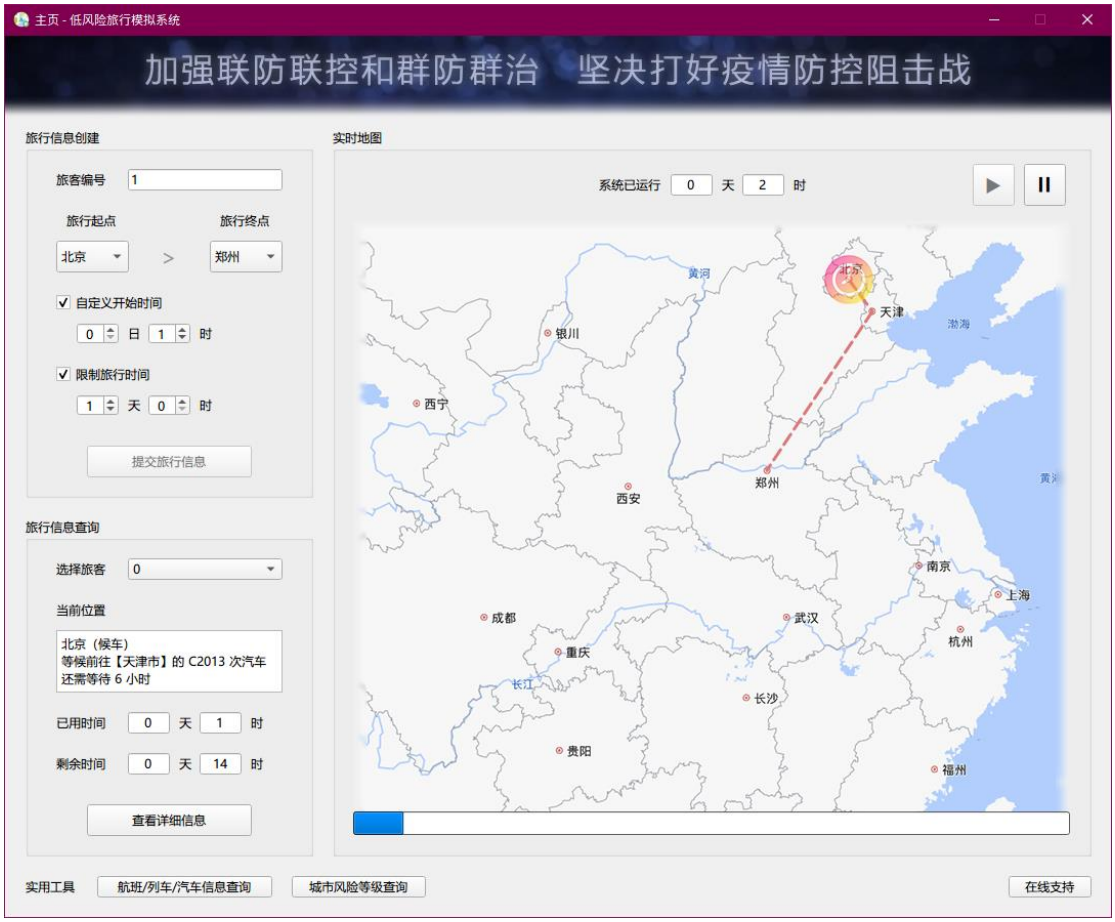
1 程序评价

1.1 从用户的角度评价

1.1.1 界面布局紧凑合理

本程序的主界面涵盖了几乎本软件的全部功能，包括**旅行信息创建**、**旅行信息查询**和**实时地图**三个操作区域，并带有区域标签，界限分明。这使任何用户打开软件后便对本软件界面结构清晰明了。

软件 GUI 使用 Qt 库开发，界面显示和布局严格遵循 Windows 平台应用开发标准。



1.1.2 用户交互体验良好、健壮性强

软件界面的每个部分责任分明，使用户操作简单且易于上手。每个输入框、按钮的作用清晰易懂，符合绝大多数用户的操作逻辑。用户执行每一个重要操作时软件都会给予相应的反馈，对于不合法的用户操作，软件会进行屏蔽并给出相应的提示信息。包括数据文件缺失；用户定义时间不合法、未找到旅行方案、选择的城市不合法的错误反馈；以及旅行终点为高风险地区的温馨提示。

1.1.3 信息显示直观性、实时性强

当旅客提交了旅行请求并且找到合适方案时，系统会弹出旅行方案预览框，该预览框会

复现旅客的旅行信息，如旅行发起时间、旅行起点和旅行终点。并且会在下方方案展示框中展示完整的旅行行程，包括在某城市的停留时间，每次发车时间，以及乘车时长，可以说 100%地描述了旅客的旅行轨迹，非常直观。

当有旅客时，主界面的旅行信息查询区域会实时显示旅客的状态，包括旅客的位置，旅客还需候车/乘车多久，还会展示旅客已经旅行了多久，整个旅行还剩余多久。

地图展示界面能够实时反映旅客的旅行情况，地图会将旅客途径城市用虚线连接，并且对旅客的不同状态，以不同的图标加以区分，直观性强。当旅客处于乘车状态时，图标运动平滑，可达到 60 帧的运动效果。

当有旅客正在旅行时，地图底部会自动出现一个进度条，反映旅客旅行进度。当旅行结束后自动消失。

当切换旅客时，界面所有元素会立即刷新，显示相应的旅客信息，没有任何延迟和卡顿。

1.1.4 功能丰富、输出最优解

本系统在课程设计基础要求下继续添加新功能，改进基础功能。系统支持时间的暂停/继续；支持用户自定义旅行开始时间，支持用户对旅行进行限时，支持考虑交通工具的风险；支持旅客之间的无缝切换，以及旅行信息的实时展示；支持完整日志输出；支持查询城市风险等级，支持单独查询车次。

