

数据结构课程设计

旅行模拟查询 系统的设计

概要设计

班 级	班内序号	学 号	姓 名
2017211302	06	2017211145	张津源
2017211303	04	2017211172	杨卉帆
2017211304	21	2017211219	范乾一

2019.3 – 2019.6

目录

1	系统概述.....	3
1.1	系统任务.....	3
1.1.1	系统目标.....	3
1.1.2	运行环境.....	3
1.2	需求规定.....	3
1.2.1	功能需求.....	3
1.2.2	性能需求.....	4
1.2.3	数据要求.....	4
2	软件结构图.....	5
2.1	模块结构图.....	5
2.2	模块清单.....	5
3	模块功能及特点.....	6
3.1	模块 1（控制函数模块）	6
3.2	模块 2（搜索算法模块）	6
3.3	模块 3（图形化界面与日志记录模块）	6
4	用户界面.....	7
4.1	初始界面.....	7
4.2	行程查询界面.....	8
4.3	行程管理界面.....	14
5	功能需求、数据结构和模块.....	18
5.1	功能需求与模块关系.....	18
5.2	数据结构与模块关系.....	19
5.2.1	主要数据结构.....	19

5.2.2	数据结构和模块的交叉引用表.....	21
-------	--------------------	----

1 系统概述

1.1 系统任务

1.1.1 系统目标

本软件旨在根据旅客的要求设计出行路线并输出，同时系统能模拟旅客所在的地点和状态，目标是旅客提供完美的出行计划。

1.1.2 运行环境

IDE: Microsoft Visual Studio 2017

编译器: gcc -std=c++11

系统: Windows 10, Mac

1.2 需求规定

1.2.1 功能需求

城市总数不少于 10 个

建立汽车、火车和飞机的时刻表（航班表）

- 有沿途到站及票价信息
- 不能太简单（不能总只是 1 班车次相连）

旅客的要求包括：起点、终点、途经某些城市和旅行策略

旅行策略有：

- 最少费用策略：无时间限制，费用最少即可
- 最少时间策略：无费用限制，时间最少即可
- 限时最少费用策略：在规定的时间内所需费用最省

旅行模拟查询系统以时间为轴向前推移，每 10 秒左右向前推进 1 个小时(非查询状态的请求不计时)；

不考虑城市内换乘交通工具所需时间

系统时间精确到小时

建立日志文件，对旅客状态变化和键入等信息进行记录

某旅客在旅行途中可更改旅行计划，系统应做相应的操作

用图形绘制地图，并在地图上反映出旅客的旅行过程

1.2.2 性能需求

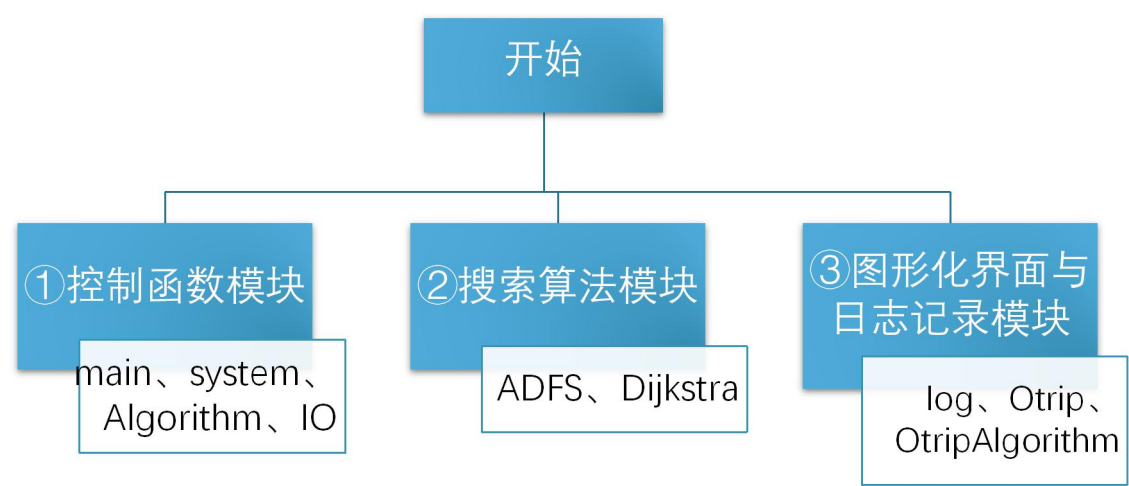
使用 Dijkstra 算法计算最优旅行策略，时间复杂度为 $O(n^2)$ ，空间复杂度 $O(n)$ 。

