铁路最优路线问题

摘要

本文解决的是铁路路线选取的最优化问题，旨在研究当前铁路网络条件下的最优出行路线。需要解决的问题时任意给定两地，分析得出两地间的最优路线，制定出行方案，首先对题目所给数据进行统计学分析，然后为解决不同的问题建立了两个个不同的模型。

针对问题一：建立了多目标最优化模型。首先利用 *Excel* 软件和 *MATLAB* 语言整理得出任意两地点车站间距离，然后删选不重复的车站并对其进行编号，结合 *floyed* 算法并利用 *C++* 语言编写出一个求两地最短路径的算法。根据所得出的最短路径，建立以路线所需时间，换乘次数，票价为目标的多目标最优化模型，得出最终路线为丹东—北京—宜昌东，天津—北京西—拉萨，白城—沈阳—青岛，历程数分别为 2969 公里，3884 公里，1908 公里（具体方案见问题一解答）。

针对问题二：建立了 *Hopfeild* 神经网络模型。将问题映射到 *Hopfeild* 网络上，将 6 个城市视为神经元，确定三个约束条件为：

约束 1：在同一时间,不能访问多于一个城市；

约束 2：对每个城市只访问一次；

约束 3：使总的旅行距离最小。

建立能量函数，其最小值对应最短路径。确定旅行顺序后，利用问题一中的系统确定旅行方案。最终方案为：宜昌—南京—无锡—苏州—上海—杭州—宜昌。所乘车次分别为 D3078-D3135-G7209-G7215-G7363-K529.共需费用 627 元（详见问题二求解）。

关键词： 多目标最优化， *Hopfeild* 神经网络， *floyed* 算法

1. 问题重述

1.1 问题背景

铁路既是社会经济发展的重要载体之一，同时又为社会经济发展创造了前提条件。近几年来，在全社会客运量稳步上升的同时，长期以来铁路承运了大量旅客。相对于其他的运输方式铁路具有时间准确性高、运输能力大、运行比较平稳、安全性高等优点。同时火车也成为了旅途的首选交通运输工具。

虽然目前铁路网络已经比较发达，但是仍然有很多地方之间并没有直接到达的铁路。并且在节假日期间，一些热门路线的火车票总是一票难求。在这种情况下，需要考虑两个站点之间的最优铁路路线问题，即先从起点站到换乘站，再从换乘站到目的站。

1.2 需要解决的问题

问题一：给出任意两个站点之间的最优铁路路线问题的一般数学模型和算法。若两个站点之间有直达列车，需要考虑直达列车票已售罄情况下最优的换乘方案。根据附录数据，利用你们的模型和算法求出一下起点到终点的最优路线：丹东→宜昌、天津→拉萨、白城→青岛。

问题二：假设你从打算从宜昌出发乘火车到上海、南京、杭州、苏州、无锡旅游最后回到宜昌，请确定相关数学模型，给出整个行程的最优路线。

2. 模型假设

假设一：按照一般习惯，所选路线最多转车两次；

假设二：列车行驶过程中不会出现因临时停车、错车而延误时间；

3. 符号说明

符号	释义
$t_{ij}(l_i)$	乘l_i次列车从i站到j站所用的乘车时间
$t_{ij}(l_1,l_2)$	l_2次列车出发时刻与l_1次列车到达时刻之间的时间差
$p_{ij}(l_i)$	乘l_j次列车从i站到j站所用的票价
$N_{\max}$	最大换乘次数
$x_{ij}^{l_i}$	表示从i站到j站是否乘坐了l_i次列车
x,y	分别为城市编号
u_{xi}	神经元xi的输入值
u_0	神经元函数的斜率

4. 问题分析

随着交通事业的发展，铁路局所开通的铁路线路也逐渐增多，针对乘客需求、需要建立一个合适的模型，解决两个站点之间的最优铁路路线问题。对于最优路线的选取本文从时间和票价两个角度分析，而走完选定路线所用的时间包含经过所有站点的时间和转车所用的时间，全程所需的总车费即为票价总和，二者的权重可由乘客自行确定，对于不同需求选择不同的路线。

针对问题一，本问题需要解决的核心问题即是最短路径的选择，将已知的铁路线路信息转化为相应的线路矩阵处理，利用*MATLAB*语言与$C++$语言建立了两个站点之间的最优铁路路线选取系统，信息导入，结合 *floyed* 算法并根据乘客的不同要求寻找出一条最优路径。

针对问题二：本文研究的旅游路径是一个封闭回路的数学模型。这一问题涉及到平面上的点的遍历问题，即要寻找一条行走路线最短（尽可能花费最少）但又可以行遍图上所有地点的路径。

