北京工业大学

2023- 2024学年 第1学期

信息学部 计算机科学与技术

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称： | 数据结构课程设计 | | |
| 报告性质： | 课程设计报告 | | |
| 学号： | 21071003 | 姓名： | 高立扬 |
| 任课教师： | 杜永萍 | 课程性质： | 必 修 |
| 学分： | 2 | 学时： | 60 |
| 班级： | 210710 | 成绩： |  |
| 教师评语： |  | | |

2023年 11月 日

摘要

课设选题为：第1题 学习课程计划的辅助编排系统。用Python语言进行开发，前端开发插件为Pyqt5。程序包括了后台数据，后端代码和前端代码，收集数据和处理数据用时半天，后端代码书写及debug用时一周，前端代码书写及debug和最终调优用时两周。灵活运用了大二下学期《数据结构与算法》课所学，并利用了python语言开发特色，顺利完成了课设基本要求功能和附加功能，并额外根据实际应用添加了部分功能。最终呈现出一个较为简洁且美观的学习辅助编排系统，它可以进行课程编排调整和方案的保存读取，而保存和读取都会与本地的数据实时交互。界面底部加入了提示框，它会无时不刻地提示用户如何使用，是本软件一大人性化设计，大大提升了其交互性。

该程序可以辅助编排适用于计算机科学与技术专业本科生，四年共八学期的每学期课程辅助编排，即通过后台总课程数据，生成八个学期，每个学期建议学生修读的课。用户可以依此更快速便捷地进行后续每学期的人工课程编排，本程序不提供针对于每一个学期的详细编排，这是因为，结合本科两年多的课表来看，每学期课表的详细编排不仅需要考虑到北京工业大学所有专业所有课程的时间与空间冲突，还要考虑到学生劳逸结合问题，还要考虑到两节课之间是否存在相似关系（这样的两门课连一起上会降低学习效果）……种种复杂的因素，在当前我的编程学习和认知的情况下，是不能完成的，这也会导致每学期详细课程安排效果十分不乐观，与实际课表相差甚远，并且工程量极大，不符合课设的考查初衷。为了在本程序中体现到辅助用户手动编排具体学期课程的观念，在课程数据中加入了课程开设限制，以作为对用户手动排课的指示；还可以为每节课安排教室，同样体现该理念。

Python编程的灵活性在于其各种软件包，本软件就用到了pandas和numpy软件包，极大地简化了后端代码，并减少了相关运算时间开销，使得程序更加流畅。用到了networkx和matplotlib来绘图。代码总行数约1600行，总大小（含一张背景图）为2.5MB不到（图片占2.34MB）。

目录

[1 需求分析 1](#_Toc7290)

[1.1 程序功能 1](#_Toc1727)

[1.1.1 基本功能 1](#_Toc9703)

[1.1.2 扩展功能 1](#_Toc23096)

[1.1.3 隐含功能 2](#_Toc17622)

[1.2 需要处理的数据 2](#_Toc30585)

[1.2.1 后端 2](#_Toc11397)

[1.2.2 前端 3](#_Toc31027)

[1.3 开发环境 3](#_Toc6055)

[1.4 用户界面设计 3](#_Toc5894)

[2. 数据结构设计 4](#_Toc4134)

[2.1 主要数据结构 4](#_Toc26810)

[2.1.1 队列类 5](#_Toc15363)

[2.1.2 课程类 5](#_Toc13587)

[2.1.3 边类 6](#_Toc22042)

[2.1.4 图类 6](#_Toc29205)

[2.2 程序架构 7](#_Toc30179)

[2.2.1 整体架构 7](#_Toc9325)

[2.2.2 课表生成模块 8](#_Toc26464)

[2.2.3 绘制拓扑图模块 9](#_Toc32128)

[2.2.4 读存模块 9](#_Toc32299)

[2.2.5 人工模块 10](#_Toc21307)

[3. 详细设计 10](#_Toc11847)

[3.1 数据结构 10](#_Toc12317)

[3.1.1 图类成员函数setEdge() 10](#_Toc19614)

[3.2 后端 12](#_Toc16452)

[3.2.1 拓扑排序topology() 12](#_Toc15427)

[3.3 前端 14](#_Toc27678)

[3.3.1 人工删课deleteClass() 14](#_Toc14067)

[3.3.2 人工换课 exchangeCourse() 15](#_Toc13683)

[3.3.3 人工添课 addCourse() 16](#_Toc4713)

[4. 测试 16](#_Toc20954)

[4.1 正常运行测试 16](#_Toc12668)

[4.1.1 课表生成测试 16](#_Toc5604)

[4.1.2 拓扑图绘制测试 18](#_Toc1838)

[4.1.3 保存课表测试 19](#_Toc7803)

[4.1.4 读取课表测试 20](#_Toc775)

[4.1.5 课时调整测试 21](#_Toc21055)