1.2.3 数据要求

汽车、火车及飞机的时刻表及对应的数据结构必须合理。

2 软件结构图

2.1 模块结构图




2.2 模块清单

编号	模块名称	模块内容
1	控制函数模块	main.cpp、system.cpp、Algorithm.cpp、IO.cpp
2	搜索算法模块	ADFS.cpp、Dijkstra.cpp
3	图形化界面与日志记录模块	Otrip.py、OtripAlgorithm.py、log.json


3 模块功能及特点

3.1 模块 1（控制函数模块）

 包含 main.cpp、system.cpp、Algorithm.cpp、IO.cpp

 功能：


- 存储用户输入的信息
- 根据用户输入的要求，选择相应的策略函数，求解出相应的最佳路线

 特点：

本模块将要解决的问题分层次进行解答，通过一层层的函数调用，实现复杂问题简单化。

3.2 模块 2（搜索算法模块）


 包括 ADFS.cpp 和 Dijkstra.cpp

 功能及特点：

对于数据规模为 23 个城市和 11686 个行程的数据集，本程序采用 Dijkstra 算法、Astar 算法、剪枝技术和松弛操作等方法，在此规模上能进行有效高速地搜索，求解出最优解。

3.3 模块 3（图形化界面与日志记录模块）

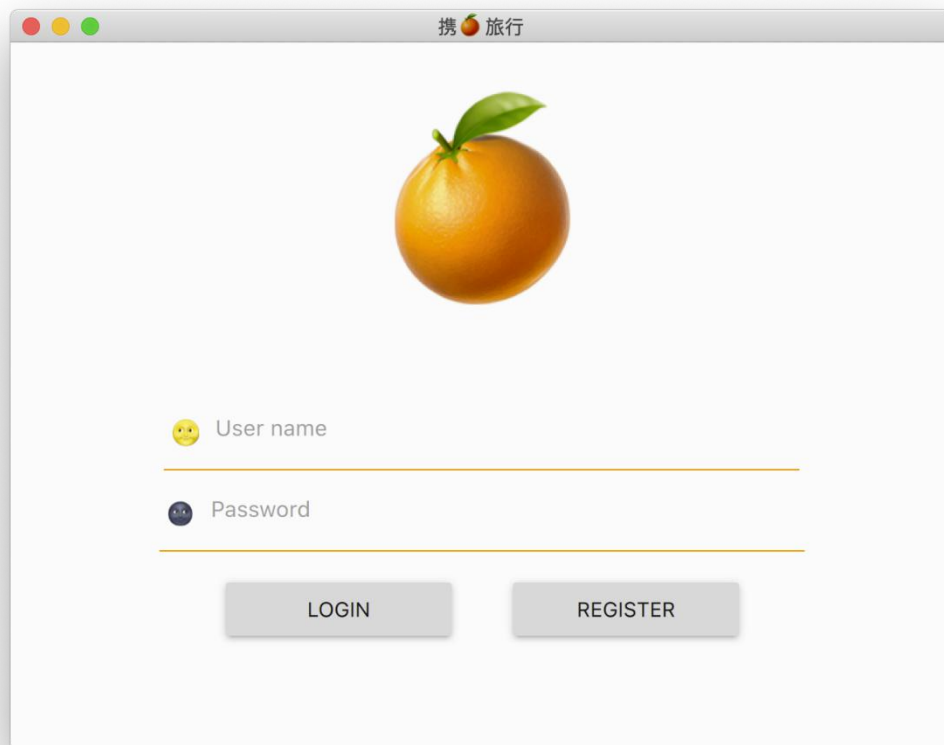
 包括 Otrip.py、OtripAlgorithm.py、log.json

 完成图形化界面的绘制和日志的记录

4 用户界面

4.1 初始界面

- 开始运行程序，首先显示初始界面。
用户可选择登陆已有账号，或注册一个新账号。
- （本程序实现多用户信息的存储，及多用户同时使用的功能。）



用户在界面下方可点击选择进入“行程管理”或“行程查询”两个不同的界面。

The screenshot shows a web application window titled "携程旅行" (Ctrip Travel). The interface is divided into two main sections: "行程管理" (Trip Management) on the left and "行程查询" (Trip Search) on the right.

