**华中科技大学网络空间安全学院**

**程序设计综合课程设计**

**报告**

题目: 武汉市地铁交通系统

班 级： 信息安全1901

学 号： U201911657

姓 名： 李文重

成 绩：

指导教师： 朱建新

**完成日期：** 2021**年 3月 25日**

**目 录**

[一、系统需求分析 1](#_Toc9999)

[二、 总体设计 3](#_Toc24917)

[三、数据结构设计 5](#_Toc4451)

[1、数据文件存储结构 5](#_Toc26440)

[2、 程序中存储结构 5](#_Toc27488)

[3、 使用的数据结构 6](#_Toc26095)

[4、 几个特殊的数据的存储 6](#_Toc22830)

[四、详细设计 8](#_Toc12384)

[1、 数据处理部分 8](#_Toc19177)

[2、请求响应部分 10](#_Toc16928)

[3、 用户界面部分 12](#_Toc21116)

[4、 信息安全部分 12](#_Toc8214)

[五、系统实现 13](#_Toc22826)

[六、运行测试与结果分析（黑体小2加粗居中） 22](#_Toc11891)

[复杂度分析 28](#_Toc14969)

[附录一、参考文献 33](#_Toc2673)

[附录二、主要源程序片段 39](#_Toc14175)

[附录三、程序使用说明 49](#_Toc12106)

# 一、系统需求分析

本次课程设计综合实践是要求我们用C实现一个武汉地铁路径规划系统。

从底层来说，是要求我们能够在一个比较简化的地铁运行模型里面，模拟出武汉地铁的运行状态，包括不同线路和不同车次。从表层来说，是要求我们能够实现一个能够完成用户一般使用地铁交通的需求，比如一个人想要随便乘坐地铁遍历。或者为需要出行指南的用户提供寻找比较优秀路径的指导，并且能够提供比较优秀的用户体验。

因此设计中，将程序分为前端和后端是我的基本思路。

在设计后端的时候，注意到使用的是简化模型，这意味着实际上不需要很多的数据，基本都是站点信息，线路信息之类的，我只额外使用了一个经纬度坐标信息。而程序中的运算时，数据虽然种类不少，但基本上都是比较容易去维护的。而且，正因为数据量比较少，算法的复杂度对于用户的体验的影响就少了许多。当n=100的时候，,的实际差距并不大，因此我把精力组要投入到了优化用户体验的方面，也即前端方面。

而后端要求能够完成的任务主要有这些：

1.选择地图上的任意一个始发站点，然后浏览其相邻站点，或在相邻站点换线，实现站点浏览。这个过程实际上就是对于乘地铁遍历的时候的状态转移的考验，实际上这个也是后面所有的功能的基础。

2.计算出武汉地铁每个站的可行线路以及乘车时间表，以方便用户了解最近一班车什么时候到。这个实际上考察的是对于列车运行信息的维护。但这个地方如果实际的去维护每一辆列车，难度比较大，而且没有必要。而简化的数据约定则为使用数学方法完成它创造了条件。这样实际上就变成了一个带有bool型判断的数学公式。

3.求解1-3条总实际时间/体感时间/路程（票价）尽量短的合理路线，要求到达终点站点时间在该线路的正常运行时间范围内，否则为不可到达； 并且获得个各条线路应当能够在按照某种顺序排序输出。

4.支持对于多种信息的修改，比如时间或者拥挤度。并且通过某种方式利用这些信息，这个实际上还是对于状态转移的各种判断。

除了以上方面之外，还要求了用户界面以及信息安全方面的设计。

用户界面方面自然就是为了提升用户的体验，而且一个自由度比较低的界面(比如只由按钮构成)本身就可以帮助保护信息的安全。

而信息安全方面，说实在的我并没有想到这个要求是为了维护哪些信息的安全，本身输入的信息就基本都是互联网上可以找到的。而且整个工程实现的是对于地铁的一个普适的系统。并没有涉及到太多的私密信息，比如用户信息或者交易信息。所以我认为主要的方面在于对于程序本身的保护，也即对于程序实现方法的保护，所以我主要在优化这个部分。

# 总体设计

根据上文中的描述，程序应该分成前端和后端。而且本着尽可能模块化设计以及降低耦合度的标准。将程序分成了若干个部分，各个部分之间通过不同的接口完成调用。

因此我把程序分为了

数据部分：文件部分，解密部分；

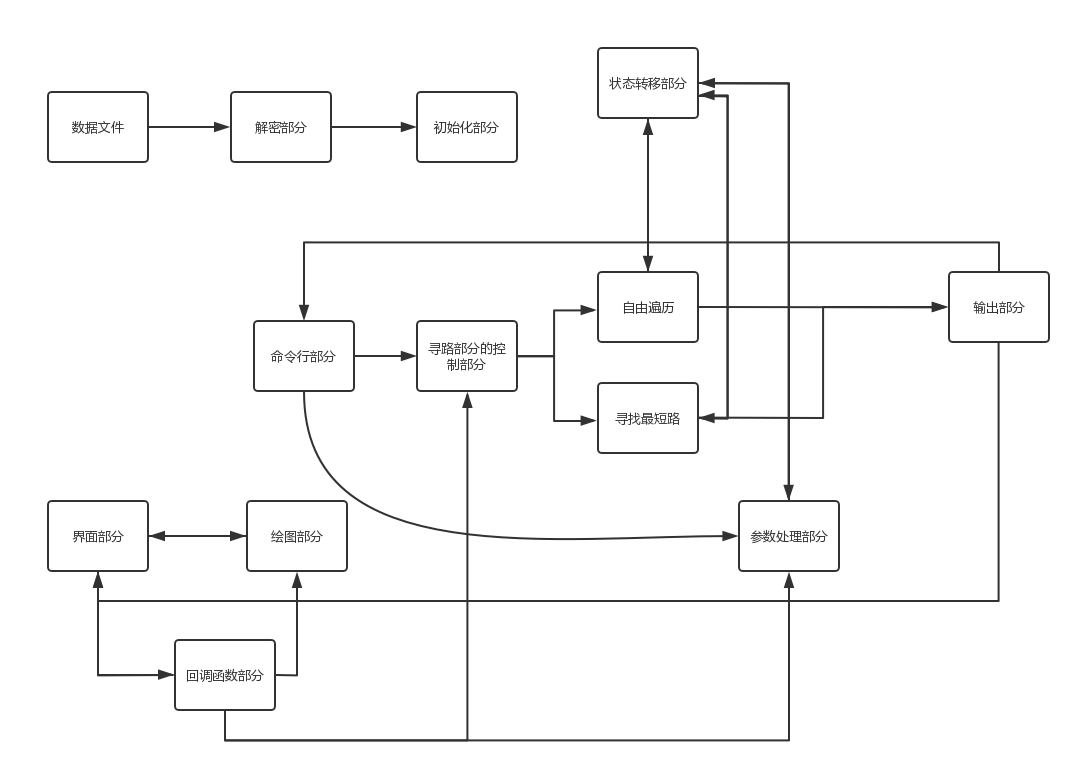
主控制部分：初始化部分，命令行界面（有GUI之后不再使用）；

数据处理部分：状态转移部分，参数处理部分（修改拥挤度，计算费用等等）；

寻路部分：自由遍历部分，寻找最短路部分，控制部分，输出部分；

GUI(gtk)部分：界面部分（主要通过glade实现），回调函数部分，绘图部分(Cairo实现)。

各个模块之间的关系如下图：



后端部分

前端部分

图2-1武汉市地铁模拟系统系统框架图

# 三、数据结构设计

## 1、数据文件存储结构

使用.txt 文本存储信息，而其中的信息进行了加密，需要程序进行解密才能够使用。因此数据的保密是基于代码的保密的

分别有：

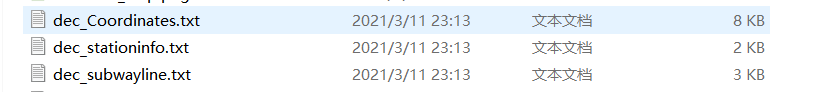


图3-1数据文件存储结构

加密存储经纬度信息，站点信息，线路信息

## 程序中存储结构

使用的结构体（具体作用可见注释和变量英文）：

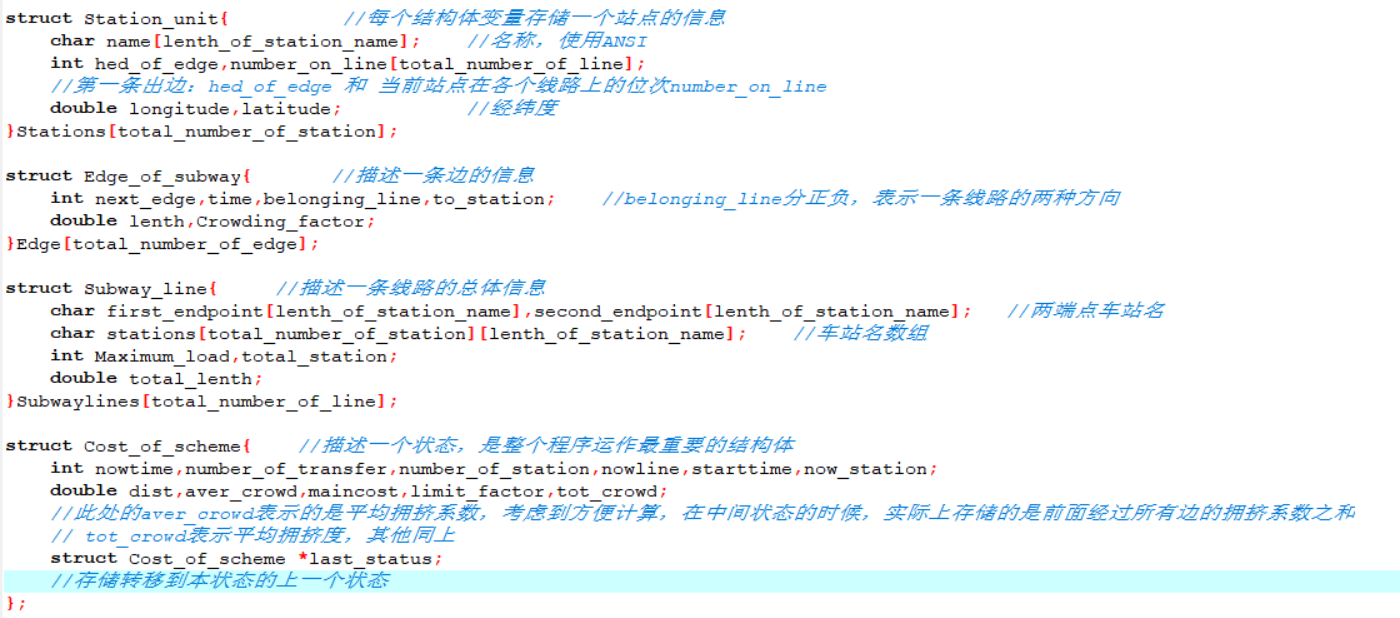
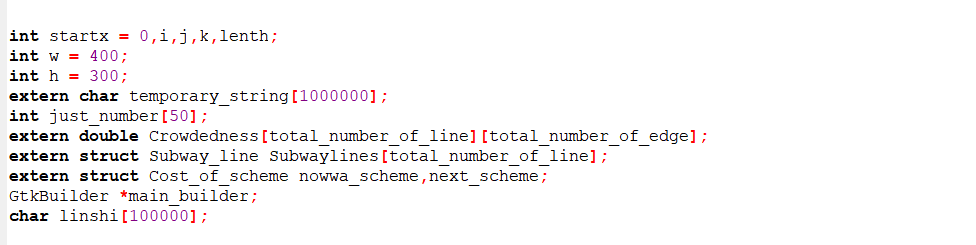
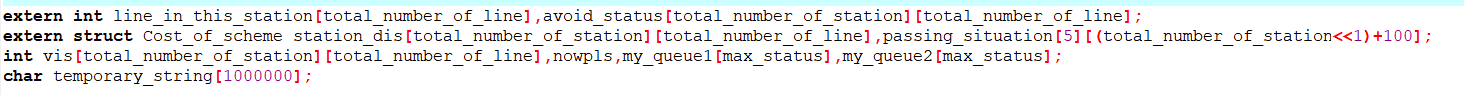