[4.1.6 添加课程/新拓扑关系测试 22](#_Toc14706)

[4.1.7 删除课程测试 24](#_Toc25334)

[4.1.8 调换课程测试 25](#_Toc28674)

[4.2 错误用例 25](#_Toc30533)

[4.2.1 添加课程错误用例 25](#_Toc13501)

[4.2.2 删课错误用例 26](#_Toc10049)

[4.2.3 换课错误用例 26](#_Toc8418)

[4.3 极端情况 27](#_Toc2728)

[5. 总结与提高 28](#_Toc10952)

[5.1 体会与收获 28](#_Toc19904)

[5.2 问题与解决 29](#_Toc18373)

[5.3 自我评估 29](#_Toc10784)

# 1 需求分析

1.1 程序功能

1.1.1 基本功能

表1.1 基本功能

|  |  |  |
| --- | --- | --- |
| 需求编号 | 需求 | 功能 |
| 1 | 调研有关计算机专业应该开设的课程 | 实现了读写课程数据集“course.csv”的函数 |
| 2 | 建立构造课程的有向无环图 | 实现了图类及有向无环图的构建，提供图的增删边、改边、拓扑排序 |
| 3 | 求解拓扑子集划分的参考解 | 实现了含随机数的随机拓扑排序，并确保了分出8个划分（对应八个学期） |
| 4 | 适度拉长修业的时间 | 实现了课程随机失活算法，确保了每学期的课时不会过多，控制在课时限制的四分之三及以上 |
| 5 | 建立有向无环图 | 实现了前端页面左上角可以生成拓扑图的按钮，点击后即可读取当前课表内的课，绘制拓扑图，拓扑图可以反映出课程之间的拓扑关系，便于课程的先后修关系查看和后续人工编排 |
| 6 | 课程编排结果 | 实现了编排算法确保每学期课时不超出限制，并将编排结果打印在信息提示区和课表区，十分直观，便于查看 |
| 7 | 人工调整及界面支持 | 实现了课程的增删改算法，相应的功能区域均有文字标识，各个功能的使用方法在信息提示区有实时的提示，交互性强 |

1.1.2 扩展功能

表1.2 扩展功能

|  |  |  |
| --- | --- | --- |
| 需求编号 | 需求 | 功能 |
| 1 | 把某些课程调整到后续学期且考虑到因此产生的拓扑子集变化 | 在排八个学期的课程的时候，若检测到课时即将超过限制，则会通过课程随机冻结的方法，配合排课队列，在确保课时不超限制的同时，保证了课程的拓扑关系 |
| 2 | 每学期学时数的上限值 | 实现了调节每学期最高学时设置框，设置完毕后拓扑排序即可以此重新规划 |
| 3 | 自由选择同一学期内的感兴趣的课程 | 通过随机筛选，配合课时限制调整，再加上人工编排，可以使用户筛选出同一学期感兴趣的课程 |
| 4 | 输入有向弧的顶点对 | 此部分在课程的“增”操作中可以实现，在尝试添加课程的时候，会通过拓扑排序来判断新加入的键值对是否存在环 |
| 5 | 以文件的形式存储 | 保存图数据、课程编排数据、课程列表数据、拓扑子集数据 |
| 6 | 以备再次启动本辅助编排系统时导入 | 读取图数据、课程编排数据、课程列表数据、拓扑子集数据 |

1.1.3 隐含功能

表1.3 隐含功能

|  |  |  |
| --- | --- | --- |
| 需求编号 | 需求 | 功能 |
| 1 | 拓扑排序对环的判断 | 拓扑排序需要加入对环的判断，以避免原始课程数据可能在编写拓扑关系时出错，导致问题越来越严重影响程序后续编写 |
| 2 | 建图更简便 | 建图不可通过一行一行的setEdge()来实现，这样过于复杂碍眼，且当原始数据发生改变时，需要修改后台代码，应当以读取文件的自动化方式来建图 |
| 3 | 实时交互 | 前端用户的实时交互都应当在后端进行同步并保存为本地的临时文件，便于后续保存读取功能的正常运转 |
| 4 | 交互反馈 | 前端增删改操作无论成功与否，都应当对用户进行对应的提示和帮助，避免程序因为错误而终止，实现程序只能由用户主动关闭 |
| 5 | 读取操作 | 排课方案的读取不应当局限于仅读取课表，还应当读取上一次保存后的图数据，拓扑子集数据等，这样用户在读取排课方案之后，仍能确保正确地修改课表并再次保存 |

1.2 需要处理的数据

1.2.1 后端

全部后端文件及代码对应关系如下所示：

①“courses\_ori.csv”是最原本手动收集整理的课程数据备份；“courses.csv”是每次程序启动建图所用到的课程数据文件；“courses\_save.csv”是每次保存排课方案所保存的课程数据文件。前二者的数据构成如下表1.4：

