**数据结构课程设计报告：旅行模拟查询系统设计**

组员：王博琛 2017211526 夏东林2017211516 马诚宇2017211525

**一、软件需求分析**

（一）需求分析：

当今时代，交通越来越发达，出行方式越来越越多并且越来越方便，无论是出差开会、还是跨地旅游都不是什么难事儿。而目前，出行线路的选择和规划，常常令出行者头疼不已。同样，出行目的的不同和出行方式的不同也让出行规划变得复杂多样。因此，急需一款软件：存储各个城市的交通信息（即交通方式时间价格表），并且能够根据用户提出的出行时间（出发时间，到达时间）出行策略（速度最快，价格最低，限时间价格最低）对用户旅行做出规划。

（二）问题描述：

城市之间有三种交通工具（汽车、火车和飞机）相连，某旅客于某一时刻向系统提出旅行要求，系统根据该旅客的要求为其设计一条旅行线路并输出；系统能查询当前时刻旅客所处的地点和状态（停留城市/所在交通工具）。并且模拟出用户出行的时间路线关系。

（三）功能设计：

城市选择了：北京，青岛，济南，石家庄，大同，太原，西安，榆林，郑州，开封。

系统时间采用按小时推进，交通方式起始时间和到达时间四舍五入到整点。并且不考虑换乘交通工具所花费时间。

城市的交通方式有飞机，火车（动车+高铁），客运汽车，不同的城市方式不一。构建timetable，按照携程网上的真实票价，时间，花费时间（舍入到整点，整小时）。确定起点和终点以后，可以根据用户要求选择途径的城市（按顺序选择）。用户可选择三种旅行策略：速度最快，花销最低，限时花销最低。软件可以记录并输出用户的旅行日志，即用户模拟旅行状态。

综上：用户选择起点与终点，选择最早出发时间，选择途径的城市，并且选择三种旅行策略（限时花销最低则选择最长时间），软件根据用户所提出的信息，计算出较优的出行路线，并输出花费，总时间，交通方式信息。若不存在路线则输出无路线。并且可以根据出行路线画出地图，模拟出行过程。

