四川大学计算机学院 （软件学院、 智能科学与技术学院）

实验报告

学号：x 姓名： x 专业： 人工智能 班级： 233040701 第 12 周

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 课程名称 | 数据结构与算法分析课程设计 | | 实验课时 | 4课时 |
| 实验题目 | 排课软件 | | 实验时间 | 2024/11/21 |
| 实验目的和要求 | **基本要求**  根据输入的待排序课程序列文件，输出一种合法的排课顺序方案。  1 输入数据的合理性判断  2. 用文本存诸输入数据与输出各学期的课表：  3. 实现排课策略  **已实现的扩展要求**   1. 判断有没有环 2. 得到所有拓扑排序结果 | | | |
| 实验环境 | Clion2024、MinGW11.0、C++17 | | | |
| 算法描述 | **Csv数据加载逻辑：**  首先将文件读入，再按行分割为vector<char>。再按空格分割，创建顶点节点，根据是否已经排号课放入对应的列表（需要排课的和无需排课的需要分开放，只有需要排课的才需要拓扑排序），创建边并放入边所在的列表。  至于字符串分割操作，我将其整合、封装为了一个函数。利用标准库中的istringstream与vector<string>来实现字符串分割操作。  代码逻辑如下：  **构造函数逻辑**   1. **读取文件内容**：将输入的字符串内容按行分割成 vector<char>。 2. **解析课程数量**：读取第一行，按空格分割，获取每个学期需要安排的课程数量，并存储在 classesPerSemesters 列表中。 3. **创建顶点节点**：遍历后续的每一行，按空格分割，创建 Node 对象，并根据课程是否已经安排固定学期将其放入对应的列表。 4. **区分顶点列表**：    * 如果课程有固定学期（fixedSemester 不为0），则将其添加到 fixedNodes 列表中。    * 如果课程没有固定学期，则将其按索引存储在 nodes 列表中。 5. **创建边**：对于每个顶点，解析其前置课程索引，创建 Edge 对象，并将其添加到 edges 列表中。如果前置课程索引为0，则跳过不创建边。   **参数说明**   * content: 包含图信息的字符串，格式为每行代表一个课程，包含课程索引、名称、学分、固定学期以及前置课程索引列表。   **成员变量更新**   * classesPerSemesters: 存储每个学期需要安排的课程数量。 * nodes: 存储所有顶点节点的指针数组。 * fixedNodes: 存储有固定学期的顶点节点列表。 * edges: 存储所有边的列表。   **拓扑排序逻辑：**  首先遍历顶点与边集合，查找入度为0的节点，取出该顶点放入result节点，接着删除从这个顶点发出的有向边。循环这个过程，直到所有的顶点与边集合为空。  但在实际上，由于我的列表使用的是vector容器，不擅长删除操作，于是这一步我其实是将这个元素设置为NULL，以便提高性能。另外，由于这个图比较稀疏，我没有使用邻接表的形式，而是单独设置一个边的集合与点的集合，来优化内存占用率。  为了得到每学期课表，只需要将拓扑排序的结果结合以排序的课程按照要求放入每学期即可。  具体代码介绍如下：  **类概述**  Graph 类用于表示一个有向图，并提供了拓扑排序的功能，以便于安排课程表。该类通过顶点和边的集合来表示图，并支持将课程安排到每个学期。  **构造函数**   * Graph(const string& content): 根据输入的内容初始化图。内容应该包含每学期的课程数量、节点信息（包括课程名称、索引、学分和固定学期）以及边信息（前置课程索引）。   **公共方法**   * vector<vector<Node\*>> arrange(): 根据拓扑排序的结果，将课程安排到每个学期。返回一个二维向量，其中每个子向量代表一个学期的课程列表。 * void dispose(): 清理图中的资源，释放内存。   **私有方法**   * static void getOne(...): 递归函数，用于实现拓扑排序并获取所有可能的路径。该方法会找到所有入度为0的节点，并将它们添加到路径中，然后递归地调用自身，同时删除这些节点和它们发出的边。 * static vector<Edge\*> findInEdge(...): 查找给定节点的所有入边。 * static vector<Edge\*> findOutEdge(...): 查找给定节点的所有出边。 * static vector<Node\*> findNodeZeroInDegree(...): 查找图中所有入度为0的节点。 * static void deleteNode(...): 将指定节点设置为nullptr，以模拟删除操作。 * static void deleteEdge(...): 将指定边设置为nullptr，以模拟删除操作。   **辅助类型**   * Recorder: 一个函数类型，用于记录拓扑排序的结果。 * StopSignal: 一个函数类型，用于确定是否停止递归。   **成员变量**   * vector<int> classesPerSemesters: 存储每学期的课程数量。 * vector<Edge\*> edges: 存储图中的所有边。 * vector<Node\*> nodes: 存储图中的所有节点。 * vector<Node\*> fixedNodes: 存储有固定学期的节点。   **排课逻辑：**  排课逻辑有很多种策略与实现方法。我的逻辑是尽量先将课排到下午，然后再是早上、然后晚上，并且相同的课尽量隔天上，并且计算量尽量少。我按照川大的课表（时间表:早上4节课，下午5节课，晚上3节课）进行排序。