运用 *Hopfeild* 神经网络算法来解决旅行路线最佳问题。首先我们通过查询资料了解到 6 座城市的纬度坐标，将问题映射到 *Hopfeild* 网络上，将 6 个城市视为神经元，任何一个城市在最终路径上的访问次序可用一个 *N* 维向量来表示，因

此每个城市需要6个神经元表示，我们不妨规定第i个城市为$Ai(x、y)$，$x、y$表示城市的经纬度，以宜昌东为起点，神经元（城市）视为1，其余神经元视为0，为了表示所有城市，可以用一个5×5的矩阵表示，必须保证每行每列只有一个1，这里选取的是能量函数，其最小值对应最短路径。确定旅行顺序后，利用问题一中的系统确定旅行方案。

5. 问题一的求解

5.1 多目标最优化模型的准备

要求出任意两个站点之间的最优铁路路线，从路线所需时间（包括乘车时间和换程的等候时间）、乘车票价以及换乘次数这三方面的因素考虑，建立多目标铁路旅客乘车方案优化模型，并根据路网结构特点结合 *floyed* 算法利用C进行求解。

5.2 多目标最优化模型的建立[1]

首先，分别解释每一个因素对最优铁路路线的影响。

1．**路线所需时间**包括乘车时间和换程的等候时间。

乘车时间除了与所选线路、始发终到站里程有关外，还与所选列车种类（快、慢车）有关。用$t_{ij}(l_i)$表示乘l_i次列车从i站到j站所用的乘车时间

$$t_{ij}(l_i)=t_{l_i}^2(j)-t_{l_i}^2(i)$$

换乘间隔时间为从l次列车换乘l_2次列车的时间，即l_2次列车出发时刻与l_1次列车到达时刻之间的时间差，记为

$$t_{ij}(l_1,l_2)$$

2．**票价费用**因列车等级、席作的不同而各异，列车的等级可分为：动车组、直列车、特快列车、快速列车及普快列车等。我国铁路网中，任何铁路线路上的两站点间的铁路运价均可以在铁路列车时刻表上查询得到，在研究过程中，票价选用各种类的最低票价（如普通列车选用硬座票价）进行计算。那么乘l_j次列车从i站到j站所用的票价为$p_{ij}(l_i)$

3．**换乘次数**是指乘客在完成一次出行过程中所换列车的次数。由于换乘次数是乘客能直接感知的，因此换乘次数是直接影响乘客路径选择的一个重要因素。一般情况下，乘车时间差不多时换乘次数越少的路径被选择的概率越大。铁路中转换乘的旅客在途时间长、疲劳程度高，中转换乘耗时耗力，因此换乘方案中换乘次数一般是一个不大的值，设最大换乘次数为$N_{\max}$，为了方便计算旅客

乘车方案，最大换乘次数值可取 2。

换乘模型的构建

设 $x_{ij}^{l_i}$ 表示从 i 站到 j 站是否乘坐了 l_i 次列车，

$$x_{ij}^{l_i}=\begin{cases}1, \text{乘} l_i \text{次列车从} i \text{ 站到} j \text{ 站}\\ 0, \text{否则不乘}\end{cases}$$

o、d 分别为旅客旅行的始发站、终到站。则换乘次数受限时 o 至 d 的多目标乘客乘车方案模型为：

$$\begin{cases}\min z_1=\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\sum\limits_{l_i,i=1}^{N}\left(t_{l_i}^{1}(j)-t_{l_i}^{2}(i)\right)x_{ij}^{l_i} & (1)\\ \min z_2=\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\sum\limits_{l_1,l_2}t_{ij}(l_1,l_2) & (2)\\ \min z_3=\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\sum\limits_{l_i,i=1}^{N}p_{ij}(l_i)x_{ij}^{l_i} & (3)\\ \min z_4=\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\sum\limits_{l_i,i=1}^{N}x_{ij}^{l_i}-1 & (4)\end{cases}$$

$$s.t.\begin{cases}\sum\limits_{j=1}^{N}\sum\limits_{l_i,i=1}^{N}x_{oj}^{l_i}=1 & (5)\\ \sum\limits_{j=1}^{N}\sum\limits_{l_i,i=1}^{N}x_{im}^{l_i}-\sum\limits_{j=1}^{N}\sum\limits_{l_i,i=1}^{N}x_{mj}^{l_i}=0; m\neq o,d; \forall m\in M & (6)\\ \sum\limits_{i=1}^{N}\sum\limits_{l_i,i=1}^{N}x_{id}^{l_i}=1 & (7)\\ \sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\sum\limits_{l_i,i=1}^{N}x_{ij}^{l_i}-1\le N_{\max} & (8)\\ x_{ij}^{l_i}\in\{0,1\}; \forall i,j=1,2,\cdots,N & (9)\end{cases}$$