**二、概要设计**

（一）核心函数定义及接口说明：

***1、console.py***模块：

**def input\_file():**读取文件timetable并存储。

返回值：city\_num 城市数

city\_squence 城市序列

city\_matrix 序列矩阵

table 时刻表

**def input\_travel\_info(city\_squence,city\_num):**读取用户信息

参数： city\_squence 城市序列

city\_num 城市数

返回值：start\_point 起点

destination 目的地

starttime 开始时间

time\_limit 时间限制

**def find\_start\_num(I,j,city\_matrix,city\_num):**寻找到表中对应城市时刻表

参数： city\_matrix

city\_num

返回值：city\_matrix[i-1][city\_num-1]+1或city\_matrix[i][j-2]+1 timetable中对应位置

**def min\_money(city\_num,city\_matrix,start\_point,destination,table)：**计算最少费用路径

参数： start\_point 起点

destination 终点

返回值：path 路径

**def min\_time(city\_num,city\_matrix,start\_point,destination,starttime,table)：**计算最少时间路径

参数： starttime 开始时间

返回值：path 路径

**def min\_time\_limit(city\_num,city\_matrix,start\_point,destination,starttime,table,time\_limit)：**计算限时最少费用路径

参数： time\_limit 最长限制时间

返回值：path 路径

**def calculate\_time(city\_num,starttime,table,path)：**计算路径总共花费时间

参数： path 路径

返回值：time\_sum

**def calculate\_money(city\_num,table,path)：**计算路径总共花费钱数

参数： path 路径

返回值：money\_sum

***2、travel.py***模块：

**def setupUi(self,MainWindow)：**创建程序窗口

**def calculate(self)：**窗口按键计算触发事件，根据读入信息计算路径并输出

**def simulate(self)：**窗口案件模拟出发事件，根据路径信息模拟

***3、simulate.py***模块：

**def processinfo(self)：**模拟履行过程

**def paintEvent(self,event)：**画图函数

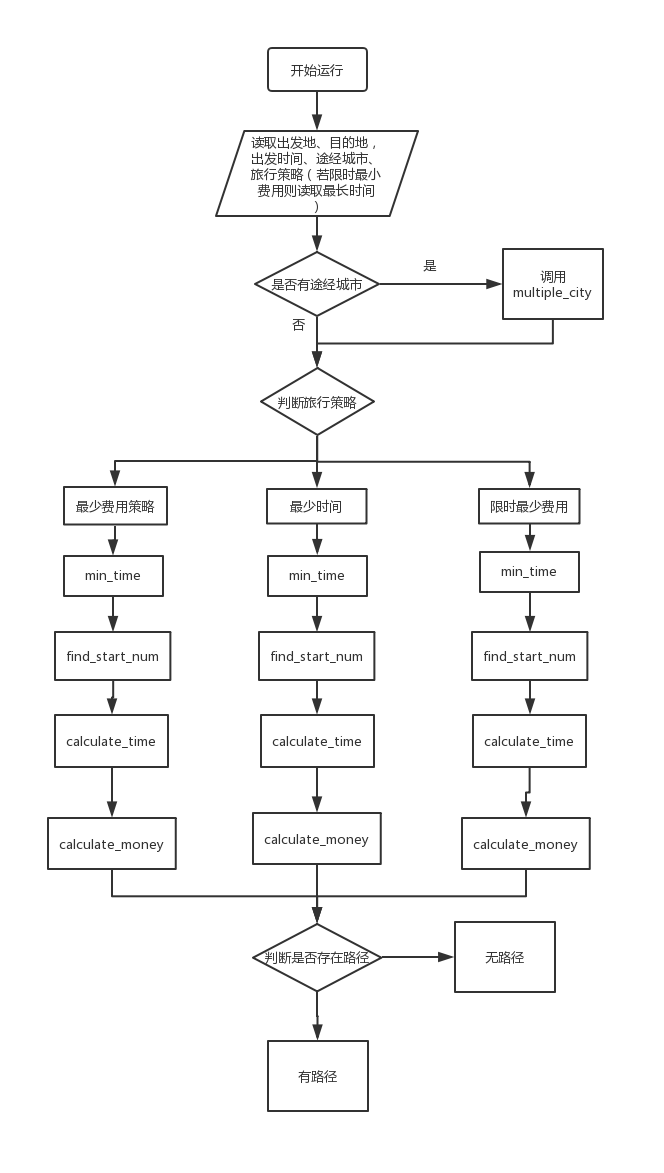
**def timerEvent(self,e)：**动态时间控制函数

**def accelerate(self)：**按键加速事件

**def inaccelerate(self)：**按键减速对应事件

***4、run.py***模块：运行整个程序，构造窗口，读取信息，输出计算信息

（二）算法流程图



1. **详细设计**

（一）数据结构说明和数据字典：

1、算法模块：

(1)使用名为city\_num的int变量存储该路线文件中的城市个数。

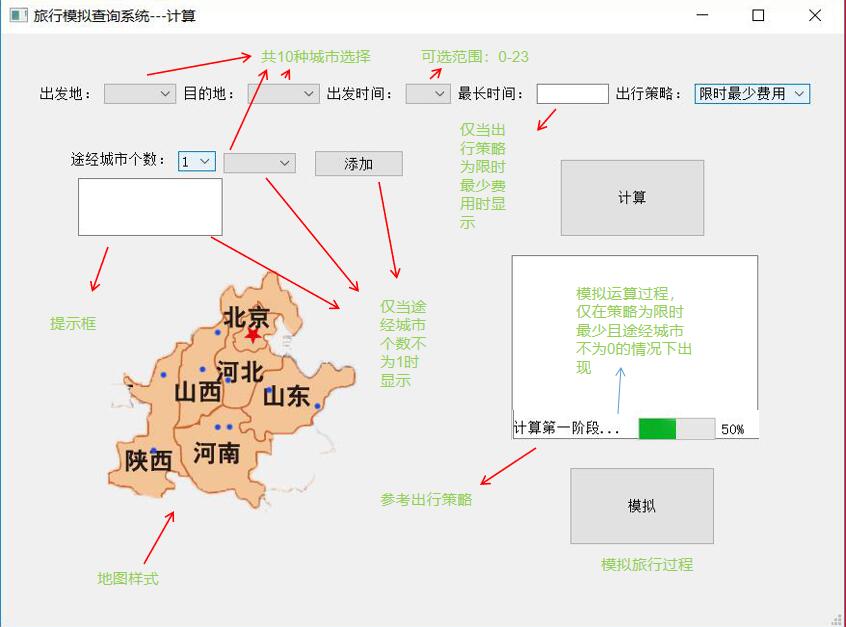
(2)系统将不同城市抽象为从0开始的、小于city\_num不同的数字。相当于 ∃ 一一对应的f, f将城市映到{ i | i < city\_num, i ∈ N+}。使用名为city\_squence的list变量存储数字i与城市的对应关系。列表的每一项为一个字符串，存储城市的名字。在系统中，数字i代表city\_squence[i]存储的城市。

(3)使用名为table的list变量存储飞机、汽车、火车的时刻表。table列表的每一项是一个列表，table中的每一项存有交通工具类型、车次名称、发车时间、到达时间、价格、总时间。table每一项的顺序是优先根据出发地城市对应的数字、其次根据到达地城市对应的数字从小到大排序的。

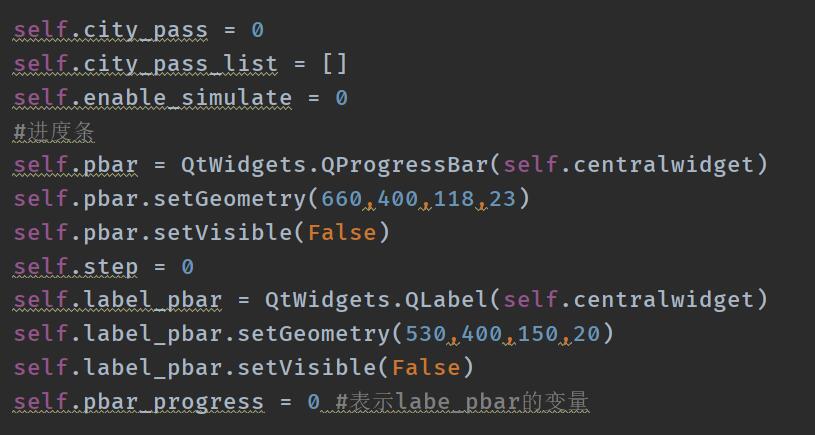
(4)使用名为city\_matrix的二维列表存储不同起点终点对应table中的序号。city\_matrix[i][j] (i≠j) 表示: max { k | table[k] ∈ i地到j地的车次}(i地到j地有路时) 或max { k | k ∈ city\_matrix[a][b] a <= i b < j } (i地到j地有路时)。

2、图形化模块：

(1)计算窗口：



组件功能示意如上图，程序内其余变量功能如下：



city\_pass：用于存储当前已添加的途经城市个数

city\_pass\_list：将所有途经城市以一个列表形式传递给其余函数

enable\_siulate：为0表示此时不允许点击模拟按钮，为1表示允许

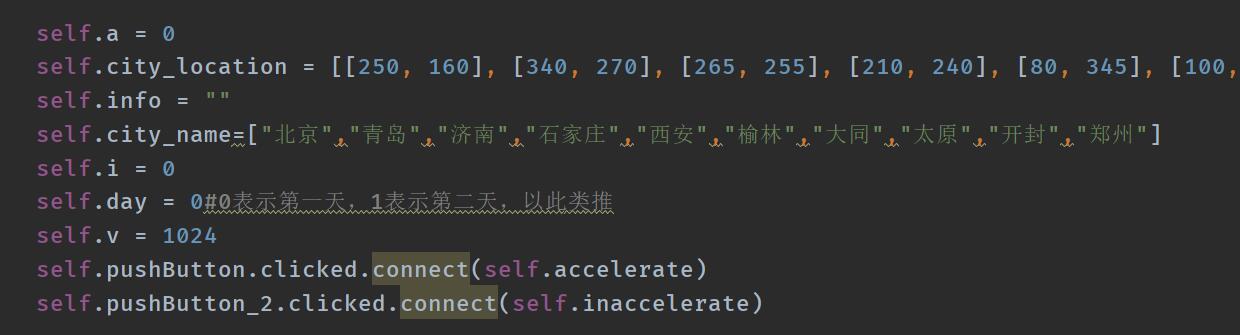
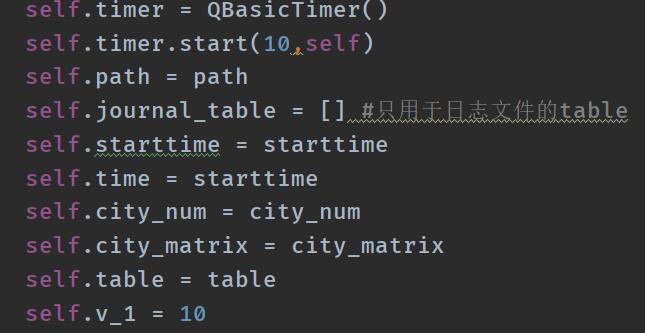
step：用于向进度条pbar设值

pbar\_progress：用于表示进度条的上限

(2)模拟窗口：



组件功能示意如上图，程序内其余变量功能如下：



timer：计时器

path，starttime，city\_num,city\_matrix,table：从计算窗口传入的参数

journal\_table：用于输出日志文件的路径表

a：用于提供作图的速率参数

city\_location：10个城市在地图上的坐标

Info：用于设置每一行的日志文件

city\_name：10个城市的名称

i：在输出日志文件中用于表示当前为第几条路径

day：在输出日志文件中用于表示当前为第几天

v：日志文件输出速度相关参数

v\_1：日志文件输出速度相关参数

1. 各模块设计说明：

**1、算法模块:**

(1)函数：

① ***calculate\_time：***

计算时间，设该函数为f，f把path（路线，即乘车车次和顺序）和starttime（出发时间）映到{ time | time ∈ Z}。time为从出发时间开始，依次乘车，到乘车结束的时间。