图3-2结构体存储结构

还有一些其他的变量：





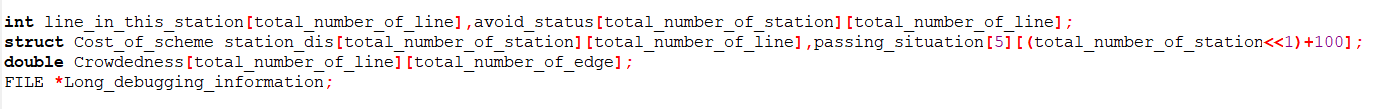


图3-3其他变量存储结构

## 使用的数据结构

因为数据比较少，而且绝大多数都是链式的结构因此主要使用的数据结构就是链表。然后图部分的存储是使用了邻接表的方式，考虑到边的数量并不很多，因此这样存是比较方便和简洁的。

## 几个特殊的数据的存储

1. 关于编码，在.txt文件和程序中所使用的编码是ANSI，但是由于gtk只支持输出utf-8的汉字编码，因此通过g\_locale\_to\_utf8函数进行转化。
2. 关于拥挤度，我是采用了时间 \* 线路的一个二维数组来完成的，这样最大的好处是查询方便，但是相对的，修改起来稍微复杂度要高一些，但是考虑到实际上高一些也只是几千数量级的所以也完全可以接受。
3. avoid\_status数组是我为了寻找最短路所引入的一个应当避免走的状态的集合。借助这个数组我就可以完成找次短路的问题。实际上相当于是对于原来的图的所有边进行了重新赋权。
4. 为了方便最短路的运算，实际上最短路永远是指maincost 一栏的值最小，因此需要根据type来决定怎么维护maincost。
5. 考虑到需要输出方案，而且进一步可能需要输出路线上经过的所有状态。因此引入了指向前一个状态的指针last\_status，这样就可以反向的获得遍历找到的路径。将整个路径装入一个数组里面。

# 四、详细设计

## 1、 数据处理部分

分为一般的数据处理部分：包括用路程计算费用的函数，修改拥挤度的函数，寻找经过本站的所有车次并且找出下一趟到达的时间函数的函数，加边函数等等。这些函数往往实现比较简单没有涉及到太多的计算，而且这些函数往往都不去调用其他的函数，因此实际上他只和前端或者很少的后端控制类函数交互。这些函数基本都在database.c中，其中有比较详细的注释。

另一类就是初始化相关的函数，比如对三个文件进行解密的函数，以及读入三个文件信息的函数。这些函数的一般特征是之和文件和控制函数交互，而且往往只会执行一遍。他们的作用是将具体的数据导入到程序的内存里面，以方便进行后续的操作，还有一些数组，变量的置为0/无穷大的操作。这些主要在database.c里面，其中有比较详细的注释。

最后一类就是涉及到状态变化的函数，这些函数以Calculate\_the\_cost为核心，综和当前状态，寻路类型，拥挤度，各种限制来求得经过某一条边转移之后的新状态。主要是辅助最短路过程的，实际上主要由各种判断和数学计算构成，在时间的复杂度上是O(1)的。

在这个地方针对规避高拥挤度线路，我额外设计了一种模式，也就是模式3，对于拥挤度取exp再取exp作为拥挤修正系数。想到这个模式是因为我有一个同学在测试带有拥挤度上界模式的时候，曾经有这样一组数据。在8点多前往一个只能通过8号线到达的站点，但是此时设计的拥挤度上界小于此时的8号线拥挤度0.65。结果就是，这个人从8点开始等，一直等到15:00拥挤度下降才去乘车。从一个车站等了7个小时，我觉得如果给使用者推荐这样的线路实际上是没有什么意义的。所以我想要一个能够避免高拥挤度线路的，但是在极端情况下也可以接受高拥挤度线路的方法。考虑到这个函数在0-1中取值是递增的，而且由于斜率是递增的，所以当拥挤度比较接近1的时候，微小的变化就会导致巨大的修正系数差异。因此就可以避免高的拥挤度，尤其是避免特别高的拥挤度。并且虽然修正系数比较大，但还是比无穷大小得多，所以在没有其他方案的时候，还是可以走高拥挤度路线的。这样就避免了等8个小时不上车的问题。在实际的测试中，基本上每次他都会指出拥挤度最小但路程偏长的路线。只有在可能另一条线路比本条要多十几站乃至几十站的时候次啊会选择拥挤度大但路程偏短的路线。

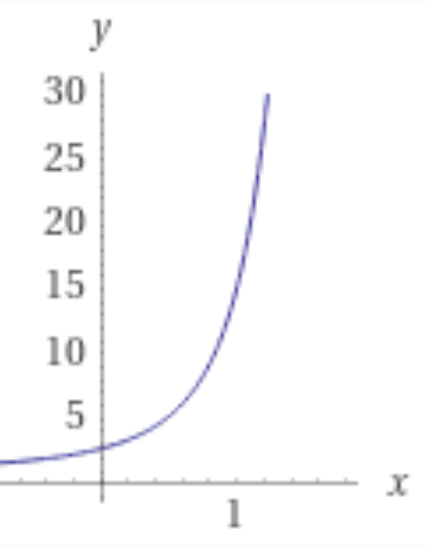


图4-1 函数在[0,1]上的图像

## 2、请求响应部分

自由遍历部分就是响应客户的要求，调用Calculate\_the\_cost进行一步的改变。比较简单。

主要是后面的推荐路线部分，我采用的是SPFA算法加上避免重复路的方法来完成寻找三条路线的要求。

SPFA是 [Bellman-Ford算法](https://baike.baidu.com/item/Bellman-Ford%E7%AE%97%E6%B3%95" \t "https://baike.baidu.com/item/SPFA%E7%AE%97%E6%B3%95/_blank) 的队列优化算法的别称，是一种比较优秀的最短路算法，而且可以在存在负权值的边的图上运行。

算法思想：我们用数组d记录每个结点的最短路径估计值，用邻接表来存储图G。我们采取的方法是动态逼近法：设立一个先进先出的队列用来保存待优化的结点，优化时每次取出队首结点u，并且用u点当前的最短路径估计值对离开u点所指向的结点v进行松弛操作，如果v点的最短路径估计值有所调整，且v点不在当前的队列中，就将v点放入队尾。这样不断从队列中取出结点来进行松弛操作，直至队列空为止期望的时间复杂度O(ke)， 其中k为所有顶点进队的平均次数，可以证明k一般小于等于2，与[bfs](https://baike.baidu.com/item/bfs/542084" \t "https://baike.baidu.com/item/SPFA%E7%AE%97%E6%B3%95/_blank)算法比较，[复杂度](https://baike.baidu.com/item/%E5%A4%8D%E6%9D%82%E5%BA%A6/9716772" \t "https://baike.baidu.com/item/SPFA%E7%AE%97%E6%B3%95/_blank)相对稳定且小得多。但在稠密图中复杂度比[迪杰斯特拉算法](https://baike.baidu.com/item/%E8%BF%AA%E6%9D%B0%E6%96%AF%E7%89%B9%E6%8B%89%E7%AE%97%E6%B3%95/4049057" \t "https://baike.baidu.com/item/SPFA%E7%AE%97%E6%B3%95/_blank)差。

下面是算法的伪代码：

1. ProcedureSPFA;
2. Begin
3. initialize-single-source(G,s);
4. initialize-queue(Q);
5. enqueue(Q,s);
6. while not empty(Q) do begin
7. u:=dequeue(Q);
8. for each v∈adj[u] do begin
9. tmp:=d[v];
10. relax(u,v);
11. if(tmp<>d[v])and(not v in Q)then enqueue(Q,v);
12. end;
13. end;
14. End;

考虑到换乘时间的问题，存储状态的时候，不仅要存储他的车站编号，也要存储现在在哪一条线路的哪个方向上。因此状态数=车站数\*（线路数\*2）。

而为了实现对于次短路的寻找，我采用了一种比较玄学的方法。就是再搜索后面的线路的时候，尽可能避开前面已经经过的状态，因此我设置了一个规避系数avoid\_index，显然当avoid\_index大的时候两天线路之间的差异会比较大，反之则会有较多重合。经过一些数据的测试，我发现avoid\_index选为15是一种比较优秀的方案。

但是这种方法也存在问题，如果出现了两个相邻车站，并且两个车站不在一个环上，那么在我的算法里面，无论怎么设置avoid\_index，最终也只会找到一组解，因为程序实际上运行的还是最短路，而其他路线是严格包括最短路的。因此就需要去重操作。在去重的时候，我采用了一个类似HASH的方法，对两个路线的一些特征值进行比较。考虑到这个HASH是puzzle\_friendly的因此不太可能会撞，所以两个特征值相同的方案一定是相同的方案。

## 用户界面部分

因为同学的推荐，我使用了GTK+来搭建用户界面。因此如果运行环境上没有gtk+的话可能会出现错误。而且gtk+还附带了一个比较好用的编辑器glade，并且gtk3也内置了cairo，我主要就是通过这两个来完成用户界面的搭建的。

具体galde框架搭建可以见：

glade

图4-2 使用glade搭建的框架文件

之后在C里面实现的过程大概就是把控件从框架文件里面“抠”下来，这样的话就可以省掉很多定义格式、嵌套关系、参数等等。但是回调函数肯定还是要手写的。这个部分我主要是参照着gtk+的用户手册完成的。

考虑到文本框只能输出单行文本，但是输出推荐路线的内容是很多的，所以我选择了使用缓冲区+text\_view，但是也是有上限的，不过已经足够使用了。

绘图部分因为直接导入出现了一些问题，所以我采用的方式是先导出到一个图片文件里面，然后调用图片文件，导入到界面里面。

## 信息安全部分

正如我在前面所说的，我认为这个工程实际上并没有很多的需要保护的数据，几乎所有的数据全部都是很容易找到的。因此我把重心放在了保护代码上。因此我可以预先把.c文件与处理成.lib文件，而.h文件因为只有函数原型所以不是很需要保护。所以可以直接保留

# 五、系统实现

开发环境配置如下：

编辑器：Dev-C++ 5.4.2

GNU环境：MSYS2

C语言编译器：：Mingw64\_bit

GTK+版本 ：gtk+-3.0

关于GTK环境的配置可以参考https://www.gtk.org/ 中的相关内容，而且文末的参考资料里面也有大量的相关内容，主要是出现一些错误之后应该怎么修复。