表1.4 courses.csv & courses\_ori.csv数据构成

|  |  |
| --- | --- |
| 表格元素 | 说明 |
| id | 课程的index |
| 课程名称 | 课程的全程 |
| 学分 | 课程对应的学分（float） |
| 学时 | 课程对应的学时（int） |
| 开设时间限制 | 该课程是否建议在上午或下午等时间段进行安排，体现了本程序能辅助用户在未来进行每一学期详细排课的理念 |

由于保存排课方案之后，上课地点也应当保存在课表数据中，所以“course\_save.csv”相较于前二者，多了一个元素“上课地点”，保存的是在排课生成时，为每节课安排的上课教室。

全部课程来源于https://jwglxt.bjut.edu.cn内的计算机科学与技术专业的教学大纲，并根据上课经验自行编辑了开设时间限制。

csv是一种表格的形式，在python中可以通过pandas插件进行操作。csv存取速度非常快，同等“内容”的csv和xls文件，读取速度相差甚远，尤其是在大数据领域，业界均多用csv来保存。经过本科两年的竞赛和学习经验，可知将近500MB的csv表格仅几秒钟就可以通过pandas全部读取完毕，而本课设用到的几十门课的xls表格，却需要十几秒才能读取，效果可见！

②“通州上课地点.csv”和“校本部上课地点.csv”，分别是通州校区可供上课的地点和校本部可供上课的地点，全部课程来自于https://my.bjut.edu.cn公示公告中2023-2024上半学年全校本科生课程安排，筛选出了信息学部的课程并进行了教室提取和去重。

③“Graph”、“Graph\_ori”和“Graph\_save”，分别是程序运行中有向无环图的快照、程序运行初建图的文件和保存排课方案时当前排课方案的有向无环图数据。它保存着一行又一行的键值对，对应了key:val，也会有单独出现的数字，代表着暂时无先修后修关系的独立课程（如心理课）。

④“model”，是保存排课方案后，对前端八个学期课表主展示区字符串数据的保存。为了区分换行和单元格，文件内采用“λ”标识符辅助读取。

⑤“sems”和“sems\_save”分别是程序运行中拓扑子集的快照，和保存排课方案后当前排课方案对应的拓扑子集。保存的是八行数字，用逗号隔开，数字即是课程id。

1.2.2 前端

上文提到的Graph,sems,model等，均需要在前端进行拷贝，便于交互操作。除此之外，便是用于前端组件交互的槽函数，它们会对鼠标的点击，键盘的输入等做出相应的交互操作。

1.3 开发环境

IDE：Visual Studio Code 版本1.84.2

操作系统：Windows 10 x64

开发语言：Python

开发环境：Python==3.10.12；pandas==2.0.2； numpy==1.24.1； pyqt5=5.15.10； networkx==30rc1； graphviz==8.1.0； matplotlib==3.7.1

1.4 用户界面设计

前端UI设计见下页图1.1。课表在UI中处于最显眼的位置，且所占面积最大。底部是信息提示区，是程序对用户的使用进行提示和反馈的区域，提供了下拉框方便用户滑动查看。课表与提示区之间是添加课程区，默认情况下不可使用，当用户生成课表之后才能使用。最右侧从上到下依次是生成课表、设置每学期最大学时、学期课程删除和学期课程调换区域，这些功能区均有文字标识，且有明显区域分布，不容易看错搞混，且按钮和输入框设置简单易懂，使得用户不需要太多提示也能看懂如何使用。信息提示区右上角是锁定按钮，当用户想要人工调整之后，点击此按钮即可激活删除课程和调换课程功能；右下角是读条框，由于六十多节点九十余边的图的计算量对于当今计算机来说很小，所以大部分情况读条都是瞬时到100%。UI左上角是生成课表内所有课程的拓扑图。上方是Menu菜单，点开后可以读取或保存当前排课方案。



图1.1 UI设计

1. 数据结构设计
   1. 主要数据结构

所有的数据结构均在“classes.py”文件内定义。由于Python的设计理念，无需声明变量类型，可以通过赋值后自动赋予对应类型。Python的数组为list，而list本身就是STL，可以通过插入的第一个元素来决定其类型，因此无法进行vector数组的实现；而Python又没有指针，因此无法通过链表的方式来构建图类，这使得本来就难以书写的红黑树更难在Python中实现。

在与老师交流时，已经获得使用Python自带STL：list和dict的权限，即利用list和dict进行图类代码的书写。dict，也就是字典，它实现图类的代码和大二下《数据结构与算法》课程中的链表实现大不相同，这意味着我无法直接誊抄PPT中已写好且恰好可以完成本课设中所有图操作的代码，且需要完全独立构思（写这部分代码时我没有查阅任何资料，官方手册除外）如何用字典来实现对应的图操作功能，符合本课程不允许使用STL的初衷——学生理解并掌握STL并熟练运用的初衷。