② ***calculate\_money:***

计算费用，设该函数为f，f把path映到{ money | money ∈ Z}。money为乘车的总费用。

③ ***get\_point：***

计算起点终点，设该函数为f，f把一个数字num和city\_matrix，映到ZxZ。设第一个整数为i，第二个为j。i、j为table[num]中车次的起点、终点。

④ ***find\_start\_num:***

1. j表示两个城市，分别为起点终点。设该函数为f，f把i，j映到 { n | n ∈ Z}。其中n = 1 + max{city\_matrix[a][b] (a <= i, b < j) }。

(2)策略：

①最少费用策略：无时间限制，费用最少即可（函数：min\_money）

算法的思路与《数据结构》课本中的Dijkstra算法相似。城市、车次构成的图的边长为城市间价格最低的直达车次的价格。跟据实际情况：所选城市和车次构成的图连通性较好（两城市之间没有直达车的情况较少），导致使用课本中描述的Dijkstra算法未到达的城市在两次循环后就为0；几乎要进行城市数量次循环，才能找到起点到大部分城市的最短路。并且考虑到编程的简单，所以未对已找到最短路的城市进行存储。同时，考虑到两城市间有多趟车次，并且要输出所坐的车次，所以使用一个列表对从起点到达某城市所乘坐的车次进行存储。

②最少时间策略：无费用限制，时间最少即可（函数：min\_time）

与（1）相似，只是城市、车次构成的图的边长为到达某城市的最早时间。

③限时最少费用策略：在规定的时间内所需费用最省（函数：min\_time\_limit）

算法的外层循环与（1）相似，内层借鉴了动态规划的思想，相当于并行多个动态规划：循环搜索在某城市第i小时（i=0,1,2,…,限制时间-1）出发，经过j小时（j=1,2,…,限制时间-i）到达相邻城市的最少费用。

④计算途径城市路线的算法：（函数：multiple\_city）

该函数能计算确定城市顺序的最少费用、最少时间和限时最少费用策略。

对于最少费用策略，只需按顺序依次计算各个城市间的最小费用，让后将路线按顺序连起来，就得到带有途径城市的最少费用策略算法。

对于最少时间策略，按顺序依次计算各个城市间的最小时间，将上一段路的到达时间作为下一段路的开始时间，让后将路线按顺序连起来，就得到带有途径城市的最少费用策略算法。

对于限时最少费用策略，使用动态规划算法，依城市次序求i小时（i = 1, 2, …, 限制时间-1）到中途城市的最少费用，最后求得i小时（i = 1, 2, …, 限制时间）到终点到最少费用。

(3)所使用算法的正确性

①路径中每个点到起点的距离是最短的，该路径就是最短的。反证法：假设路径中有一点a到起点b的距离为x1，a到终点c的距离为x2，bc的最短路程为x。而x1不是最短的，那么，存在a到b有更短的x3。有x3 + x2 < x1 + x2 = x，所以x不是最短路径，矛盾。□

②对于最少时间策略，有结论：如果开始时间t1 < t2，，∃函数f，f将开始时间、起点终点映到一个（使用最少时间策略策略得到路线的）道达时间。起点终点相同的情况下有f(t1) <= f(t2)。证明简单，不再赘述。

③动态规划

跟具不同城市划分不同阶段，跟具不同的到达时间状态。这里的状态具有无后效性（即马尔可夫性）。

定义：xk状态k的状态变量，Xk为阶段k允许的状态集合，即在城市k可能的时间。设决策u为从一状态转移到另一状态的函数，即xk+1 = ux(xk)。阶段指标d为从一个开始时间开始从一城市到另一城市所需时间。指标函数Vk, n，即在k城市，n小时到的最少费用。

最优值函数fx+1(xk+1) = min{dk(xk, uk(xx))+fk(xk)} (k = 0, 1,.., 限制的时间）。

动态规划的最优性定理：有V0, n-1 = min{ V0, k-1(x0, p0, k-1)+min{Vk, n-1(xk, pk, n-1}}时，允许策略p\*0, n-1=(u\*0,..., u\*n-1)是最优策略。

证明：

V0, n-1(x0, p0, n-1)

= V0, k-1(x0, p0, k-1) + Vk, n-1(xk, pk, n-1)

