

# 北京邮电大学课程设计报告

课程设计 名称	数据结构课程设计		学院	计算机	指导教师	郭岗
班 级	班内序号	学 号		学生姓名	分 工	
2021211304		2021212171		杨晨	课程表、临时事物、课外活动、用户和管理员登录注册	
2021211304		2021211899		蒋礼特	模拟时钟，提醒表，日志	
2021211304		2021212484		张梓良	导航算法、地图信息编写、可视化导航界面	
课 程 设 计 内 容	<p>同学每天都有学校课程及自选课程、课外活动、临时事务等多类活动，每类活动的特点各不相同，因而，需要针对学生们的特殊需求，将每天的多类活动进行有效的管理和提醒。</p> <p>学生日程管理系统可以帮助学生管理自己的课程、各种活动和临时事务，具备课程类日程管理、课外活动日程管理和临时事务日程管理的相关功能。每天晚上系统会提醒学生第二天的所有日程；快要到活动时间时，系统会根据活动的类型进行相应的提醒和规划；也可以查看一个学期的所有日程。</p> <p>对于课程类，邻近上课时间时会进行提醒：如果是线下课程，会输出去教室的路线，如果是线上课程，会输出课程在线平台和链接；对于课外活动类，如果是线下活动，会输出到活动地点的路线（校内），如果是线上活动会输出活动的在线平台和链接；对于临时事务，要按临时事务进行地点规划完成多个事务的途经最短距离路线并输出</p>					
学 生 课程设计 报 告 （附 页）						

课 程 设 计 成 绩 评 定	<p>遵照实践教学大纲并根据以下四方面综合评定成绩：</p> <ol style="list-style-type: none"><li>1、课程设计目的任务明确，选题符合教学要求，份量及难易程度</li><li>2、团队分工是否恰当与合理</li><li>3、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果</li><li>4、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范</li></ol> <p>评语：</p>
	<p>成绩：</p> <p>指导教师签名：</p> <p>2023 年 5 月 28 日</p>

注：评语要体现每个学生的工作情况，可以加页。

# 目 录

第一章 项目概述.....	
1.1 问题描述.....	
1.2 系统设计思路.....	
1.2.1 概要设计.....	
1.2.2 详细设计.....	
1.3 功能设计与用户场景匹配.....	
1.3.1 登录、注册.....	
1.3.2 课程、活动、临时事物管理.....	
1.3.3 提醒与闹钟.....	
1.3.4 地图与导航.....	
第二章 系统架构.....	
2.1 系统模块.....	
2.1.1 模块划分.....	
2.1.2 模块依赖关系.....	
2.1.3 模块内部结构.....	
2.1.4 模块接口定义.....	
2.2 数据描述.....	
2.2.1 用户表.....	
2.2.2 课程表.....	
2.2.3 活动表.....	
2.2.4 临时事务表.....	
2.2.5 日志表和提醒表.....	
2.2.6 地图表.....	
2.3 业务流程.....	
2.3.1 管理员后台管理流程和课表管理流程.....	
2.3.2 课外活动管理流程.....	
2.3.3 临时事务管理流程.....	
2.3.4 导航流程.....	
2.3.5 时钟轮和闹钟提醒流程.....	
第三章 算法设计和算法性能分析.....	
3.1 项目进度管理的概念.....	
3.1.1 哈希算法.....	

3.1.2 快速排序算法 .....	
3.1.3 BKDRHash.....	
3.1.4 dijkstra.....	
3.1.5 tabuSearch .....	
3.2 算法复杂度分析 .....	
3.2.1 时间复杂度 .....	
3.2.2 空间复杂度 .....	
3.2.3 算法时间复杂度实例分析 .....	
3.3 算法正确性证明 .....	
3.3.1 直接证明法 .....	
3.3.2 反证法 .....	
第四章 算法测试 .....	
4.1 算法效率测试 .....	
4.1.1 测试用例设计 .....	
4.1.2 测试用例执行记录 .....	
4.1.3 测试结果 .....	
第五章 总结 .....	

# 第一章 项目概述

## 1.1 问题描述

大学中每位同学每天都有学校课程及自选课程、课外活动、临时事务等多类活动，每类活动的特点各不相同，因而，需要针对学生们的特殊需求，将每天的多类活动进行有效的管理和提醒。

学生日程管理系统可以帮助学生管理自己的课程、各种活动和临时事务，具备课程类日程管理、课外活动日程管理和临时事务日程管理的相关功能。每天晚上系统会提醒学生第二天的所有日程；快要到活动时间时，系统会根据活动的类型进行相应的提醒和规划；也可以查看一个学期的所有日程。

对于课程类，邻近上课时间时会进行提醒：如果是线下课程，会输出去教室的路线，如果是线上课程，会输出课程在线平台和链接；对于课外活动类，如果是线下活动，会输出到活动地点的路线（校内），如果是线上活动会输出活动的在线平台和链接；对于临时事务，要按临时事务进行地点规划完成多个事务的途经最短距离路线并输出

## 1.2 系统设计思路

### 1.2.1 概要设计

采用 B/S 架构，前端使用 Qt，后端使用 C++ 语言开发，数据库使用本地 txt 文件。

功能模块，主要包含用户管理、课程管理、活动管理、临时事务管理、时间管理、路径规划、提醒等模块。

需要的数据库表，有用户表、课程表、活动表、临时事务表、提醒表等，记录用户及其课程、活动、事务等相关信息。

### 1.2.2 详细设计

#### 用户管理

用户注册：管理员使用添加功能，输入学号、密码、姓名等信息，确认信息无误后添加用户至用户表

登录验证：用户输入学号和密码，后台验证信息与用户表记录匹配后允许登录

用户信息修改：管理员可以修改用户的信息，包括密码，姓名。

#### 课程管理

添加课程：用户输入课程信息如课程名称、上课时间、上课地点等，后台检测冲突后添加课程记录至课程表。

查看课程：可以通过一个表格化的界面展示课表

查询课程：通过课程名字，取出属于该课程的信息进行展示。

修改课程：用户选择要修改的课程，输入新的课程信息，后台检测冲突后修改课程表相应记录。

删除课程：用户选择要删除的课程，后台删除课程表相应记录。

#### 活动管理

添加活动：用户输入活动信息如活动名称、活动类别、活动时间、活动地点等，后台检测冲突后添加活动记录至活动表。

查看活动：可以通过一个表格化的界面展示活动

查询活动：通过活动名字，取出属于该活动的信息进行展示。

修改活动：用户选择要修改的活动，输入新的活动信息，后台检测冲突后修改活动表相应记录。

删除活动：用户选择要删除的活动，后台删除活动表相应记录

### **临时事物管理**

添加临时事物：用户输入临时事物信息如临时事物名称、事物时间、事物地点等，后台检测冲突后添加临时事物记录至临时事物表。

查看临时事物：可以通过一个表格化的界面展示事物表

查询临时事物：通过临时事物名字，取出属于该临时事物的信息进行展示。

修改临时事物：用户选择要修改的临时事物，输入新的临时事物信息，后台检测冲突后修改临时事物表相应记录。

删除临时事物：用户选择要删除的临时事物，后台删除临时事物表相应记录

### **提醒与闹钟**

每天早上 8 点提醒当天所有事情，每天晚上会提醒第二天的所有事情。会提前一小时提醒事情

添加提醒：用户输入信息如名称、时间、地点等，后台添加提醒记录到提醒表中。

查看提醒：可以通过一个表格化的界面展示提醒表

查询提醒：通名字，查找到详细的提醒信息进行展示。

修改提醒：用户选择要修改的提醒，输入新的提醒信息，后台修改提醒表相应记录。

删除提醒：用户选择要删除的提醒，后台删除提醒表相应记录

### **导航**

根据地点数量的不同选用不同的算法进行最短路径求解。

当是单点到单点的情形时，采用小根堆优化的 **dijkstra** 算法进行求解。

当是单点到多点且多点数量不超过 10 的情形时，采用 **dfs** 搜索算法进行求解。

当是单点到多点且多点数量超过 10 的情形时，采用启发式的禁忌算法进行求解。

采用以上不同的算法均是基于在不同情形下能达到尽可能低的时间复杂度的考虑。

## **1.3 功能设计与用户场景匹配**

### **1.3.1 登录、注册**

学生首次使用该系统需要系统管理员为其注册账号，这点是为了防止用户进行恶意注册，同时也符合真实校园的情况。注册成功后可以登录使用系统中的其他功能。登录功能可以验证用户的账号和密码，保证系统安全性。

### 1.3.2 课程、活动、临时事物管理

学生可以通过该系统添加、查看、修改和删除自己的课程安排。管理员在后台，也可以看到课程的安排，可以随时发布新的课程和考试，或调整已有的课程安排。

学生可以通过该系统添加、查看、修改和删除自己的课外活动、临时事物安排。可以避免与课程或其他活动冲突，提高规划效率。系统可以推送活动前提醒，方便学生准时参加活动。

将课程表、临时事物、课外活动都用表格的形式呈现出来，便于查看时间，设计了冲突检测算法，可以避免课程冲突，提高学习效率。系统可以推送课前提醒，并给出导航，方便学生准时出发上课

### 1.3.3 提醒与闹钟

学生可以通过该系统添加、查看、修改和删除自己的闹钟提醒。系统会在每天晚上，每天早上，和事情开始前 1 小时推送事情提醒，方便学生准时参加活动。

### 1.3.4 地图与导航

地图建立在真实校园地图的基础上，当有导航请求时，会根据用户的需求采用不同的导航算法进行导航，同时导航具有可视化窗口，能够动态输出导航路线。

## 第二章 系统架构

### 2.1 系统模块

#### 2.1.1 模块划分

登录模块：用户登录和注册功能

用户模块：用户个人信息管理功能

课外活动模块：课外活动的添加、查看、修改和删除等功能

课程表模块：课程安排的添加、查看、修改和删除等功能

临时事务模块：临时事务的添加、查看、修改和删除等功能

闹钟提醒模块：提醒和导航功能

导航模块：路径规划和导航功能

时间模块：时间管理和换算功能

日志模块：系统日志和操作日志记录功能

管理员模块：系统管理和用户管理功能

#### 2.1.2 模块依赖关系

登录模块和用户模块存在依赖，用户登录需要验证用户信息。

课外活动模块、课程表模块和临时事务模块相互独立，但都依赖用户模块获取用户信息，此外，课外活动模块、课程表模块和临时事务模块相互关联，用以冲突。

闹钟提醒模块依赖课外活动模块、课程表模块和临时事务模块获取用户日程信息。

闹钟提醒模块依赖时间模块来判断提醒时间和具体的提醒信息。

导航模块依赖提醒模块获取用户当前导航目的地。

日志模块依赖系统的各个功能模块和时间模块记录相应的日志信息。

管理员模块可以管理用户的基本信息和课程表模块。

#### 2.1.3 模块内部结构

登录模块：包含登录注册、管理员窗口、用户窗口子模块

课程表模块：包含课程表管理、课程表窗口、用户信息子模块

课外活动模块：包含活动、活动管理、课外活动窗口、活动排序窗口、用户信息子模块

临时事物模块：包含临时事务、临时事物管理、临时事物窗口、临时事务排序窗口、用户信息子模块

闹钟提醒模块：包含模拟时钟、提醒表窗口、当前时间的提醒、所有的提醒、提醒按时间排序

日志模块：包含输出函数、模拟时钟、日志文件名

导航模块：包含地图信息初始化模块，地点名 hash 模块，单点到单点导航模块，单点到多点导航模块 1（点数 $\leq 10$ ），单点到多点导航模块 2（点数 $> 10$ ）