主要的数据结构可以参考下表2.1：

表2.1 主要数据结构

|  |  |  |  |
| --- | --- | --- | --- |
| 数据结构 | 表达形式 | 逻辑结构 | 存储结构 |
| 队列类 | class MyQ | 线性 | 顺序 |
| 课程类 | class courses | 线性 | 顺序 |
| 边类 | class Edge | 树形 | 链接 |
| 图类 | class Graph | 图 | 散列 |

2.1.1 队列类

队列类MyQ，的主要作用是服务于图的拓扑排序，在构造时需要输入队列的大小，来提前分配内存空间，避免了list列表无限添加元素的情况，同时队列通过对首尾下标进行取余操作实现了循环队列，避免了队列插入删除错误和溢出。

队列类的主要成员如下表2.2所示

表2.2 队列类

|  |  |  |
| --- | --- | --- |
| 变量/函数 | 功能 | 实现 |
| self.q | 存储结构 | Python列表表达式 |
| self.front | 队列头部下标 | int型变量，初始化为0 |
| self.rear | 队列尾部下标 | int型变量，初始化为0 |
| self.mSize | 队列最大存储容量 | 在构建时传参 |
| def enQueue | 元素进入队列 | 元素插入到队列尾部 |
| def deQueue | 元素离开队列 | 元素从队列头部弹出 |
| def getFront | 获取队列头部 | 返回list[front] |
| def print | 打印队列（调试用） | Python允许直接print(list) |
| def isEmpty | 判断队列是否为空 | 判断头尾指针数值是否重叠 |

如下图2.1所示，不同于C++中自定义队列的实现，在Python中我用list列表表达式来对存储结构进行内存初始化，而None代表“不是任何类型”，此时当其插入第一个元素之后，便会赋予该元素的类型，比如插入一个int型变量或常数，那么list就会成为int型，只能存储int型的量，这称为“弱类型”。在后续拓扑排序中，将会用于八个学期课程规划，所以下方list的初始化，等效于C++中的“MyQueue<courses> q(size);”。

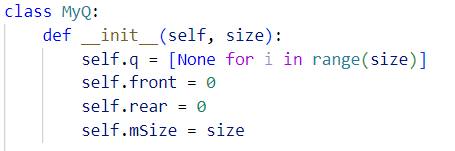


图2.1 队列类的构造函数

2.1.2 课程类

课程类courses，是图节点的重要构成，是图类中字典键值对的“值”。课程类包含了一门课程的所有关键信息。课程类在算法应用时，用list进行总体存储。课程类的主要成员如下表2.3所示

表2.3 课程类

|  |  |  |
| --- | --- | --- |
| 变量/函数 | 功能 | 实现 |
| self.Cid | 课程id | 传参 |
| self.Cname | 课程名字 | 传参 |
| self.Ccredit | 课程学分 | 传参 |
| self.Climit | 课程开设时间限制 | 传参 |
| self.Csem | 课程建议修读学期 | 后续算法生成 |
| self.Chour | 课程学时 | 传参 |
| self.Cplace | 上课地点 | 后续算法生成 |
| def \_\_str\_\_ -> str | 打印类时直接输出关键信息而不是地址号 | 重载 |
| def \_\_le\_\_等 | 重载自定义类的小于等于等大小逻辑判断 | 重载 |
| set & get函数 | set & get | - |

课程类有一个成员“课程开设时间限制”，便是摘要中提到的，为了后续每一学期详细排课所准备的成员，它决定了某个课程是否建议早上开设，或者是否建议上下午均可开设，或者是否建议晚上开设。虽然在本程序中没有用到（具体原因摘要已点明），但是体现了本程序的可扩展理念。

2.1.3 边类

边类Edge，是链接图节点的桥梁，是有向无环图中的有向边。在下文图类中函数的实现起到了重要的作用。边类的主要成员如下表2.4所示

表2.4 边类

|  |  |  |
| --- | --- | --- |
| 变量/函数 | 功能 | 实现 |
| self.Efrom | 有向边始点 | 传参 |
| self.Eto | 有向边重点 | 传参 |
| self.Eweight | 有向边权重 | 传参 |
| def \_\_str\_\_ -> str | 用于调试，将边类关键信息直接打印而不是输出其地址 | 重载 |

值得一提的是，由于Python赋值时决定变量类型的特点，Efrom和Eto并不用手动规定类型；而对于排课而言，边权作用不大，而这里特地加入了边权，是为了辅助下文图类中相关函数。

2.1.4 图类

图类Graph是本课设的灵魂所在，是排课系统的核心驱动类。其内部主要依靠Python的字典dict所形成的键值对来建图，并通过其函数对键值对的访存修改等进行维护。图类的主要成员如下表2.5所示