≤ V0, n-1(x0, p0, n-1) + min { V0, n-1(x0, p0, n-1) }

≤ min {V0, n-1(x0, p0, n-1) } + min { V0, n-1(x0, p0, n-1) }

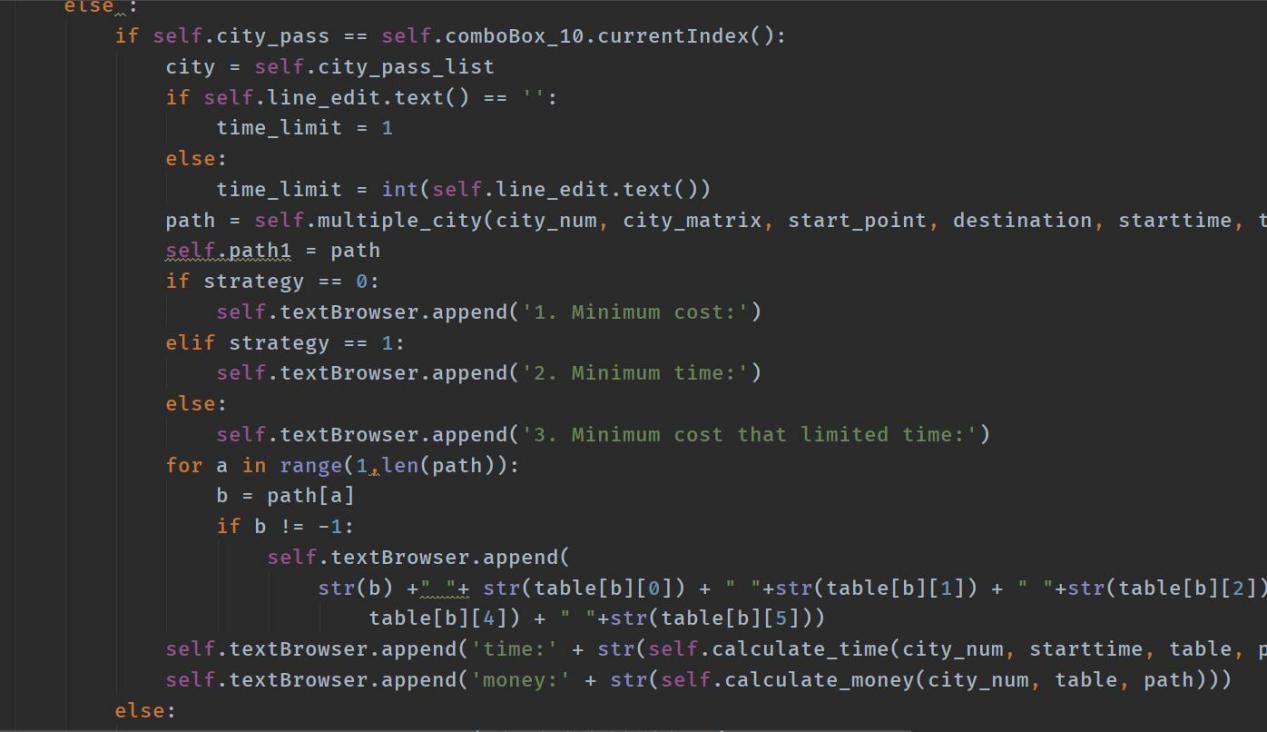
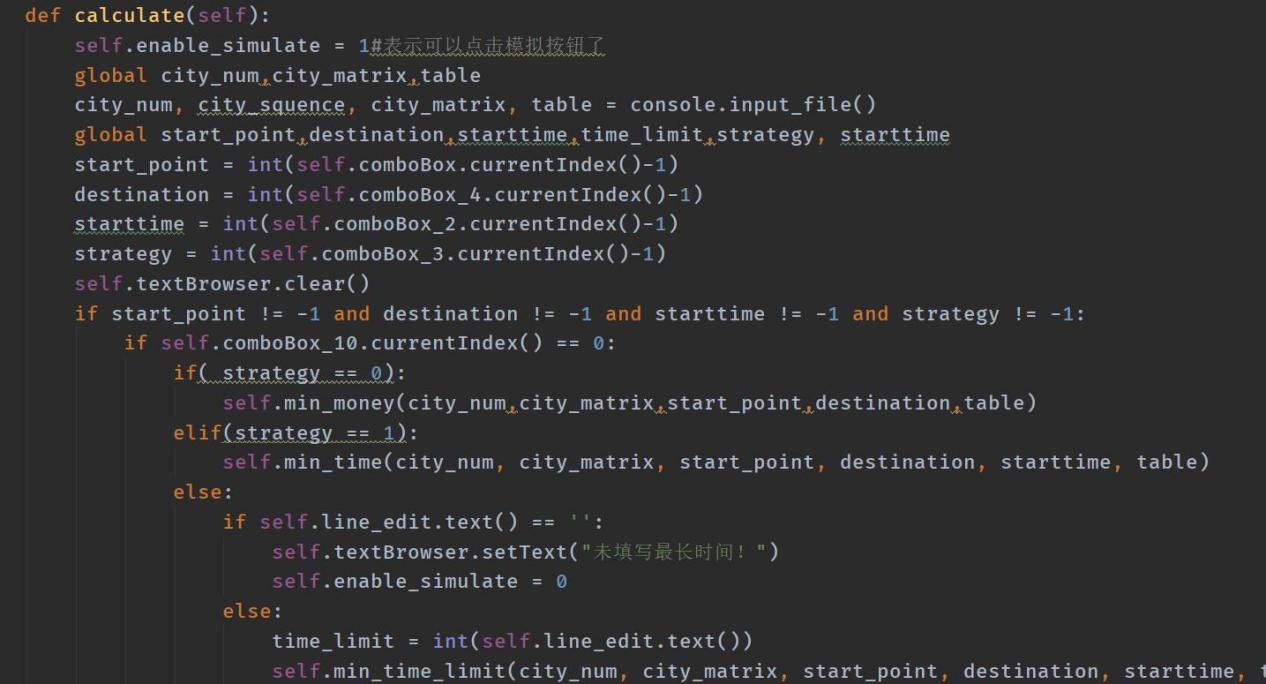
≤ V0, n-1(x0, p\*0, n-1)

同时，p\*0, n-1是最优策略。□

**2、图形化模块：**

(1)计算窗口：

①核心模块***calculate：***



该模块通过点击“计算”按钮进入，并将允许点击模拟按钮设置为允许，并将用户选择的出发地、目的地、出发时间、最长时间、策略存进相应变量，并将textBrowser中内容清空。

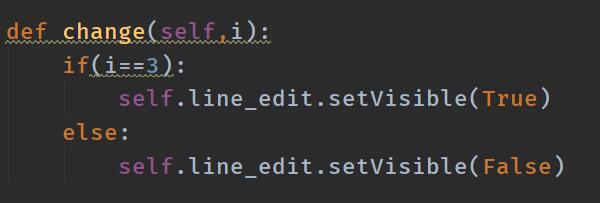
之后判断用户选择项目是否有效，若不有效则在textBrowser中输出提示信息。如果有效，则首先判断途经城市是否为0。若为0，则根据策略直接调用三种策略函数进行计算，若不为0，则调用multiple\_city函数进行计算。

②模块***simulate：***



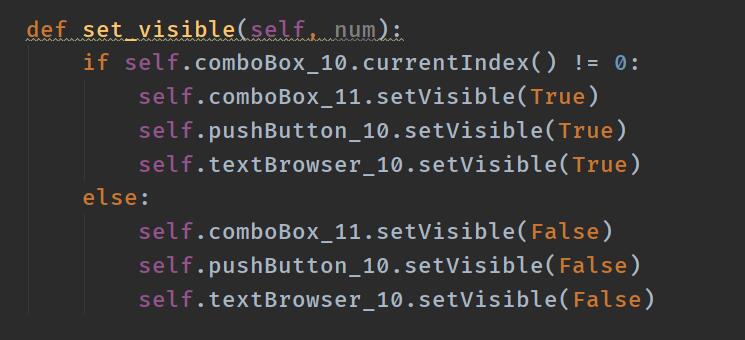
该模块通过点击模拟按钮进入，flag的作用是判断是否存在路径，如果不存在则为0，存在则为1。如果存在则进入模拟窗口，为0则输出错误信息。