### 2.1.4 模块接口定义

#### 登录模块

登录验证函数 `authenticate()`

管理员窗口内，添加学生函数 `addStudent()`，删除学生函数 `deleteStudent()`，修改学生信息函数 `modifyStudent()`

用户窗口内，展示学生界面函数 `StudentWindow()`

#### 课程表模块

表格化展示课表函数 `createCourseTable()`，更新课表函数 `updateCourseTable()`，添加课程函数 `onAddCourseClicked()`，查找课程函数 `onFindCourseClicked()`，修改和删除课程函数 `onCourseCellClicked()`。

#### 课外活动模块

表格化展示活动函数 `createTempTable()`，更新活动函数 `updateTempTable()`，添加活动函数 `onAddTempClicked()`，查找活动函数 `onFindTempClicked()`，修改和删除活动函数 `onTempCellClicked()`

#### 临时事物模块

表格化展示事物函数 `createActivityTable()`，更新活动函数 `updateActivityTable()`，添加活动函数 `onAddActivityClicked()`，查找活动函数 `onFindActivityClicked()`，修改和删除活动函数 `onActivityCellClicked()`

#### 闹钟提醒模块

暂停函数 `PAUSE()`，继续函数 `CONTINUE()`，添加删除更改提醒函数 `add\del\change`，获取当前周数、星期、小时等信息的函数 `getXXX()`。提醒结构体 `Alarm`。

#### 日志模块

日志文件的名字 `filename`，日志输出函数 `log()`，时钟变量 `clk_time`

#### 导航模块

单点到单点导航函数 `dijkstra()`，单点到多点导航函数 `tabuSearch()`，单点到多点导航函数 `dfsTsp()`

显示单点到单点导航结果函数 `show_dijk()`，显示多点到多点导航结果函数 `show_dfs()`，显示多点到多点导航结果函数 `show_tabu()`;

## 2.2 数据描述

### 2.2.1 用户表

这里是 `UserAccount.txt` 的数据格式：

	学号（账号）	密码	姓名
类型	string	string	string
长度	任意	任意	任意
是否可为空	不	不	不
格式说明	纯数字学号	任意字符	任意字符

样例数据：

2021212171 123456 ZhangSan

2021212484 123 LiSi

数据格式设计思路：

学号和密码作为登录信息，不能为空。姓名作为辅助信息，也不能为空。

学号作为账号，应当唯一。故类型选 string，长度为 10，以学号规则限定长度。

密码类型选 string，可以存放加密后的密码信息。

姓名类型选 string，可以容纳大多数姓名。

数据格式统一，如姓名的格式控制为“姓+名”。

txt 格式设计以后续的数据操作和查询为主。如提供学号和密码可以登录系统等。

txt 中只提供必要的用户信息，不宜过度设计导致冗余。可以在系统中根据需求增加更丰富的用户信息。

这里是 AdminAccount.txt 的数据格式

	账号	密码
类型	string	string
长度	任意	任意
是否可为空	不	不
格式说明	任意字符	任意字符

样例数据：

admin 123456

数据格式设计思路：

账号和密码作为管理员登录信息，不能为空。

账号应当唯一，用于管理员识别。类型选 string。

密码类型选 string，可以存放加密后的密码信息。

表中只包含一个管理员账号，方便管理和操作。如果需要可以添加更多管理员账号，但账号应唯一。

表结构设计以后续的管理员登录和操作为主。

表中只提供必要的管理员账号信息，不宜过度设计导致冗余。

可以在系统中根据需求，为每个管理员账号分配不同的权限等更详细的信息。

### 2.2.2 课程表

这里是 courses.txt 的数据格式：

	课程名字	上课周	星期几上课	第几节上课	老师	上课地点	考试周	星期几考试	第几节考试	考试地点
类型	string	int	string	int	string	string	int	int	int	string
长度	任意	整数	汉字编码的长度 2	整数	任意	任意	整数	整数	整数	整数
是否可为空	不	不	不	不	不	不	不	不	不	不
格式说明	课程名	起始周-结束周	周[一二三四五六日]	开始节-结束节	老师名	地点名	考试周	周[一二三四五六日]	开始节-结束节	地点名

样例数据：

计网 1-9 周二 6-7 高占春 教三楼中心 17 周一 3-4 教三楼中心

数电 1-16 周三 3-4 金老师 主楼 17 周一 1-2 主楼

数据格式设计思路：

数据标准化，上课周、考试周数使用数字，时间使用“节次-节次”格式。

确保每个字段不为空，以免出现 Null 值。

字段长度考虑实际需要，不宜过长或过短。

字段名要便于理解，反映其含义。

表中只包含一个学期的课程信息，便于查询。

### 2.2.3 活动表

这是集体活动 group\_activities.txt 的数据格式

	活动名字	活动周	星期几活动	第几节活动	活动地点
类型	string	int	string	int	string
长度	任意	整数	汉字编码的长度 2	整数	任意

是否可为空	不	不	不	不	不
格式说明	课程名	起始周-结束周	周[一二三四五六日]	开始节-结束节	地点名

样例数据:

聚餐 1-15 周五 8-10 北门

小组作业 1-15 周三 6-7 图书馆

吃麦麦 1-15 周一 12-13 麦当劳

这是个人活动 XX\_activities.txt 的数据格式

	活动名字	活动周	星期几活动	第几节活动	活动地点
类型	string	int	string	int	string
长度	任意	整数	汉字编码的长度 2	整数	任意
是否可为空	不	不	不	不	不
格式说明	活动名	起始周-结束周	周[一二三四五六日]	开始节-结束节	地点名

样例数据:

晨跑 1-16 周四 1-1 体育场

吃早点 1-16 周四 2-2 综合食堂西门

数据格式设计思路:

根据题目要求, 这里需要两个实体, 一个是 group\_activities 表示集体活动, 一个是个人活动 XX\_activities

确定属性: 每个实体都有活动名字、活动周、星期几活动、第几节活动和活动地点这 5 个属性

这里可以用活动名字作为主键, 因为同一个活动的其他属性值是唯一的。

group\_activities 和 XX\_activities 是两种不同的实体, 之间没有关系, 它们的表结构和属性都一致, 只是记录的活动数据不同。

### 2.2.4 临时事物表

这是临时事物 XX\_tempevents.txt 的数据格式

	事物名字	事物周	星期几	第几节	分钟	事物地点
类型	string	int	string	int	int	string
长度	任意	整数	汉字编码的长度 2	整数	整数	任意
是否可为空	不	不	不	不	不	不
格式说明	事物名	起始周-结束周	周[一二三四五六日]	开始节-结束节	整数，范围 0-59	地点名

样例数据：

找朋友 1 1 周一 16 2 学五公寓

干杂工 1 1 周一 16 8 科研楼

吃夜宵 1 1 周一 16 1 学生食堂西门

晚跑 1 1 周一 16 30 体育场

休息 1 会 1 周一 16 5 学三公寓

数据结构设计思路：

根据题目要求，每个人都要有自己的临时事物安排，定义为 XX\_tempevents.txt

确定属性：每个实体都有事物名字、事物周、星期几、第几节，分钟和事物地点这 6 个属性，设计分钟是考虑到一个时间段内，可能会有多个临时事物进行  
这里可以用事物名字作为主键，因为同一个事物的其他属性值是唯一的

### 2.2.5 日志表和提醒表

日志输出格式表

输出信息	操作时间	用户操作
数据类型	string	string
输出格式	[周数 星期 小时]	用户+操作类型+数据

日志输出样例：

This is our log.

[0 0 2] :: 登陆成功，为用户

[0 0 2] :: 登录 2021211899 jlt

[0 0 3] :: 用户暂停计时  
 [0 0 3] :: 用户继续计时  
 [0 0 8] :: 用户查找课程: 书店  
 [0 0 10] :: 用户查找课程: 数电  
 [0 0 10] :: 用户查找课程: 数电  
 [0 0 12] :: 用户获取提醒列表  
 [0 0 14] :: 用户暂停计时

提醒格式表

数据信息	提醒序号	活动名字	活动时间	活动地点
数据类型	int	string	string	string
数据格式	x	name	周数+星期+时	place

提醒输出样例:

提醒序号	活动名字	活动时间	活动地点
0	写报告	1 周一早上八点	教三楼中心位置
1	取快递	2 周三下午六点	中邮驿站
2	晚跑	3 周五晚上八点	体育馆

## 2.2.6 地图表

这是地图点集 node.txt 的数据格式

	地点名	地点坐标
类型	string	pair<int, int>
长度	任意	整数对
是否可为空	不	不
格式说明	地点名	(x, y)

样例数据:

北邮锦江酒店 (115, 128)  
 学十一公寓 (155, 128)  
 学十公寓西门 (169, 128)  
 学十公寓南门 (197, 158)  
 学十公寓东门 (235, 121)  
 学九公寓东门 (169, 142)  
 学九公寓南门 (146, 158)