行程管理 (Trip Management) Section:

- 来自:** A dropdown menu.
- 前往:** A dropdown menu.
- 途径 1:** A dropdown menu.
- 途径 2:** A dropdown menu.
- 途径 3:** A dropdown menu.
- 出发时间:** Input fields for "日" (Day) and "时" (Hour).
- 时间限制:** Input fields for "天" (Days) and "时" (Hours).
- 金钱限制:** An input field for "元" (Yuan).
- 策略选择:** Two radio buttons: "金钱优先" (Money Priority) and "时间优先" (Time Priority).
- 按钮:** A large "行程查询" (Trip Search) button.

行程查询 (Trip Search) Section:

- 总金钱消耗:** 四颗恒星 (Total Money Spent: 4 Stars).
- 抵达时间:** 三百年 (Arrival Time: 300 Years).
- 按钮:** "地图" (Map) and "接受行程" (Accept Trip).
- 详情 (Details):** Two entries showing trip details:
 - Entry 1:** 来自: 半人马α座 (From: Centaurus Alpha), 前往: 地球 (To: Earth), 出发时间: [0,0,0] (Departure Time: [0,0,0]), 抵达时间: [1461,0,0] (Arrival Time: [1461,0,0]), 车号: 第一舰队 (Vehicle Number: First Fleet).
 - Entry 2:** 来自: 半人马α座 (From: Centaurus Alpha), 前往: 地球 (To: Earth), 出发时间: [0,0,0] (Departure Time: [0,0,0]), 抵达时间: [400,0,0] (Arrival Time: [400,0,0]), 车号: 第二舰队 (Vehicle Number: Second Fleet).

At the bottom of the window, there are two tabs: "行程管理" (Trip Management) and "行程查询" (Trip Search), with the latter being the active tab.

4.2 行程查询界面

1. 在上图所示的界面中，点击下方“行程查询”按钮。
2. 单击几个输入框的箭头。用户可通过下拉菜单，选择起点、终点、途径城市、出发时间、时间限制、金钱限制和优先策略方案。
3. 用户输入要求完毕后，点击界面左侧下方的“行程查询”按钮，即可在界面右侧得到相应的最优行程规划。

行程查询示例：

携程旅行

来自: 广州

前往: 北京

途径 1: 乌鲁木齐

途径 2: 哈尔滨

途径 3: 拉萨

出发时间

614

时间限制

天

时

金钱限制

元

☒ 金钱优先

☐ 时间优先

行程查询

总金钱消耗: 2450

抵达时间: 12,14,34

地图

接受行程

详情

来自: 广州
前往: 郑州
出发时间: 6,17,10
抵达时间: 7,8,58
车号:Z14

详情

来自: 郑州
前往: 乌鲁木齐
出发时间: 8,5,0
抵达时间: 9,11,15
车号:Z41

详情

来自: 乌鲁木齐
前往: 兰州
出发时间: 9,12,8
抵达时间: 10,11,34
车号:K1584

行程管理

行程查询

9

若输入的出发时间早于当前时间，系统会有“非法输入”提示。示例图如下：

携程旅行

非法输入：所查询最早发车时间早于当前时间！

途径 1:

▼

途径 2:

▼

途径 3:

▼

出发时间

日

时

时间限制

天

时

金钱限制

元

☒ 金钱优先

☐ 时间优先

行程查询

详情

来自：半人马座
前往：地球
出发时间：[0,0,0]
抵达时间：[1461,0,0]
车号:第一舰队

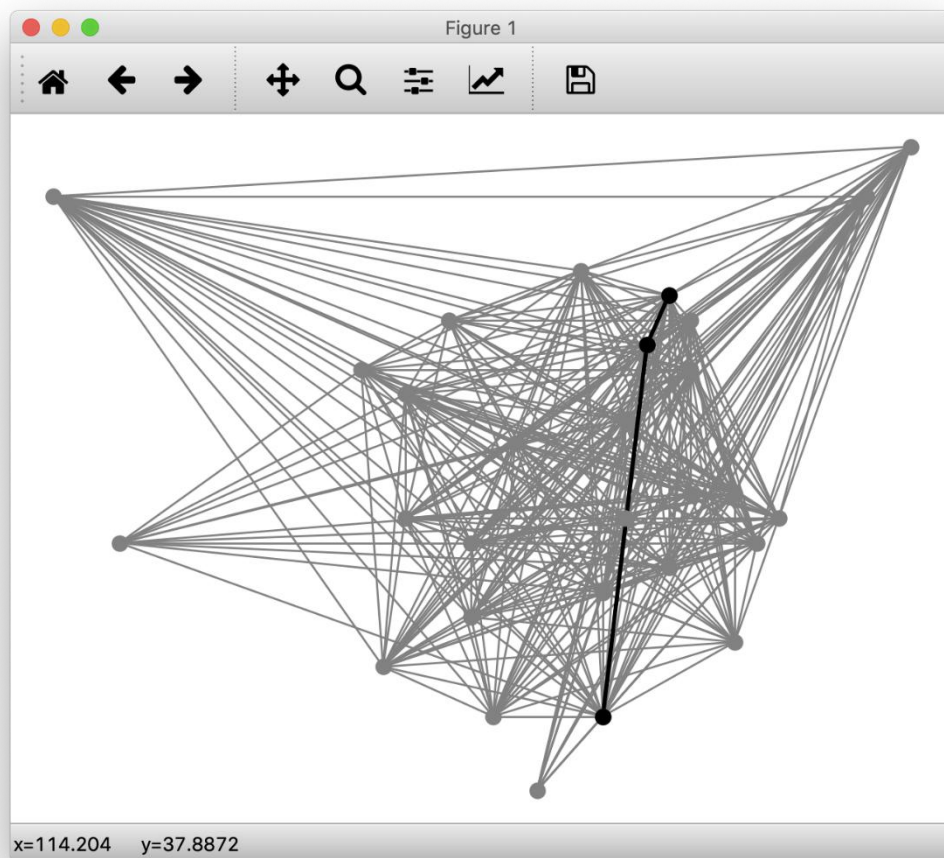
详情

来自：半人马座
前往：地球
出发时间：[0,0,0]
抵达时间：[400,0,0]
车号:第二舰队

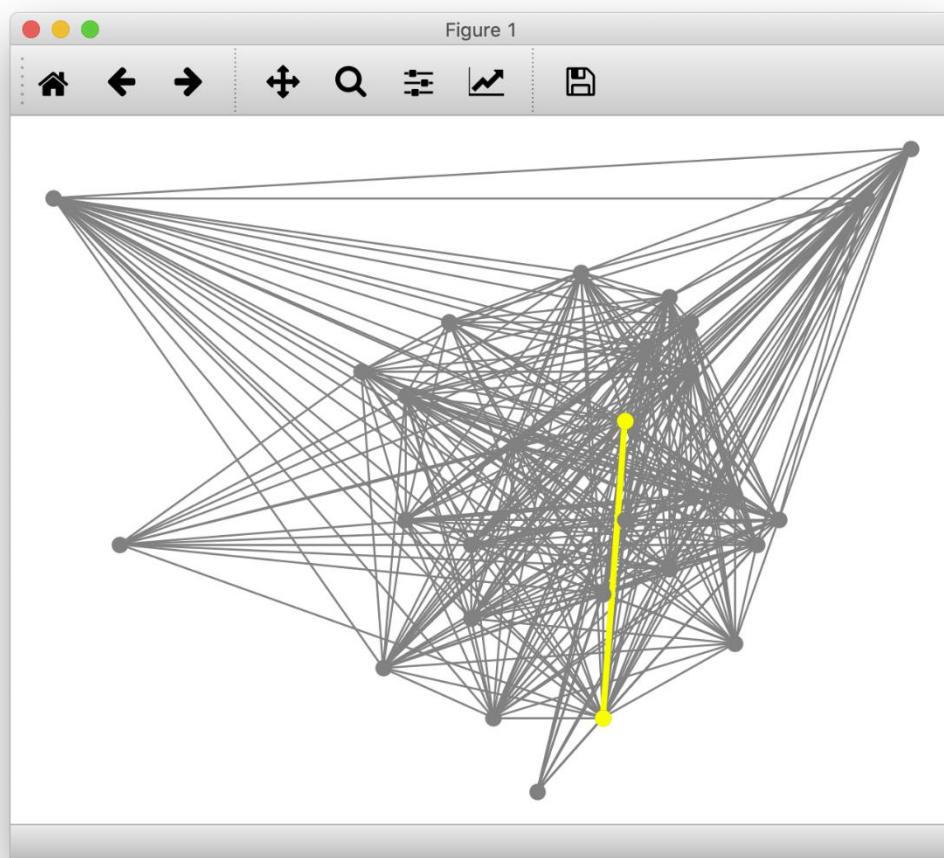
行程管理

行程查询

- 右侧“行程查询”界面包含“地图”、“接受行程”、“详情”三类按钮。
- 点击“地图”，显示包含全部城市的地图，上用黑色粗线条标记系统提供的最优行程路线。



