exit Sub
Greska:
  MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdRefresh_Click()
  On Error GoTo Greska
  pkp1.Refresh
  Call Chekiranje
  Exit Sub
  Greska:
  MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdUpdate_Click()
  On Error GoTo Greska
  frmPTTtrejoni.pkp1.UpdateRecord
  frmPTTtrejoni.pkp1.Recordset.Bookmark = pkp1.Recordset.LastModified
  Call Chekiranje
  frmPTTtrejoni.Show
  frmUlize.Hide
  Exit Sub
  Greska:
  MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub cmdClose_Click()
  On Error GoTo Greska
  Screen.MousePointer = vbDefault
  Unload Me
  Exit Sub
  Greska:
  MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub pkp1_Error(DataErr As Integer, Response As Integer)
  On Error GoTo Greska
  MsgBox "Problem with database!", vbCritical, "Greska"
  Response = 0
  ' Throw away the error
  Exit Sub
  Greska:
  MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub pkp1_Reposition()
Screen.MousePointer = vbDefault
On Error Resume Next
' This will display the current record position for datasets and snapshots
pkp1.Caption = "Record: " & (pkp1.Recordset.AbsolutePosition + 1)
End Sub

Private Sub pkp1_Validate(Action As Integer, Save As Integer)
    On Error GoTo Greska
    Select Case Action
        Case vbDataActionMoveFirst
        Case vbDataActionMovePrevious
        Case vbDataActionMoveNext
        Case vbDataActionMoveLast
        Case vbDataActionAddNew
        Case vbDataActionUpdate
        Case vbDataActionDelete
        Case vbDataActionFind
        Case vbDataActionBookmark
        Case vbDataActionClose
            Screen.MousePointer = vbDefault
    End Select
    Screen.MousePointer = vbHourglass
    Exit Sub
    Greska:
        MsgBox "Restart the program!", vbCritical, "Warning!"
    End Sub

Private Sub Form_Unload(Cancel As Integer)
    Screen.MousePointer = vbDefault
    Exit Sub
    Greska:
        MsgBox "Restart the program!", vbCritical, "Warning!"
    End Sub

Private Sub Form_Load()
    On Error GoTo Greska
    txtID(0).Text = frmOptjenica.TextID(0).Text
    Call Chekiranje
    Exit Sub
    Greska:
        MsgBox "Restart the program!", vbCritical, "Warning!"
    End Sub

Private Sub Timer1_Timer()
    On Error GoTo Greska
    Call Pretraga(frmOpttreni, "")
    Timer1.Enabled = False
    Exit Sub
    Greska:
        MsgBox "Restart the program!", vbCritical, "Warning!"
    End Sub

Public Sub Chekiranje()
    If txtDupla(9).Text = "1" Then
        CheckUkidanje.Value = 1
    Else
        CheckUkidanje.Value = 0
    End If
    If txtDupla(9).Text = "2" Then
        CheckDvosmera.Value = 1
    Else
        CheckDvosmera.Value = 0
    End If
End Sub

Form: Password

Private Sub cmdPotvrda_Click()
txtSifra.SetFocus
Sifra = txtSifra.Text
If Sifra <> "" Then
    MsgBox "INCORRECT PASSWORD", vbInformation, "UNAUTHORIZED ACCESS"
End If
End Sub
frmPozadina.Show
frmPTTjedinica.Show
frmPTTrejoni.Show
Unload Me
End Sub