这是地图边集 edge.txt 的数据格式

	地点名	地点名
类型	string	string
长度	任意	任意
是否可为空	不	不
格式说明	地点名	地点名

样例数据：

北邮锦江酒店~学十一公寓

学十一公寓~北邮锦江酒店

学十一公寓~学十公寓西门

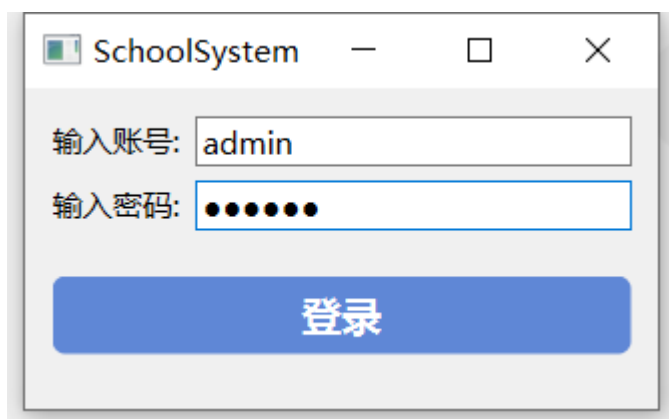
学十公寓西门~学十一公寓

学十公寓西门~路口 1

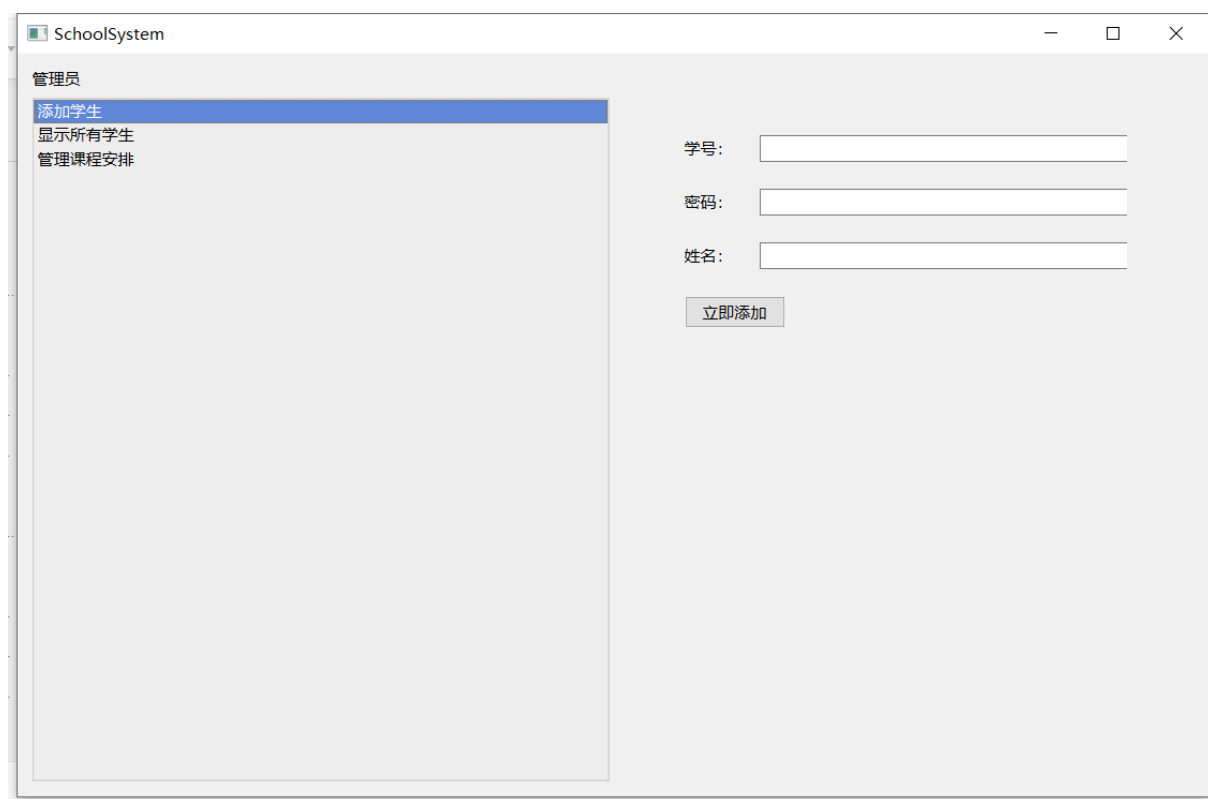
## 2.3 业务流程

### 2.3.1 管理员后台管理流程和课表管理流程

在登录界面输入管理员账号和密码（存储在 AdminAccount.txt 中），如图



登录成功后，进入管理员界面



“添加学生”界面，可以进行添加学生，如图

This is a close-up view of the "Add Student" form. It contains three input fields: "学号:" (Student ID) with the value "123456789", "密码:" (Password) with masked characters represented by seven dots, and "姓名:" (Name) with the value "test". Below the fields is a button labeled "立即添加" (Add Immediately).

点击“立即添加”，系统就会创建好一个学生账户，存储在 UserAccount.txt 中



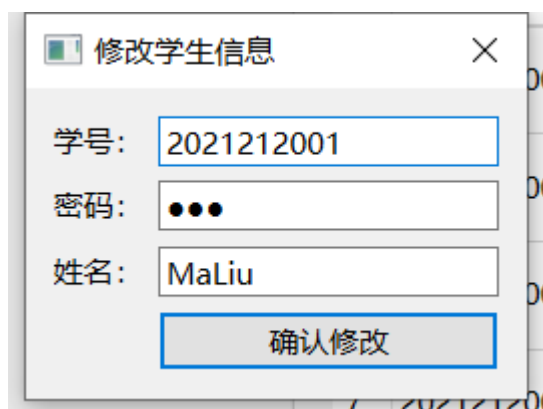
UserAccount.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

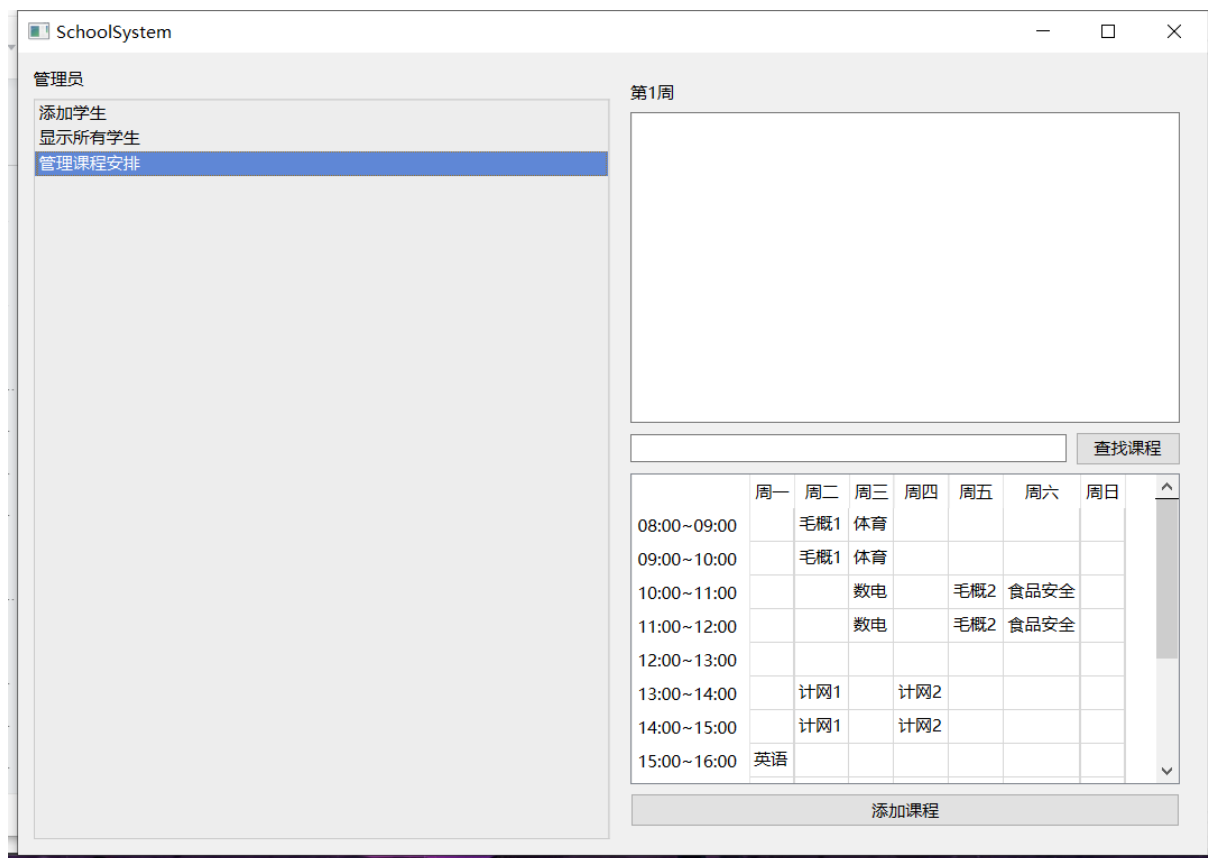
```
2021212171 123456 ZhangSan
2021212484 123 LiSi
2021211899 jlt jlt
2021212000 123 WangWu
2021212001 123 MaLiu
2021212002 123 LiFei
2021212003 123 Tom
2021212004 123 Jerry
2021212005 123 Mike
2021212006 123 yyy
123456789 123456 test
```



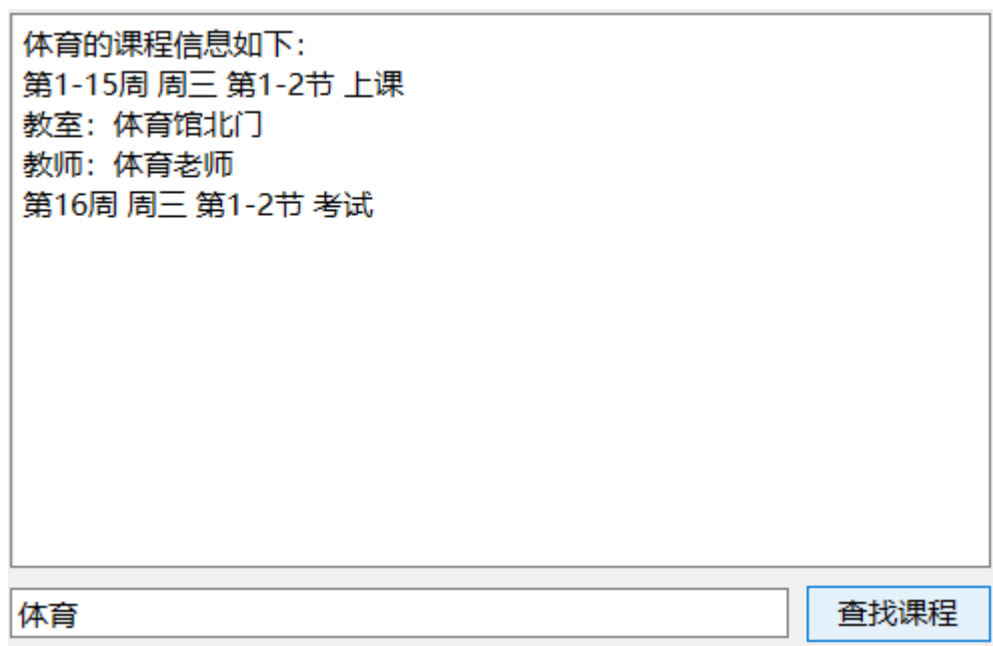
在“显示所有学生”界面里，管理员可以看到系统中存储的所有学生的信息



可以对任意一个学生进行修改和删除，同时 UserAccount.txt 中，会同步更新  
在“管理课程安排界面”，管理员可以查看系统中已有的所有课程



可以在“查找框”输入课程名字，点击“查找课程”按钮，就可以查看课程的详细信息

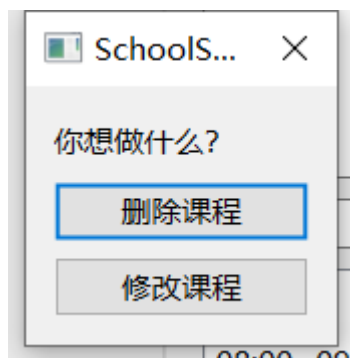


添加“添加课程”按钮，可以进行添加课程，courses.txt 文件会同步更新



The image shows a Windows-style dialog box titled "添加课程" (Add Course). It contains several input fields and spinners for course details. The fields are: "课程名称:" (Course Name) with a text box containing "课程名称"; "教师姓名:" (Teacher Name) with a text box containing "教师姓名"; "教室:" (Classroom) with a text box containing "教室"; "开始周:" (Start Week) with a spinner box containing "1"; "结束周:" (End Week) with a spinner box containing "1"; "星期几上课:" (Day of the week) with a spinner box containing "1"; "开始时间 (节):" (Start Time) with a spinner box containing "1"; "结束时间 (节):" (End Time) with a spinner box containing "1"; "考试时间 (周):" (Exam Week) with a spinner box containing "1"; "考试日期 (星期几):" (Exam Date) with a spinner box containing "1"; "考试开始时间 (节):" (Exam Start Time) with a spinner box containing "1"; "考试结束时间 (节):" (Exam End Time) with a spinner box containing "1"; and "考试教室:" (Exam Classroom) with a text box containing "考试教室". At the bottom, there are two buttons: "OK" and "Cancel".

点击表格中，某一个课程，会弹出修改和删除的对话框

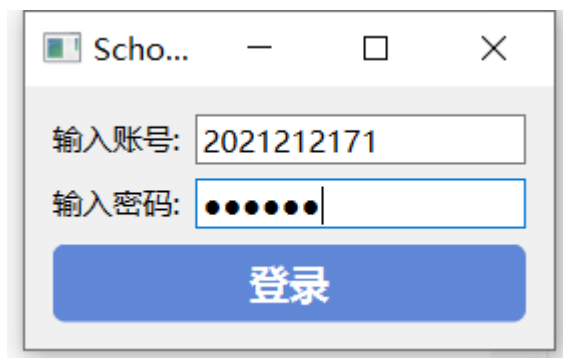


The image shows a small Windows-style dialog box titled "SchoolS...". It contains the text "你想做什么?" (What do you want to do?). Below the text are two buttons: "删除课程" (Delete Course) and "修改课程" (Modify Course). The "删除课程" button is highlighted with a blue border.