模型目标函数式 $(1)\sim(4)$ 分别为总乘车时间最小、换乘间隔时间最小、总票价费用最少、换乘次数最少。式 $(5)\sim(7)$ 保证从始发站 o 到终到站 d 为一条路径，

式(8)表示换乘次数满足最大换乘次数$N_{\max}$的限制，式(9)是0−1变量约束。

5.3 多目标最优模型的求解

首先利用$C++$语言得出从丹东至宜昌东，天津至拉萨，白城至青岛的最短路径。

表 5.3-1:最短路径中转表

起点站	中转站	终点站
丹东	北京	宜昌东
天津	北京西	拉萨
白城	沈阳	青岛

结合多目标最优化模型，利用$MATLAB$语言求的最优方案如下：

表 5.3-2:路线选择最优方案

车次	发战/到站	发站时间/到站时间	总时间/停留时间	总里程	总票价
K28	丹东	18：31	总 1 天 6 小时 18 分	2629 公里	319
	北京	08：31			
T57		11：48	停 3 小时 17 分		
	宜昌东	00：49			
T5684	天津	13：46	总 2 天 0 小时 56 分	3884 公里	390.5
	北京西	15：38			
T27		20：00	停 4 小时 22 分		
	拉萨	14：42			
K1302	白城	02：32	总 17 小时 58 分	1908 公里	577
	沈阳	09：28			
G1266		11：59	停 2 小时 31 分		
	青岛	20：30			

综上所述即为最优路线的选择。

6. 问题二的求解

6.1 *Hopfeild* 神经网络模型的建立

本文考虑的旅游路线问题是从一个城市出发，对每个要游览的城市都游览一遍，且每个城市只经过一次。基于这种思想，构建模型如下：

用网络框架表示旅游路线问题，每个神经元对应于一个按次序访问的城市。由于各神经元的输出在0到1之间，两个极端取值就对应着是访问还是不访问。例如本问中的5个城市，如果某个城市被第3个访问，城市可以表示成向量$(0,0,1,0,0$。对5个城市的地图，可以创建一个包括$5\times5=25$个神经元的$Hopfield$网络。这意味着，一条路线可以表示成一个$n\times n$的神经元输出V_{xi}的状态矩阵V_s（列表示路线中的不同位置，行表示城市。约定元素V_{xi}的第1个下标表示行，第2个下标表示列）。

$$V_s\equiv\begin{bmatrix}V_{11}&V_{12}&V_{13}&V_{14}&V_{15}\\V_{21}&V_{22}&V_{23}&V_{24}&V_{25}\\V_{31}&V_{32}&V_{33}&V_{34}&V_{35}\\V_{41}&V_{42}&V_{43}&V_{44}&V_{45}\\V_{51}&V_{52}&V_{53}&V_{54}&V_{55}\end{bmatrix}$$

假设5个城市$P=(p_1,p_2,p_3,p_4$，应该以下面的顺序访问：$p_3\Rightarrow p_1\Rightarrow p_4\Rightarrow p_2\Rightarrow p_5$，则状态矩阵为$V_p$。

$$V_p\equiv\begin{bmatrix}0&1&0&0&0\\0&0&0&1&0\\1&0&0&0&0\\0&0&1&0&0\\0&0&0&0&1\end{bmatrix}$$

在创建能量函数时,必须清晰地表达给定问题的所有约束和目标。

约束1：在同一时间,不能访问多于一个城市；

约束2：对每个城市只访问一次；

约束3：使总的旅行距离最小。

能量函数为：

$$E=\frac{A}{2}\sum_{i=1}^{N}\left(\sum_{x=1}^{N}V_{xi}-1\right)^2+\frac{B}{2}\sum_{i=1}^{N}\left(\sum_{x=1}^{N}V_{xi}-1\right)^2+D\sum_{x=1}^{N}\sum_{y=1}^{N}\sum_{i=1}^{N}d_{xy}V_{xi}V_{y,i+1}\quad(1)$$

式中：x,y分别为城市编号；i为某城市被访问的次序；d_{xy}为两个城市间的距离；N为城市的数目；A,B分别为影响状态矩阵对同一行、同一列各元素之间的权重；D为路径顺序上的相邻元素之间的权重。对上述能量函数求偏导数，得到相应的状态方程如下：

$$\frac{du_{xi}}{dt}=-A\left(\sum_{i=1}^{N}V_{xi}-1\right)-B\left(\sum_{i=1}^{N}V_{xi}-1\right)-D\sum_{y=1}^{N}d_{xy}V_{y,i+1}\quad(2)$$

$$V_{xi}=\frac{1}{2}\left(1+\tanh\left(\frac{u_{xi}}{u_0}\right)\right)\quad x,i=1,2\cdots,N \tag{3}$$