主要宏，变量，函数原型及注释如下：

database.h

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. 函数名：lines\_that\_pass\_this\_station
3. 描述：查找有哪些线路通过本站
4. 函数原型： void lines\_that\_pass\_this\_station(int stationid);
5. 输入参数：站id
6. 输出参数：无，但是会修改 line\_in\_this\_station 数组，初始或不通过这个节点为-1，否则为最近一班车到达的时间 ，考虑到线路有负数，故加10存储
7. 日期：2021.3.4
8. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
9. **void** lines\_that\_pass\_this\_station(**int** stationid,**int** now\_time);
11. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
12. 函数名：change\_Crowdedness
13. 描述：修改一个线路一段时间的拥挤度（并非系数，系数需要经过计算）
14. 函数原型： int change\_Crowdedness(int now\_line,int frotime,int totime,double new\_crowdedness);
15. 输入参数： now\_line 要修改的线路 ,此处默认两个方向的拥挤度相同，因此没有偏移存储
16. fromtime totime 要修改的时间区间
17. new\_crowdedness 修改后的拥挤系数(0.0-1.0)
18. 添加了对传入参数越界和fromtime，totime大小反向的判断
19. 输出参数： 如果程序正确修改了，就返回1
20. 如果出现错误，就返回0
21. 日期：2021.3.5
22. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
23. **int** change\_Crowdedness(**int** now\_line,**int** fromtime,**int** totime,**double** new\_crowdedness);
25. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
26. 函数名：add\_edge
27. 描述：在两个站间添加有向边
28. 函数原型： void add\_edge(int frompoint,int topoint,double Crowding\_factor,int belonging\_line,double lenth)
29. 输入参数： 含义见英文，类型见函数定义
30. 输出参数：无
31. 日期：2021.2.25
32. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
33. **void** add\_edge(**int** frompoint,**int** topoint,**double** Crowding\_factor,**int** belonging\_line,**double** lenth);
35. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
36. 函数名：Name\_find\_station
37. 描述：使用汉语车站名找到它的id 但此处的复杂度为 O(number\_of\_station)
38. 函数原型： int Name\_find\_station(char \*stationname);
39. 输入参数：一个车站的名字 stationname
40. 输出参数：这个车站的序号 ，如果不存在就返回0
41. 日期：2021.2.28
42. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
43. **int** Name\_find\_station(**char** \*stationname);
45. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
46. 函数名：Calculate\_the\_fare
47. 描述：根据给出的公里数，计算分段票价。使用向上取整的方法。
48. 函数原型： int Calculate\_the\_fare(double total\_km);
49. 输入参数：一个路程的总公里数
50. 输出参数：收取的车票价格
51. 日期：2021.2.25
52. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
53. **int** Calculate\_the\_fare(**double** total\_km);
55. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
56. 函数名：get\_modified\_coefficient
57. 描述：计算当前边在当前时间，当前优先级类型下的权值修正系数
58. 函数原型：double get\_modified\_coefficient(struct Edge\_of\_subway \*now\_edge,int typ,int time);
59. 输入参数：当前边now\_edge，当前类型typ，当前时间 time
60. 其中typ == 1  表示不理会拥挤度，默认为1.0 找最少时间的方案
61. 如果typ == 2 就使用宽松、一般拥挤、拥挤的约定 （默认方法）
62. 如果typ == 3 就使用对拥挤系数取exp再取exp作为修正系数，这样可以保证会很大程度上规避拥挤，但是也不会出现因为等拥挤而等几个小时的问题
63. 如果typ == 4 就设定一个可以容忍的上线limit\_factor，不走拥挤系数大于他的边，但是由于没有传入这个上限，所以返回当前拥挤系数，在调用函数里判断是否不能走
64. 输出参数：修正系数 ,如果当前不可行就返回一个极大的数
65. 日期：2021.2.25
66. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
67. **double** get\_modified\_coefficient(**struct** Edge\_of\_subway \*now\_edge,**int** typ,**struct** Cost\_of\_scheme \*now\_status);
69. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
70. 函数名：Calculate\_the\_cost
71. 描述：计算当前边在当前时间，当前优先级类型下的消耗，再结合传进来的之前状态，获得走这条边之后的状态
72. 函数原型： struct Cost\_of\_scheme Calculate\_the\_cost(struct Edge\_of\_subway \*now\_edge,struct Cost\_of\_scheme \*now\_situation,int typ);
73. 输入参数： 当前边now\_edge，当前状态now\_situation
74. 对于typ:
75. case 1，2，3，4的含义见get\_modified\_coefficient 函数前的注释
76. case 5 最低票价（最小里程）
77. case 6 最少换乘下最低票价
78. 输出参数：返回走这条边之后的状态
79. 日期：2021.3.5
80. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
81. **struct** Cost\_of\_scheme Calculate\_the\_cost(**struct** Edge\_of\_subway \*now\_edge,**struct** Cost\_of\_scheme \*now\_status,**int** typ);
83. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
84. 函数名：if\_scheme\_equal
85. 描述：判断两个方案是不是相同
86. 函数原型： int if\_scheme\_equal(struct Cost\_of\_scheme a,struct Cost\_of\_scheme b);
87. 输入参数：两个  Cost\_of\_scheme型变量
88. 输出参数：0 不相等
89. 1 相等
90. 日期：2021.3.5
91. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
92. **int** if\_scheme\_equal(**struct** Cost\_of\_scheme a,**struct** Cost\_of\_scheme b);



97. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
98. 函数名：init\_station
99. 描述：导入车站信息，到结构体数组Stations里
100. 函数原型： void init\_station(void)
101. 输入参数：无
102. 输入文件： ./data/stationinfo.txt   里面应该是所有站点（无重复）的信息（名称）
103. 输出参数：无
104. 日期：2021.3.5
105. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
106. **void** init\_station();
108. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
109. 函数名：init\_line
110. 描述：导入线路信息信息，到结构体里。
111. 函数原型： void init\_line(void)
112. 输入参数：无
113. 输入文件： ./data/subwayline.txt   里面应该是所有线路的信息，包括长度、线路经过站点个数、容量
114. 输出参数：无
115. 日期：2021.2.25
116. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
117. **void** init\_line();
119. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
120. 函数名：init\_Crowdedness
121. 描述：初始化各个线路在每个时间段上的拥挤系数。采用简化约定中的结果
122. 函数原型： void init\_Crowdedness()；
123. 输入参数：无
124. 输出参数：无
125. 日期：2021.3.5
126. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
127. **void** init\_Crowdedness();
129. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
130. 函数名：init\_coordinates
131. 描述：输入各个车站的经纬度作标
132. 函数原型： void init\_coordinates()；
133. 输入参数：在./data/Coordinates.txt中输入车站的经纬度坐标
134. 输出参数：无
135. 日期：2021.3.10
136. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
137. **void** init\_coordinates();
139. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
140. 函数名：init
141. 描述：在main.c里面调用这个函数，在从这个函数来调用其他初始化函数，实现初始化过程。本函数是一个控制函数。
142. 函数原型： void init()
143. 输入参数：无
144. 输出参数：无
145. 日期：2021.2.25
146. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
147. **void** init();


151. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
152. 函数名：Traverse\_in\_line
153. 描述：在当前状态下，沿当前线路移动一站 。返回的是移动后的状态
154. 函数原型： struct Cost\_of\_scheme Traverse\_in\_line(struct Cost\_of\_scheme now\_status,int aim\_line);
155. 输入参数： now\_situation 当前状态，主要是车站id和时间以及路程
156. aim\_line 要移动的列车线路，其中正负表示不同的方向。
157. 输出参数： 移动一条边之后的状态
158. 日期：2021.3.2
159. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
160. **struct** Cost\_of\_scheme Traverse\_in\_line(**struct** Cost\_of\_scheme now\_status,**int** aim\_line);
162. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
163. 函数名：spfa
164. 描述：使用spfa算法计算两点间的最短路，最短指使得maincost最小。 设计最短路状态的时候，使用目前站+目前线路来确保找到最优。
165. 并且引进了avoid\_status来减少走重复方案的可能
166. 函数原型： int spfa(struct Cost\_of\_scheme now\_status,int typ);
167. 输入参数：now\_status 最初的状态
168. typ 寻找最短路的模式
169. 输出参数：1 表示运行到了结尾
170. 日期：2021.3.5
171. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
172. **int** spfa(**struct** Cost\_of\_scheme now\_status,**int** typ);
174. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
175. 函数名：Find\_the\_path
176. 描述：通过在spfa里的存储转移方案的指针，反向找到整个路径上的所有状态
177. 函数原型： int Find\_the\_path(int aimstation,int pls);
178. 输入参数：aimstation 目标车站的编号，最终要从这里倒序找出整个路径
179. 输出参数：返回当前路径的长度 ，如果没有路径就返回0
180. 日期：2021.3.5
181. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
182. **int** Find\_the\_path(**int** aimstation,**int** pls);
184. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
185. 函数名：print\_the\_path
186. 描述：输出一条寻路方案
187. 函数原型： int print\_the\_path(int pls,int typ);
188. 输入参数：pls 输出的这个方案所存储的位置
189. typ 这个方案的寻路类型
190. 输出参数：正常结束输出0
191. 日期：2021.3.5
192. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
193. **int** print\_the\_path(**int** pls,**int** typ);
195. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
196. 函数名：find\_shortest\_path
197. 描述：寻路算法的总控制函数，在这个函数里面
198. 函数原型： int find\_shortest\_path(int aimstation,struct Cost\_of\_scheme now\_status,int typ);
199. 输入参数：aimstation 目标车站的编号，最终要从这里倒序找出全部路径
200. now\_situation 初始的状态，重要的是当前车站id和时间 ，其他的都应该清零
201. typ  寻路的类型
202. 输出参数：在屏幕上完成多条路线的输出
203. 日期：2021.3.5
204. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
205. **int** find\_shortest\_path(**int** aimstation,**struct** Cost\_of\_scheme now\_situation,**int** typ);