表2.5 图类

|  |  |  |
| --- | --- | --- |
| 变量/函数 | 功能 | 实现 |
| self.graph | 图结构存储 | 字典dict |
| self.numVertex | 节点数 | - |
| self.numEdge | 边数 | - |
| self.Indegree | 入度 | list |
| self.Outdegree | 出度 | list |
| self.Mark | 点遍历表 | list |
| self.saveTo | 将图存储为文件要用到 | str |
| def isEdge | 判断是否为边 | 通过判断边权重是否为0，因为在建图时边权即为课程的学分，若边权为0则说明该边不存在 |
| def toVertex | 返回边的终点 | 返回Eto |
| def firstEdge | 寻找某节点的第一条边 | 通过dict来寻找某节点的第一条边 |
| def nextEdge | 寻边的下一条边 | 原理同上 |
| def setEdge | 建立有向边 | 构建字典键值对，并更新对应的出度入度表等 |
| def canDel | 判断某节点是否可删 | 通过判断出入度来确定一个节点是否可以返回而不影响先修课后修课的拓扑关系 |
| def addVertex | 单独添加节点 | 单独添加一个节点，并更新对应数据 |
| def delEdge | 删除某节点及其所有关联边 | 循环寻节点，首先判断节点是否存在，然后删除所有关联边，最后删除节点。更新对应出入度等 |
| def printAll | 打印整个图（调试用） | 循环遍历并打印 |
| def sortG | 字典排序，确保后续排课正确进行 | 遍历字典每一个key，并根据课程id进行升序排列 |

由于用字典来实现图的相关操作，因此在实际操作当中存在一定的难度，而且PPT中的链表建图已完全不可参考。在编写图类时，相关函数我都是自主思考构思，创立了一个“test.ipynb”对想法进行一步一步的验证和实验，最终通过一遍又一遍的debug和极端情况测试，终于完善了图类的构建。好几次编写时我都卡住了很久，不会写，但是我坚持自主完成，不上网搜别人的思路和代码，我认为在此付出的时间和辛苦都是值得的。

通过用字典来实现链表，我不但对字典的认知更上一层楼，与此同时还强化了对链表的认识和理解，体验到了二者各自的便利和不足。链表（单向）由于其链式结构，在访存时时间复杂度为O(n)，而字典由于其底层为红黑树，因此时间复杂度为O(logn)；链式结构可以通过指针十分便捷地判断边表的末尾，而字典则需要通过键值对长度等辅助量，侧面地实现遍历边表……

* 1. 程序架构

2.2.1 整体架构

程序分为Ui、前端、后端和数据，以及一个main函数用于链接槽函数和启动程序。可以通过下表2.6来概括

表2.6 整体架构概括

|  |  |  |  |
| --- | --- | --- | --- |
| 程序架构 | 文件 | 功能 | 功能模块划分 |
| Ui | Ui\_main.py | 交互组件绘制和定义 | - |
| 前端 | interact.py | 槽函数，负责前后端交互 | 课表生成模块  人工模块  读存模块  绘制拓扑图模块 |
| 后端 | algorism.py | 对前端操作的后端反馈 | 建图模块  拓扑模块 |
| 数据 | 1.2中有提到 | 程序的数据驱动源泉 | - |
| main函数 | main.py | 链接槽函数并启动程序 | - |

值得一提的是，后端算法文件我将其命名为algorism而不是algorithm，这并不是拼写错误，algorism是来自于9世纪波斯数学家阿尔•花剌子米名字的拉丁语翻译“Algorismus”，后来随着计算机科学与技术的发扬，algorism的异体词algorithm出现并普及。将其取名为algorism，寓意是我首次用Python自主编写并完成课设。

部分文件在本报告中没有提到，它们是见证了我在完成课设过程中思考与实验的纪念，因此予以保留。

着重介绍前后端，前后端的各个模块之间有着密切的联系，见下图2.2所示



图2.2 模块之间关系

该图为在Ui界面的辅助下，模块之间的关系。可见，各个模块之间没有特别复杂的关系，基本上来说就是人工模块和读存模块之间的交互。

2.2.2 课表生成模块

该模块构成见下表2.7

表2.7 课表生成模块

|  |  |  |  |
| --- | --- | --- | --- |
| 细分模块 | 函数 | 功能 | 实现 |
| 读存模块的一部分 | load\_data | 读取数据 | 借助pandas插件进行数据读取 |
| 建图模块 | init\_graph | 建图 | 通过文件读取进行图的构建，保存相关参数 |
| 拓扑模块 | topology | 划分拓扑子集，即八个学期课程 | 借助队列和图内遍历操作，配合课时限制等算法进行课程合理分配规划 |
| 建图模块 | addPlace | 为课程添加上课地点 | 读取上课地点信息并随机分配到每节课程 |
| - | printSem | 打印八个学期课程分配方案（调试使用） | 遍历list |