🎨 点击“详情”按钮，可得到该车次的示意图。对应路线由黄色线条标记。



点击“接受行程”按钮，即可成功添加该行程。

携程旅行

成功添加由北京发往广州的列车。

途径 1:

途径 2:

途径 3:

出发时间

日

10

时间限制

天

时

金钱限制

元

☒ 金钱优先

☐ 时间优先

行程查询

详情

来自: 北京
前往: 石家庄
出发时间: 0,23,50
抵达时间: 1,3,30
车号:K5215

详情

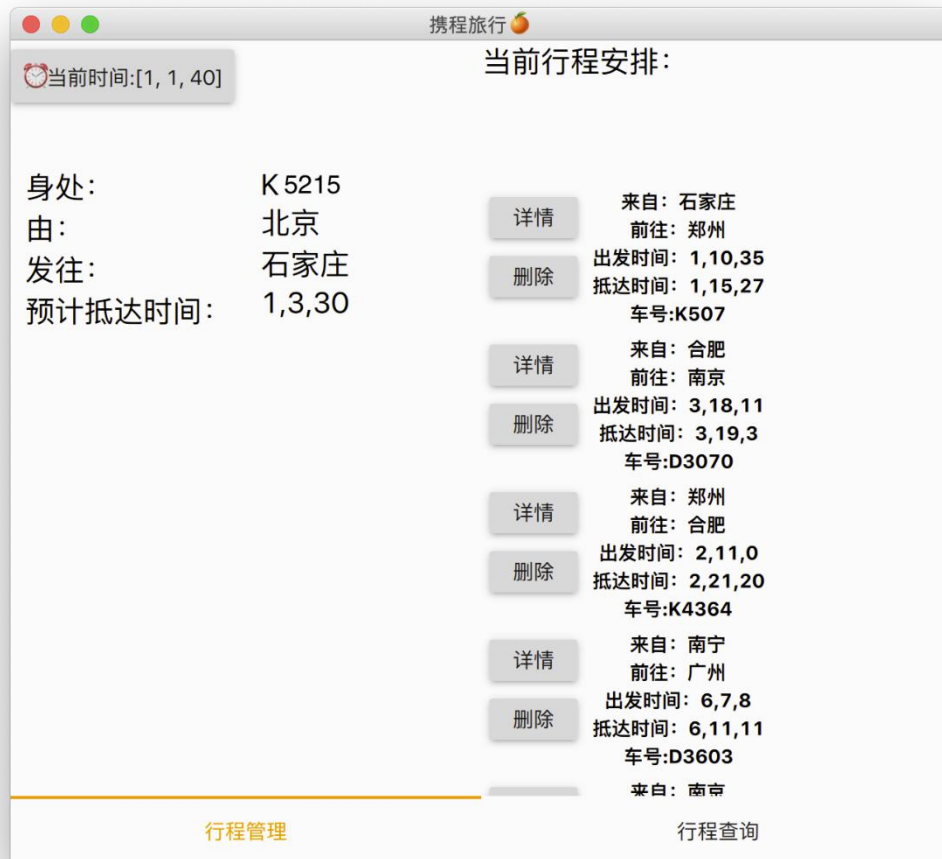
来自: 石家庄
前往: 广州
出发时间: 1,7,9
抵达时间: 2,5,30
车号:T369