Form: List of Streets

Private Sub cmdWord_Click()
On Error GoTo Greska
Dim word As Object
Set word = CreateObject("Word.application")
With word
    .Visible = True
    .Documents.Add
    .selection.Font.Name = "Arial"
    .selection.Font.Size = "14"
    .selection.typeText = "City:" & frmPTTjedinica.txtPJT(t(3)).Text & Space(150)
    .selection.typeText = "Area:" & frmPTTjedinica.txtPTTjed(2).Text & Space(150)
    .selection.typeText = "Date:" & Date$ & Space(150)
    .Selection.Font.Bold = True
    .Selection.Font.Size = "18"
    .Selection.typeText = "R E P O R T" & Space(100)
    .Selection.Font.Bold = False
    .Selection.Font.Size = "14"
    .Selection.typeText = "The order of streets in which the postman needs to traverse the network: " & frmPTTjedinica.txtRejoni(1).Text & Space(20)
    .Selection.typeText = "------------------------------" & Space(130)
    .Selection.Font.Size = "12"
    For j = 2 To t
        .Selection.typeText = (j - 1) & ". " & Ulica(Tura(j - 1), Tura(j))
        .Selection.typeText = Space(150 - Len(Ulica(Tura(j - 1), Tura(j))))
    Next j
End With
word.quit
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Private Sub ListUlica_Click()
On Error GoTo Greska
    'If Zoom <> Val(frmPTTjedinica.txtZoom(4).Text) Then
    Zoom = Val(frmPTTjedinica.txtZoom(4).Text)
    frmCrtanje.HScroll2.Value = Zoom
    pro = 2999
    frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKlizacH(5).Text)
    frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKlizacV(6).Text)
    'End If
    ' frmCrtanje.HScroll1.Value = Val(frmPTTjedinica.txtKlizacH(5).Text)
    ' frmCrtanje.VScroll1.Value = Val(frmPTTjedinica.txtKlizacV(6).Text)
    10 Rem
Dim X, Y, x1, y1 As Integer
Dim Cvor, Cvor1, Cvor2, zarez, j, odu As Integer
Dim pronadjeno As Boolean
For j = 2 To 8
    Cvor = Trim(Mid$(ListUlica.Text, j, 1))
    If Cvor = "" Then
        Cvor1 = Trim(Mid$(ListUlica.Text, 2, j - 2))
        zarez = j
    End If
End Sub
If Cvor = "") Then Cvor2 = Trim(Mid$(ListUlica.Text, zarez + 1, j - 1 - zarez))
Next j

' MsgBox Cvor1 & " " & Cvor2
If ListUlica.ListIndex = 0 Then
    frmCrtanje.Picture1.CLS
    frmCrtanje.Picture1.DrawMode = 1
    If frmCrtanje.mnu3PrikazCvorova.Checked = True Then frmCrtanje.Picture1.Circle (CvorX(Tura(1)), CvorY(Tura(1))), 120, 225
For j = 2 To t
    If Tura(j - 1) = 99 Or Tura(j - 1) = 98 Or Tura(j) = 99 Or Tura(j) = 98 Then GoTo 20
    frmCrtanje.Picture1.ForeColor = RGB(0, 0, 0)
    frmCrtanje.Picture1.DrawMode = 1
    frmCrtanje.Picture1.DrawWidth = 7
    frmCrtanje.Picture1.Line (CvorX(Tura(j - 1)), CvorY(Tura(j - 1)))-(CvorX(Tura(j)), CvorY(Tura(j)))
End If
    ' frmCrtanje.Picture1.DrawWidth = 2
    frmCrtanje.Picture1.DrawMode = 4
    ' frmCrtanje.Picture1.ForeColor = RGB(255, 0, 0)
    frmCrtanje.Picture1.Line (CvorX(Tura(j)), CvorY(Tura(j)))-(CvorX(Tura(j)), CvorY(Tura(j)))
    If frmCrtanje.mnu3PrikazCvorova.Checked = True Then frmCrtanje.Picture1.Circle (CvorX(Tura(j)), CvorY(Tura(j))), 120, 225
    If Tura(j) < 10 Then
        odu = 90
    Else
        odu = 140
    End If
    frmCrtanje.Picture1.CurrentX = CvorX(Tura(j)) - odu
    frmCrtanje.Picture1.CurrentY = CvorY(Tura(j)) - 90
    frmCrtanje.Picture1.Print Tura(j)
End If
Next j
End If
End If
End If
End If
' fmCrtanje.Picture1.Circle (CvorX(Tura(1)), CvorY(Tura(1))), 120, 225
End If
    frmCrtanje.Picture1.DrawStyle = 1
    frmCrtanje.Picture1.DrawMode = 14
Exit Sub
Greska:
    MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Module: Word
Public word As Object
Public Sub WordText(strText As String)
    Set word = CreateObject("Word.application")
End Sub

Module: Search
Public Sub Pretraga(Forma As Form, Strelica As String)
    On Error GoTo Greska
    ' Dim ptRejoni As Recordset
    Dim strIDPTTjed As String
    ' Dim strIDrejon As Field
    Dim i2 As Integer
    Dim strFind As String
    Dim strProzil As String
    Dim strKriterij As String
    i2 = 0
'strKriterij = InputBox("Unesi polje pretrage !", "Polje")
'strFind = InputBox("Unesi traženu vrednost polja; " & strKriterij, "Potraga")
'strFind = Trim(strFind)
strKriterij = "ID"
strFind = frmPTTjedinica.txtID(0).Text