修改和删除时，courses.txt 会同步更新

### 2.3.2 课外活动管理流程

退出登录后，使用学生的学号和密码进行登录（存储在 UserAccount.txt 中）



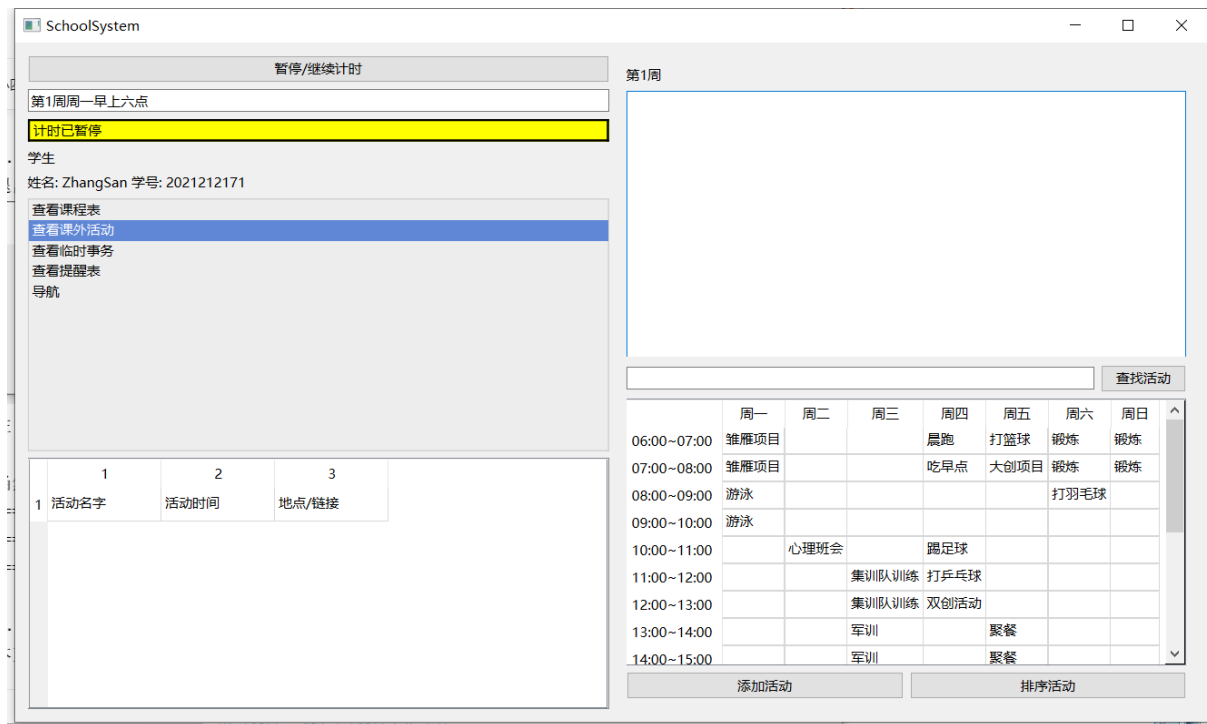
Scho...

输入账号: 2021212171

输入密码: ●●●●●●

登录

在“查看活动”界面，可以看到所有的活动安排



SchoolSystem

暂停/继续计时

第1周周一早上六点

计时已暂停

学生

姓名: ZhangSan 学号: 2021212171

查看课程表

查看课外活动

查看临时事务

查看提醒表

导航

1	2	3
活动名字	活动时间	地点/链接
1		

第1周

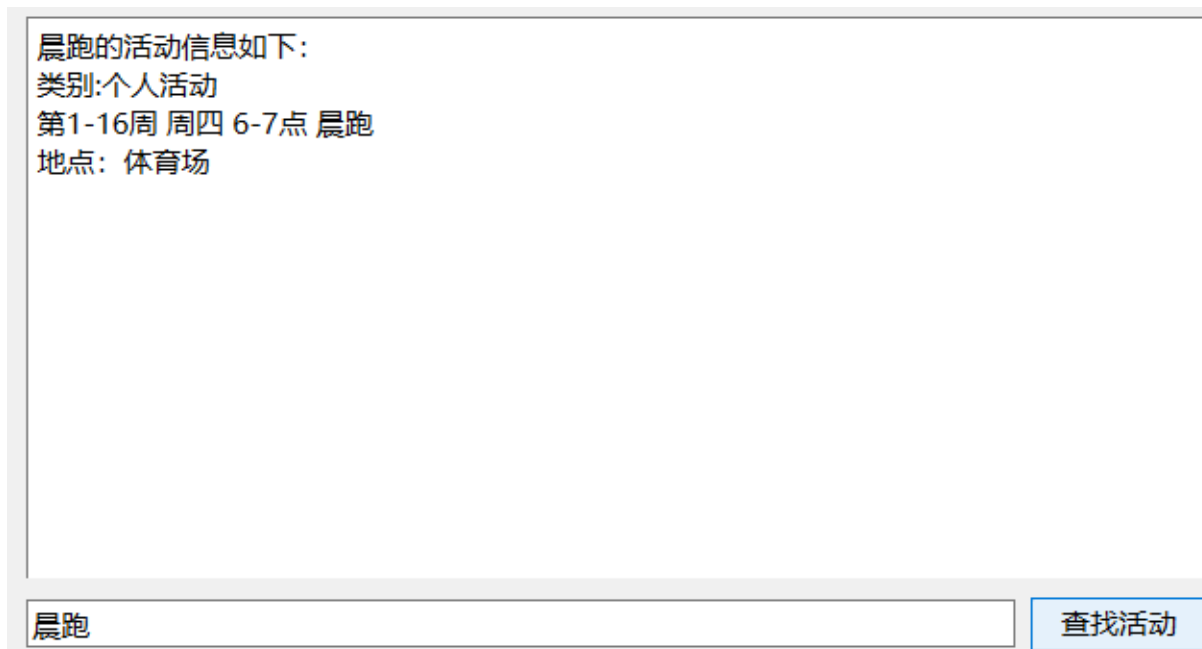
	周一	周二	周三	周四	周五	周六	周日
06:00~07:00	雏雁项目			晨跑	打篮球	锻炼	锻炼
07:00~08:00	雏雁项目			吃早点	大创项目	锻炼	锻炼
08:00~09:00	游泳					打羽毛球	
09:00~10:00	游泳						
10:00~11:00		心理班会		踢足球			
11:00~12:00			集训队训练	打乒乓球			
12:00~13:00			集训队训练	双创活动			
13:00~14:00			军训		聚餐		
14:00~15:00			军训		聚餐		

查找活动

添加活动

排序活动

在“查找框”输入活动名字，点击“查找活动”按钮，可以看到活动的详细信息



晨跑的活动信息如下:

类别:个人活动

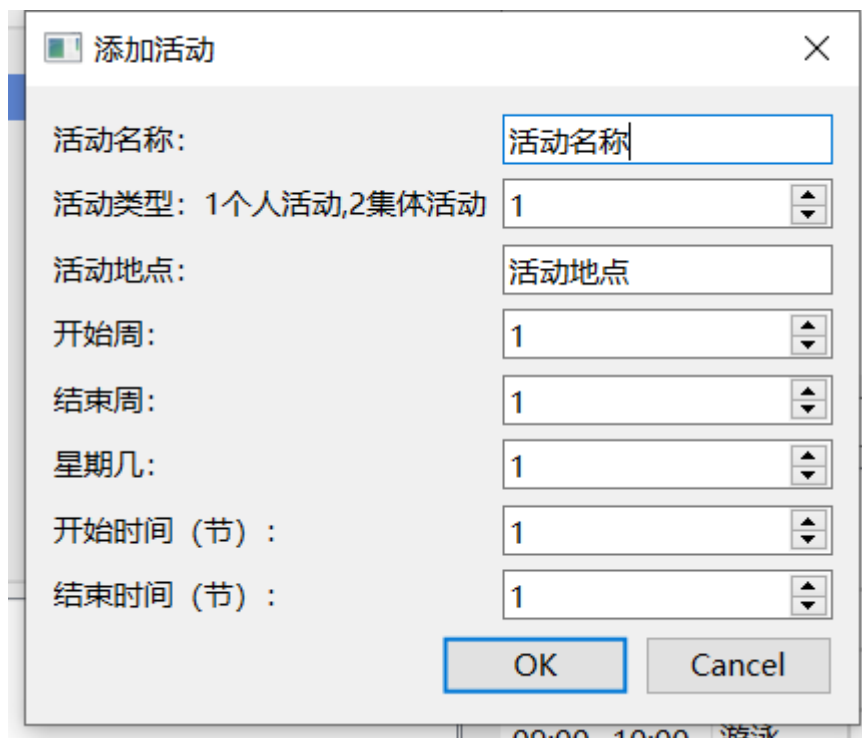
第1-16周 周四 6-7点 晨跑

地点: 体育场

晨跑

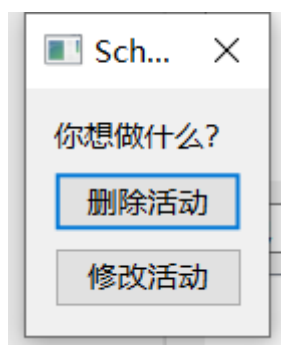
查找活动

点击“添加活动”按钮，可以添加新活动，txt 文件会同步更新



The image shows a Windows-style dialog box titled "添加活动" (Add Activity). It contains several input fields and dropdown menus for defining a new activity. The fields are: "活动名称:" (Activity Name) with a text box containing "活动名称"; "活动类型:" (Activity Type) with a dropdown menu showing "1" and a label "1个人活动,2集体活动"; "活动地点:" (Activity Location) with a text box containing "活动地点"; "开始周:" (Start Week) with a dropdown menu showing "1"; "结束周:" (End Week) with a dropdown menu showing "1"; "星期几:" (Which Day of the Week) with a dropdown menu showing "1"; "开始时间 (节):" (Start Time (Class)) with a dropdown menu showing "1"; and "结束时间 (节):" (End Time (Class)) with a dropdown menu showing "1". At the bottom are "OK" and "Cancel" buttons.

点击表格中，任意一项活动，可以进行修改或删除，txt 文件会同步更新



The image shows a small context menu titled "Sch..." with a close button (X). It contains two options: "删除活动" (Delete Activity) and "修改活动" (Modify Activity). The "删除活动" option is highlighted with a blue border.

