数据结构课程设计自评表(2025)

组号：

组员：2023210887-孙明皓，2023210890-胡振宇，2023210976-刘梓宸

|  |  |
| --- | --- |
| 自评项 | 情况说明 |
| 基本信息 | 使用C++开发，框架为QT+MYSQL，景区+校园数量300个，地图包含78个点，202条边，用户数10+（可注册添加）  git仓库地址<https://github.com/ruguiima/travel-system-BUPT.git> |
| 旅游推荐 | 实现的功能：   * 用户可以按照“学习，美食，旅游，运动，热度，评分”任意一个标准进行topk优先队列**排序**   + 排序算法：堆排序   + 数据结构：top\_k\_algorithm类，site\_recommend类   + 算法分析：     - 性能：时间复杂度为O(n+nlogn)     - 优点：堆排序是不完全排序，通过构建一个最小堆选出前topk个最大的元素，其余的元素保持无序，速度快     - 缺点：堆排序时不稳定排序，在排序时会出现分数相同的数据在排序后的位置不同 * 用户可以搜索景点名称进行查找，进行**模糊匹配**   + 字符串匹配算法：KMP算法   + 数据结构：kmp\_search类，site\_recommend类   + 算法分析：     - 性能：时间复杂度为O(m+n),空间复杂度来源于部分匹配表的O(m)     - 优缺点：kmp算法能提高匹配效率，但是对于这个功能的场景，用户输入的内容不长，要匹配的目标字符串也不长，实际上并不能发挥出算法的优势。 * 用户双击选择景点可以直接跳转到该景点对应旅游日记的搜索结果，便于查阅 |
| 路线规划 | 实现的功能：   * 用户可以输入场所名称定位自己在校园中的当前位置，和添加想去的目的地   + 查找算法：**哈希表**   + 数据结构：campus\_nav类   + 算法分析     - 性能：哈希表时间复杂度O(1)     - 优缺点：       * 哈希表查询效率非常高为O(1)常数级时间       * 由于哈希表实现过为复杂，且该功能为路线规划的前置需求并非关键算法，直接套用了C++标准库std::hash       * 哈希表只能实现精确查询，不能进行关键词和模糊查找。弥补方法：使用QCompleter库增加自动补全功能，辅助用户输入正确场所名称。 * 用户确定当前位置和目的地后，可以选择规划策略（距离优先/步行时间优先/混合时间优先）规划路线   + 最短路径算法：**贪心（大框架）+Dijkstra（具体实现）**   + 数据结构：campus\_nav类，route\_plan类   + 预处理：在进入页面时，读取地图数据（点+边）并对应三种策略计算不同的权值，构建出三张图，便于后续用户操作直接调用   + 算法分析     - 性能：k（目的地数量）\* O(n logn) = O(k n logn) （使用堆优化）     - 优缺点：       * 相对于传统TSP问题求法非常高效，只需要线性对数级时间即可求得       * 贪心求出的路径不一定为真正的最短路径，可能陷入局部最优陷阱，但实际需求并非需要真正最优的线路，只需要一个大致近似解，考虑到程序性能和实现困难度，此算法可以接受 * 计算出的路径可以可视化的呈现，绘制在地图中   + 数据结构：campus\_nav类   + 方法：遍历返回的路径数组（前后两点间视为直线），调用QPainter库，通过转换实际经纬坐标和像素坐标绘制路线 |
| 场所查询 | 实现的功能：   * 用户可以自定义查询范围和选择场所类型，查询附近的场所，范围排序基准为**实际步行距离**   + 最短路径算法：**Dijkstra**   + 数据结构：campus\_nav类，route\_plan类   + 方法：route\_plan::search\_place函数会使用Dijkstra算法计算附近各点距离添加到数组中，此时即已按距离远近顺序添加，在达到指定范围时退出循环返回数组，再按场所类型做一次筛选即为最终结果   + 算法分析     - 性能：O(n logn) （使用堆优化）     - 优缺点：算法比较高效 * 查询得到的场所同样可以可视化地标明位置，展示在地图中 * 用户可以一键添加查询得到的场所至目的地列表，后续可直接导航 |
| 旅行日记 | 实现的功能：   * 用户可以输入旅游目的地，对目的地相关的旅游日记根据热度，评   分以及**日记的综合得分**进行排序   * + 排序算法：**堆排序**   + 数据结构：diarywindow类，和top\_k\_algorithm类   + 算法分析     - 性能：整体时间复杂度为*O*(*n*+*n*log*n*)=*O*(*n*log*n*)（构建堆+排序）     - 优缺点：堆排序为不稳定排序，不同的排序量的时候排序结果不完全一致，但是优点是可以不进行完全的排序，直接获取顺序靠前的结果。 * 用户可以按景点名称或者日记名称进行**模糊搜索**   + 字符串匹配算法：**kmp算法**   + 数据结构：diarywindow类，kmp\_search类   + 算法分析     - 性能：时间复杂度为 *O*(*m*+*n*)，分为预处理和与模板串匹配两个过程，空间复杂度来源于部分匹配表的 *O*(*m*)     - 优缺点：kmp算法本身的设计是尽可能的提高匹配效率但是对于这个功能的场景，每次进行匹配的目标字符串都不长，实际上并不能发挥出算法的优势。 * 可以按日记内容进行全文检索   + 字符串匹配算法：kmp算法   + 数据结构：diaryread类，kmp\_search类   + 算法分析     - 基本同上，但是日记内容一般字符量大，kmp的优势可以更好地体现。 * 可以对旅游日记进行压缩存储下载，并且可以**通过客户端**以本地浏览的形式**打开**   + 编码算法：**哈夫曼编码/解码**   + 数据结构：diaryread类，HuffmanCoding类   + 算法分析     - 性能分析：时间复杂度 *O*(*n*+*k*log*k*)，分为统计频率和构建哈夫曼树两个过程，空间复杂度*O*(*k*)，取决于对字符个数（字符频率的存储）     - 优缺点：哈夫曼编码是一种无损压缩，而且实现原理简单，但是该功能面对的是中文场景，中文文章的不同字符数量大，在相同文字量的时候与英文比较无法体现哈夫曼编码在数据压缩上的优势。 * 可以自己撰写对应景点的旅游日记，并且可以**上传图片**   + 数据结构：writewidget类 * 用户可以根据浏览和查询所有用户的旅游日记，旅游日记的浏览量即为该日记的热度，每位同学浏览完可以对旅游日记**进行评分**（0-5分制）   + 数据结构：diaryread类，update\_database类 |
| 智能体  开发使用 | 一个智能体用来学习QT，mysql，git相关工程知识；  一个智能体提供系统设计思路、辅助开发、编写测试用例；  一个智能体帮助完成任务分解，便于三人协作；  一个智能体帮助完成文档和PPT制作； |
| 其它AI  使用情况 | 即时聊天机器人(chatGPT, deepseek等）；  IDE的 ai插件，如 GitHub copilot； |
| 成员分工 | 刘梓宸：   * 爬取、编写景区和校园数据以及旅游日记内容； * 完成关键公用算法，如自实现topK排序（使用堆排序），KMP算法； * 搭建旅游推荐界面。   胡振宇：   * 设计初期预想的整体数据结构； * 搭建数据库，存储景区和校园，旅游日记数据以及用户信息； * 代码实现QT连接数据库，搭建用户登录注册界面； * 完成旅游日记模块的算法编写和界面搭建。   孙明皓：   * 爬取，设计地图数据； * 完成校园导航模块（路线规划+场所查询）的算法编写和界面搭建。 * 进行最终整体系统调试，优化算法和用户体验。 |