首先将这个学期的进行预处理，将课时在4节及以上的课不断拆分出课时刚好为2的子课时。我创建了一个类bundle来表示在一个时间片段里的课，首先设置bundle的课时limit为5（下午），不断尝试添加（由于一周的工作日只有5天，因此最多生成5个bundle示例）。然后再将bundle的limit设置为4(早上)，重复上述过程，如果还有剩余，就将bundle的limit设置为3，继续尝试。如果最终都尝试完了，还有课没有被排到，就说明课过多了，排课失败。  现在，我们得到了若干bundle实例。为了使相同的课尽量隔天上，我只需要为bundle设置不同的工作日，由于在事先预处理的时候，**相同的课是相邻的（课时过长的课被拆分为多节），这意味着含有相同课的bundle也是相邻的**。有了这样的信息，我们只需要让bundle不要在工作日里相邻就行，这可以利用映射(2\*i)%5来实现，这个映射将0-4映射到0,2,4,1,3。  代码说明如下：  **类概述**  SemesterTable 类用于实现课程表的排课逻辑。它遵循特定的策略，将课程尽量安排在下午，其次是早上和晚上，并尽量使相同的课程隔天进行。该类基于四川大学的课表结构（早上4节，下午5节，晚上3节）进行排课。  **公共静态方法**   * static vector<Bundle> put(vector<Node\*> classes): 接收一个学期中的课程列表，并返回一个包含排课结果的 Bundle 实例列表。   **排课逻辑**   1. **预处理课程**：将课时在4节及以上的课程拆分为课时为2的子课程，以便更好地适应时间表。 2. **创建 Bundle 实例**：首先设置课时限制为5（下午），尝试将课程添加到 Bundle 中，最多生成5个 Bundle 实例。 3. **调整课时限制**：将课时限制依次调整为4（早上）和3（晚上），重复添加课程到 Bundle 的过程。 4. **安排工作日**：为了使相同的课程隔天进行，利用映射 (2\*i)%5 为 Bundle 设置不同的工作日。 5. **排课失败检查**：如果所有课程都尝试排课后仍有剩余，则表示课程过多，排课失败。 6. **排序 Bundle 实例**：根据 Bundle 的总课时进行排序，以便于后续的课表安排。 7. **分配时间段**：将排序后的 Bundle 实例分配到下午、早上和晚上。   **辅助方法**   * static bool isRest(const vector<Node\*>& table): 检查课程列表中是否还有未排课的课程。   **Bundle 结构体**   * int day: 表示工作日，用于确定课程安排在哪一天。 * dayType dayType: 枚举类型，表示一天中的时间段（早上、下午、晚上）。 * int totalDuration: Bundle 中所有课程的总课时。 * vector<Node\*> classes: 包含在 Bundle 中的课程列表。   **其他**   * Bundle 实例的创建和排课过程中，通过将课程指针设置为 nullptr 来模拟删除操作。 * 如果生成的 Bundle 实例数量超过15，则表示排课失败。   **Table 类文档**  **类概述**  Table 类用于创建和打印整个学期的课程表。它使用 SemesterTable 类的排课逻辑来安排每个学期的课程。  **构造函数**   * Table(const string& filename): 使用指定的文件名初始化 Graph 对象，并获取每个学期的课程列表。   **公共方法**   * vector<vector<Bundle>> arrange(): 调用 SemesterTable::put 方法来安排每个学期的课程，并返回排课结果。 * void print(): 打印整个学期的课程表，包括每个学期的 Bundle 实例和其中的课程。 * void dispose(): 释放内存。   **私有方法**   * string getDayType(dayType dayType): 将 dayType 枚举转换为对应的字符串描述（早上、下午、晚上）。   **成员变量**   * vector<vector<Node\*>> semesterClasses: 存储每个学期的课程列表。 * vector<vector<Bundle>> arranged: 存储排课结果，每个子向量代表一个学期的 Bundle 实例列表。 | | | |
| 源程序清单 | main.cpp 程序入口  graph.h 实现图结构，实现拓扑排序  Utils.h 辅助类，实现文件读取，字符串切割  Table.h 实现将一个学期的课表进行排课 | | | |
| 运行结果 |  | | | |
| 实验运行情况分析(包括算法、运行结果、运行环境等问题的讨论) | **实验结果：**  实验最终运行结果与预期相符。  **问题的讨论：**  对于如何判断图是否有环，这个比较简单。若当前无法找到一个入度为0的节点，如果边集中还有边，就说明这时候是有环的。  而对于得到所有拓扑排序的算法，也不算难，可以使用递归实现，普通的拓扑排序如下：  首先遍历顶点与边集合，查找入度为0的节点，取出该顶点放入result节点，接着删除从这个顶点发出的有向边。循环这个过程，直到所有的顶点与边集合为空。  只需要修改几个步骤就可以实现：  首先遍历顶点与边集合，查找入度为0的所有节点，分别取出该顶点放入result节点，接着删除从这个顶点发出的有向边。递归循环这个过程（因为不同的递归过程的状态是不同的），直到所有的顶点与边集合为空。  但由于总体情况实在太多，于是我设置了一个阈值，只取前1000个。 | | | |
| 指导老师  评议 |  | | | |
| 成绩评定： | 指导教师签名： | | |