

**数据结构课程设计**



**姓 名**  **邱天泽**

**学 院 国际学院**

**专 业 智能科学与技术**

**班 级 2022215123**

**学 号 2022213649**

**班内序号 10**

**指导教师 刘咏彬**

**2024年 4 月**

目录

[一、实验题目 3](#_Toc165240165)

[二、简介与数据结构 4](#_Toc165240166)

[三、算法和主要思想 5](#_Toc165240167)

[四、工作流程 5](#_Toc165240168)

[五、运行实例展示 8](#_Toc165240169)

**数据结构课程设计大作业**

# 一、实验题目

1. 问题描述：

当一个用户从甲地到乙地时，由于不同需求，就有不同的交通路线，有人希望以最短时间到达，有人希望用最少的换乘次数等。请编写一北京地铁线路查询系统，通过输入起始站、终点站，为用户提供两种决策的交通咨询。

1. 设计要求：

1. 提供对地铁线路进行编辑的功能，要求可以添加或删除线路。

2. 提供两种决策：最短时间，最少换乘次数。

3. 中途换乘站换乘耗时为5分钟，地铁在除始发站外每一站停留1分钟。

4. 按照始发站时间、地铁时速及停留时间推算之后各个线路的地铁到站时间。

5. 该系统以人机对话方式进行。系统自动获取当前时间，用户输入起始站，终点站以及需求原则（需求原则包括最短时间，最少换乘次数），系统输出乘车方案：乘几号线，距离，时间，费用，换乘方法等相关信息。

1. 课程设计报告需要包含以下内容：

1. 描述程序使用的所有数据结构，并列出其对应的抽象数据类型。

2. 描述程序中所用的算法及其对应的主要思想。

3. 尽可能详细地描述整个地铁线路系统的设计与实现的方法及工作的流程。

4. 多种决策对应的程序运行的示例。

5. 代码源码。

实验报告需要包含目录，一号标题为黑体三号字，二号标题为为黑体小三号字，三号标题为宋体小四号字，中文正文为宋体五号字数字和字母为Times New Roman，1.5倍行距，首行缩进两字符。

实验报告以word文档形式提交，命名格式为“学号\_姓名”，同源码文件打包，命名为“数据结构课程设计大作业\_学号\_姓名”,于9月10日23：59之前提交至指定邮箱。

# 二、简介与数据结构

此作业采用Tauri框架进行开发，其中前端使用Vue 3，后端使用Rust语言。

1. 基本信息
2. 线路数组

线路名称为数组，数组的每个元素都为Line类型。

|  |  |
| --- | --- |
| id | Int |
| name | String，线路名称 |
| speed | Int，线路的运行速度（km/h） |
| is\_loop | Bool，是否是环线 |
| is\_deleted | Bool，为真时不计入规划 |

1. 站点列表

站点列表为哈希表（具体来说是HashMap），来实现直接用站点名来获得站点信息。其键为站点名称，其中每个元素都为Station类型。

|  |  |
| --- | --- |
| id | Int |
| name | String，站点名称 |
| lines | Line的数组 |
| neighbors | Neighbor的数组，此时为空 |

1. 站名和站距数组

站名、站距为二位数组，第一维是线路，第二维是站点。

1. 规划信息
2. 站点列表

站点列表为哈希表（具体来说是HashMap），此时，通过遍历，站点列表的neighbors列得到填充。Neighbor 的数据结构为：{ name: String, line: Int /\*line数组中的下标\*/, distance: Int }。

1. 最优路过程数据列表

站点列表为哈希表（具体来说是HashMap），用以实现直接用站点名来获得过程数据。其键为站点名称，其中每个元素都为DijkData类型。

|  |  |
| --- | --- |
| name | String，站点名称 |
| lines\_on | Line的数组 |
| station\_from | String，最优来路的站点名称 |
| value | Int，最优权值 |
| known | Bool，为真时表示已知 |

# 三、算法和主要思想

1. 最短路Dijkstra算法

Dijkstra算法是一种经典的算法，用于在图中找到从单一源点到所有其他顶点的最短路径。这一算法由荷兰计算机科学家艾兹格·迪科斯彻（Edsger W. Dijkstra）在1956年提出，并被广泛应用于网络路由和地理信息系统中。其核心思想是采用贪心策略，即通过局部最优选择逐渐构建全局最优解。