上述函数在模块内被按顺序调用，最终保存下图、八个学期排课方案以及全部课程列表。

数据集采用了csv表格形式，这种表格读存速度非常快，可以大大减少用户等待时间；建图操作采用通过文件读取，十分便于维护；

拓扑排序为本模块的核心，采用了队列和图的遍历操作来进行拓扑排序，需要八个循环，时间复杂度介于O(n^2)和O(n)之间。为了实现课设中“避免课程过于紧凑，不要超过学时限制”，采用了许多策略来使课程“失活”：①专业选修课一共有25个，按照实际情况来讲，本科生不可能将其全修完，肯定是有选择性得修读，按照教学计划来讲，最少修9门，因此对另16门进行随机“失活”，按照 失活课程数 = -0.072每学期课时限制 + 36 的公式来进行“失活”，并配合随机数使排课方案丰富多样②当某个学期排课课时超过限定课时的四分之三时，会有二分之一的概率随机使当前预安排的课程“失活”，在下一学期排课时备用③当课时达到限制，立刻退出循环。上述方法即保证了课程编排方案的多样性，还保证了必修课必定会安排在课表中，且符合要求，不会扰乱拓扑顺序。

2.2.3 绘制拓扑图模块

绘制拓扑图模块用到了python的networkx插件进行拓扑图构建，用到了graphviz插件进行画面优化，用到了matplotlib插件进行绘图，有了插件的辅助，绘图代码书写十分简单顺畅。绘图效果如下图2.3，绘制的拓扑图可以很好地反映出各个课程之间的拓扑关系，比如17号课程代表了“C语言课”，可以看出它在计算机科学与技术专业中的重要地位。

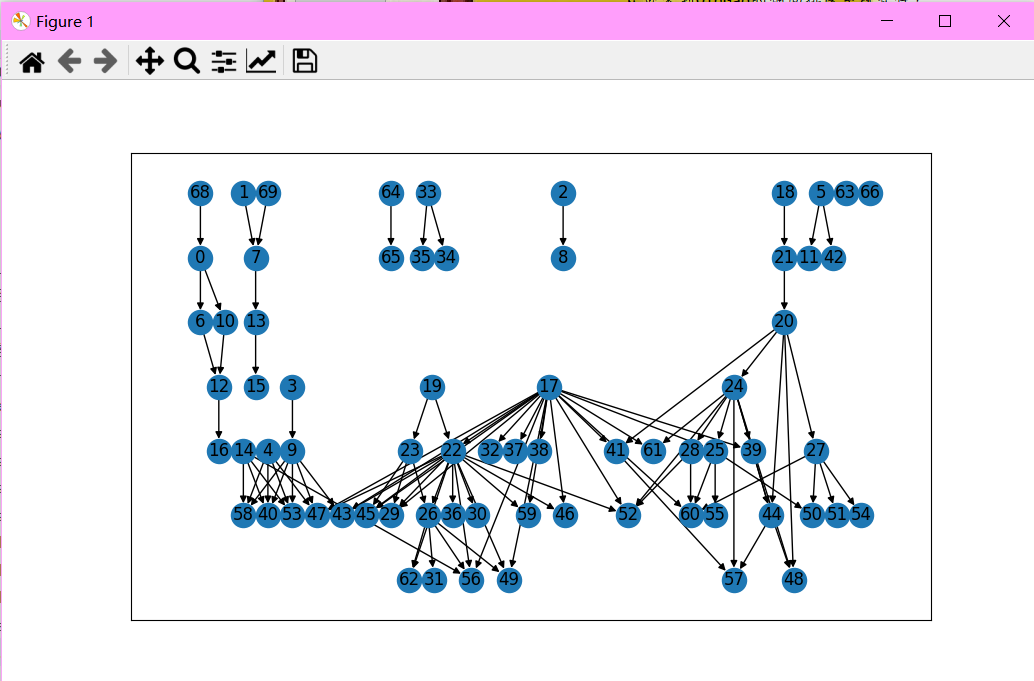


图2.3 拓扑图绘制示例

2.2.4 读存模块

该模块核心构成见下表2.8，另还有读存功能融合到了人工模块当中，也包括了2.2.2中的load\_data()函数也是读存模块一部分构成。读存模块会在用户通过Ui界面进行相关操作时，频繁地更新后台文件，从而确保无误地实现最终需求——课表的保存和读取，且确保读取后可以再次编辑并保存。

表2.8 读存模块

|  |  |  |
| --- | --- | --- |
| 函数 | 功能 | 实现 |
| optionSave | 保存排课方案 | 读取快照文件和课表，保存到本地 |
| optionLoad | 读取排课方案 | 读取本地排课方案文件，载入程序 |

2.2.5 人工模块

该模块构成见下表2.9，人工模块是用户对课表进行调整修改的核心所在，用户可以通过Ui中的交互组件，在人工模块的帮助下完成前后端相应的反馈。同时它也包含了部分组件失效和生效的逻辑，这确保了用户能正确使用所有功能。