③模块***change：***



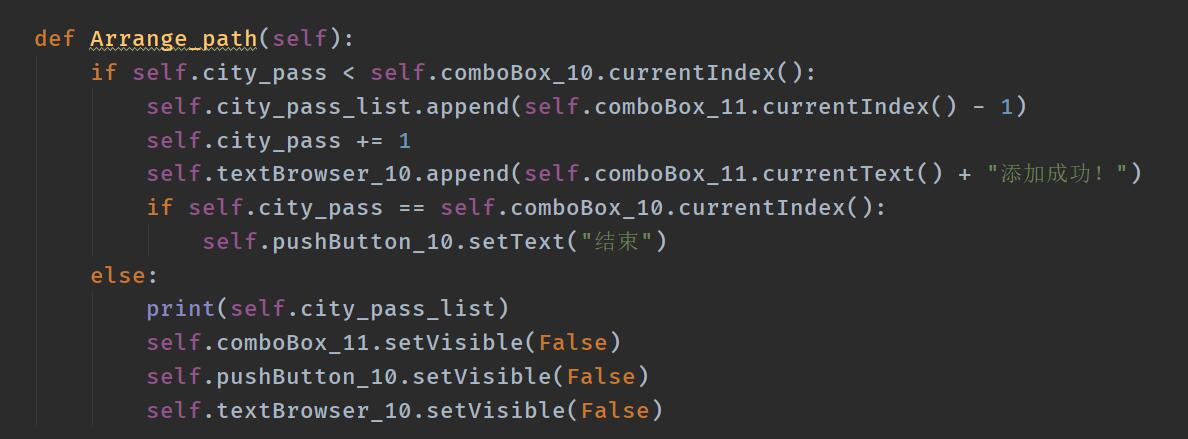
该模块通过设置策略进入，如果为限时最少策略，则将最长时间的编辑框设为可见，否则不可见。

④模块***set\_visible：***



该模块通过设置途经城市个数进入，如果不为0则将对应的添加城市框、添加按钮和提示信息框设为可见，否则不可见。

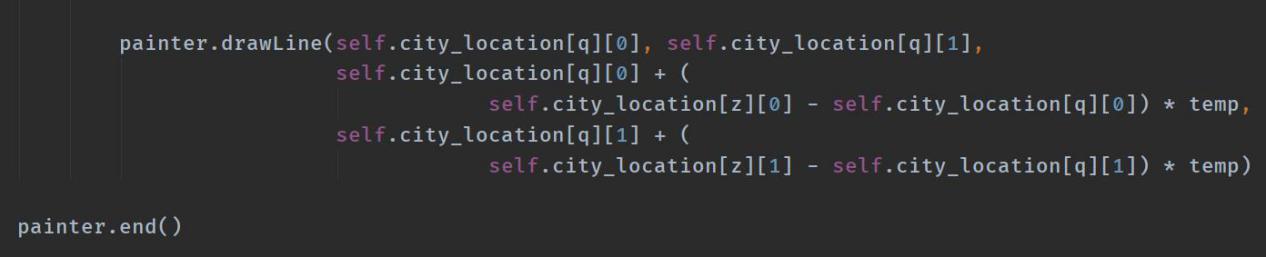
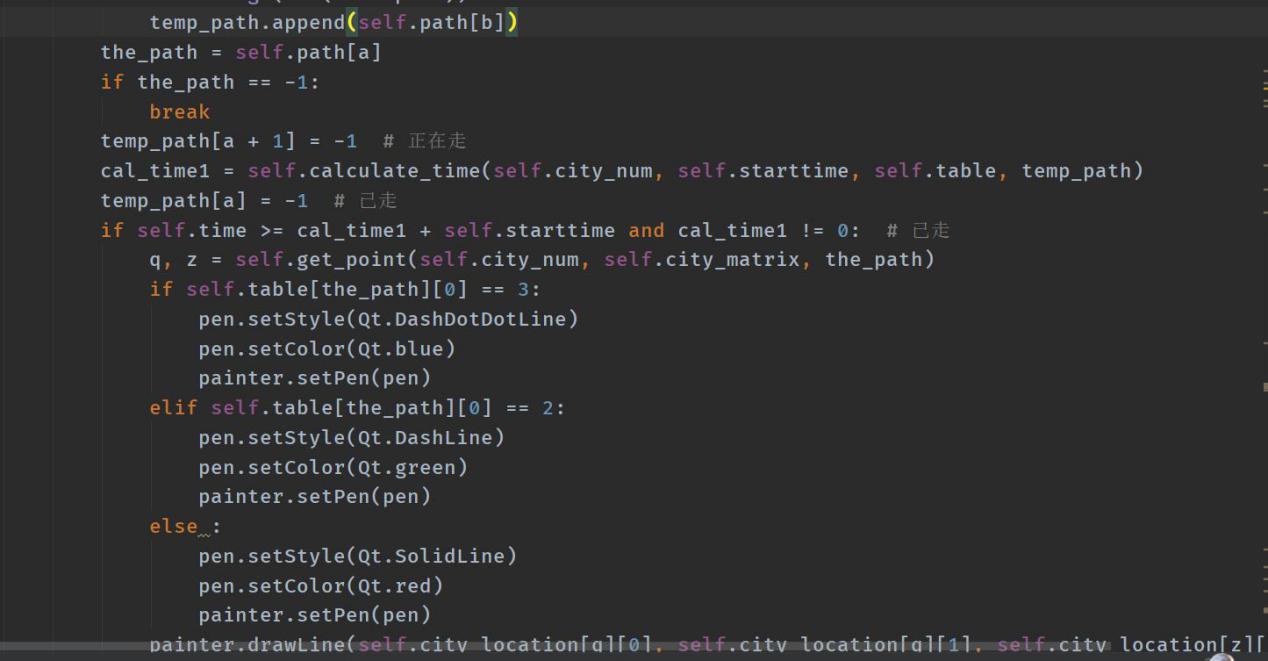
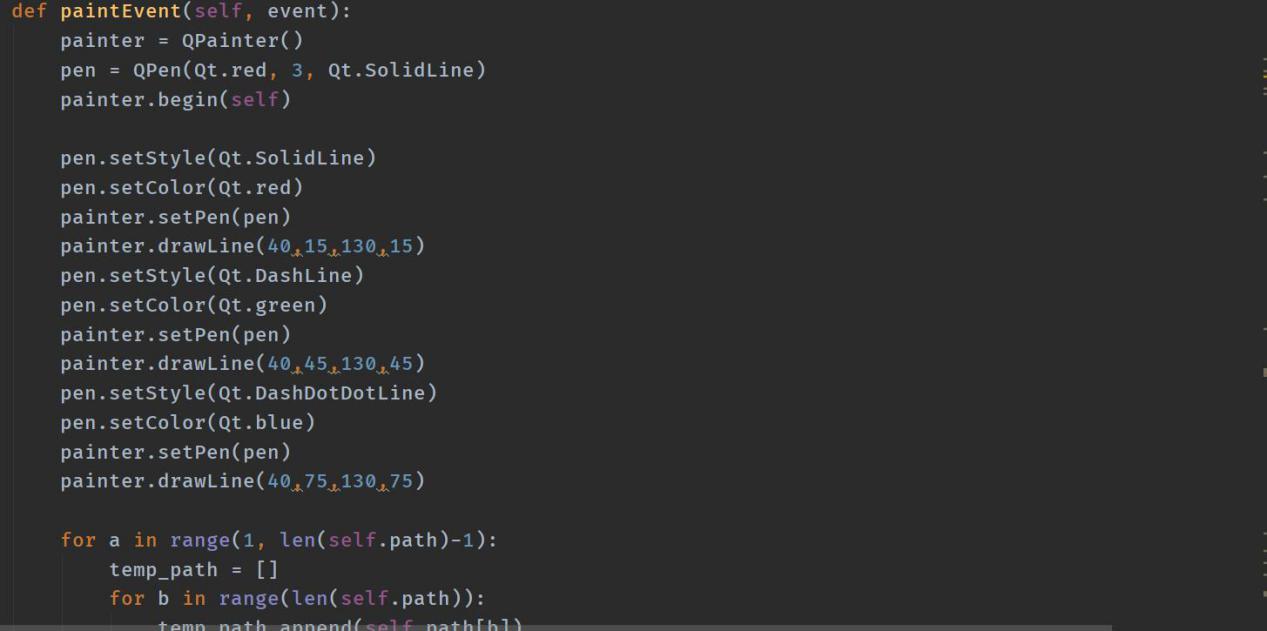
⑤模块***Arrange\_path：***