Dijkstra算法在运行时维护两个顶点集合：一个是已经确定最短路径的顶点集，另一个是尚未确定最短路径的顶点集。初始时，源点的最短路径设为0（其他顶点设为无限大），并被归入已确定集。算法不断从未确定集中挑选一个距离已确定集中某顶点最近的顶点，更新它到源点的最短路径，并将其转移至已确定集。同时，还需更新所有可通过该新纳入顶点而达到的、仍在未确定集中的顶点的最短路径。这一过程持续进行，直至所有顶点都进入已确定集。

总的来说，Dijkstra算法的核心在于不断地探索未处理的最近顶点，并优化从源点到该顶点的路径，从而有效地解决了单源最短路径问题。

本项目中，Dijkstra算法用于求解换乘次数最少或用时最短的最优路线。

# 四、工作流程

1. 总体介绍

应用采用Tauri框架进行开发，其中前端使用Vue 3，后端使用Rust语言。后端因采用了Rust语言编写，而具有高安全性和高稳定性。前端采用Adobe Spectrum Web Component作为其组件库，从而具有了很好的界面质量和可扩展性。使用Tauri框架进行开发，使得应用有跨平台性，可以在Windows、macOS、Linux、Android、iOS环境下运行。

1. 基本流程

|  |  |
| --- | --- |
|  |  |

用户首先在“站点列表”界面选择起始站点和终点站点，然后选择规划模式，或者取消选择。

|  |  |
| --- | --- |
|  |  |

然后，应用将跳转到“规划行程”界面，并显示此次出行的用时、票价、里程，及完整的乘坐流程、换乘点位。

|  |  |
| --- | --- |
|  |  |
|  |  |

如果北京地铁未来开通了新线路，用户也可以自己进行线路的新建与删除。新建时，需要输入线路的名称、运行速度、站点和站间距信息。已有的站点会被标记为换乘站，从而实现新线路和原有现网之前的互相通达。

1. 具体实现方式：站点信息

首先批量新建线路相关的各种信息并存于数组。

然后，从station\_names、station\_distances数组中提取站点和站间距信息。这两个数组中的信息是从官网爬取后，经过ChatGPT进行自动格式转换（从有分隔的纯文本，到符合Rust语法的线路为第一维的二维数组）而形成的。

从station\_names数组批量进行站点的更新。如果stations这一HashMap中没有同名站点，新建站点；如果有同名站点，标记站点经过了一条新线。

完成这些操作后，将结果反映到前端界面中。

然后执行init\_graph函数来进行地铁网络的构建，也就是给所有的站点标记neighbors，来完成站点之间的相互连接。

1. 具体实现方式：路径规划

将前文描述的地铁网络、起点站终点站信息、用户选择的规划模式，以及线路列表传入do\_dijkstra函数。

初始化记载DijkData数据的HashMap，称为dijk。对每个站点，默认lines\_on为经过的所有线路，value为无穷大，known为假，station\_from为空。起始点的value为线路到站时间和现在时间的时间差，秒为单位。

执行循环。遍历dijk，找出known为假的value最小的站点v。如果找不到这样的站点，退出当前循环。将这个点的known设为真。

在v的所有neighbors中，对于known为假的部分，更新value。

具体而言，对于“最少换乘次数”模式，如果判断neighbor所在的线路和当前站点所在的线路不同，给value增量设置的很大，比如10000，来保证在非必要的时候都不换乘；如果在相同线路上，给value增量设为1。