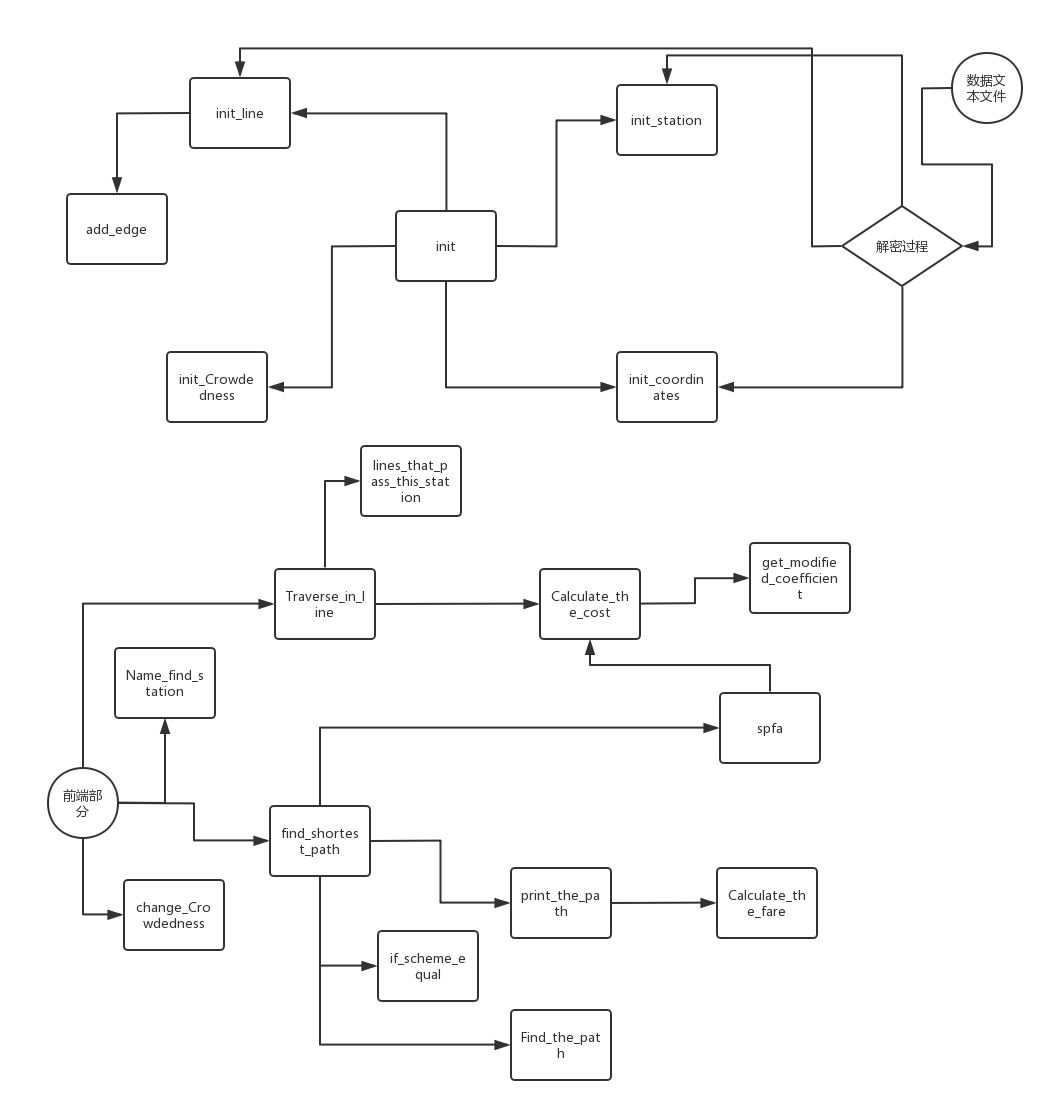


图5-1 后端运行中的函数调用关系图

# 六、运行测试与结果分析（黑体小2加粗居中）

GUI下的界面如图所示：

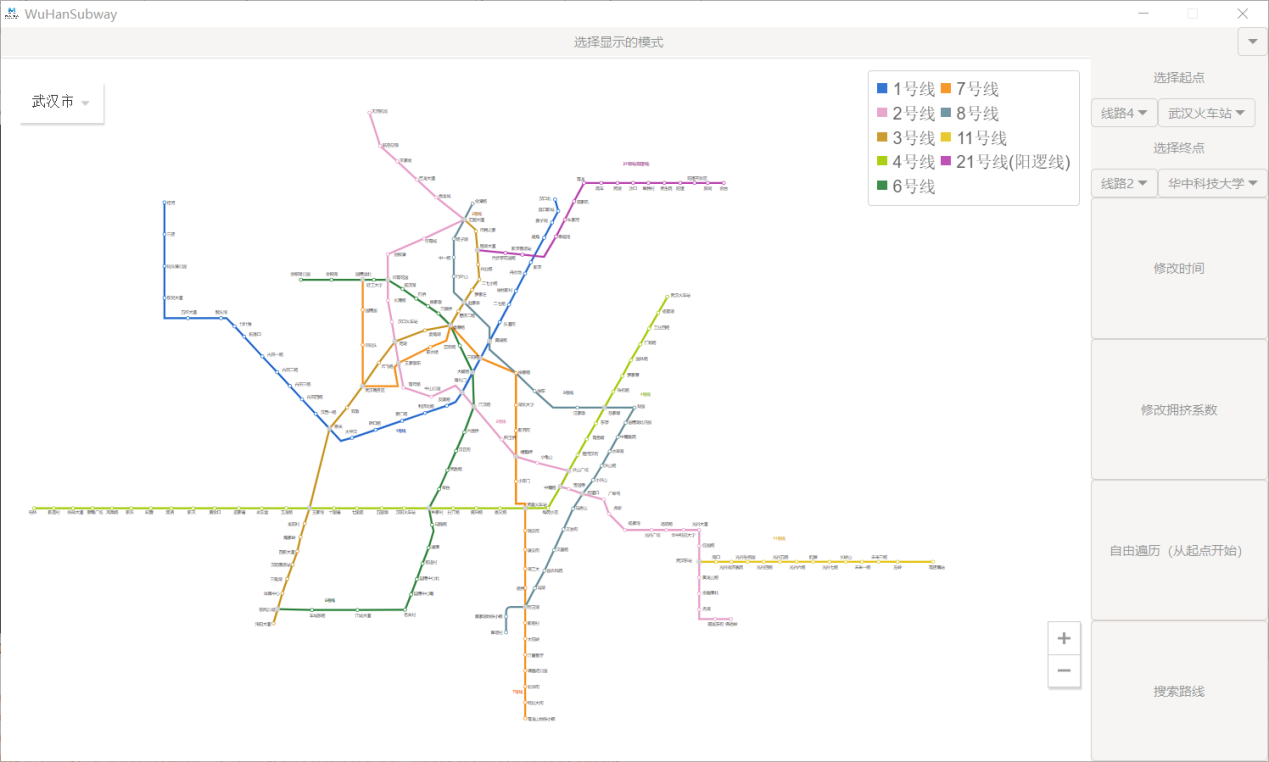


图6-1 基础GUI界面

除了基础窗口之外其他窗口如图：

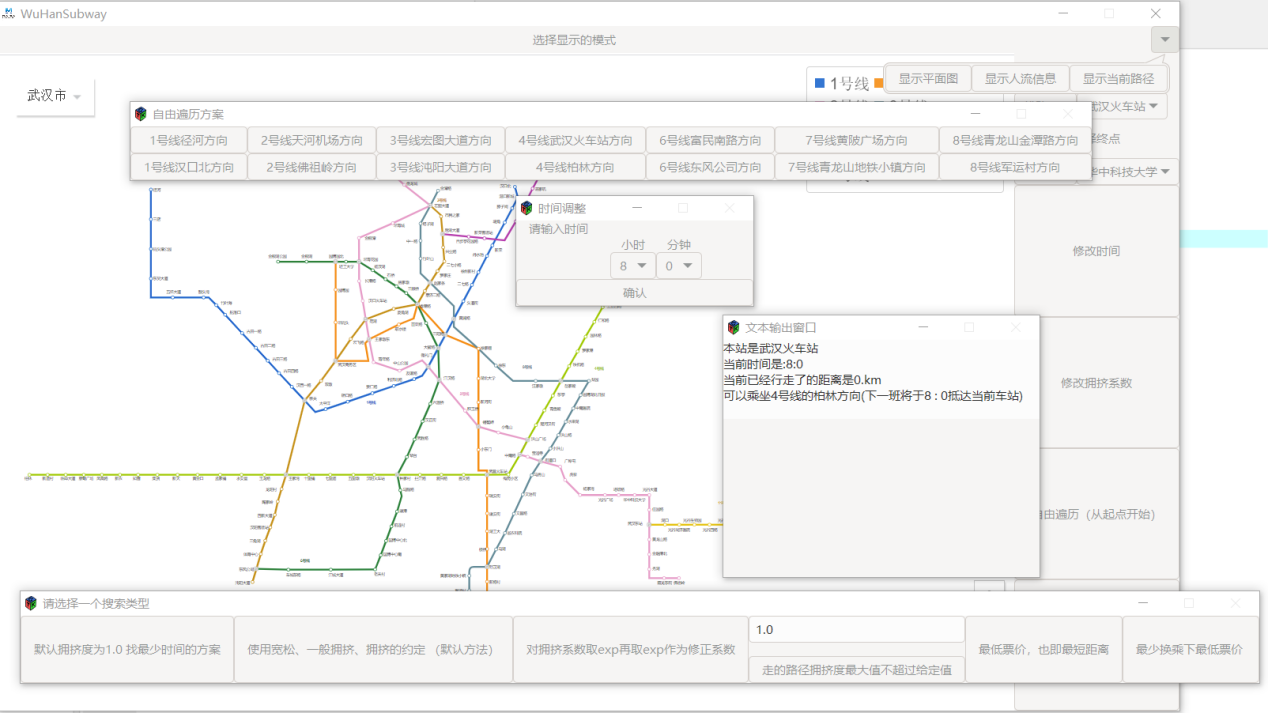


图6-2主要GUI窗口界面一览

具体的使用方法，请见附件三中的使用说明。

首先系统提供了拥挤度图像化显示，是使用的红到黄到绿，因此表示拥挤度从高到低。效果如图：

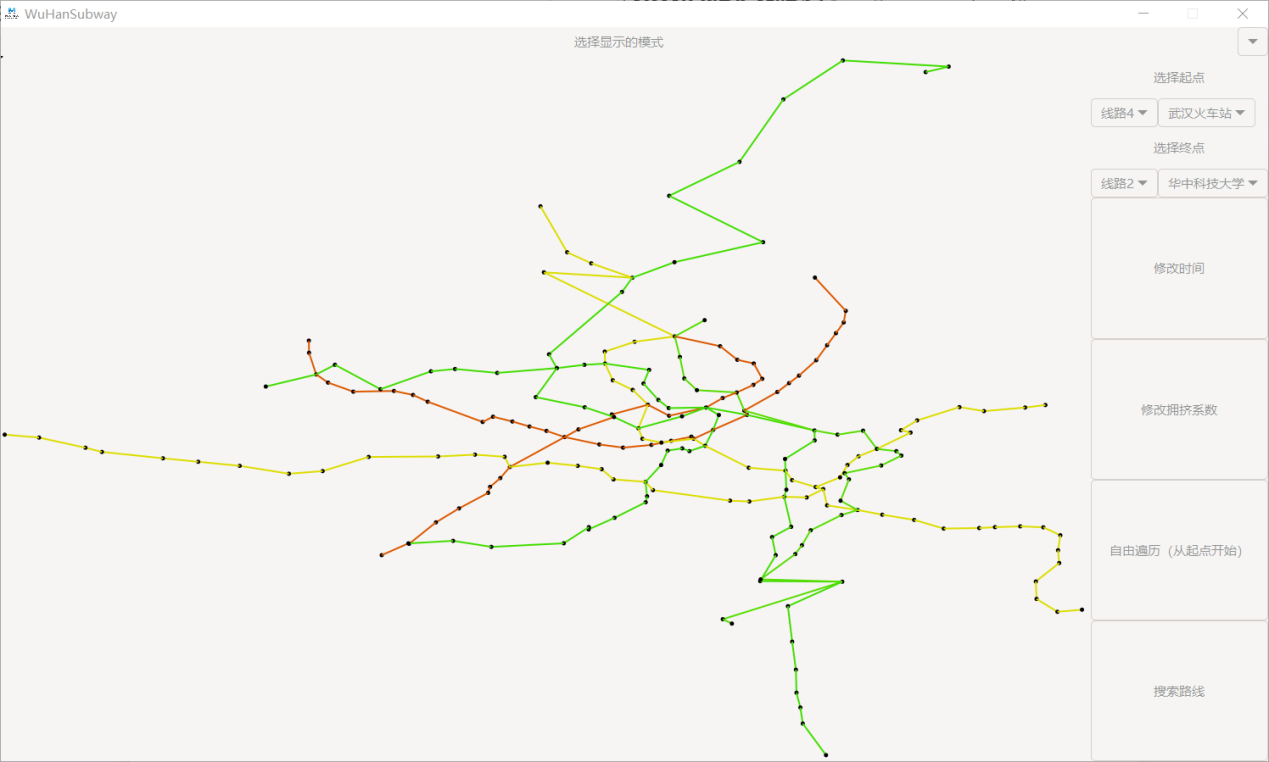


图6-3 拥挤度可视化效果图

修改2号线在当前时间下的拥挤度为1.0之后的效果：

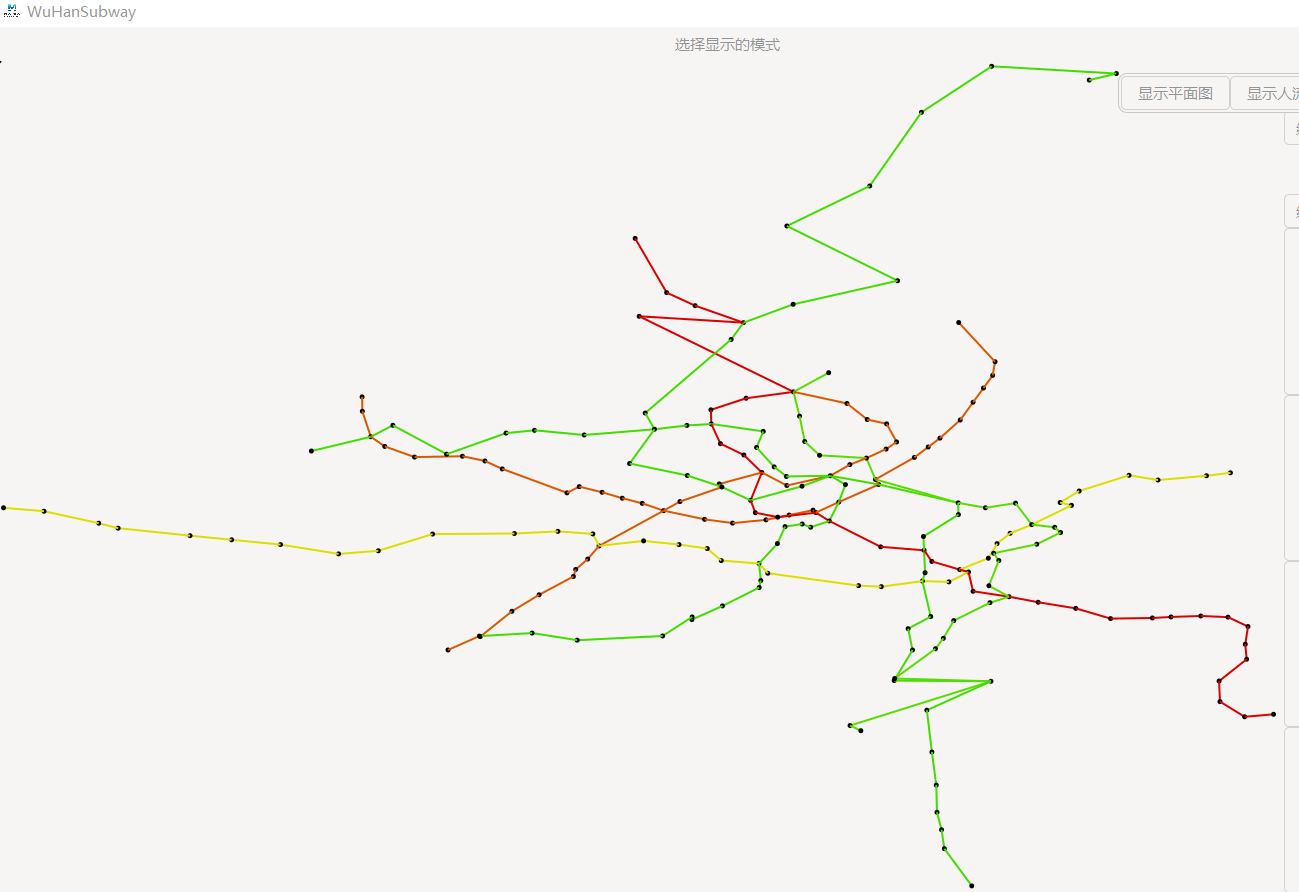


图6-4 拥挤度可视化效果图

修改时间到2号线1.0拥挤度区间之外，图片即同图6-3。

对于自由遍历部分，尝试从武汉火车站到华中科技大学，出发时间为8:00，如图：



图6-5 自由遍历效果图

程序可以预测下一班车的时间，但注意到因为存在换乘时间，实际上下一班车不一定是可以赶得上的。如图：



图6-6 预测班车效果图

推荐最短路，使用实例1，即2号线的宝通寺到6号线的石桥，效果如图（并且展示了最短路线的地图，选择展示哪一个可以通过右上角控制选择），此时已经按照体感时间排序，可通过切换宏来实现按照最小票价排序，并且已经显示了大部分的维护信息（包括拥挤度，时间，路程/票价，换乘站等）：



图6-7 寻找最短路效果图

然后考虑两个特殊情况。

1. 只有一条路线可以到达，其他路线都会包含此条路线。见图6-8：
2. 因为时间不符或者拥挤度不符而不能到达或延迟到达。见图6-9，图6-10（本图是修改了8:00后一段时间内2号线的拥挤度为1.0，同时寻路时设置容忍上线为0.9）：



图6-8 特殊情况图1



图6-9 特殊情况图2

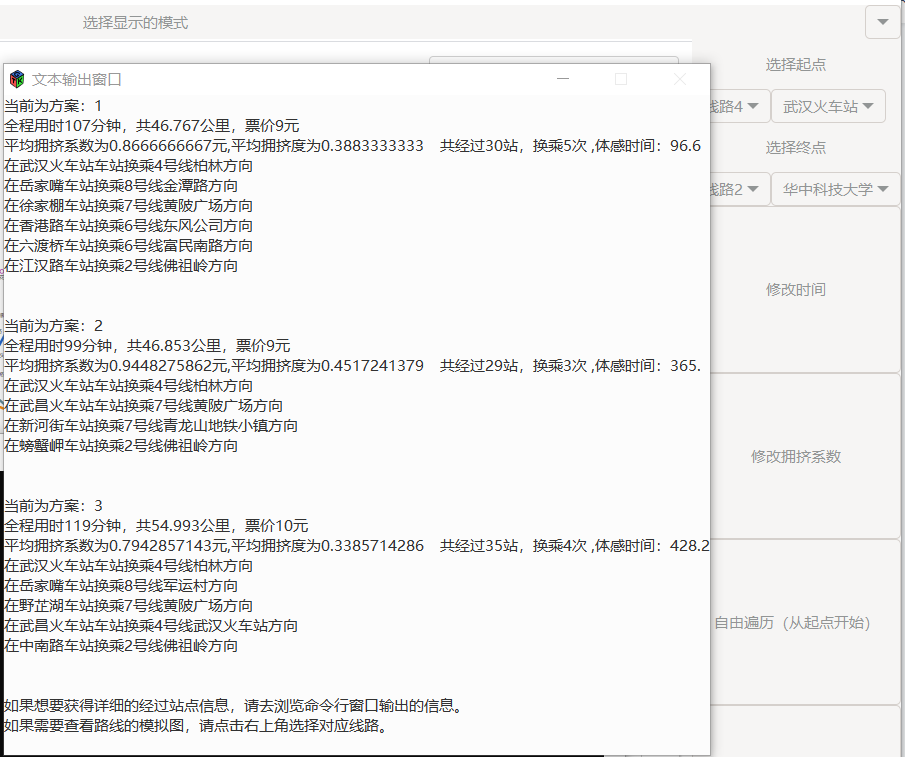


图6-10 特殊情况图2

图6-10 注，正是因为2号线在8；00到9；00间是不能通行的，因此推荐路线才会多换乘了很多次，最后乘上比较早的2号线班次。之所以不停留在某一站而是乱转，是因为其他线路不但可以通行，而且他们的修正系数都是小于1的，因此一个体感时间的最短路应该包含尽可能多的乘车部分。故出现此路线。

## 复杂度分析

在数据从数据文件读入（初始化）的时候，复杂度和文件大小是正相关的，这是因为全过程中文件指针并没有出现跳跃或者回到前面的情况。同样的GUI界面初始化也是常数的。除此之外初始化拥挤度数组的复杂度是O(60(min/h)\*24(h/day)\*8(线路)\*2(线路方向))的。

在接受前端的站点信息的时候，需要将站点名称转化为站点的id，此过程最坏情况相当于遍历所有站点，复杂度O(N（站点数）)，但是显然可以使用排序二分法、平衡树法、HASH法等来优化，但考虑N仅仅几百，所以优化在实际上效果可以忽略。

理论上SPFA算法的最坏时间复杂度是O(NM)，N为图中节点个数，M为图中边的个数，它的策略是将每个成功松弛过的点不重复的放入队列，但事实上，根据图的结构的不同，每个节点入队的次数是无法确定的，只知道在连通图中，它的入队次数大于等于1小于M,所以SPFA算法的时间复杂度是在O(N)~O(NM)不等，也即是O(kE)，k为一个1-M的常数。但是实际上，因为本图算是一个稀疏图，因此spfa中的常数k应该是很小的，在测试中，几乎所有情况下，他小于log(N)，因此在本问题中，可以认为spfa的复杂度~O(Nlog(N))