该模块通过点击添加途径城市按钮进入，首先根据city\_pass变量和实际途经城市个数判断是否输入完毕，未输入完毕则继续向city\_pass\_list添加城市并输出提示信息，输入完毕后将三个组件设为不可见。

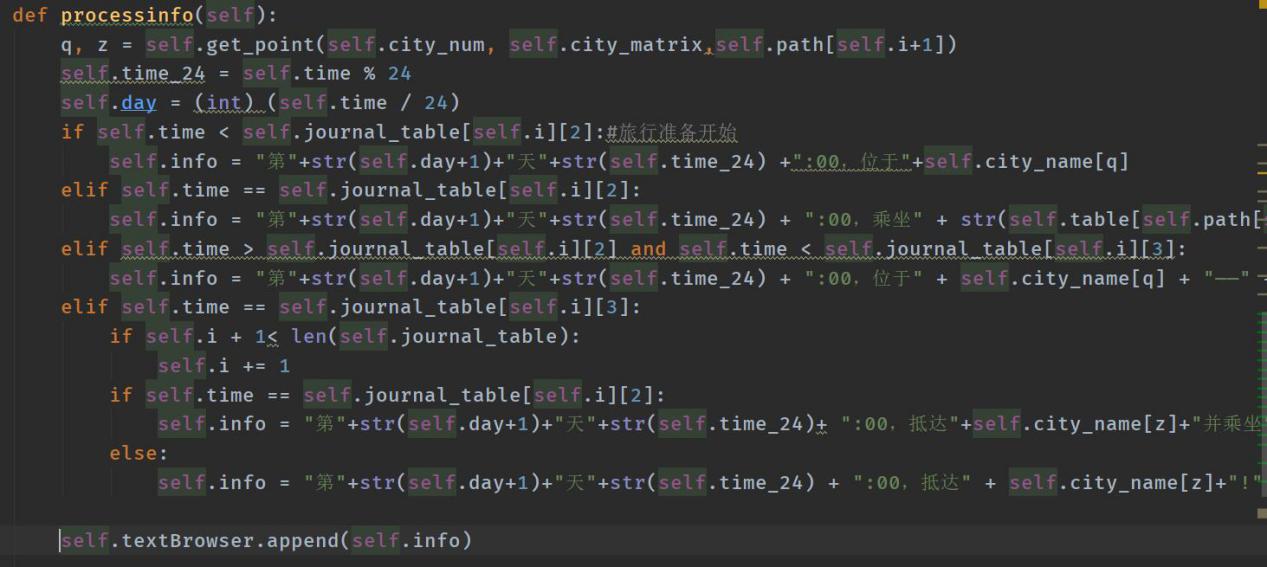
(2)模拟窗口：

①核心模块PaintEvent:



该模块用于绘制旅行线路，首先将窗口左上角的图例画出。之后进入主循环，设置两个变量temp\_path和the\_path储存路径，temp\_path为当前路径，the\_path为当前的路径对应于时刻表上的行数。内层设置两个条件，如果判断当前时间下当前路径已经走完，则直接将线路画出，如果当前时间正处于路径途中，则设置一个temp变量作为画线的速率，并根据时间动态画出线路。