经过手工测试和理论证明，本软件核心算法所求得的解为**全局最优解**。

1.1.5 软件兼容性强

经过全面的测试，本软件支持 Windows XP 及所有更高版本的 x64 体系架构操作系统，均可正常运行。

经过 UI 设计优化，本软件支持在高分屏设备以及高缩放桌面环境下显示，不会造成界面显示异常、无法操作等情况。

1.1.6 产品后期支持完善

本软件提供后期技术支持，用户可以在软件主界面点击“在线支持按钮”获得在线技术支持，其网址为 http://www.myzhang.site/data-structure-course-design-support/。在此可下载软件的最新版本、缺失的数据文件、技术文档以及使用说明等。为用户提供满意的服务。

1.2 从开发者的角度评价

1.2.1 系统架构稳固

本软件开发使用面向对象程序设计方法（C++实现），**分层体系架构**。采用分层体系架构的目的是：修改某一层的功能和具体实现方法时，不对其它层次产生影响，因此系统架构稳固。

1.2.2 系统模块划分清晰、可扩展性好

本软件的源代码分为 9 个.cpp 源文件，每个文件实现一个核心功能，尽量实现了**高内聚**和**低耦合**。降低了模块间的依赖性和相关性，易于维护和调试。

本软件的源代码中，尽量使用**类型别名**、**宏定义**和**常量表达式**，当对某些功能或常量进行修改时，不会造成代码结构的变化。当加入新功能时，基本无需修改现有源代码内容。

1.2.3 程序安全性高

1. 本软件遵循**接口和实现分离**的原则，并严格控制类属性和类方法的**访问权限**，使得一个类仅有提供的服务对外是可见的，其内部属性和实现对其它模块完全透明。

【例】系统控制类 Control 的定义，对类属性和类方法的访问权限进行了严格的控制：

```
class Control {
private:
    bool sysPause;                          // 系统状态
    std::thread* sysClock;                  // 系统主时钟线程
    std::thread* travelStatusRefresher;     // 系统刷新器线程
    void checkProcessExist() const;         // 判断是否已有此程序实例运行
    void startLogSystem();                  // 启动日志系统
    ...（略）
public:
    CityList cityList;                      // 城市列表
    Schedule schedule;                      // 时刻表
    InstanceList instanceList;              // 旅行实例列表
    ...（略）
    void pause();                           // 系统暂停
    void proceed();                         // 系统继续
    ...（略）
};
```

2. 本软件中对一个类的某些方法加以 **`const` 限定符**，使得该类的对象的内部属性得到保护而不被意外修改。

【例】Vehicle 类中使用的 const 限定符：

```
class Vehicle {
private:
    ...（略）
    City* srcCity, * dstCity;                   // 交通工具始发站和终到站
    Time startTime, runTime;                    // 交通工具开点和运行时间
public:
    ...（略）
    City* const getSrcCity() const;             // 获取交通工具始发站
    City* const getDstCity() const;             // 获取交通工具终到站
    Time getStartTime() const;                  // 获取交通工具开点
    ...（略）
};
```

3. 程序编制过程中尽量使用常量表达式而不是宏定义，以提高安全性和可靠性。

4. 开发过程中严格进行内存管理，坚决**避免内存泄漏**，提高安全性和性能。

【例】程序调试中对内存占用的监视，体现其重视程度：

1.2.4 程序性能较高

程序的性能体现在两方面，一是语言层面的程序执行性能；另一方面是算法的性能。下面分别阐述：

1. 程序设计语言的性能

本程序完全采用 C++语言编写，目的是兼顾开发效率和程序性能。因为 C++语言既支持面向对象的程序设计方法，又不失 C 语言所具有的高性能的特点。

在程序的交互层，为了安全性，采用 C++ STL 标准模板库；在程序的算法层，一律采用 C 风格原生数组，以及 C 原生链表，以尽最大限度提高程序性能。

2. 算法的性能

程序核心算法为 Dijkstra 算法，为了进一步优化算法性能，特采用了**堆优化**，并改用**邻接表**存储图结构，降低时间复杂度和空间复杂度。经过压力测试，算法在 **10000 个城市+1000000 个车次**的巨型规模数据量下，响应时间仅为 **2~3s**！而对于常规数据量，算法更是能做到**微秒级响应**。

1.2.5 编码风格规范统一、可读性强

在源代码编写过程中，严格遵循编码规范，对变量名，函数名，类名有着严格的要求，并且名称表意清晰。严格把握代码缩进，增强逻辑性。代码中有大量注释，提高其可读性。

2 程序改进意见

2.1 数据文件存储方式需要改进

本软件的数据文件以文本方式存储，其安全性和规范性得不到有力保障，若数据文件被恶意篡改，程序便不能正常工作。因此本人拟在后续发行版本中采用数据库存储程序数据，提高安全性和稳定性。

2.2 CPU 占用率需要优化

本软件在运行时 CPU 占用率在 1.5%左右，会造成设备功耗较大，其原因是为了追求地图模块的高精度刷新，本人拟在后续发行版本中优化地图刷新策略，降低 CPU 占用率。

2.3 算法性能仍有提升空间

为寻求最优解，本软件未采用启发式算法；但是在数据规模较大时，采用启发式算法会更具优势。

2.4 跨平台

本软件目前仅支持在 Windows x64 平台上运行。得益于 Qt 跨平台的特性，本软件的跨平台开发应该没有太大的困难，本人后续会做这项工作。