输出方案的复杂度近似于输出字符串的长度，实际上因为换乘似乎不会太多，字符串就不会太长。而绘图之类的操作，复杂度等于路线经过站点数<字符串长度，因此输出的总复杂度是O(|temporary\_string|)。七、总结

这次课程设计应该说是我第一次独立设计一个大工程，原来写过最大的也就是数独这种的。确实对于我来说是一个很大的挑战，再加上我寒假的时候大意了，没有写。这个挑战就又更大了。

相比于我认识的几个同学，我觉得我写这份工程的时候心态的变化还是比较平稳的，这应该八成归功于我一边写一边写注释，以及尽可能拆分模块的思想，不仅降低了出错概率而且能够极大的方便我进行调错，这应该说是一种工程的思维而不仅仅是科学的思维，我现在在做的更像一个码农而不是一个科学家。用一种尽可能平静的心态去学习和写代码，而不是和自己怄气。

其实注意到如果仅仅要求实现前90分的内容，我的总工作量应该比现在的少了很多。从代码量上来看，应该少了一半又不止，而且因为GTK+对于我完全是一个全新的内容，虽然也是用C，但是实际上还是大相径庭的。而且似乎因为我电脑的一些配置被我改过，所以安装GTK的时候还是费了不少功夫，甚至一度让我想放弃写用户界面。我一直担心一种情况，就是用户界面中我想要的一个功能并不能通过gtk实现，这样如果我前面写了很多，但是最后却被卡住，不能全部实现，这也很难受。但是我最终还是决定去做，一方面是为了完成作业，另一方面也是为了能够让自己多学一点，多去了解一点，以免毕业直接失业。

至于为什么选择gtk而不是其他的方法，应该说我是被我的语文阅读能力，我的同学坑了。我理解的题目要求是全部用C实现，结果就直接放过了qt等等一系列的语言，选择了一个比较难而且没有基础的语言。另一方面就是我的同学以为我要重新造轮子，所以只给我大概讲了下GTK最基本的东西，并没有告诉我还有很多别人做好的包可以拿来用，甚至是别人写好了的界面。不过归根到底还是因为我自己确实水平不够，而且了解的不多。

因为我从来没有了解过gtk，所以这次在写gui的时候，我是边学边写的。这次学习新语言和之前是完全不一样的，之前的时候，我都是依据某一本教材或者某个顺序，去由浅入深的学习。这样对底层实现的了解肯定会更清晰一些，但是这对于一个两周就要上交的作业来说太奢侈了。因此我这次主要是由需求去找方法，能实现就行，不求甚解 这样的方法去完成我的gui部分。所以要特别感谢一下https://developer.gnome.org/ 上的gtk+帮助手册，正是借助这上面的介绍和大量的博客，我才能完成我的大作业。

而且这次编程实践，也让我深深的认识到了设计的重要性。按照一个预先设计好的方案来编程是事半功倍的，不仅使得思路更明晰，写的更快。最重要的是能够让我几乎不“回滚”代码，而且代码间的关系也更加清晰。如果说有什么美中不足的，就是在我设计的时候，并没有直接写到实验报告里面，而是仅仅靠脑补。如果我能提前做出结构图，流程图等等，肯定可以让我的效率更高。

最后一个我想要说的收获就是，虽然作业是以个人为单位的，但是如果有一些同学、朋友能够互通有无，教学相长的话，是非常好的。因为大家面对的问题是相同的，甚至解法也有相似之处，这时候交流可以促进共同提升。孔子云:“不愤不启，不悱不发。”相对的，当一个人“愤”，“悱”的时候也确实需要一点启发。或许没有人启发，在学习的时候就会走上歧途，浪费很多时间，甚至直接走向错误的理解。比如我在学习gtk的时候就有很多是向大佬们询问学习的，如果没有大佬们的帮助，我可能理解回调函数就理解错了，然后就可能出现程序的问题了。