②核心模块***processinfo：***

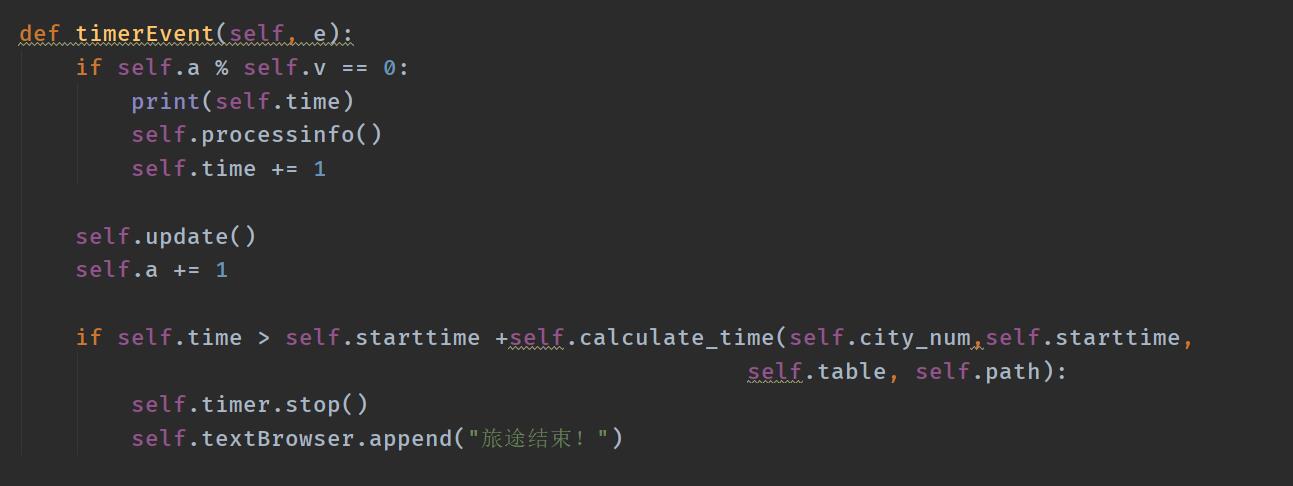


该模块用于输出日志文件，首先算出当前路径下的起点和终点，新设置一个24小时制的时间用于输出，并用day变量计算出当前是第几天。

之后根据时间的不同情况可以分为四大类：在某地等待出发、出发、路途中、到达。

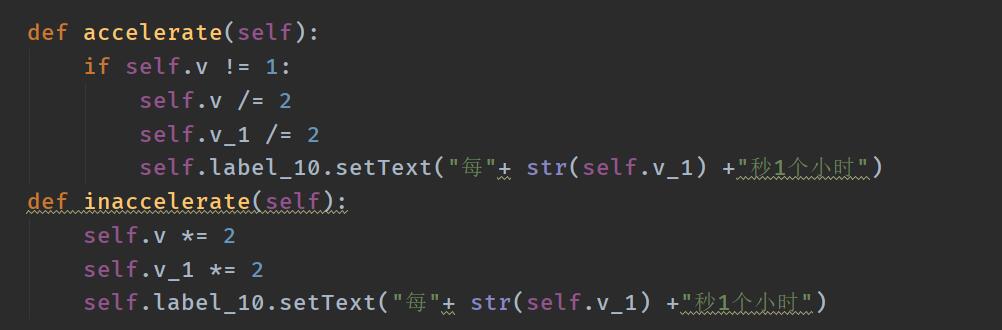
对于每一类分别设置info字符串的内容，并在最后打印在textBrowser上。

③核心模块***timerEvent：***



该模块用于模拟时间的推进，a和v分别为速率参数，根据加速和减速按钮的点击设置不同的v，a叠加到一定数目时，输出日志文件并利用update()函数模拟一个时间，进入到模块paintEvent中画图。直到当前时间超过了旅行的总时间，关闭时间定时器并输出日志文件。