式中：u_{xi}为神经元xi的输入值；u_0为神经元函数的斜率。

为了模拟计算，首先将状态方程(2)和(3)离散化为差分方程：

$$u_{xi}(n+1)=u_{xi}(n)+h\left(-A\left(\sum_{i=1}^{N}V_{xi}-1\right)\right)-B\left(\sum_{i=1}^{N}V_{xi}-1\right)-D\sum_{y=1}^{N}d_{xy}V_{y,i+1} \tag{4}$$

$$V_{xi}=\frac{1}{2}\left(1+\tanh\left(\frac{u_{xi}}{u_0}\right)\right)\quad x,i=1,2\cdots,N \tag{5}$$

式中：$A=B=1.5$（为了保证对称性），$D=1.5$，h为步长，$h=0.5$，$u_0=0.02$。

初始化U：U在0附近取值。

虽然(2)式是单调递增函数，已证明随着时间的推移，有

$$\frac{dE}{dt}\le 0,\ \frac{dE}{dt}=0$$

的关系。但是，考虑到上述能量曲线和状态矩阵的变化，不能用后者判别是否到达极小值，而是检查是否得到合法回路。其次，记录合法回路的长度，要求第$r+1$次路径的长度必须小于或等于第r次路径的长度，因为局部极小很多，除非得到全局最优解，连续两次合法路径的长度相等的概率几乎等于零，所以利用以上要求不难获得全局最优解。第r次计算路径长度的主要步骤为：

步骤1：若$r=1$，设初始化矩阵为$V(N,N)$，其中N为城市个数。令迭代次数$count=0$，转入步骤2。若$r>1$,对$V\left(N,N\right)$任一行进行一个微小的扰动，令迭代次数$count=0$，转入步骤2；

步骤2：对$x,i=1,2,..,N$的式(4)、(5)进行1次迭代运算,记录迭代次数$count=count+1$；

步骤3：检查超时。如果$count$达到迭代次数的上限时，终止第r次循环,计算路径长度，令$r=r+1$转入步骤0，否则，转入步骤4；

步骤4：判别合法路径

当 V 中的每个元素是0（即$V<0.01$）或1（即$V>0.99$）时，继续执行；否则，转入步骤1；

当每行每列恰有一个1时，继续执行步骤5。否则，转入到步骤2；

步骤5：计算路径长度，从求得的状态矩阵，计算对应的回路长度；

步骤6：求最短的回路长度。若$r=1$，记录求得的回路长度L。若$r>1$，第r次求得的回路长度L不大于第$(r-1)$次求得的回路长度时，记录求得的回路长度L，结束第r次运算，令$r=r+1$转入步骤0。否则，转入步骤2。

6.2 *Hopfeild* 神经网络模型的求解

通过求解模型的出环绕旅行方案为：最终方案为：宜昌—南京—无锡—苏州—上海—杭州—宜昌。所乘车次分别为 D3078-D3135-G7209-G7215-G7363-K529. 共需费用 627 元。

7. 模型的评价

7.1 模型的优点

1） 针对多种因素对最有路线的影响，本文建立了多目标最优化路线模型，将具有约束条件的但目标函数转化为多目标优化模型，是模型更加合理，结果更加精确。
2） 针对最佳游路线问题，建立了神经网络模型，通过网络状态的动态演化逐步趋向稳定而自动搜索出最优解，并运用$0-1$变量直观的体现出了结果。
3） 本文在建立模型的过程中将模型与算法结合，可以较为精确的计算出结果。

7.1 模型的缺点

本文在建立多目标最有路线模型是，没有考虑到同一城市可能存在不止一个火车站的情况，可能导致结果存在误差。

8. 模型的改进与推广

多目标优化问题在工程应用等现实生活中非常普遍并且处于非常重要的地位，这些实际问题通常非常复杂、困难，是主要研究领域之一。*Hopfeild* 神经网络模型广泛应用于优化问题和联想记忆中，具有非常积极的意义。

9. 参考文献

[1]【作者】 杨信丰；刘兰芬；李引珍；何瑞春；
多目标铁路旅客乘车方案优化模型及算法研究[J]；
交通运输系统工程与信息；2013 年 10 月第 13 卷 5 期。