如果说这次综合课程设计最大的问题，我想应该是拖延症了。暂且不说为什么我寒假仅仅安了各环境，几乎什么都没写。就是开学以后，上课和课下我的进度也是十分的缓慢，我印象里我的一个室友去找老师检查的时候，我的程序还不能通过编译。我觉得这主要是因为我太懒了以致于几乎没有花课余时间去写代码，最多做一些改错和注释的工作。不过好在有惊无险，基本我的所有事前的设想都按时完成了，虽然我是在最后一堂课才去找老师检查的，而且在检查前的10min的时候，我还在改一个莫名其妙的bug。之后写实验报告也是一拖再拖，我写下这行文字的时候，距离提交平台结束提交也只有不到半天。（下次一定不拖.jpg）

总的来说，这次还是收获满满的，从态度、思维到技术，我都感觉到了很大的进步，而且也感受到了和大佬们在设计、技术和码力上的差距。我完全没想到只有4分的GUI竟然几乎所有同学都写了，而且不少同学花在GUI上的精力甚至超过核心部分。因此我更加怀疑我投入的时间是不是和收获成正比。

# **附录一、参考文献**

<https://www.gtk.org/docs> gtk官网

<https://github.com/wingtk/gvsbuild> gvsbuild资料

<https://docs.python.org/zh-cn/3/contents.html> Python文档内容

https://yunyaniu.blog.csdn.net/article/details/90238729?utm\_medium=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control&dist\_request\_id=0be47993-7d82-42db-800d-5484df54d22f&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control GTK+：GTK+的简介、安装、使用方法之详细攻略

<https://www.dazhuanlan.com/2019/12/09/5dedab60ca269/> Windows平台下安装GTK

<https://blog.csdn.net/qq_43037374/article/details/83343604> Windows下通过Dev-C++配置GTK+3.0编程环境

https://blog.csdn.net/qize9956/article/details/55223047?utm\_medium=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-5.control&dist\_request\_id=5a6def11-a29c-4586-86a8-a7570e481017&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-5.control C语言 + GTK3+ Visual Studio 2015环境配置 以及第一个GTK界面程序

<https://blog.csdn.net/drsonxu/article/details/86596550> GTK+系列---Windows下的GTK+开发平台搭建（Win7 64位）

<https://blog.csdn.net/weixin_38184741/article/details/111316014> Linux:更新 /usr/share/glib-2.0/schemas 目录

<https://blog.csdn.net/pfysw/article/details/81190404> GTK+ 3 基础知识学习

<https://blog.csdn.net/ygyoe/article/details/79654057> GTK+入门教程

<https://blog.csdn.net/lianghe_work/article/details/47041153> Linux之GTK系列教程

<https://blog.csdn.net/tennysonsky/article/details/42778635> GTK帮助文档的使用

<https://houye.xyz/2015-02/gtkresource/> Gtk+2.0 Gtk+3.0文档，手册，资源分享

<https://www.bookstack.cn/read/GTK-3-Chinese-Reference-Manual/README.md> [GTK+3 参考手册](https://www.bookstack.cn/books/GTK-3-Chinese-Reference-Manual" \o "GTK+3 参考手册)

<https://developer.gnome.org/gtk-tutorial/2.90/> GTK+ 2.0 Tutorial

<https://developer.gnome.org/references> API 参考

<https://developer.gnome.org/gtk3/stable/> GTK+ 3 Reference Manual

<https://www.cnblogs.com/cy163/archive/2007/06/16/785341.html> [Gtk对于通常的gui程序，大家想做的事就是做一点事件处理(包括各种计算、文件操作等)，然后在界面上显示出来](https://www.cnblogs.com/cy163/archive/2007/06/16/785341.html)

<https://www.cnblogs.com/zjutzz/p/10959816.html> [基于gtk的imshow:用gtk读取并显示图像](https://www.cnblogs.com/zjutzz/p/10959816.html)

<https://blog.csdn.net/lianghe_work/article/details/47103039?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_title-0&spm=1001.2101.3001.4242> GTK常用控件之图片控件( GtkImage )

https://blog.csdn.net/lianghe\_work/article/details/47152271?ops\_request\_misc=&request\_id=&biz\_id=102&utm\_term=gtk%20%20%E7%94%BB%E5%9B%BE&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduweb~default-0-47152271.first\_rank\_v2\_pc\_rank\_v29\_10 GTK进阶学习：绘图事件

<https://blog.csdn.net/zegeyzz/article/details/14891737> GTK常用接口——行编辑、复选框、下拉框

<https://blog.csdn.net/tennysonsky/article/details/43019923> GTK入门学习：glade的使用

<https://huanhaoadam.wordpress.com/2009/06/23/linux%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0%EF%BC%88%E4%B8%89%E5%8D%81%E4%BA%8C%EF%BC%89%EF%BC%8D%EF%BC%8Dgtk%E4%B8%AD%E7%9A%84%E4%B8%8B%E6%8B%89%E8%8F%9C%E5%8D%95/> [Linux学习笔记（三十二）－－GTK中的下拉菜单](https://huanhaoadam.wordpress.com/2009/06/23/linux%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0%ef%bc%88%e4%b8%89%e5%8d%81%e4%ba%8c%ef%bc%89%ef%bc%8d%ef%bc%8dgtk%e4%b8%ad%e7%9a%84%e4%b8%8b%e6%8b%89%e8%8f%9c%e5%8d%95/)

<https://www.cnblogs.com/rain-blog/p/gtk-menu.html> [GTK 菜单的创建详解](https://www.cnblogs.com/rain-blog/p/gtk-menu.html)

<https://stackoverflow.com/questions/55105682/gtkwidget-aka-struct-gtkwidget-has-no-member-named-window> [‘GtkWidget {aka struct \_GtkWidget}’ has no member named ‘window’](https://stackoverflow.com/questions/55105682/gtkwidget-aka-struct-gtkwidget-has-no-member-named-window)

https://www.baidu.com/s?ie=utf-8&f=8&rsv\_bp=1&ch=&tn=baidu&bar=&wd=gtk\_text\_buffer\_emit\_insert%3A+assertion+%27g\_utf8\_validate+%28text%2C+len%2C+NULL%29%27+failed&oq=gtk\_text\_buffer\_emit\_insert%253A%2520assertion%2520%2526%252339%253Bg\_utf8\_validate%2520%28text%252C%2520len%252C%2520NULL%29%2526%252339%253B%2520f&rsv\_pq=ad74a9ef0004985e&rsv\_t=4150xLngT%2FX5T4k0RltyENX4u0g969XzgZdSBKhk4ZyDK8FQhTEF4Tq3yGw&rqlang=cn&rsv\_enter=1&rsv\_btype=t&rsv\_dl=tb&inputT=3512 错误信息

<http://blog.sina.com.cn/s/blog_bd0647d80102vyb4.html> GTK多行文本的使用

https://blog.csdn.net/zegeyzz/article/details/14894113?utm\_medium=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control&dist\_request\_id=1328627.10415.16153613702754635&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control   GTK常用接口——绘图

<https://blog.csdn.net/learnerz/article/details/7200860> GTK简单画图程序

<https://blog.csdn.net/sjin_1314/article/details/17634111> GTK编程基础----状态栏、文本输入构件、组合框

https://blog.csdn.net/lastking/article/details/66513?utm\_medium=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control&dist\_request\_id=1328603.59620.16152092989403163&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control GTK+2.0 中的容器控件与布局技巧

<https://blog.csdn.net/a87b01c14/article/details/52100193?spm=1001.2014.3001.5502> PyGobject（九十三）Cairo系列——简介&简单示例

<https://www.jianshu.com/p/854756d3f54d> GTK+3系列教程6：UI神器Glade

<https://blog.csdn.net/a87b01c14/article/details/52035551> PyGobject（十二）布局容器之Button篇——Gtk.MenuButton

<https://blog.csdn.net/a87b01c14?type=blog> sanxiaochengyu blog

<https://baike.baidu.com/item/itoa/4747365?fr=aladdin> itoa

<https://www.jb51.net/article/71485.htm> 对比分析C语言中的gcvt()和ecvt()以及fcvt()函数

<https://www.cnblogs.com/bluestorm/p/3168719.html> [C语言itoa()函数和atoi()函数详解(整数转字符C实现)](https://www.cnblogs.com/bluestorm/p/3168719.html)

<https://blog.csdn.net/nanfeibuyi/article/details/80811498> C语言 整数与字符串的相互转换

<http://c.biancheng.net/c/strcat.html> C语言strcat()函数：字符串连接（拼接）

<https://blog.csdn.net/u013693952/article/details/93460908> C语言rewind()函数：将文件指针重新指向文件开头

<https://blog.csdn.net/yang889999888/article/details/73442356> 大整数运算库gmp安装及使用

<https://blog.csdn.net/tigerisland45/article/details/51458507> C语言实现的RSA算法程序（使用GMP）

<https://blog.csdn.net/violet_echo_0908/article/details/51176497> RSA算法详解及C语言实现

<https://blog.csdn.net/leumber/article/details/80451223> RSA算法C语言实现

<https://bbs.csdn.net/topics/80076243> 在gtk中如何使一个button可见而不可用!请高手指点!

<https://www.runoob.com/git/git-workflow.html> Git 教程

<http://www.codeinword.com/> 代码高亮块

# 附录二、主要源程序片段

宏及主要数据结构定义部分：

1. #define lenth\_of\_station\_name 50
2. #define total\_number\_of\_station 330
3. #define total\_number\_of\_edge 3000
4. #define total\_number\_of\_line 25
5. #define max\_status 200000
6. #define inf (10000.0)
7. #define Inf (100000000.0)
8. #define Bidirectional\_line\_offset 10
9. #define avoid\_index 15
10. #define trans\_time(x,y) (x\*60+y)
11. #define eqs  (0.0000001)
12. #define Image\_Max\_X  1285
13. #define Image\_Max\_Y   832
14. #define eexp(x) exp(exp(x))
15. #define can\_you\_reach\_here    printf("Hello?\n")
16. #define outt(x) printf("%s = %d\n",getVariableName(x),x);
17. #define hh  puts("")
18. #define my\_swap(a,b) (a)^=(b)^=(a)^=(b)
19. #define getVariableName(x)  #x
20. #define ANGLE(ang)  (ang \* acos(-1.0) / 180.0)
21. //#define debug                    //通过定义此宏来输出调试信息
22. #define use\_gui                    //通过定义此宏来开启GUI界面
23. //#define  Long\_debugging\_information      //通过定义此宏来将一些调试信息输出到文件./data/Long\_debugging\_information.txt里面
24. //#define gui\_print\_detailed\_information              //通过定义此宏来输出详细的线路通过站点信息到gui中的文本输出框  ，内容很长
25. #define maincost\_ascending          //当定义这个宏的时候通过maincost的最小来完成输出路线排序，否则使用路程最短来排序
27. **struct** Station\_unit{          //每个结构体变量存储一个站点的信息
28. **char** name[lenth\_of\_station\_name];    //名称，使用ANSI
29. **int** hed\_of\_edge,number\_on\_line[total\_number\_of\_line];
30. //第一条出边：hed\_of\_edge 和 当前站点在各个线路上的位次number\_on\_line
31. **double** longitude,latitude;           //经纬度
32. }Stations[total\_number\_of\_station];
34. **struct** Edge\_of\_subway{       //描述一条边的信息
35. **int** next\_edge,time,belonging\_line,to\_station;    //belonging\_line分正负，表示一条线路的两种方向
36. **double** lenth,Crowding\_factor;
37. }Edge[total\_number\_of\_edge];
39. **struct** Subway\_line{     //描述一条线路的总体信息
40. **char** first\_endpoint[lenth\_of\_station\_name],second\_endpoint[lenth\_of\_station\_name];   //两端点车站名
41. **char** stations[total\_number\_of\_station][lenth\_of\_station\_name];    //车站名数组
42. **int** Maximum\_load,total\_station;
43. **double** total\_lenth;
44. }Subwaylines[total\_number\_of\_line];
46. **struct** Cost\_of\_scheme{    //描述一个状态，是整个程序运作最重要的结构体
47. **int** nowtime,number\_of\_transfer,number\_of\_station,nowline,starttime,now\_station;
48. **double** dist,aver\_crowd,maincost,limit\_factor,tot\_crowd;
49. //此处的aver\_crowd表示的是平均拥挤系数，考虑到方便计算，在中间状态的时候，实际上存储的是前面经过所有边的拥挤系数之和
50. // tot\_crowd表示平均拥挤度，其他同上
51. **struct** Cost\_of\_scheme \*last\_status;
52. //存储转移到本状态的上一个状态
53. };
55. **int** line\_in\_this\_station[total\_number\_of\_line],avoid\_status[total\_number\_of\_station][total\_number\_of\_line];
56. **struct** Cost\_of\_scheme station\_dis[total\_number\_of\_station][total\_number\_of\_line],passing\_situation[5][(total\_number\_of\_station<<1)+100];
57. **double** Crowdedness[total\_number\_of\_line][total\_number\_of\_edge];
58. **FILE** \*Long\_debugging\_information;