④模块***accelerate***与inaccelerate：



两个模块通过点击加速和减速按钮进入，分别将v与v\_1减少或增加，并输出提示信息在label上。

在加速模块中，由于timerEvent模块中v作为取余的对象，不能为小数，所以设置一个条件，如果v已经减小到1则无法再加速。

1. **范例执行结果及测试情况说明**

（一）样例一

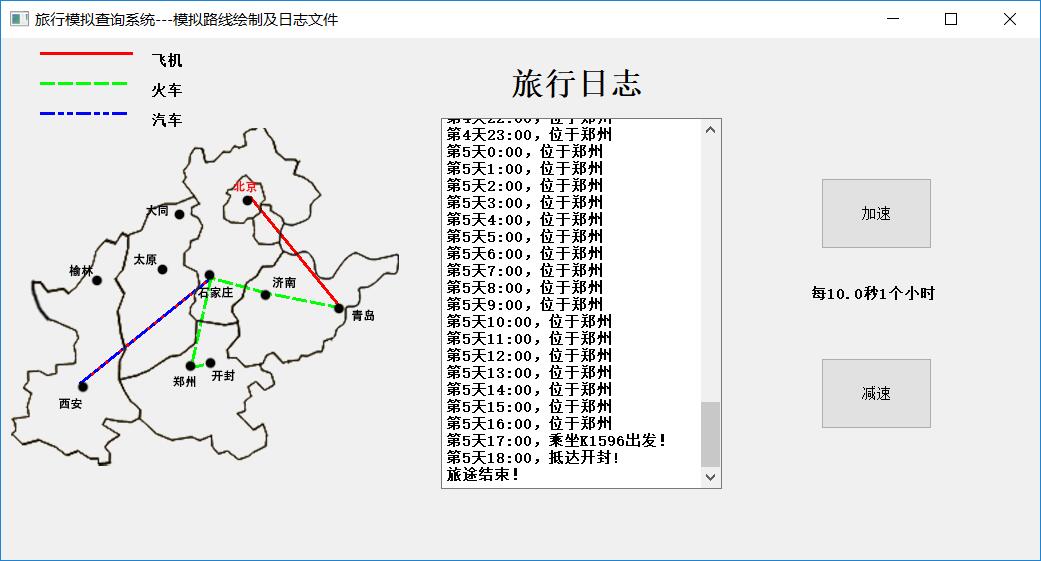
出发地：北京

目的地：开封

出发时间：14:00

策略：最少时间

途经城市：青岛、济南、石家庄、西安、郑州



（二）样例二

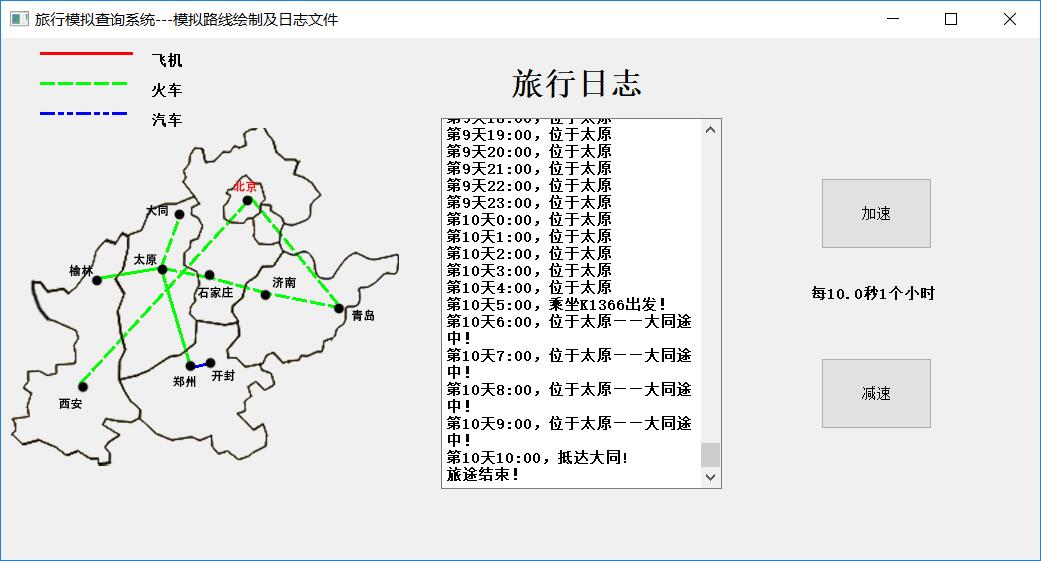
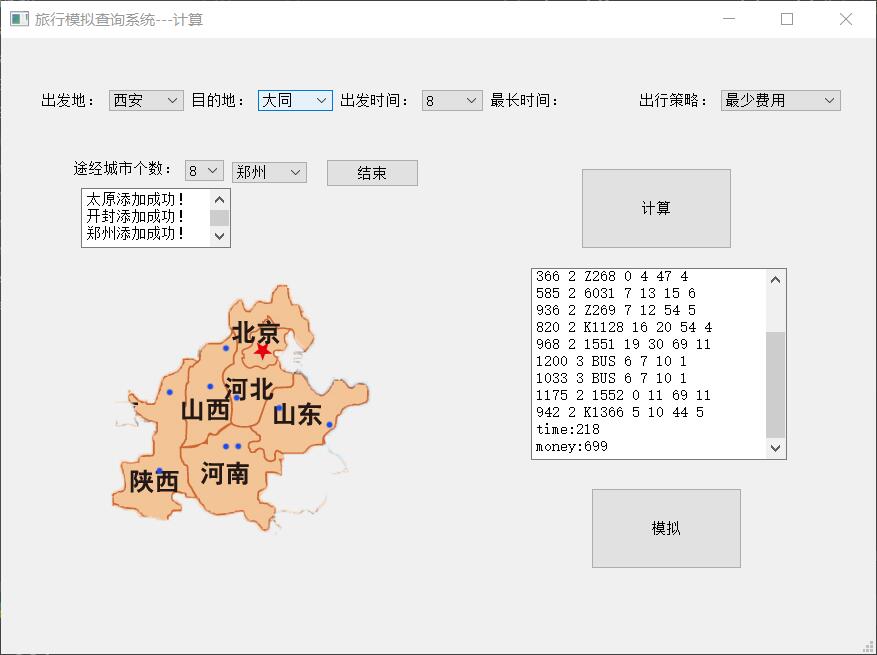
出发地：西安

目的地：大同

出发时间：8

策略：最少费用

途经城市：北京、青岛、济南、石家庄、榆林、太原、开封、郑州



（三）样例三

出发地：北京

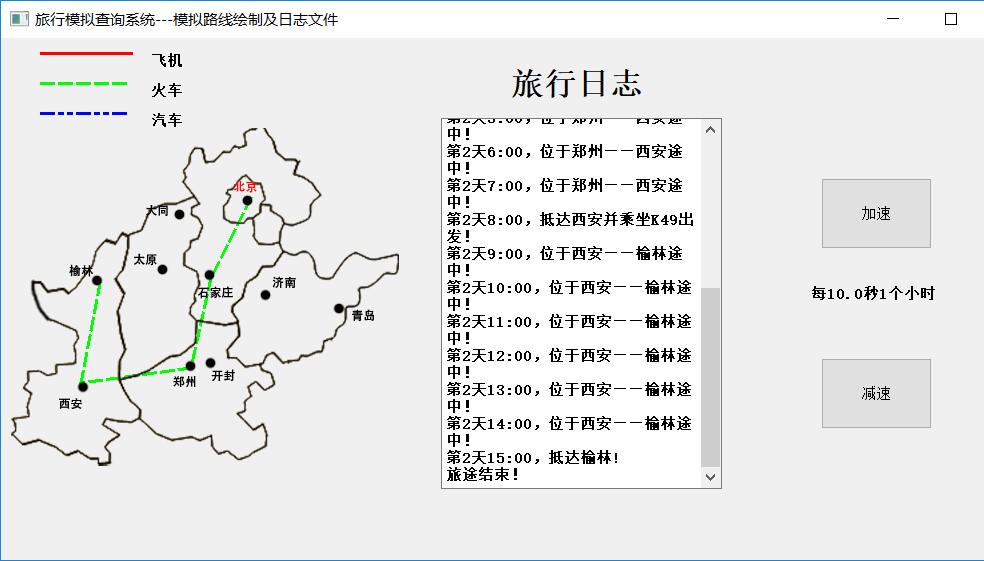
目的地：榆林

出发时间：20

策略：限时最少费用

最长时间：20

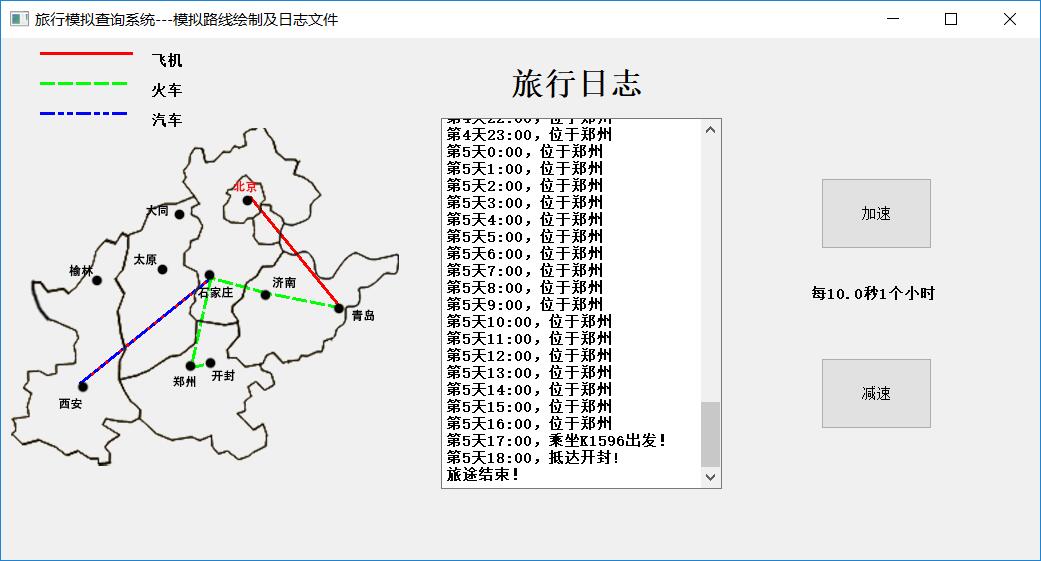
途经城市：西安



**五、用户使用说明**



在计算窗口中选择出发地、目的地、出发时间、出行策略，途经城市。如果出行策略为限时最少费用策略，则再输入想要限定的最长时间，点击计算按钮，稍等片刻，等待提示框中输出路径，再点击模拟按钮进行模拟。



在模拟窗口中，可以点击加速或减速按钮来调整输出的速度。

参考文献：

[1]卓新建.运筹学[M].北京邮电大学出版社:北京,2013.

[2]《运筹学》教材编写组.运筹学[M].清华大学出版社:北京,2012.