点击“排序活动”按钮，可以对所有活动进行排序，可按时间排序，也可按类型排序

12 SchoolSystem

按类型排序

	活动名称	活动类型	开始时间	结束时间
1	晨跑	个人活动	第1周 周四 6:00	第16周 周四 6:00
2	吃早点	个人活动	第1周 周四 7:00	第16周 周四 7:00
3	雏雁项目	集体活动	第1周 周一 6:00	第15周 周一 7:00
4	游泳	集体活动	第1周 周一 8:00	第6周 周一 9:00
5	吃麦麦	集体活动	第1周 周一 17:00	第15周 周一 18:00
6	心理班会	集体活动	第1周 周二 10:00	第8周 周二 10:00
7	聚餐	集体活动	第1周 周二 15:00	第15周 周二 16:00
8	集训队训练	集体活动	第1周 周三 11:00	第15周 周三 12:00
9	军训	集体活动	第1周 周三 13:00	第15周 周三 14:00

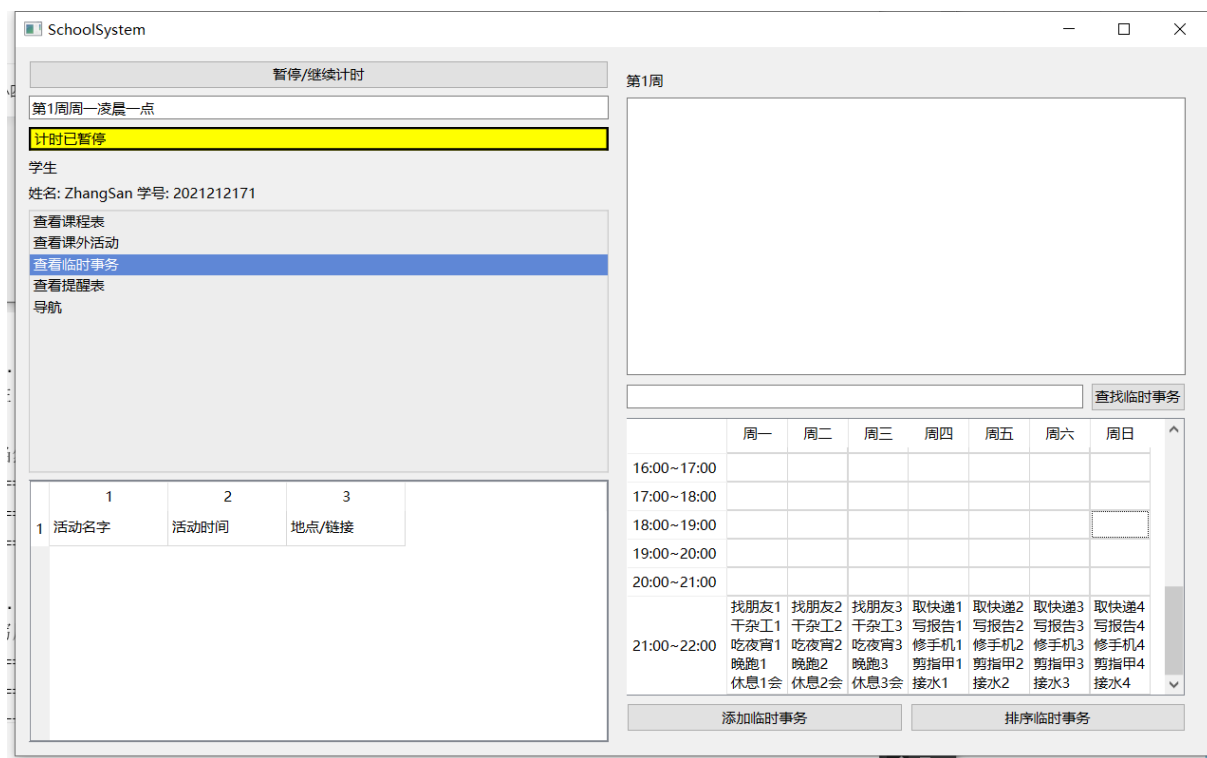
SchoolSystem

按时间排序

	活动名称	活动类型	开始时间	结束时间
1	雏雁项目	集体活动	第1周 周一 6:00	第15周 周一 7:00
2	游泳	集体活动	第1周 周一 8:00	第6周 周一 9:00
3	吃麦麦	集体活动	第1周 周一 17:00	第15周 周一 18:00
4	心理班会	集体活动	第1周 周二 10:00	第8周 周二 10:00
5	聚餐	集体活动	第1周 周二 15:00	第15周 周二 16:00
6	集训队训练	集体活动	第1周 周三 11:00	第15周 周三 12:00
7	军训	集体活动	第1周 周三 13:00	第15周 周三 14:00
8	晨跑	个人活动	第1周 周四 6:00	第16周 周四 6:00

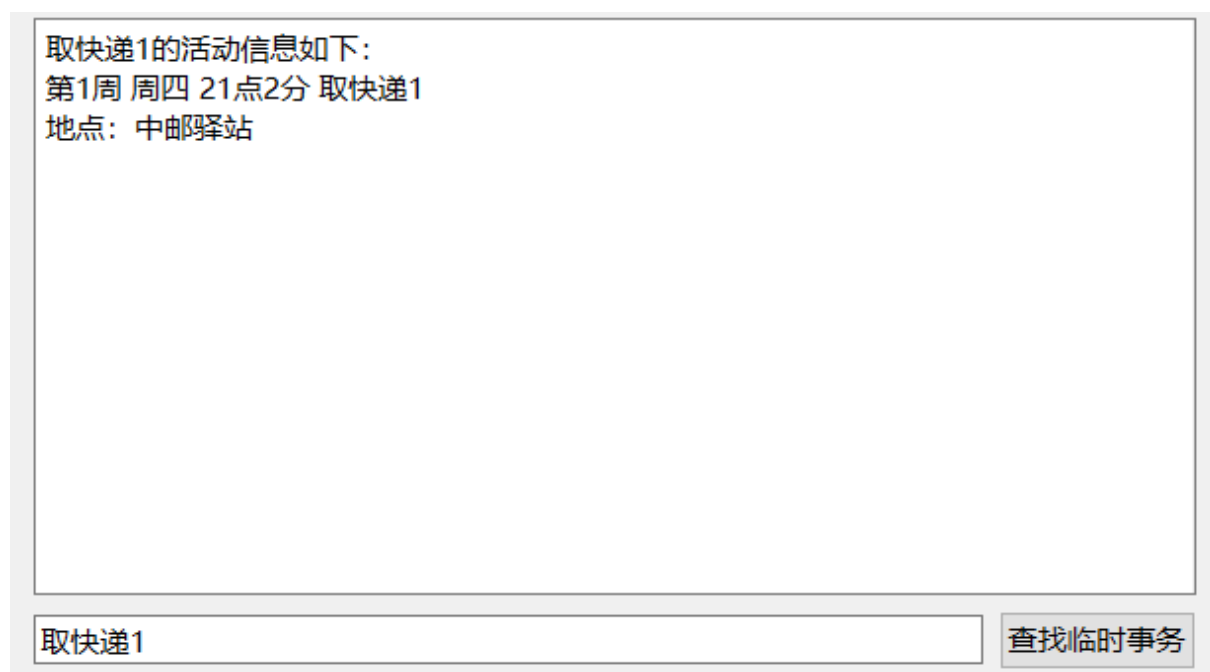
### 2.3.3 临时事物管理流程

在“查看临时事物”界面，学生可以查看所有的临时事物安排

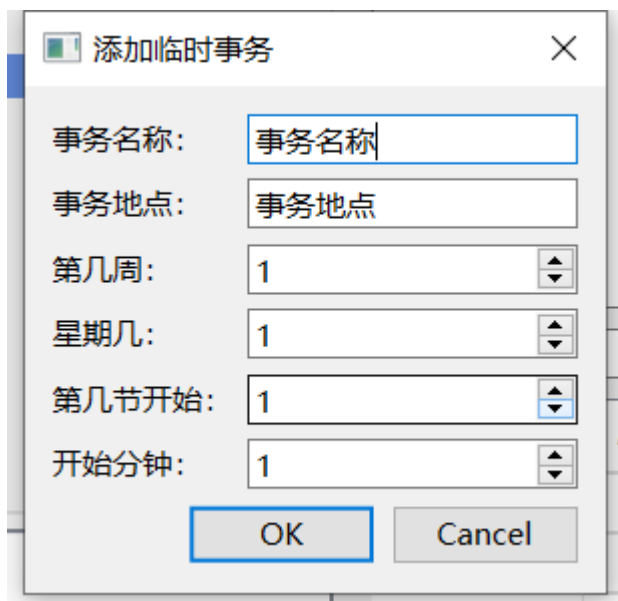


同一个时间段内，可以有多个临时事物

在“查找框”内输入临时事物名称，点击“查找临时事物”按钮，可以查看临时事物的详细信息

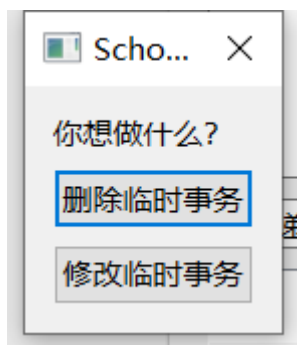


点击“添加临时事物”按钮，可以添加新的临时事物，txt 文件会同步更新



The screenshot shows a dialog box titled "添加临时事务" (Add Temporary Task). It contains several input fields: "事务名称:" (Task Name) with a text box containing "事务名称"; "事务地点:" (Task Location) with a text box containing "事务地点"; "第几周:" (Which Week) with a spinner box set to "1"; "星期几:" (Which Day of the Week) with a spinner box set to "1"; "第几节开始:" (Which Class Starts) with a spinner box set to "1"; and "开始分钟:" (Start Minute) with a spinner box set to "1". At the bottom, there are "OK" and "Cancel" buttons.

点击表格中，某个临时事物，可以对临时事物进行修改和删除，txt 文件会同步更新



The screenshot shows a dialog box titled "Scho...". It contains the text "你想做什么?" (What do you want to do?). Below this text, there are two buttons: "删除临时事务" (Delete Temporary Task) and "修改临时事务" (Modify Temporary Task). The "删除临时事务" button is highlighted with a blue border.