表2.9 人工模块

|  |  |  |
| --- | --- | --- |
| 函数 | 功能 | 实现 |
| schemeGenerate | 生成排课方案 | 读取全局变量后，调用课表生成模块，并将生成的内容反馈到Ui |
| setHour | 设置每学期课时限制 | 获取用户设置内容，并记录在全局变量，以供下一次课表生成 |
| deleteClass | 课表课程删除 | 获取用户需要删除课程的id，搜索课程后进行删除，需要调用图的成员函数；更新后台数据 |
| activate | 激活/失活Ui组件 | 设置组件的激活状态，以便某些功能性Ui组件在点击后能获得正确的反馈，使目标组件激活/失活 |
| addCourse | 向课程数据中添加用户设置的课程 | 读取用户添加的课程信息，首先调用拓扑模块；更新后台数据 |
| exchangeCourse | 学期之间的课程调换 | 获取拓扑顺序，检查是否课程可调换至目标学期，并进行Ui界面课程调换展示与课时调整；更新后台数据 |

1. 详细设计

本课设众多数据结构与算法中，有部分代码技术含量小，内容比较基础，在本部分将会详细介绍实现难度大，在课设中具有重要价值的部分。

* 1. 数据结构

3.1.1 图类成员函数setEdge()

该函数是有向无环图构建的核心函数，由于采用Python编写，且用字典来模拟链表，这就意味着在写代码时无法参考PPT中已写好的C++代码，因此值得在报告中进行单独的详细的介绍。

该函数全部代码，如图3.1所示。该函数位于“classes.py”约170行。

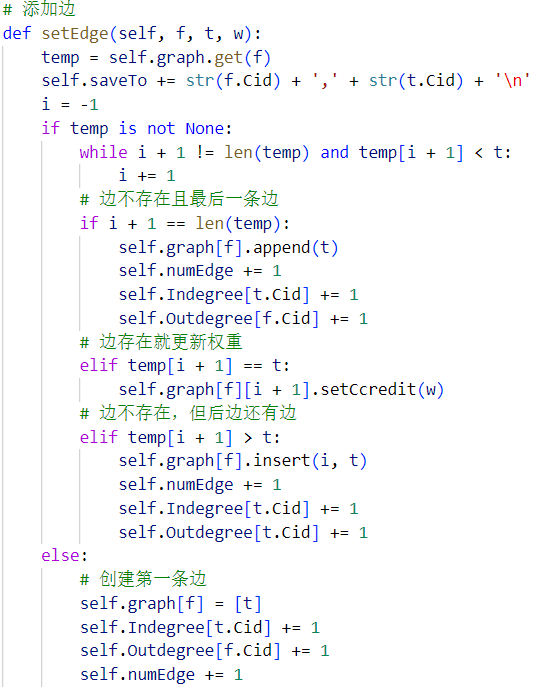


图3.1 setEdge代码

该函数的流程图可见下图3.2

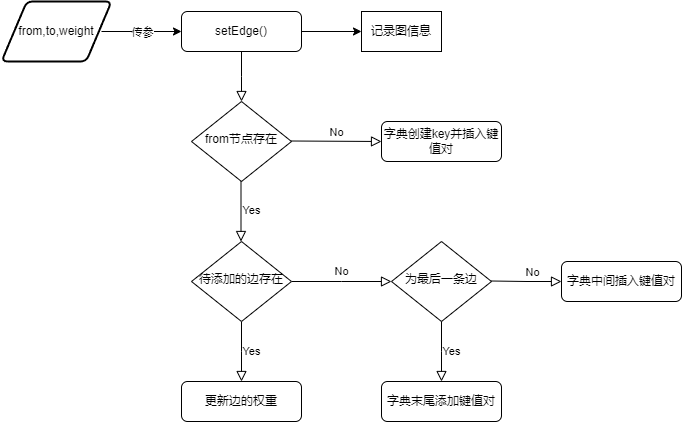


图3.2 setEdge流程图

从函数代码和流程图可见，该函数首先会记录下图的数据，该数据会在图构建完毕之后保存到Graph快照文件内。参数f和t代表了一条边的from和to，均为courses类，weight则为to课程的学分。该算法首先会判断dict字典中是否存在from，若没有则会创建该key且插入{key:val}，否则会检测待添加的边是否存在，如果存在则仅更新weight，否则会按课程的id来选择字典中{key:val}的插入位置，确保了每个key内的所有val的id升序排列，模拟了链表。

在建边的同时，图内成员变量也会进行对应的更新，从而确保后续算法的正确运转。

* 1. 后端

3.2.1 拓扑排序topology()

该函数是八个学期拓扑子集划分和核心函数，也是后续人工加课的必备函数，否则用户自行创建的课可能存在循环关系，导致无法划分。拓扑关系划分会根据进度来推进Ui界面的进度条，由于计算速度快，因此有概率能看见读条，大部分情况只能看到读条从0%瞬间推进到100%。该函数的流程图如图3.3所示