附录

C++最优路径选择系统源程序：

```
#include <fstream>
#include <iostream>
#include <string>
#include <cstdio>
#include <iomanip>
#include <cstdlib>
#define SIZE_view 50                    //2867
#define SIZE_line 100                   //4683
#define SIZE_way 300                    //49369
#define MAXNODE 30      //定义最多节点数
#define MAXCOST 10000
using namespace std;
struct view_info      /*城市信息结构*/
{
    int id;
    char name[20];
    int code;
    //char shortname[20];
    //char LName[100];//经过此站的铁路线名称
}views[SIZE_view];

struct line_info//铁路线信息结构
{
    int Lid;//代表车次
    //char LName[20];
    int start_id;    //始发站
    int end_id;      //终点站
    int dist;        //铁路线长
    //char sign[5];          //通行标志
}lines[SIZE_line];

struct way_info      //铁路度的信息结构
{
    int station1;//起点站
    int station2;//终点站
    int dist;//里程
}ways[SIZE_way];
```

```
struct path_info    //用于最短路径的查询
{
    int count;
    int path[SIZE_view];
};

int view_count, line_count, way_count;     //用来存储文件中有多少条记录

//typedef int INTARRAY[SIZE_view];
//INTARRAY *dist_list = new int[SIZE_view][SIZE_view];



int dist_list[SIZE_view][SIZE_view];     //定义一个数组

struct path_info path_list[SIZE_view][SIZE_view];   //定义一个 path_info 结构体变量，



void readviews();
void readways();
void readlines();
void search();
void addview();
void addway();
void addline();
void shortest_path();
void floyed();
void adddata(int menu);

void main()
{
    readviews();
        cout<<endl<<endl;
    readlines();
        cout<<endl<<endl;
    readways();
    while (1)
    {
        int menu;
        cout << endl << endl;
        cout << "              全国铁路运输网经由系统" << endl;
        cout << "**************************************" << endl;
```

```
cout << "            1、增加车站信息" << endl;
cout << "            2、增加铁路线信息" << endl;
cout << "            3、查询车站信息" << endl;
cout << "            4、查询最短路径" << endl;
cout << "            5、退出界面" << endl;
cout << "**************************************" << endl;
cout << "请选择你要的操作代码（1-5):" << endl;
cin >> menu;
while (menu<1 || menu>5)
{
    cout << "error!please enter again:";
    cin >> menu;
}
switch (menu)
{
case 1:
case 2:
    adddata(menu);
    break;
case 3:
    while (1)
    {
        search();
        cout << "do you want to continue?(y/n)" << endl;
        char con;
        cin >> con;
        if (con == 'y');
        else
            break;
    }
    break;
case 4:
    while (1)
    {
        shortest_path();
        cout << "do you want to continue?(y/n)" << endl;
        char con;
        cin >> con;
        if (con == 'y');
        /*              addline();*/
        else
            break;
    }
    break;
```

```
            case 5:
                cout << "谢谢使用，再会！" << endl;
                exit(1);
            }
        }
}
void readviews()
{
        int i;
        ifstream infile("views.txt", ios::in);    //打开文件
        //吧文件中的个数付给 view_count
        /*    infile>>view_count; */
        if (!infile)              //打开文件失败
        {
            cerr << "can't open views.txt!" << endl;
            exit(1);
        }
        //infile>>view_count;//先读入文件个数
        for (i = 0;; i++)
        {
            infile >> views[i].id >> views[i].name >> views[i].code;//>> views[i].shortname >>
views[i].LName;
            if (i != 0 && views[i].id == 0)break;
        }
        view_count = i;
        //view_count=i;//给出源文件中车站的个数
        infile.close();
        //下面是测试用的代码
        cout << setiosflags(ios::left);
//      cout << setw(8) << "id" << setw(9) << "name" << setw(8) << "code" << setw(12) <<
"shortname" << setw(10) << "LName" << endl;
        for (i = 0; i<view_count; i++)
        {
//          cout << setw(8) << views[i].id << setw(9) << views[i].name << setw(8) <<
views[i].code << setw(12);
//          cout << views[i].shortname << views[i].LName << endl;
        }
        cout << resetiosflags(ios::left);
}
void readways()//读文件 ways.txt
{
        int i;
        ifstream infile("ways.txt", ios::in);//打开文件
        /*    infile>>way_count;*/        //把文件中的记录付给 way_count
```

```
    if (!infile)              //打开文件失败
    {
        cerr << "can't open ways.txt!" << endl;
        exit(1);
    }
    for (i = 0;; i++)
    {
        infile >> ways[i].station1 >> ways[i].station2 >> ways[i].dist;
        if (i != 0 && ways[i].station1 == 0)break;
    }
    way_count = i;
    infile.close();
    //测试用，输入路段的信息