点击“排序临时事物”按钮，可以对所有的临时事物进行排序，可以按照类型（字典序），也可以按照时间排序

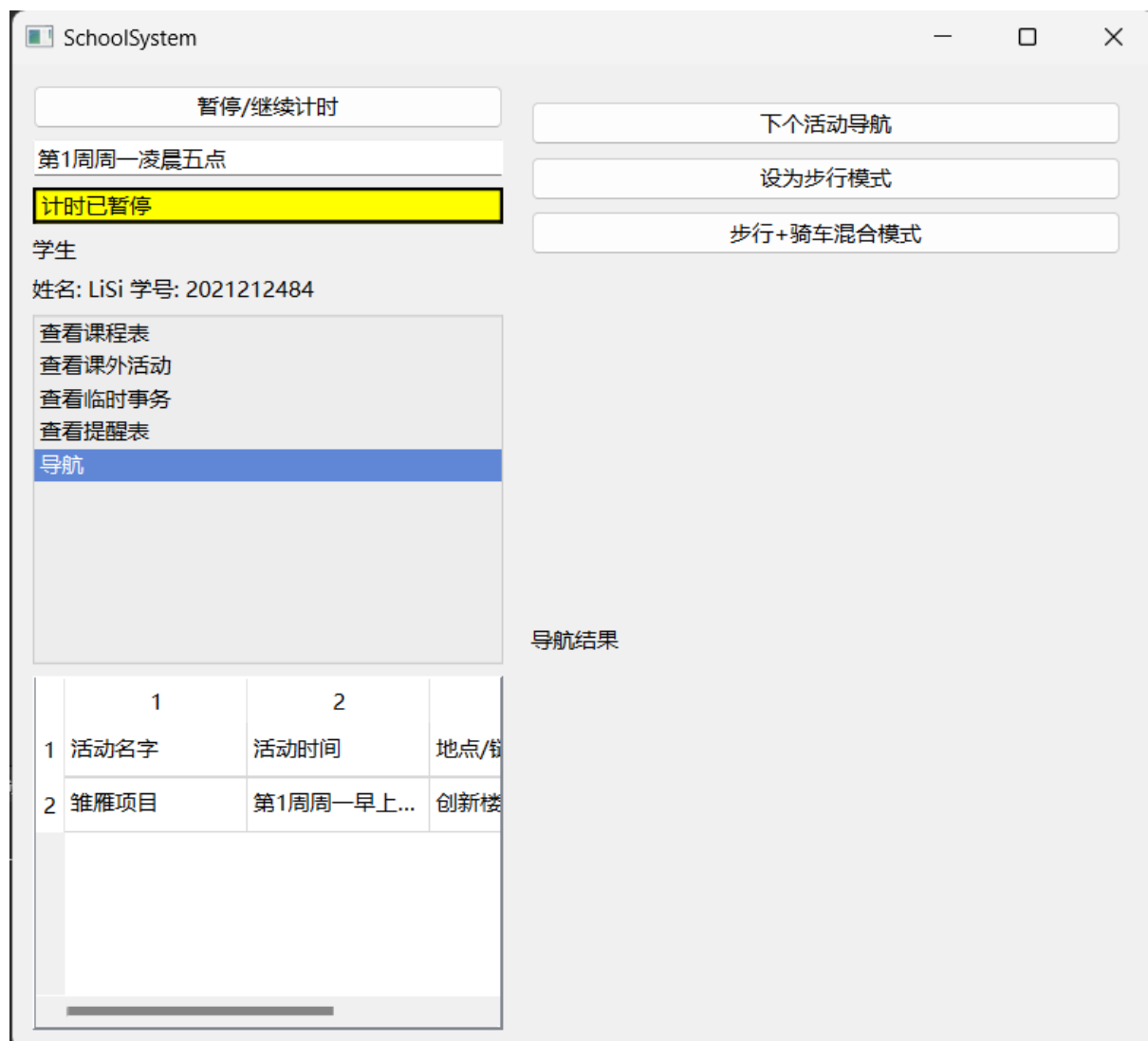


SchoolSystem			
按类型排序			
	事物名称	事物时间	事务地点
1	休息1会	第1周 周一 21点5分	学三公寓
2	休息2会	第1周 周二 21点5分	学三公寓
3	休息3会	第1周 周三 21点5分	学三公寓
4	修手机1	第1周 周四 21点1分	物美超市
5	修手机2	第1周 周五 21点1分	物美超市
6	修手机3	第1周 周六 21点1分	物美超市
7	修手机4	第1周 周日 21点1分	物美超市
8	写报告1	第1周 周四 21点8分	主楼

SchoolSystem			
按时间排序			
	事物名称	事物时间	事务地点
1	吃夜宵1	第1周 周一 21点1分	学生食堂西门
2	找朋友1	第1周 周一 21点2分	学五公寓
3	休息1会	第1周 周一 21点5分	学三公寓
4	干杂工1	第1周 周一 21点8分	科研楼
5	晚跑1	第1周 周一 21点30分	体育场
6	吃夜宵2	第1周 周二 21点1分	学生食堂西门
7	找朋友2	第1周 周二 21点2分	学五公寓
8	休息2会	第1周 周二 21点5分	学三公寓

### 2.3.4 导航流程

在“导航”界面，学生可以查看前往下一次事物地点的导航



在显示导航结果前可以选择导航模式，导航模式分为步行模式和步行+骑车混合模式两种。

选择好导航模式后，点击“下个活动导航按钮”，就会弹出动态的导航路线。

下面是步行模式下的导航结果



下面是步行+骑车混合模式下的导航结果



其中红色路线部分表示采用骑车方式，灰色路线部分表示采用步行方式。

同时导航界面的左下角可以实时显示出道路拥塞度，道路拥塞度是选择骑车还是步行的一个重要指标。

### 2.3.5 时钟轮和闹钟提醒流程

在系统运行时，首先点击获取“提醒列表”

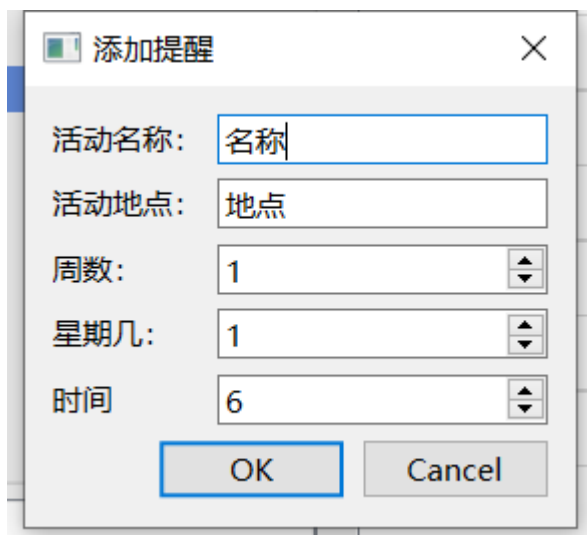
提醒序号	活动名字	活动时间	地点/链接
1	雏雁项目	1周一早上六点	创新楼
2	游泳	1周一早上八点	游泳馆
3	英语	1周一下午三点	教一楼西门
4	晚跑1	1周一晚上九点	体育场
5	找朋友1	1周一晚上九点	学五公寓
6	吃夜宵1	1周一晚上九点	学生食堂西门
7	干杂工1	1周一晚上九点	科研楼
8	休息1会	1周一晚上九点	学三公寓
9	毛概1	1周二早上八点	教三楼中心位置
10	心理班会	1周二上午十点	教三楼中心位置
11	计网1	1周二下午一点	教三楼中心位置
12	聚餐	1周二下午三点	北门
13	干杂工2	1周二晚上九点	科研楼
14	吃夜宵2	1周二晚上九点	学生食堂西门
15	晚跑2	1周二晚上九点	体育场
16	休息2会	1周二晚上九点	学三公寓
17	找朋友2	1周二晚上九点	学五公寓

添加提醒
删除提醒
修改提醒
获取提醒列表

用户添加提醒:

用户通过点击应用界面的按钮添加新的提醒。

用户提供提醒的具体时间、日期、提醒的名字与地点。系统将新的提醒存储在提醒表中。

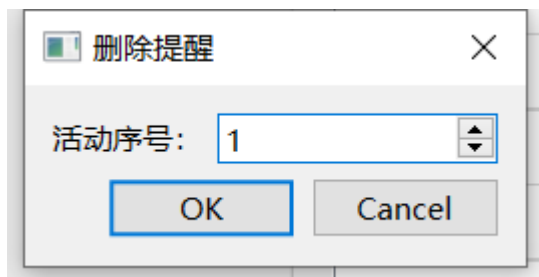


A dialog box titled "添加提醒" (Add Reminder) with a close button (X) in the top right corner. It contains five input fields: "活动名称:" (Activity Name) with the text "名称", "活动地点:" (Activity Location) with the text "地点", "周数:" (Week Number) with the value "1", "星期几:" (Day of the Week) with the value "1", and "时间:" (Time) with the value "6". At the bottom are two buttons: "OK" and "Cancel".

用户删除提醒:

用户通过点击应用界面的按钮选择要删除的提醒。

系统从提醒表中移除该提醒。

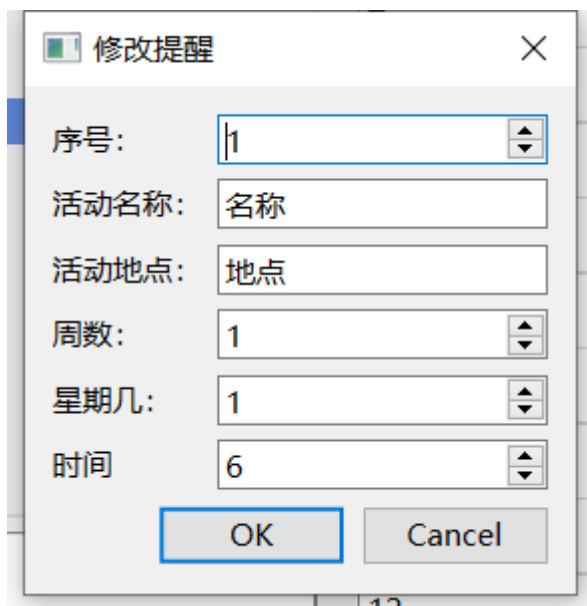


A dialog box titled "删除提醒" (Delete Reminder) with a close button (X) in the top right corner. It contains one input field: "活动序号:" (Activity ID) with the value "1". At the bottom are two buttons: "OK" and "Cancel".

用户更改提醒:

用户通过点击应用界面的按钮选择要更改的提醒。

用户提供新的时间、日期或内容等更新后的信息。系统更新提醒表中相应提醒的信息。



A dialog box titled "修改提醒" (Modify Reminder) with a close button (X) in the top right corner. It contains six input fields: "序号:" (ID) with the value "1", "活动名称:" (Activity Name) with the text "名称", "活动地点:" (Activity Location) with the text "地点", "周数:" (Week Number) with the value "1", "星期几:" (Day of the Week) with the value "1", and "时间:" (Time) with the value "6". At the bottom are two buttons: "OK" and "Cancel".

获取提醒表内容:



用户通过点击应用界面的按钮请求获取提醒表的内容。  
系统返回提醒表中所有提醒的列表。

提醒序号	活动名字	活动时间	地点/链接
1	雏雁项目	1周一早上六点	创新楼
2	游泳	1周一早上八点	游泳馆
3	英语	1周一下午三点	教一楼西门
4	晚跑1	1周一晚上九点	体育场
5	找朋友1	1周一晚上九点	学五公寓
6	吃夜宵1	1周一晚上九点	学生食堂西门
7	干杂工1	1周一晚上九点	科研楼
8	休息1会	1周一晚上九点	学三公寓
9	毛概1	1周二早上八点	教三楼中心位置
10	心理班会	1周二上午十点	教三楼中心位置
11	计网1	1周二下午一点	教三楼中心位置
12	聚餐	1周二下午三点	北门
13	干杂工2	1周二晚上九点	科研楼
14	吃夜宵2	1周二晚上九点	学生食堂西门
15	晚跑2	1周二晚上九点	体育场
16	休息2会	1周二晚上九点	学三公寓
17	找朋友2	1周二晚上九点	学五公寓

添加提醒
删除提醒
修改提醒
获取提醒列表

事件发生前一小时提醒：

系统每隔一段时间检查当前时间与提醒表中提醒时间的差距。

如果有提醒的时间距离当前时间正好一个小时，系统发送提醒通知给用户。

第	取 策 地	取	16	17	18	19	20

在每天的固定时间，系统检查提醒表中当天日期与当前日期相符的提醒。系统发送提醒通知给用户，展示当天的活动。



暂停/继续计时

第1周周一早上八点

计时已暂停

学生  
姓名: ZhangSan 学号: 2021212171

查看课程表

查看课外活动

查看临时事务

查看提醒表

导航

	1	2	3
1	活动名字	活动时间	地点/链接
2	游泳	第1周周一早上八点	游泳馆
3	雏雁项目	第1周周一早上六点	创新楼
4	游泳	第1周周一早上八点	游泳馆
5	英语	第1周周一下午三点	教一楼西门
6	晚跑1	第1周周一晚上九点	体育场
7	干杂工1	第1周周一晚上九点	科研楼

每日提醒明日活动:

在每天的固定时间,系统检查提醒表中明天日期与当前日期相符的提醒。

系统发送提醒通知给用户,展示明天的活动。

暂停/继续计时

第1周周一晚上八点

计时已暂停

学生

姓名: ZhangSan 学号: 2021212171

[查看课程表](#)  
[查看课外活动](#)  
[查看临时事务](#)  
[查看提醒表](#)  
[导航](#)