对于“最短时间”模式，将value增量先设为两个站点之间的所用时间（距离/速度，秒为单位），然后如果是换乘站，再加300（5分钟），否则再加60（1分钟）。

然后回溯整个过程，完成路径的输出。同时完成时间、路程的计算，并通过公式得出票价的数据。

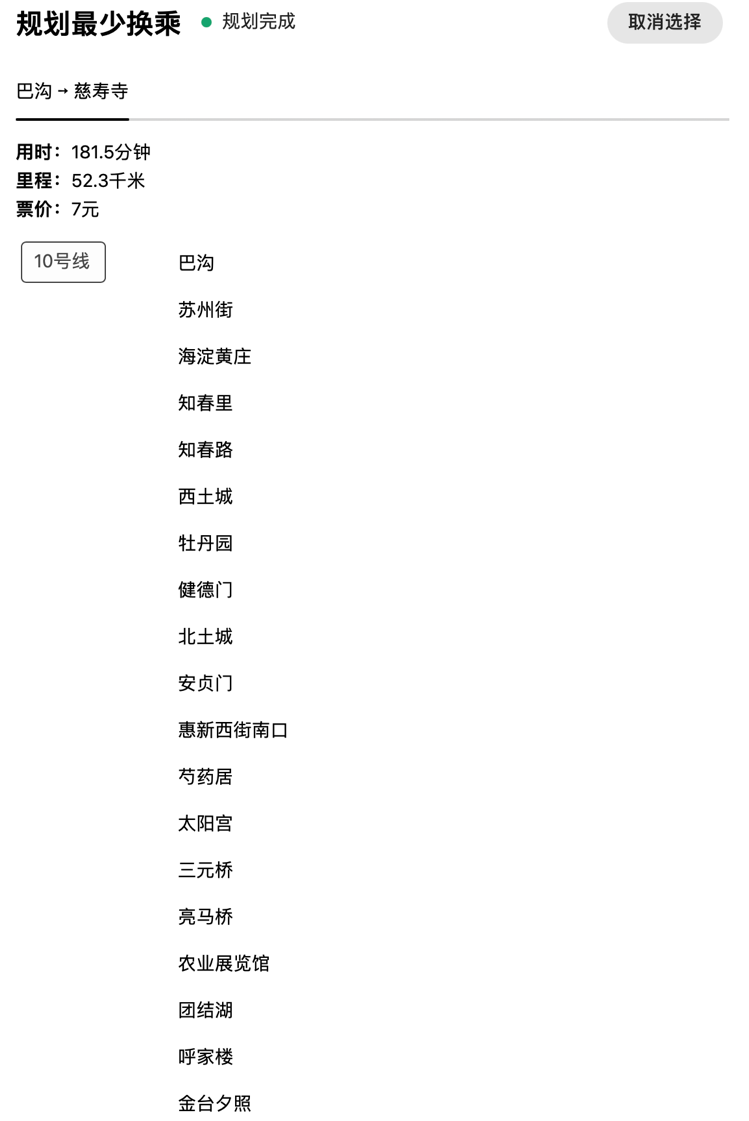
# 五、运行实例展示

|  |  |
| --- | --- |
|  |  |

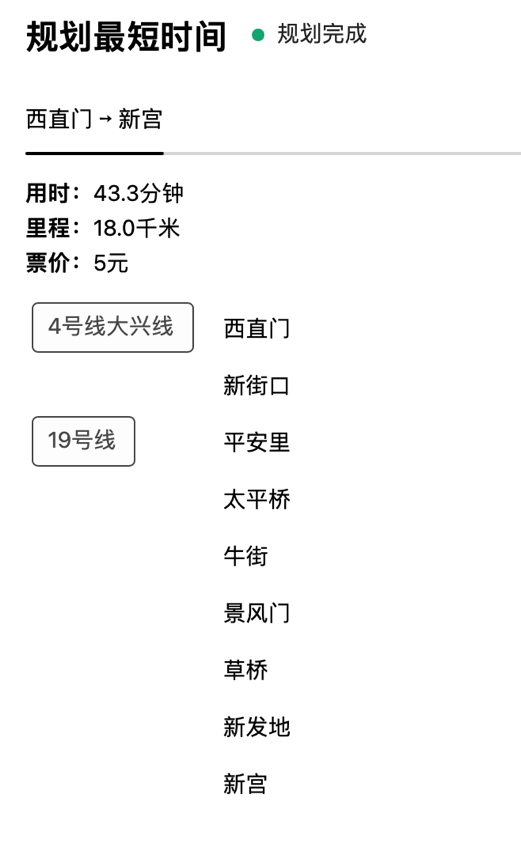
对于简单的无换乘情况，没有问题。

|  |  |
| --- | --- |
|  |  |

对于复杂的多次换乘，对于最少换乘次数来说没有问题。



最短换乘有效的其他例子。



对于最短时间，成功选用了快线，符合最短时间要求。票价计算无误。