行程管理

行程查询

4.3 行程管理界面

- ✚ 点击下方“行程查询”按钮，界面如下图所示。
- ✚ 界面左侧为当前信息（身处车次，车次起点、终点、预计抵达时间），右侧为此时已添加的行程安排。



- 左侧界面上方的“当前时间：XX”为一个按钮，按一次可使时间加快 10 倍。
共有四档：
真实时间流速：模拟时间流速 = 1s:1min； 1s:10min； 1s:100min； 1s:1000min。



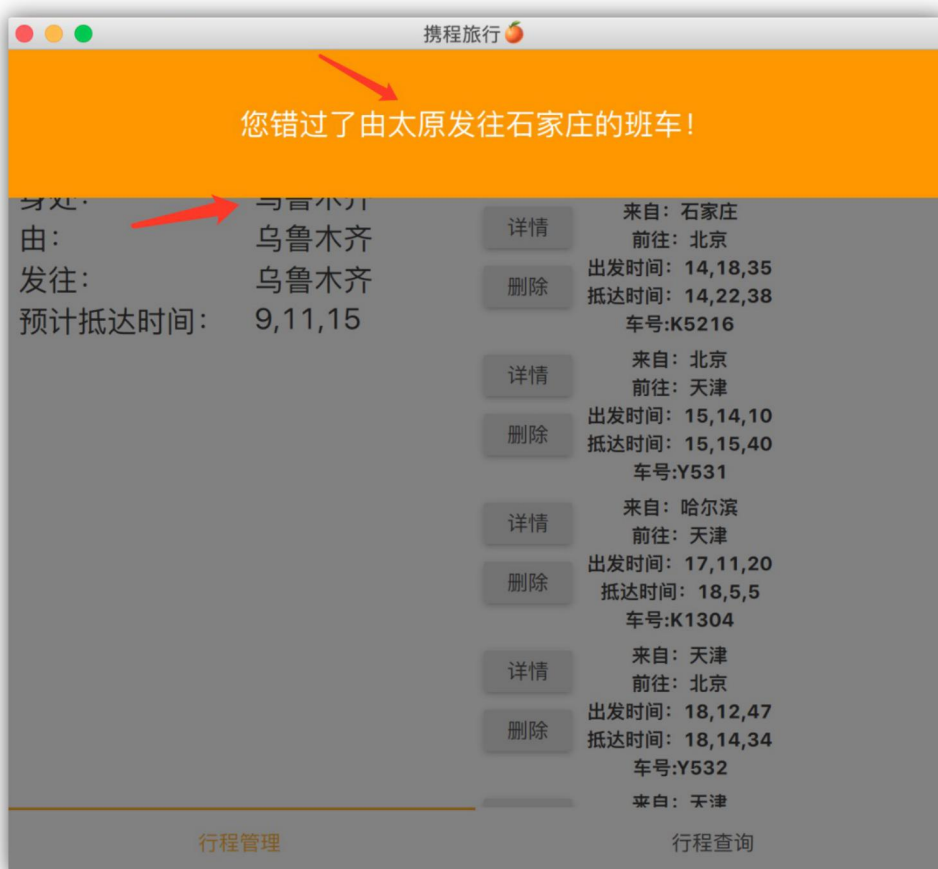
✚ 若当前时间在行程安排的时间内，即用户在某班次的行程中，会有系统提示。示例如下：



- ✚ 若用户在旅行途中更改旅行计划，可在此界面进行操作。
用户可通过点击某车次的“删除”按钮，删除本次行程中的该车次。



- ✚ 然后，用户可在行程查询界面，输入新要求，进行查询、添加车次的操作。
如果删除车次后，未再次购票，错过班车时，会弹出如下错误提示：



5 功能需求、数据结构和模块

这一章建立功能和模块、数据结构和模块的交叉引用表。

- 功能和模块的交叉引用表表明功能需求到模块的分配，也就是说，功能需求与实现该功能的模块集合的对应关系。
- 数据结构和模块的交叉引用表表明模块和数据结构的生成关系和使用关系。