	1	2	3
1	活动名字	活动时间	地点/链接
2	毛概1	第1周周二早上八点	教三楼中心位置
3	心理班会	第1周周二上午十点	教三楼中心位置
4	计网1	第1周周二下午一点	教三楼中心位置
5	聚餐	第1周周二下午三点	北门
6	干杂工2	第1周周二晚上九点	科研楼
7	吃夜宵2	第1周周二晚上九点	学生食堂西门

时间轮暂停:

用户通过应用界面或命令请求将时间轮暂停。

系统停止发送提醒通知和执行每日提醒功能。

暂停/继续计时

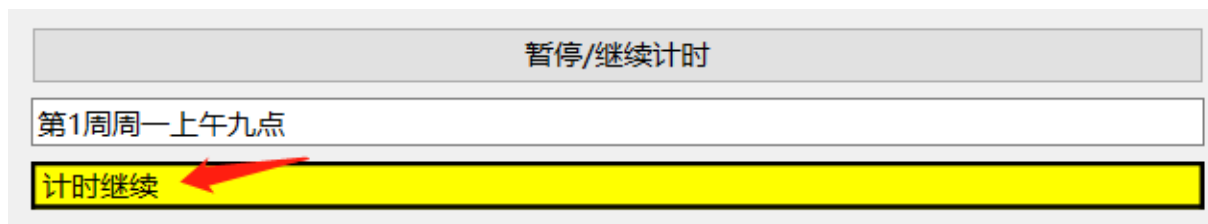
第1周周一上午十一点

计时已暂停

时间轮继续:

用户通过应用界面或命令请求将时间轮继续。

系统恢复发送提醒通知和执行每日提醒功能。



## 第三章 算法设计和算法性能分析

### 3.1 算法设计思路

#### 3.1.1 哈希算法

问题分析：学生需要通过输入课程（课外活动、临时事物）名字，查找到对应课程（课外活动、临时事务）的详细信息

思路设计：考虑使用哈希查找，构建一个以 `string-vector<string>` 为键值对的哈希表

```
// 哈希表，用于存储课程信息
```

```
MyMap<string, vector<string>> courseMap; // 课程名字到课程信息的映射
```

算法实现:

```
// 定义模板类 MyMap
template<typename K, typename V>
class MyMap {
private:
    // 定义默认桶数量
    static const size_t defaultBucketCount = 10;

    // 定义内部结构 KeyValuePair, 用于存储键值对
    struct KeyValuePair {
        K key;
        V value;

        KeyValuePair(const K& key, const V& value) : key(key), value(value) {}
    };

    // 使用 vector 和 list 实现哈希表, 每个桶包含一个 list
    std::vector<std::list<KeyValuePair>> buckets;
    // 桶的数量
    size_t bucketCount;
    // 哈希表中元素的数量
    size_t itemCount;

    // 计算给定键的哈希值
    size_t hash(const K& key) const {
        std::hash<K> hasher;
        return hasher(key) % bucketCount;
    }
};
```

采用 c++11 带有的 `std::hash` 作为哈希函数, 能够很好地避免冲突, 冲突时, 采取的解决办法是链地址法, 用一个桶 `list` 来维护

### 3.1.2 快速排序算法

问题分析: 学生需要对课外活动 (临时事物) 进行排序, 更直观地查看活动的安排

思路设计: 考虑使用 c++11 自带的 `std::sort`, 根据不同的关键字, 进行排序

```
// 根据下拉框的选项对活动进行排序

if (index == 0) {
    // 按时间排序

    this->logtxt->log("用户按时间对课外活动排序 ");
    std::sort(activities.begin(), activities.end(),
        [](const Activity &a, const Activity &b) {
            if (a.start_week != b.start_week)
            {
                return a.start_week < b.start_week;
            }
            if (a.day != b.day)
            {
                return a.day < b.day;
            }
            return a.start_period < b.start_period;
        });
} else {
    // 按类型排序

    this->logtxt->log("用户按类型对课外活动排序 ");
    std::sort(activities.begin(), activities.end(),
        [](const Activity &a, const Activity &b) {
            if (a.activity_type != b.activity_type) {
                return a.activity_type < b.activity_type;
            }
            else
            {
                if (a.start_week != b.start_week)
                {
                    return a.start_week < b.start_week;
                }
                if (a.day != b.day)
                {
                    return a.day < b.day;
                }
                return a.start_period < b.start_period;
            }
        });
}
```

采用 c++ 自带的 `std::sort` 快速排序函数，可以极大程度地提高系统的稳定性，有效地节约排序时间

### 3.1.3 BKDRHash

问题分析：通过输入的地点名快速查找到地点在邻接表的表头中的下标

思路设计：采用发生冲突可能性较小的 BKDRHash+线性探测的冲突解决办法  
算法实现：

```
unsigned int Graph::BKDRHash(char* str)
{
    unsigned int seed = 131; // 31 131 1313 13131 131313 etc..
    unsigned int hash = 0;

    while (*str)
    {
        hash = hash * seed + (*str++);
    }

    return (hash & 0x7FFFFFFF) % N;
}

unsigned int Graph::getIndex(string name)
{
    char c[50];
    strcpy(c, name.c_str()); // 将 string 转为 char *
    unsigned int index = BKDRHash(c);
    while (array[index].place_name != name)
    {
        index = (index + 1) % N;
    }
    return index;
}
```

### 3.1.4 dijkstra

问题分析：找到单点到单点的最短路径

思路设计：采用经典的 dijkstra 算法+小根堆进行优化  
算法实现：

```
void Graph::dijkstra(string src, string dest, int mode)
{
    unsigned int s = getIndex(src);
    unsigned int d = getIndex(dest);
    if ((mode == 0 && dis1[s][d].cnt) || (mode == 1 && dis2[s][d].cnt)) // 若已经求得最短
    路径则直接输出
    {
        return;
    }

    // 小根堆优先队列
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq; // 小
    根堆维护距离
    pq.push(make_pair(0, s));
    // 将起点入队

    int temp_dis[N]; // 存放源点到各个结点的最短距离
    memset(temp_dis, INF, sizeof(temp_dis)); // 初始化距离为 INF
    temp_dis[s] = 0; // 到自己的距离为 0

    int visit[N]; // 记录是否已经找到到 i 的最短距离
    memset(visit, 0, sizeof(visit)); // 初始化为没有找到

    int prev[N]; // 记录最短路径的前驱节点,
    int transportation[N]; // 交通方式
    double congestion[N]; // 拥堵程度
    memset(prev, -1, sizeof(prev)); // 每个节点的前驱节点初始化为-1
    memset(transportation, 0, sizeof(transportation));
    memset(congestion, 0, sizeof(congestion));

    while (!pq.empty())
    {
        // 堆不空
        int u = pq.top().y;
        visit[u] = 1;
        pq.pop();
        Node* temp = array[u].head;
        default_random_engine e;
        uniform_int_distribution<int> d(50, 500);
        e.seed(time(0));
```

### 3.1.5 dfsTsp

问题分析：找到单点到多点的最短路径，点数较少

思路设计：这是一个典型的 TSP 问题，当点数较少时，可以采用 dfs 搜索枚举出所有可能的排列，找出最优解

算法实现：



```
// 利用 dfs 算法求解 TSP 问题

void Graph::dfs(vector<int>& nums, vector<int>& visited, vector<int>& permutation,
vector<int>& best_path, int& lowest_cost, int mode)
{
    if(permutation.size() == nums.size() + 1)
    {
        // 判断是否找到了一个更优解
        if(calcPathLen(permutation, mode) < lowest_cost)
        {
            best_path = permutation;
            lowest_cost = calcPathLen(permutation, mode);
        }
        return;
    }

    for(int i = 0; i < nums.size(); i++)
    {
        if(!visited[i])
        {
            visited[i] = 1;
            permutation.push_back(nums[i]);

            // 递归搜索
            dfs(nums, visited, permutation, best_path, lowest_cost, mode);

            // 回溯
            permutation.pop_back();
            visited[i] = 0;
        }
    }
}
```

```
vector<int> Graph::dfsTsp(string src, vector<string> dst, int mode)
{
    // 初始化当前解
    vector<int> cur_path;
    for(int i = 0; i < dst.size(); i++)
    {
        cur_path.push_back(getIndex(dst[i]));
    }

    vector<int> visited(cur_path.size(), 0); permutation, best_path;
```

### 3.1.6 tabuSearch

问题分析：找到单点到多点的最短路径，点数较多

思路设计：这是一个典型的 TSP 问题，当点数较多时，再使用 dfs 搜索枚举出所有可能的排列，找出最优解的话时间复杂度会十分爆炸，此时采用启发式的 tabu 算法进行求解，该算法在点数较少时性能不如 dfs，但是在点数较多时性能远高于 dfs

算法实现：

```
vector<int> Graph::tabuSearch(string src, vector<string> dst, int mode)
{
    // 初始化当前解
    vector<int> cur_path;
    cur_path.push_back(getIndex(src));
    for (int i = 0; i < dst.size(); i++)
    {
        cur_path.push_back(getIndex(dst[i]));
    }

    // 随机生成一个新排列
    // srand(unsigned(time(0))); // 更改随机种子
    random_shuffle(cur_path.begin() + 1, cur_path.end());
    vector<int> best_path = cur_path;

    // 初始化禁忌表和禁忌期
    memset(tabu, 0, sizeof(tabu));
    memset(tabulen, 0, sizeof(tabulen));

    // 设置算法参数
    int maxiter = 5000;

    // 迭代搜索
    for (int iter = 0; iter < maxiter; iter++)
    {
        // 生成候选解并选择其中的一个最优解
        vector<int> new_path;
        int cnt = 0;
        while (cnt < 500)
        {
            new_path = generateCandidate(cur_path);
            if (!isInTabuList(new_path))
            {
                break;
            }
            cnt++;
        }

        // 更新禁忌表和禁忌期
        updateTabuList(cur_path, new_path);
    }
}
```

## 3.2 算法复杂度分析

### 3.2.1 时间复杂度

#### 哈希算法

`insert(key, value)` 函数:

在 `hash` 函数中计算 `key` 的哈希值, 时间复杂度  $O(1)$ 。

在 `buckets` 中查找对应的 `list`, 时间复杂度  $O(1)$ 。

在 `list` 中查找是否存在相同的 `key`, 时间复杂度  $O(n)$ ,  $n$  为 `list` 的大小。

如果不存在, 向 `list` 尾部插入新元素, 时间复杂度  $O(1)$ 。

所以, `insert(key, value)` 函数的时间复杂度是  $O(n)$ 。

`at(key)` 函数:

计算 `key` 的哈希值, 时间复杂度  $O(1)$ 。

在 `buckets` 中查找对应的 `list`, 时间复杂度  $O(1)$ 。

在 `list` 中查找是否存在 `key`, 时间复杂度  $O(n)$ ,  $n$  为 `list` 的大小。

所以, `at(key)` 函数的时间复杂度也是  $O(n)$ 。

`erase(key)` 函数:

计算 `key` 的哈希值, 时间复杂度  $O(1)$ 。

在 `buckets` 中查找对应的 `list`, 时间复杂度  $O(1)$ 。

在 `list` 中查找是否存在 `key`, 时间复杂度  $O(n)$ ,  $n$  为 `list` 的大小。

如果存在, 从 `list` 中移除该元素, 时间复杂度  $O(1)$ 。

所以, `erase(key)` 函数的时间复杂度也是  $O(n)$ 。

其他函数的时间复杂度:

`operator[]`:  $O(n)$

`contains()`:  $O(n)$

`size()`:  $O(1)$

`begin()`:  $O(1)$

`end()`:  $O(1)$

所以总的来说, 这个哈希表的时间复杂度是  $O(n)$ 。对任意一个操作来说, 其时间复杂度与 `list` 的大小成正比。

在实际测试中, 由于 `c++11` 提供的 `std::hash` 具有很好的性能, 所以很少会出现 `list` 很长的情况, 因此, 实际测试中, 这个哈希表的查找效率非常高