    //          cout<<setiosflags(ios::left);
    //          cout<<setw(12)<<"station1"<<setw(12)<<"station2"<<"dist"<<endl;
    //          for(i=0;i<way_count;i++)
    //          {
    //              cout<<setw(12)<<ways[i].station1<<setw(12)<<ways[i].station2;
    //              cout<<ways[i].dist<<endl;
    //          }
    //          cout<<resetiosflags(ios::left);
}
void readlines()//读文件 lines.txt
{
    int i;
    ifstream infile("lines.txt", ios::in);//打开文件
    //把文件中的记录付给 line_count
    if (!infile)              //打开文件失败
    {
        cerr << "can't open lines.txt!" << endl;
        exit(1);
    }
    /*    infile>>line_count; */
    for (i = 0;; i++)
    {
        infile >> lines[i].Lid /*>> lines[i].LName */>> lines[i].start_id;
        infile >> lines[i].end_id >> lines[i].dist; //>> lines[i].sign;
        if (i != 0 && lines[i].Lid == 0)break;
    }
    line_count = i;
    infile.close();
    //下面的代码为测试时用的
    cout << setiosflags(ios::left);
//  cout << setw(8) << "Lid" << setw(12) << "LName" << setw(10) << "start_id";
```

```
//      cout << setw(10) << "end_id" << setw(8) << "dist" << "sign" << endl;
        for (i = 0; i<line_count; i++)
        {
//              cout << setw(8) << lines[i].Lid /*<< setw(12) << lines[i].LName */<< setw(10) << lines[i].start_id;
//              cout << setw(10) << lines[i].end_id << setw(8) << lines[i].dist; //<< lines[i].sign << endl;
        }
        cout << resetiosflags(ios::left);
        //这里输出文本中的信息

}
void search()       //查询车站信息（所在的铁路线）
{
        cout << "Please enter the station name:";
        char sta_name[20];
        cin >> sta_name;            //输入要查询的名字
        cout << endl;
        //      ifstream infile("views.txt",ios::in);      //读文件
        // /*  infile>>view_count;*/       //读出文件记录的个数
        //      if(!infile)             //打开文件失败
        //      {
        //              cerr<<"can't open views.txt!"<<endl;
        //              exit(1);
        //      }
        int i, mark;
        //      for(i=0;i<view_count;i++)
        //
        infile>>views[i].id>>views[i].name>>views[i].code>>views[i].shortname>>views[i].LName
;
        //      infile.close();
        for (i = 0; i<view_count; i++)
        {
                if (strcmp(sta_name, views[i].name) == 0)
                {
                        cout << "the station informations is:\n" << endl;
                        cout << setiosflags(ios::left);
                        cout << setw(8) << "id" << setw(9) << "name" << setw(8) << "code" << setw(12)
<< "shortname" << setw(10) << "LName" << endl;
                        cout << setw(8) << views[i].id << setw(9) << views[i].name << setw(8) << views[i].code << setw(12);
//                      cout << views[i].shortname << views[i].LName << endl;
                        cout << resetiosflags(ios::left);
                        break;
```

```
            }
            mark = i;
        }
        if (mark == view_count - 1)     //若没找到，输出提示
            cout << "sorry, the station is not in here!" << endl;
}
void addview()
{
        cout << "Please enter the new view's informations:" << endl;    //输入新的车站信息
        cout << "id(id>" << views[view_count - 1].id << "):";
        cin >> views[view_count].id;
        while (1)
        {
            if (views[view_count].id<views[view_count - 1].id)
            {
                cout << "你输入的数据不合法！请重新输入:";
                cin >> views[view_count].id;
            }
            else
                break;
        }
        cout << "name:";
        cin >> views[view_count].name;
        cout << "code(code>" << views[view_count - 1].code << "):";
        cin >> views[view_count].code;
        while (1)
        {
            if (views[view_count].code<views[view_count - 1].code)
            {
                cout << "你输入的数据不合法！请重新输入:";
                cin >> views[view_count].id;
            }
            else
                break;
        }
//      cout << "shortname:";
//      cin >> views[view_count].shortname;
//      cout << "LName:";
//      cin >> views[view_count].LName;
        ofstream outfile("views.txt", ios::app);     //打开 views 文件，并写入数据
        /*    outfile<<view_count<<endl;*/
        if (!outfile)
        {
            cerr << "can't open views.txt!";
```

```
            exit(1);
        }
        outfile << endl << views[view_count].id << " " << views[view_count].name << " ";