5.1 功能需求与模块关系

	① 控制函数模块	② 搜索算法模块	③ 图形化界面与日志记录模块
功能 A：求解最少费用策略： 无时间限制，费用最少即可	✓	✓	
功能 B：求解最少时间策略： 无费用限制，时间最少即可	✓	✓	
功能 C：求解限时最少费用策略： 在规定的时间内所需费用最省	✓	✓	
功能 D：旅行模拟查询系统以时间为轴向前推移，每 10 秒左右向前推进 1 个小时(非查询状态的请求不计)			✓
功能 E：建立日志文件，对旅客状态变化和键入等信息进行记录			✓
功能 F：某旅客在旅行途中可更改旅行计划，系统应做相应的操作	✓	✓	✓
功能 G：用图形绘制地图，并在地图上反映出旅客的旅行过程			✓

5.2 数据结构与模块关系

5.2.1 主要数据结构

数据结构 A: Schedule——时刻表（航班表）

- 全局量；数据类型为 json
- 含义：汽车、火车或飞机的时刻表（航班表）
- 定义：

key	value	含义及说明
number	string	（汽车或火车的）车次，（飞机的）航班号
from	string	起始城市
to	string	抵达城市
start	list	此段行程起始时间
end	list	此段行程结束时间
duration	list	此段行程持续时间
price	int	金钱花费
type	string (e.g. K/T/C)	对应车次/航班号的类型

- 说明：
从面向对象的角度看，每一个 Schedule 可看成是一个抽象类型。
安排一个 Schedule 的过程，可看成这个 Schedule 实例化的过程，需要给其加上出发时间的属性。
所以一个实际 Schedule 的完整定义还需添加例如{"data",[1,1,1]}的键值对，表示这个 Schedule 实例化成了一个于距离指定开始时间一天一时一分时出发的 Schedule。

- 举例：

```
{
  "number": "C2181",
  "from": "北京",
  "to": "天津",
  "start": [0, 0, 1],
  "end": [0, 0, 31],
  "duration": [0, 0, 30],
  "price": 54,
  "type": "C"
}
```

数据结构 B: D——database (数据集)

- 数据类型: 元素类型为 json 的线性表
- 存储内容: 数据集里所有的 Schedule, 即所有汽车、火车和飞机的时刻表 (航班表)
- 定义: `vector<json> D;`

数据结构 C: G——graph

- 数据类型: 邻接表形式的有向图
- 含义: D 的以城市为节点的邻接表形式
- 部分源代码: (仅列出两个城市)

```
json G;  
G = {  
    "Beijing":{  
        {"from": "Beijing", "to": "Shanghai", "number": "4817", "start": [0, 16, 30],  
         "end": [1, 7, 30], "duration": [0, 15, 0], "price": 341, "type": "N"},  
        {"from": "Beijing", "to": "Shanghai", "number": "CZ6412", "start": [0, 6, 25],  
         "end": [0, 8, 40], "duration": [0, 2, 15], "price": 720, "type": "AC"}  
    },  
    "Shanghai":{  
        {"from": "Shanghai", "to": "Wuhan", "number": "000756", "start": [0, 10, 50],  
         "end": [0, 22, 50], "duration": [0, 12, 0], "price": 262, "type": "N"},  
        {"from": "Shanghai", "to": "Wuhan", "number": "009675", "start": [0, 11, 56],  
         "end": [0, 23, 56], "duration": [0, 12, 0], "price": 230, "type": "N"}  
    },  
    .....  
}
```

数据结构 D: status

- 局部量, 数据类型为 json
- 含义: 图上各个节点的标记信息 (对于某用户的行程安排);
包括节点的抵达时间 t, 抵达金钱花费 m, 抵达时所经过的 Schedule 序列。

🚦 举例：（初始值）

```
json status;  
status = {  
    "Beijing" : { "t":0, "m":0, "s":[] }  
}
```

5.2.2 数据结构和模块的交叉引用表

🚦 说明：

“C”表示生成关系，即在一个模块中生成一个数据结构。

“U”表示使用关系，即一个模块中使用某数据结构。

	① 控制函数模块	②搜索算法模块	③图形化界面与日志记录模块
数据结构 A: Schedule	C	U	U
数据结构 B: D	C	U	
数据结构 C: G	C	U	U
数据结构 D: status		C	