#### 快速排序算法

`std::sort()` 使用快速排序算法, 其时间复杂度为  $O(n\log n)$ 。

其中,  $n$  为待排序序列的大小。

快速排序算法包含两步: 分区: 选择一个枢纽元, 并将所有小于枢纽元的元素放置到左边, 大于枢纽元的元素放置到右边。这一步的时间复杂度为  $O(n)$ 。

递归调用：递归对左右两边的序列进行快速排序。这一步的时间复杂度为  $2T(n/2) = O(n\log n)$ ， $T(n)$  为快速排序的时间复杂度。

所以，`std::sort()` 的总时间复杂度为  $O(n) + O(n\log n) = O(n\log n)$

### BKDRHash 算法

BKDRHash 是一种性能较优、冲突较少的字符串哈希算法。虽然我们采用的线性探测的冲突解决办法，但经过测试，对于地图地点名称来说，用该算法作为 hash 函数，几乎不会出现冲突的情况，所以算法复杂度近似  $O(1)$ 。

### dijkstra 算法

作为经典的最短路径求解算法，它的时间复杂度为  $O(n^2)$ ， $n$  为图的点数。但此处我们采用了小根堆的排序进行优化，可以让每一次找最近点的复杂度从  $O(n)$  降低到  $O(\log n)$ ，所以最终的算法复杂度为  $O(n\log n)$ 。

### dfsTsp 算法

假设求解 1 个点到  $n$  个点的最短路径。我们需要对这  $n$  个点找到它们的所有可能的排列，所需的时间复杂度为  $O(n!)$ ，同时我们需要用 dijkstra 算法求出这  $n$  个点加上起点中任意两个点之间的最短距离，所需的时间复杂度为  $O(n*(n-1)/2*n\log n)$ ，其中  $n*(n-1)/2$  表明有这么对点都需要采用 dijkstra 算法求出最短距离，而  $n\log n$  为每一次采用 dijkstra 算法的时间复杂度。实际上由于每采用一次 dijkstra 算法求出 A 到 B 点的最短距离，同时也求出了 A 到其余所有点的最短距离，同时我们采取了存储最优路径的策略，已经求解出的最短路径不会再次求解，因此实际复杂度会小于  $O(n*(n-1)/2*n\log n)$ 。所以整体算法的复杂度会小于  $O(n!+n*(n-1)/2*n\log n)$ 。

### tabuSearch 算法

根据前面我们对 dfsTsp 算法的分析，我们知道当  $n>10$  时，该算法的时间复杂度会十分大，算法性能急剧下降。因此对于  $n>10$  时，我们采用另一个性能较优的算法：tabu 搜索算法。该算法的时间复杂度和禁忌表的大小设置，以及禁忌周期长度设置有关，此处我们将禁忌表设置为  $125*125$ ，禁忌期设置为 5，此时算法复杂度能够稳定在百万级的常数水平。

## 3.2.2 空间复杂度

### 哈希算法

哈希表使用 vector 和 list 实现，vector 的空间复杂度是  $O(n)$ ，list 的空间复杂度也是  $O(n)$ 。

这个哈希表有一个固定大小的 buckets 数组，buckets 数组的大小由 bucketCount

指定，默认为 10。即使元素数量  $n$  很大，buckets 数组的大小也不会改变。

每个 bucket 包含一个 list，list 的大小会随着插入的元素数量增加而增加。所以实际上，空间复杂度主要取决于 list 们的总大小。

即使有哈希冲突，所有的元素也都会插入到列表中，所以空间复杂度仍然是  $O(n)$ 。

由于使用 list 实现，每个 bucket 中的元素数量不会超过  $n$ ，所以辅助空间的增长速率也是  $O(n)$ 。

### 快速排序算法

`std::sort()` 的空间复杂度主要来自递归调用栈，最坏情况下栈的深度可达  $O(\log n)$ ，所以空间复杂度为  $O(\log n)$ 。

除此之外，还需要  $O(1)$  的额外空间来存储枢纽元和游标。所以，总的空间复杂度为  $O(\log n)$ 。

### 3.2.3 算法时间复杂度实例分析

#### 哈希算法

`std::hash` 是 C++ 标准库提供的泛型哈希函数，它具有以下特点：

计算简单且高效，时间复杂度为  $O(1)$ 。

发生哈希冲突的概率很小，元素可以均匀地分布到各个桶中。

所以，考虑这两个特点，哈希表的时间复杂度实例分析如下：

`insert(key, value)` 函数：

哈希值计算时间： $O(1)$

查找桶时间： $O(1)$

在桶中的 list 中查找时间： $O(n/b)$ ， $b$  为桶的数量，因为元素均匀分布，每个桶中大约有  $n/b$  个元素。

插入时间： $O(1)$

所以，插入  $n$  个元素的总时间为： $O(n) * O(1) + O(n) * O(1) + O(n) * O(n/b) + O(n) * O(1) = O(n)$

`at(key)` 函数：

哈希值计算时间： $O(1)$

查找桶时间： $O(1)$

在桶中的 list 中查找时间： $O(n/b)$

所以，查找  $n$  个元素的总时间为： $O(n) * O(1) + O(n) * O(1) + O(n) * O(n/b) = O(n)$

`erase(key)` 函数：

哈希值计算时间： $O(1)$

查找桶时间： $O(1)$

在桶中的 list 中查找时间： $O(n/b)$

删除元素时间： $O(1)$

所以，删除  $n$  个元素的总时间为： $O(n) * O(1) + O(n) * O(1) + O(n) * O(n/b) + O(n) * O(1) = O(n)$

## 快速排序算法

`std::sort()` 使用快速排序算法，具体时间复杂度实例分析如下：

最坏情况：

快速排序的最坏时间复杂度为  $O(n^2)$ ，当序列初始状态就是有序的，或者枢纽元每次都选的是最小或最大元素。

此时，每次分区只得到一个非空子序列，递归层数达到  $n$  层，每层执行  $O(n)$  的工作，所以总时间复杂度为  $O(n^2)$ 。最好情况：

快速排序的最好时间复杂度为  $O(n\log n)$ ，当每次选取的枢纽元可以将序列分成两个等大的子序列。

此时，递归层数为  $\log n$  层，每层执行  $O(n)$  的工作，所以总时间复杂度为  $O(n\log n)$ 。平均情况：

快速排序的平均时间复杂度也是  $O(n\log n)$ 。这是因为：

- (1) 枢纽元的选取是随机的，平均情况下，每次可以将序列分成两个等大的子序列。
- (2) 即使枢纽元的选择不是最优的，到最底层的子问题的规模也是  $\log n$  量级的，所以总的时间复杂度仍为  $O(n\log n)$ 。所以，`std::sort()` 的时间复杂度实例分析如下：

最坏情况：  $O(n^2)$

最好情况：  $O(n\log n)$

平均情况：  $O(n\log n)$  综上，`std::sort()` 的时间复杂度主要取决于数据的初始状态，但平均情况下，时间复杂度是  $O(n\log n)$ ，表现很好。

## 3.3 算法正确性证明

### 3.3.1 直接证明法

#### 哈希算法

这个哈希表使用拉链法解决哈希冲突，将所有哈希值相同的元素存储在一个链表中。插入元素时，首先计算元素的哈希值，然后插入到对应链表的尾部。所以，元素可以正确地存储在哈希表中。

查找元素时，首先计算元素的哈希值，然后在对应的链表中顺序查找。如果找到，返回元素；如果未找到，抛出异常。所以，可以正确地查找元素。

删除元素时，首先计算元素的哈希值，然后在对应的链表中顺序查找。如果找到，删除该元素；如果未找到，抛出异常。所以，可以正确地删除元素。

由上可以推导，这个哈希表可以正确地执行插入、查找和删除操作，算法是正确的。

#### 快速排序算法

使用数学归纳法来证明快速排序算法的正确性：

当序列的大小为 1 时，序列已经有序，算法正确。

假设当序列的大小为  $n$  时，算法正确。

当序列的大小为  $n + 1$  时：

(1) 选择枢纽元，将大于枢纽元的元素移动到右边，小于枢纽元的元素移动到左边。  
则左序列的大小为  $i$ ，右序列的大小为  $n - i - 1$ 。

(2) 由归纳假设，左序列和右序列可以正确排序。

(3) 最后，将枢纽元放在中间，整个序列就被正确排序。

所以，通过数学归纳法可以证明，快速排序算法对任意大小的序列都可以正确排序，算法是正确的。

### 3.3.2 反证法

#### 哈希算法

假设这个哈希表的算法不正确，将导致：

插入元素时无法正确存储元素，导致元素丢失。但事实是，这个哈希表使用拉链法存储所有元素，不会导致元素丢失，与假设矛盾。

查找元素时无法正确查找元素。但事实是，这个哈希表使用元素的哈希值确定存储位置，再顺序查找对应的链表，可以正确查找，与假设矛盾。

删除元素时无法正确删除元素。但事实是，这个哈希表使用元素的哈希值确定存储位置，再顺序查找对应的链表，可以正确删除，与假设矛盾。

所以，通过反证法可以推导，这个哈希表的算法是正确的。



## 第四章 算法测试

### 4.1 算法效率测试

#### 4.1.1 测试用例设计

用例 1：检验哈希算法执行效率。使用不同数据量(100、1000 条)的输入，分别记录哈希算法的执行时间，评估其时间复杂度。

用例 2：检验快速排序算法执行效率。使用不同数据量(100、1000 条)的输入，分别记录快速排序算法的执行时间，评估其时间复杂度。

#### 4.1.2 测试用例执行记录

用例 1：

数据量：100 条，执行时间：0.05s

数据量：1000 条，执行时间：0.3s

时间复杂度分析： $O(n)$ ，但是由于 `std::hash` 具有较好的散列性能，实际测试中执行时间随数据量增长较慢，算法效率较高

用例 2：

数据量：100 条，执行时间：0.08s

数据量：1000 条，执行时间：1.2s

时间复杂度分析： $O(n\log n)$ ，执行时间随数据量增长速度较快，算法效率一般

#### 4.1.3 测试结果

退出码 0	用时 0.0523s
-------	------------

退出码 0	用时 0.3408s
-------	------------

退出码 0	用时 0.0802s
-------	------------

退出码 0	用时 1.255s
-------	-----------

## 第五章 总结

在合作开发这个项目和撰写这个项目设计报告的过程中，我们收获良多。

首先，这让我们进一步了解了软件工程的项目开发流程，了解了需求分析、设计思路、数据格式设计、界面设计、架构设计、算法设计等各个方面。

其次，在撰写具体功能模块时，我们用心思考问题，设计解决方案。这锻炼了我们的问题分析和解决能力。

再次，在撰写算法时，我们不仅要求解算法，还需要分析算法的复杂度和正确性。这让我思考的角度更多元、更科学。

此外，在撰写测试用例时，我们梳理了系统各个流程和功能，提高了系统整体的透视力。

总的来说，撰写这个项目设计报告让我们的技能更全面、能力更强大。我们不光学到了新的知识，更重要的是训练了自己的思考和解决问题的能力，打开了面向问题的思考角度。

这次经历让我们领悟到，好的项目设计不单单是一个程序，它更重要的意义在于训练我们的思维、提升我们的能力。这份报告讲解了知识，而最终更能激发内在的好奇与探索。

当然，我们还需要继续不断地学习和进步。只有不断的积累和挑战，才能成为一名出色的软件工程师。