If strFind <> "" Then
    strFind = strKriterij & " Like" & strFind & ""
    MsgBox strFind
    If Strelica = "RejonFind" Then
        Forma.pkpl.Recordset.FindFirst strFind
        If Forma.pkpl.Recordset.NoMatch = True Then
            frmPTTrezioni.pkpl.Recordset.AddNew
            frmPTTrezioni.txtID(0).Text = frmPTTjedinica.txtID(0).Text
        End If
    End If
    If Strelica = "P" Then
        Forma.pkpl.Recordset.FindFirst strFind
        If Forma.pkpl.Recordset.NoMatch = True Then
            MsgBox "THERE IS NO REQUESTED DATA", vbInformation, "SEARCH RESULTS"
            Forma.pkpl.Recordset.AddNew
            Forma.txtID(0).Text = frmPTTjedinica.txtID(0).Text
        End If
    End If
    If Strelica = "Levo" Then
        Proslil = Forma.txtPoc(i).Text
        Proslil = Forma.txtKraj(j).Text
        If Val(frmPTTrezioni.txtRast(3).Text) = 0 Then
            frmPTTrezioni.txtRast(3).Text = "0"
        End If
        If Val(frmPTTrezioni.txtUlica(4).Text) = "" Then
            frmPTTrezioni.txtUlica(4).Text = "Ulica"
        End If
        If Val(frmPTTrezioni.txtVreme(10).Text) = 0 Then
            frmPTTrezioni.txtVreme(10).Text = "0"
        End If
        Footer.pkpl.Recordset.FindNext strFind
        If Proslil = Forma.txtPoc(i).Text And Proslil = Forma.txtKraj(j) Then
            MsgBox "Nema vise slogova!", vbCritical, "Warning!"
            If UKUP = 0 Then frmPTTjedinica.Show
        End If
    End If
    If Strelica = "Desno" Then
        Proslil = Forma.txtPoc(i).Text
        Proslil = Forma.txtKraj(j).Text
        If Val(frmPTTrezioni.txtRast(3).Text) = 0 Then
            frmPTTrezioni.txtRast(3).Text = "0"
        End If
        If Val(frmPTTrezioni.txtRast(3).Text) = 0 Then
            frmPTTrezioni.txtVreme(10).Text = "0"
        End If
        If Val(frmPTTrezioni.txtUlica(4).Text) = "" Then
            frmPTTrezioni.txtUlica(4).Text = "Ulica"
        End If
        Footer.pkpl.Recordset.FindNext strFind
        If Proslil = Forma.txtPoc(i).Text And Proslil = Forma.txtKraj(j) Then
            MsgBox "Nema vise slogova!", vbCritical, "Warning!"
            If UKUP = 0 Then frmPTTjedinica.Show
        End If
    End If
    If Strelica = "DefMreze" Then
        Footer.pkpl.Recordset.FindFirst strFind
        If Forma.pkpl.Recordset.NoMatch = True Then
            MsgBox "THERE IS NO REQUESTED DATA", vbInformation, "SEARCH RESULTS"
        Else
            Erase Poc, Kraj; i2 = 0
        End If
        While Forma.pkpl.Recordset.NoMatch <> True
            i2 = i2 + 1
            Poc(i2) = Forma.txtPoc(i).Text
            Kraj(i2) = Forma.txtKraj(j).Text
            If Minimiziranje = "Vreme" Then
                D3(Poc(i2), Kraj(i2)) = Val(Forma.txtVreme(10).Text)
                D3(Kraj(i2), Poc(i2)) = Val(Forma.txtVreme(10).Text)
            End If
        End While
    End If
End If
Else
    D1(Poc(i2), Kraj(i2)) = Val(Forma.txtRast(3).Text)
    D1(Kraj(i2), Poc(i2)) = Val(Forma.txtRast(3).Text)
End If
D2(Poc(i2), Kraj(i2)) = Val(Forma.txtDupla(9).Text)
D2(Kraj(i2), Poc(i2)) = Val(Forma.txtDupla(9).Text)

UIica(Poc(i2), Kraj(2)) = FormaUlica(4).Text
UIlica(Kraj(i2), Poc(2)) = FormaUlica(4).Text

forma ph^1-RecordsetFindNext
strFind Wend
brGranal = i2
End If
End If
End If
End If
Exit
Sub
Greska-MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub

Module: Search1

Public Sub PretragaMreze(Forma As Form, strKriterij1 As String, strKriterij2 As String, strKriterij3 As String, strFind1 As String, strFind2 As String, strFind3 As String, pronadjeno As Boolean)
    On Error GoTo Greska
    Dim Prosli As String
    Dim Nadji As String
    Dim varBookmark As Variant
    Ima = False
    Nadji = strKriterij1 & " Like " & strFind1 & "" & " And " & strKriterij2 & " Like " & strFind2 & "" & " And " & strKriterij3 & " Like " & strFind3 & ""
    If Nadji <> "" Then
        'strFind = strKriterij & " Like" & strFind & ""
        Forma.pkp1.Recordset.FindFirst Nadji
        duplikat = 0
        If Forma.pkp1.Recordset.NoMatch = False Then
            Ima = True
            pronadjeno = False
            Forma.pkp1.Recordset.FindNext Nadji
            10
            If Forma.pkp1.Recordset.NoMatch = False Then
                pronadjeno = True
                If brcvor = 0 Then MsgBox "There is duplicat. Please, change name the region!", vbCritical, "Warning!"
                duplikat = duplikat + 1
                ' Forma.PKP1.Recordset.Delete
                ' Forma.PKP1.Recordset.FindFirst Nadji
                GoTo 10
            Else
                MsgBox "THERE IS NO DUPLICATES", vbInformation + vbOKOnly, "Obavestenje"
            End If
        End If
    End If
End Sub

Module: Definition

Public Poc(100), t, brcvor, brGrana As Integer
Public Kraj(100), Tura(200), Zoom As Integer
Public D1(100, 100) As Integer
Public D2(100, 100), D3(100, 100) As Integer
Public Cvor(100), CvorX(100), CvorY(100), cvorX1(100), cvorY1(100) As Integer
Public Ulica(100, 100), Mapa As String
Public design, dodato, Ima As Boolean
Public duplikat, brCrana1, debiñana As Integer
Public word As Object
Public design, dodato, Ima As Boolean
Public duplikat, brCrana1, debiñana As Integer

Module: Zoom

Dim msHeightRatio As Single, msWidthRatio As Single
Dim msIdealHeight As Single, msIdealWidth As Single
Dim msActualHeight As Single, msActualWidth As Single
Dim mResized As Boolean

Module: Zooming

Public Sub Zorning(zoom1 As Integer)
Dim Xcell, Ycell As Variant
'On Error GoTo Greska
frmCrtanje.HScroll1.Value = 1
frmCrtanje.VScroll1.Value = 1
Stoposto = frmCrtanje.HScroll2.Max / 2

Dim SaveMode As Integer
SaveMode = frmCrtanje.Picture1.ScaleMode
frmCrtanje.Picture1.ScaleMode = 3

If frmCrtanje.HScroll2.Value = frmCrtanje.HScroll2.Max Then pro = 1
If frmCrtanje.HScroll2.Value > 280 And pro <> 3000 Then
pro = 3000
Else
pro = 1
End If

If frmCrtanje.HScroll2.Value > frmCrtanje.HScroll2.Max - 15 Then

frmCrtanje.PictureClip1.ClipHeight = 377
frmCrtanje.PictureClip1.ClipWidth = 489
frmCrtanje.HScroll1.Value = 1
frmCrtanje.VScroll1.Value = 1
pro = 100
GoTo 20
Else

frmCrtanje.PictureClip1.StretchX = frmCrtanje.Picture1.ScaleWidth * 1
frmCrtanje.PictureClip1.StretchY = frmCrtanje.Picture1.ScaleHeight * 1
frmCrtanje.PictureClip1.ClipHeight = zoom1 / 1.297
frmCrtanje.PictureClip1.ClipWidth = zoom1
frmCrtanje.HScroll1.Value = 1
frmCrtanje.VScroll1.Value = 1

End If

10 frmCrtanje.Picture1.Picture = frmCrtanje.PictureClip1.Picture

20 frmCrtanje.Picture1.ScaleMode = SaveMode
pro = pro + 1
frmCrtanje.Picture1.ScaleMode = 1
Exit Sub
Greska:
MsgBox "Restart the program!", vbCritical, "Warning!"
End Sub