关键算法部分：

(1):

1. **int** find\_shortest\_path(**int** aimstation,**struct** Cost\_of\_scheme now\_status,**int** typ){
2. //  can\_you\_reach\_here;
3. memset(avoid\_status,0,**sizeof**(avoid\_status));
4. **struct** Cost\_of\_scheme initial\_status=now\_status;
5. spfa(initial\_status,typ);
7. **int** cnt1=Find\_the\_path(aimstation,1),i;
8. #ifdef debug
9. hh;puts("方案1:");print\_the\_path(1,typ);
10. #endif
11. **for**(i=1;i<cnt1;++i){              //走到这些状态，就要加代价，以此来找新的最短路 ，后面记得删除掉
12. avoid\_status[passing\_situation[1][i].now\_station][passing\_situation[1][i].nowline+Bidirectional\_line\_offset]=avoid\_index;
13. }
15. spfa(initial\_status,typ);
17. **int** cnt2=Find\_the\_path(aimstation,2);
18. #ifdef debug
19. hh;puts("方案2:");print\_the\_path(2,typ);
20. #endif
21. **for**(i=1;i<cnt2;++i){              //走到这些状态，就要加代价，以此来找新的最短路 ，后面记得删除掉
22. **if**(avoid\_status[passing\_situation[2][i].now\_station][passing\_situation[2][i].nowline+Bidirectional\_line\_offset]==0)
23. avoid\_status[passing\_situation[2][i].now\_station][passing\_situation[2][i].nowline+Bidirectional\_line\_offset]+=avoid\_index/2;
24. avoid\_status[passing\_situation[2][i].now\_station][passing\_situation[2][i].nowline+Bidirectional\_line\_offset]+=avoid\_index/2;
25. }
27. spfa(initial\_status,typ);
28. **int** cnt3=Find\_the\_path(aimstation,3);
29. #ifdef debug
30. hh;puts("方案3:");print\_the\_path(3,typ);
31. #endif


35. **int** min\_dis\_pls=1;
36. **for**(i=1;i<=3;++i){
37. #ifdef maincost\_ascending
38. **if**(passing\_situation[i][1].maincost<=passing\_situation[min\_dis\_pls][1].maincost) min\_dis\_pls=i;
39. #else
40. **if**(passing\_situation[i][1].dist<=passing\_situation[min\_dis\_pls][1].dist) min\_dis\_pls=i;
41. #endif
42. }
43. #ifdef debug
44. outt(min\_dis\_pls);
45. #endif
47. puts("推荐路线依次如下：");
48. print\_the\_path(min\_dis\_pls,typ);
50. **int** mid\_dis\_pls=1+((min\_dis\_pls==1)?1:0);
51. **for**(i=1;i<=3;++i){
52. #ifdef maincost\_ascending
53. **if**(passing\_situation[i][1].maincost<=passing\_situation[mid\_dis\_pls][1].maincost&&i!=min\_dis\_pls) mid\_dis\_pls=i;
54. #else
55. **if**(passing\_situation[i][1].dist<=passing\_situation[mid\_dis\_pls][1].dist&&i!=min\_dis\_pls) mid\_dis\_pls=i;
56. #endif
57. }
58. #ifdef debug
59. outt(mid\_dis\_pls);
60. #endif
61. **if**(if\_scheme\_equal(passing\_situation[mid\_dis\_pls][1],passing\_situation[min\_dis\_pls][1])==0)
62. print\_the\_path(mid\_dis\_pls,typ);
63. **int** max\_dis\_pls=6-min\_dis\_pls-mid\_dis\_pls;
64. #ifdef debug
65. outt(max\_dis\_pls);
66. #endif
67. **if**(if\_scheme\_equal(passing\_situation[max\_dis\_pls][1],passing\_situation[min\_dis\_pls][1])==0&&if\_scheme\_equal(passing\_situation[mid\_dis\_pls][1],passing\_situation[max\_dis\_pls][1])==0)
68. print\_the\_path(max\_dis\_pls,typ);
70. **return** 0;
71. }
72. :
73. **int** spfa(**struct** Cost\_of\_scheme now\_status,**int** typ){
74. **struct** Cost\_of\_scheme next\_status;
75. **int** hed=0,endd=1,i,j;
76. **for**(i=0;i<total\_number\_of\_station;++i)
77. **for**(j=0;j<total\_number\_of\_line;++j)
78. station\_dis[i][j].maincost=Inf,vis[i][j]=0;
80. my\_queue1[endd]=now\_status.now\_station;my\_queue2[endd]=now\_status.nowline+Bidirectional\_line\_offset;
81. vis[my\_queue1[endd]][my\_queue2[endd]]=1;station\_dis[my\_queue1[endd]][my\_queue2[endd]]=now\_status;
82. **while**(hed<endd){
83. hed++;
84. now\_status=station\_dis[my\_queue1[hed]][my\_queue2[hed]];
85. #ifdef debug
86. printf("%s,line:%d ,maincost :%lf\n",Stations[now\_status.now\_station].name,now\_status.nowline,now\_status.maincost);
87. #endif
89. vis[my\_queue1[hed]][my\_queue2[hed]]=0;
90. **int** now\_edge=Stations[my\_queue1[hed]].hed\_of\_edge;
91. **for**(;now\_edge;now\_edge=Edge[now\_edge].next\_edge){
92. next\_status=Calculate\_the\_cost(&Edge[now\_edge],&station\_dis[my\_queue1[hed]][my\_queue2[hed]],typ);
93. **if**(next\_status.maincost+avoid\_status[next\_status.now\_station][next\_status.nowline+Bidirectional\_line\_offset]<station\_dis[next\_status.now\_station][next\_status.nowline+Bidirectional\_line\_offset].maincost){
94. next\_status.maincost+=avoid\_status[next\_status.now\_station][next\_status.nowline+Bidirectional\_line\_offset];
95. station\_dis[next\_status.now\_station][next\_status.nowline+Bidirectional\_line\_offset]=next\_status;
97. **if**(vis[next\_status.now\_station][next\_status.nowline+Bidirectional\_line\_offset]==0) {
98. endd++;
99. my\_queue1[endd]=next\_status.now\_station;my\_queue2[endd]=next\_status.nowline+Bidirectional\_line\_offset;
100. vis[my\_queue1[endd]][my\_queue2[endd]]=1;
101. }
102. }
103. }
104. }
105. **return** 1;
106. }
107. :
108. **struct** Cost\_of\_scheme Traverse\_in\_line(**struct** Cost\_of\_scheme now\_status,**int** aim\_line){
109. **int** now\_edge=Stations[now\_status.now\_station].hed\_of\_edge;
110. **for**(;now\_edge;now\_edge=Edge[now\_edge].next\_edge){
111. **if**(Edge[now\_edge].belonging\_line==aim\_line){
112. **return** Calculate\_the\_cost(&Edge[now\_edge],&now\_status,0);
113. }
114. }
115. **return** now\_status;
116. }

(4):

1. **double** get\_modified\_coefficient(**struct** Edge\_of\_subway \*now\_edge,**int** typ,**struct** Cost\_of\_scheme \*now\_status){
2. **if**(typ==6||typ==5||typ==0) typ=2;
3. **double** ans=1.0;
4. **int** cost\_to\_station=((now\_edge->belonging\_line>0)?(Stations[now\_edge->to\_station].number\_on\_line[now\_edge->belonging\_line]-1)\*3:\
5. (Subwaylines[-1\*now\_edge->belonging\_line].total\_station-Stations[now\_edge->to\_station].number\_on\_line[-1\*now\_edge->belonging\_line])\*3);
6. cost\_to\_station-=3;
7. **int** line\_id=abs(now\_edge->belonging\_line),time=now\_status->nowtime;
8. **if**(typ==1){
9. ans=1.0;
10. } **else** **if**(typ==2){
11. ans=((Crowdedness[line\_id][time]>0.75)?1.5:((Crowdedness[line\_id][time]>0.2)?1.0:0.6));
12. } **else** **if**(typ==3){
13. ans=eexp(Crowdedness[line\_id][time]);
14. } **else** **if**(typ==4){
15. **if**(Crowdedness[line\_id][time]>=now\_status->limit\_factor){
16. ans=Inf;
17. } **else**  ans=((Crowdedness[line\_id][time]>0.75)?1.5:((Crowdedness[line\_id][time]>0.2)?1.0:0.6));
18. }
19. //if(time<trans\_time(6,0)+cost\_to\_station)  ans+=(double)(360+cost\_to\_station)/now\_edge->time;
20. //  outt(cost\_to\_station);
21. //  if(time>trans\_time(23,0)+cost\_to\_station) return Inf;
22. **return** ans;
23. }