        outfile << views[view_count].code << " "; //<< views[view_count].shortname;
//      outfile << " " << views[view_count].LName << endl;
        //在文件末尾添加
        view_count++;
        outfile.close();              //关闭文件
        cout << "Successfully!the new station is added" << endl;
        cout << "new station number is :" << view_count << endl;
}
void addway()
{
        cout << "Please enter the new way's informations:" << endl;   //输入新的车站信息
        cout << "station1:";
        cin >> ways[way_count].station1;         //station1  的 id
        cout << "station2:";
        cin >> ways[way_count].station2;          //station2  的 id
        cout << "dist:";
        cin >> ways[way_count].dist;               //路段的长度
        ofstream outfile("ways.txt", ios::app);     //打开 ways.txt 文件，并写入数据
        /*    outfile<<way_count<<endl;*/
        if (!outfile)
        {
            cerr << "can't open ways.txt!";
            exit(1);
        }
        outfile << endl << ways[way_count].station1 << " " << ways[way_count].station2 << " ";
        outfile  <<  ways[way_count].dist  <<  "  "  <<  ways[way_count].station2  <<  "  "  <<
ways[way_count].station1;
        outfile << " " << ways[way_count].dist;
        //在文件末尾添加
        way_count++;
        outfile.close();              //关闭文件
        cout << "Successfully!the new way is added" << endl;
        cout << "new station number is :" << way_count << endl;
}
void addline()
{
        cout << "Please enter the new line's informations:" << endl;   //输入新铁路信息
        cout << "Lid(Lid>" << lines[line_count - 1].Lid << "):";
        cin >> lines[line_count].Lid;
        while (1)
        {
```

```
            if (lines[line_count].Lid<lines[line_count - 1].Lid)
            {
                cout << "你输入的数据不合法！请重新输入:";
                cin >> lines[line_count].Lid;
            }
            else
                break;
        }
//      cout << "LName:";
//      cin >> lines[line_count].LName;
        cout << "start_id:";
        cin >> lines[line_count].start_id;
        cout << "end_id:";
        cin >> lines[line_count].end_id;
        cout << "dist:";
        cin >> lines[line_count].dist;
//      cout << "sign:";
//      cin >> lines[line_count].sign;
        ofstream outfile("lines.txt", ios::app);     //打开文件，并写入数据
        /*    outfile<<line_count+1<<endl;*/
        if (!outfile)
        {
            cerr << "can't open lines.txt!";
            exit(1);
        }
        outfile << endl << lines[line_count].Lid << " " /*<< lines[line_count].LName << " "*/ <<
lines[line_count].start_id;
        outfile << " " << lines[line_count].end_id << " " << lines[line_count].dist << " ";
//      outfile << lines[line_count].sign;
        //在文件末尾添加
        line_count++;
        outfile.close();              //关闭文件
        cout << "Successfully!the new line is added" << endl;
        cout << "new station number is :" << line_count << endl;
}
void floyed()     //弗洛伊德算法
{
        int i, j, k, m;
        //包含着 count 和 path[]用来存储经过的路径
        //      for(i=0;i<=view_count;i++)
        //              for(j=0;j<=view_count;j++)
        //                      dist_list[i][j]=MAXCOST;//先对任意两点的距离初始值为无穷
        for (int t = 0; t <= way_count; t++)
        {
```

```
        i = ways[t].station1;
        j = ways[t].station2;
        dist_list[i][j] = ways[t].dist;//把文件中的数据付给 dist_list[i][j]=ways[t].dist;形式
    }
    for (i = 0; i <= view_count; i++)
    {
        for (j = 0; j <= view_count; j++)
        {
            if (i == j)     //车站到本车站的距离赋值为零
            {
                dist_list[i][j] = 0;
                continue;
            }
            dist_list[i][j] = -1;     //先设置任意两点之间的距离为-1；表示这两点不通
            path_list[i][j].count = 0;   //先设置任意两点之间的的路径的车站数为零
            for (k = 0; k<way_count; k++)       //ways 文件的数据赋给 dist_list 数组、并记下
其中任两站的路径
            {
                if (ways[k].station1 == i&&ways[k].station2 == j)
                {
                    dist_list[i][j] = ways[k].dist;
                    path_list[i][j].count = 2;
                    path_list[i][j].path[0] = i;
                    path_list[i][j].path[1] = j;
                    break;
                }
            }
        }
    }
    //下面是计算最短路径的代码
    for (k = 0; k <= view_count; k++)
    {
        for (i = 0; i <= view_count; i++)
        for (j = 0; j <= view_count; j++)