26. **struct** Cost\_of\_scheme Calculate\_the\_cost(**struct** Edge\_of\_subway \*now\_edge,**struct** Cost\_of\_scheme \*now\_status,**int** typ){
27. **int** cost\_to\_station=((now\_edge->belonging\_line>0)?(Stations[now\_edge->to\_station].number\_on\_line[now\_edge->belonging\_line]-1)\*3:\
28. (Subwaylines[-1\*now\_edge->belonging\_line].total\_station-Stations[now\_edge->to\_station].number\_on\_line[-1\*now\_edge->belonging\_line])\*3);
29. **struct** Cost\_of\_scheme after\_status=\*now\_status;
30. after\_status.last\_status=now\_status;cost\_to\_station-=3;
31. //printf("\n%d\n%d\n",cost\_to\_station,now\_situation->nowtime);
32. **if**(now\_status->nowtime<trans\_time(6,0)+cost\_to\_station) {
33. **if**(typ<=4&&typ>=1) after\_status.maincost+=trans\_time(6,0)+cost\_to\_station-now\_status->nowtime;
34. after\_status.nowtime=trans\_time(6,0)+cost\_to\_station;
35. #ifdef debug
36. can\_you\_reach\_here;
37. #endif
38. }
40. **double** Modified\_Coefficient= get\_modified\_coefficient(now\_edge,typ,now\_status);
41. **if**(Modified\_Coefficient>=Inf)  after\_status.maincost+=Inf;
43. **if**(now\_edge->belonging\_line!=after\_status.nowline&&after\_status.nowline){
44. **int** waiting\_time=((after\_status.nowtime+3-trans\_time(6,0)-cost\_to\_station)%2==1)?2:1;
45. **if**(typ<=4&&typ>=1){
46. after\_status.maincost+=(3+waiting\_time);
47. } **if**(typ==6){
48. after\_status.maincost+=inf;
49. }
50. after\_status.nowtime+=(3+waiting\_time);after\_status.number\_of\_transfer++;
51. } **else** **if**(after\_status.nowline==0){
52. **int** waiting\_time=((after\_status.nowtime-trans\_time(6,0)-cost\_to\_station)%2==1)?2:1;
53. **if**(typ<=4&&typ>=1){
54. after\_status.maincost+=waiting\_time;
55. }
56. after\_status.nowtime+=waiting\_time;
57. //  outt(waiting\_time);
58. } **else** {
59. **if**(typ<=4&&typ>=1){
60. after\_status.maincost+=Modified\_Coefficient;
61. }
62. after\_status.nowtime+=1;
63. }
64. **if**(typ==5||typ==6) after\_status.maincost+=now\_edge->lenth;
65. **else** **if**(typ<=4&&typ>=1) after\_status.maincost+=2\*Modified\_Coefficient;
66. after\_status.nowline=now\_edge->belonging\_line;after\_status.now\_station=now\_edge->to\_station;after\_status.tot\_crowd+=Crowdedness[abs(after\_status.nowline)][after\_status.nowtime];
67. after\_status.nowtime+=now\_edge->time;
68. after\_status.dist+=now\_edge->lenth;after\_status.number\_of\_station++;after\_status.aver\_crowd+=Modified\_Coefficient;
69. //outt(ans\_scheme.nowtime);outt(trans\_time(23,0)+cost\_to\_station);
70. **if**(after\_status.nowtime-3>trans\_time(23,0)+cost\_to\_station){
71. after\_status.maincost=Inf;
72. }
73. #ifdef debug
74. printf("station:%s,line:%d,Modified Coefficient:%lf,first subway arrive time%d,and cost is %lf\n",Stations[after\_status.now\_station].name,after\_status.nowline,Modified\_Coefficient,cost\_to\_station,after\_status.maincost);
75. #endif
76. **return** after\_status;
77. }

# 附录三、程序使用说明

通过选择是否定义宏use\_gui并且重新编译全部程序可以开启/关闭GUI界面。

1. 命令行界面

命令行界面有比较完整的指示，只需要按照要求输入即可。命令行界面判掉了一些非法输入，但是不能保证此界面的绝对安全。

命令行界面默认输出全部信息，包括经过任何一个站点时的状态信息。

1. GUI界面

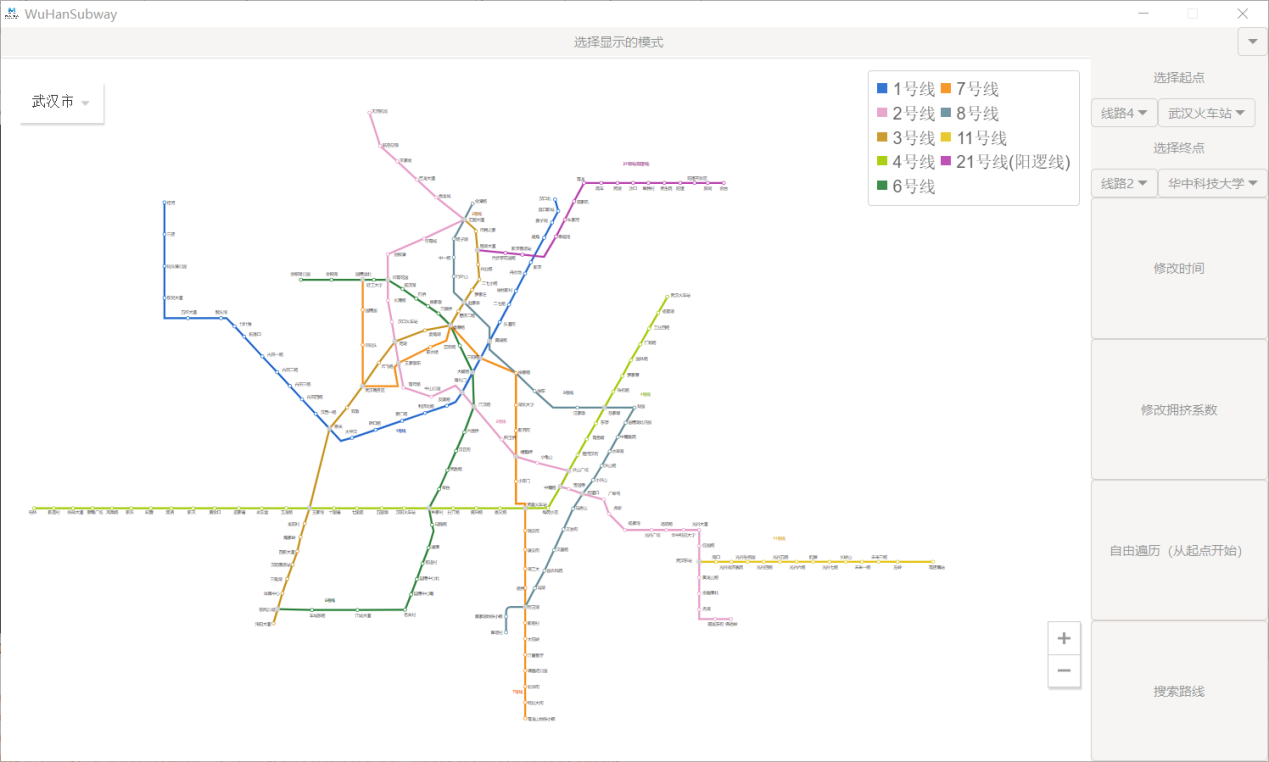


图1 基础GUI界面

除了基础窗口之外其他窗口如图：

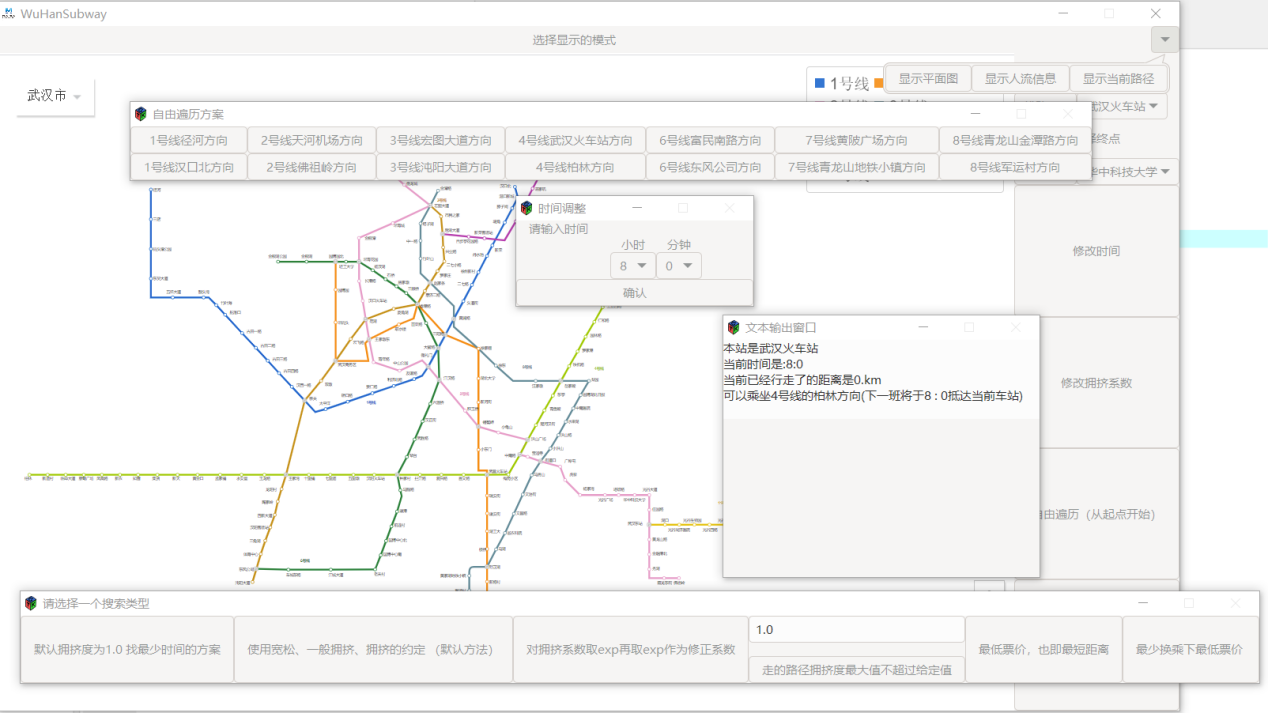


图2主要GUI窗口界面一览

此界面使用GTK+搭建，因此本机需要有gtk环境并且在对应GNU下才能完成编译运行。

除了主界面以外，还有一个文字输出界面，他是输出的主要途径，文字信息都会在这里显示。除此之外还有修改时间界面，遍历选择界面，寻路方案选择界面。

程序的主要控制按钮都在右侧，因此从上往下依次介绍。

右上角的下拉菜单中，可以选择绘图的不同方案，注意只有当一条路线存在的时候才会正确绘图，因此如果寻路之前直接显示一条路线的话是不会绘图的。 绘图中平面图是一张静态图，其他地图上的点都是使用经纬度坐标变换之后得到的。在拥挤度图中，拥挤度由低到高依次为绿，黄，红。四个寻路地图绘制的是路线，其中绿色点表示起点，红色点表示终点，紫色表示换乘站点。注意在选择的时候，要查看文本输出方案中输出的当前为方案几，并不一定按照前后顺序以此为1，2，3。

下方的三个下拉菜单，是为了选择车站，首先选择线路，然后选择车站。如果选择自由遍历的时候，会默认在第一个地址出发。但是注意，此时不会默认你已经在这个线路的车上，而是处于地铁站中，此时完全可能在0分钟内上到任何一各车次上。

再往下是修改时间按钮，此时显示的就是当前时间，因此如果不需要修改，直接点阶确定即可，否则可以选择你想要的时间，然后确定。

之后是修改路线拥挤度按钮，此时主界面会隐藏，命令行界面弹出，按照要求依次输入线路号，修改开始时间（小时+分钟），修改结束时间（小时+分钟），和修改后的拥挤度。修改之后，可以通过绘制拥挤度图来查看效果。

之后是自由遍历按钮，会弹出选择路线界面，同时在文本框中输出可选路线，如果选择不通过或者没有车次的线路，则会无效。如果线路通过并且有车次，那么按下后就会移动，如果可以不换乘，并且已经在车上，就不会换乘。

最右下角是搜索路线，点击后会弹出选择方案界面。可以选择一种方案，如果选择方案4那么就需要输入拥挤度上限（默认1.0），之后则会在目前选择的车站、时间、拥挤度下给出1-3条方案。粗略方案会输出到文本输出窗口里，详细方案则在命令行中，如果要查看某一个方案对应的路线，则可以通过右上角选择。

注意事项：

1. 因为采用了glade搭建控件的方法，当控件被destroy之后，无法重新获得控件，因此不想退出程序的时候请不要关闭任何窗口。只需要最小化或者藏在后面即可
2. 因为电脑环境不同，在其他电脑上的时候可能会出现不稳定的情况，如果出现这种情况，可以关闭界面并重新开始。
3. 绘图，显示时间等等界面需要重新按下按钮才会重置，因此在进行新的寻路之后，需要重新选择一条线路才会绘出正确图案。
4. 请注意默认开始时间为8:00，默认拥挤度为简化设定。