        {
            if (i == k || j == k || i == j)   //三个站中至两个站是相同的的话就是继续循环
                continue;
            if (dist_list[i][k] == -1 || dist_list[k][j] == -1)     //i、k 不通或者 k、j 不通继续循
环
                continue;
            if ((dist_list[i][j] == -1) || ((dist_list[i][j] != -1) && (dist_list[i][k] +
dist_list[k][j]<dist_list[i][j])))
            { //i、j 不通，或者是 i、j 通但是不是最短路径，执行下面语句
                dist_list[i][j] = dist_list[i][k] + dist_list[k][j];     //求出 i、j 的最短距离
```

```
                    //shortest[i][j]=shortest[i][k]+shortest[k][j];
                    path_list[i][j].count = path_list[i][k].count + path_list[k][j].count - 1;    //求出 i、j 路径的站的个数
                    //path_list[i][j]=k;
                    for (m = 0; m<path_list[i][k].count; m++)      //下面两个 for 语句标出 i、j 路径的每个站的 id 号，以便后面的输出最短经由路径用
                        path_list[i][j].path[m] = path_list[i][k].path[m];
                    for (m = 0; m<path_list[k][j].count; m++)
                        path_list[i][j].path[m + path_list[i][k].count] = path_list[k][j].path[m + 1];
                }
            }
        }
    }
    void shortest_path()
    {
        floyed();
        int i, k, m;
        int start_num = -1, end_num = -1;
        string start_station, end_station;//定义起始站、终点站
        //下面便是输出最短经由路径的代码
        cout << "Floyed table:\n";
        cout << "All cities in the table:\n";
        for (i = 0; i<view_count; i++)        //输出可以查看的站的名称和 id 号
        {
            cout << setiosflags(ios::left);
            cout << setw(2) << i + 1 << ":" << setw(10) << views[i].name;
            if (((1 + i) % 5) == 0 && i != 0)cout << endl;
            cout << resetiosflags(ios::left);
        }
        cout << endl;
        cout << "Please input the start station name:";
        cin >> start_station;
        for (i = 0; i<view_count; i++)
        {
            if (start_station == views[i].name)
                start_num = i;
        }
        while (start_num == -1)   //容错处理
        {
            cout << "你的输入有误，请重新输入！" << endl;
            cout << "Please input the start station name:";
            cin >> start_station;
            for (i = 0; i<view_count; i++)
```

```
        {
            if (start_station == views[i].name)
                start_num = i;
        }
    }
    cout << "Please input the end_station name:";
    cin >> end_station;
    for (i = 0; i<view_count; i++)
    {
        if (end_station == views[i].name)
            end_num = i;
    }
    while (end_num == -1)     //容错处理
    {
        cout << "你的输入有误，请重新输入！" << endl;
        cout << "Please input the end_station name:";
        cin >> end_station;
        for (i = 0; i<view_count; i++)
        {
            if (end_station == views[i].name)
                end_num = i;
        }
    }
    cout << endl << endl;
    cout << "From " << views[start_num].name << " to " << views[end_num].name;   //输出最
短经由路径
    if (dist_list[start_num + 1][end_num + 1] == -1)     //没有找到的情况的回应
        cout << " no way." << endl;
    else
    {

        cout << " distance is " << dist_list[start_num + 1][end_num + 1] << ", and path is:" <<
endl;
        k = path_list[start_num + 1][end_num + 1].path[0] - 1;
        cout << views[k].name;
        for (m = 1; m<path_list[start_num + 1][end_num + 1].count; m++)
        {
            k = path_list[start_num + 1][end_num + 1].path[m] - 1;
            cout << "->" << views[k].name;
        }
    }
    cout << endl;
}
void adddata(int menu)
```

```
{
    if (menu == 1)
    {
        while (1)
        {
            addview();
            cout << "do you want to continue?(y/n)" << endl;
            char con;
            cin >> con;
            if (con == 'y');
            else
                break;
        }
    }
    if (menu == 2)
    {
        while (1)
        {
            addline();
            addway();
            cout << "do you want to continue?(y/n)" << endl;
            char con;
            cin >> con;
            if (con == 'y');
            else
                break;
        }
    }
}
```

*MATLAB*数据筛选源程序：

```
clc;
clear;
[data,text]=xlsread('C:\Users\lenovo\Desktop\licheng.xls',1,'A1:D49369');
A=cell(49369,4);

for i=1:49369

A{i,1}=text{i,1};
A{i,2}=text{i,2};
A{i,4}=data(i);
```

```
        if      A{i,4}== 0
                A(i,:)=[];
        end
end
A;
xlswrite('距离 lines',A)
```