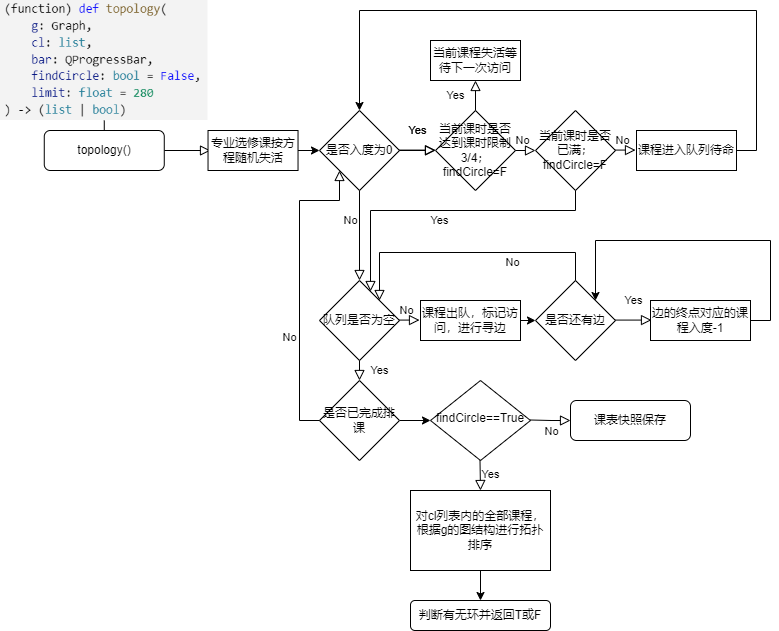


图3.3 topology流程图

重要代码及详细介绍如下：

如图3.4所示，上文分析道，北京工业大学计算机科学与技术(实验班)专业的教学计划表明，专业选修课需要至少需要修够9门课，再以课设要求中默认的每学期280课时为基准，程序内部规定最多500课时每学期为约束，建立方程对专业选修课进行动态失活，这样即确保了必修课必定能被安排在课表中，也保证了用户可以在课时范围内均能自由选择课程进行调换或删除。

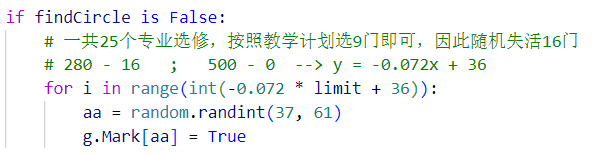


图3.4 专业选修课动态失活

如图3.5所示，每次循环均会有总课时计算，当课时即将满时，课程将会有二分之一几率失活，但是mark标记不会改变，确保了这门课不会消失在后续循环中；当课时超过限制，退出循环以节省时间开销。入度为0的点全部如队后，会推进信息提示区右下角进度条的显示，且为sems学期课程快照保存做准备。最后会通过while循环遍历某课程所关联的所有课程，将它们入度-1。如此这般，八个学期的课程就会划分完毕。

如果此时进行的功能并非排课，而是手动添加课程，那么findCircle==True，课程都将会参与拓扑排序，确保算法最后可以准确地检测到是否有环的存在。



图3.5 拓扑核心代码

算法的末尾便是收尾工作，如图3.6所示，如果findCircle==False，说明此时进行课程分配，对分配结果进行快照保存；否则此时进行的是课程添加，判断有无环后返回对应布尔值。

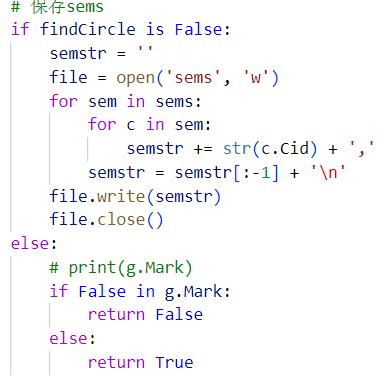


图3.6 算法收尾

* 1. 前端

前端交互算法，是我用Python语言写课设碰到的难点，根源在于对Python知识的掌握仅停留在自学水平，没有系统学习过，导致我对Python全局变量、前后端交互、深浅拷贝等认识不足，由于写这些代码时，课设已经完成了大半部分，因此无奈下我只能靠写冗余代码来弥补因知识不足产生的bug，这一点我也会在报告末尾进行自我反思和检讨，因此下面只贴出流程图。最困惑我的当属人工删课、人工换课和人工添课三个功能函数的编写，且前端交互算法中，最核心的便是这三个函数，因此下面对三者进行详细介绍。

其他函数编写较为简单，不需要过多逻辑思维，因此不在报告中夺走核心函数的关注度。

3.3.1 人工删课deleteClass()

人工删课函数流程图如下图3.7所示

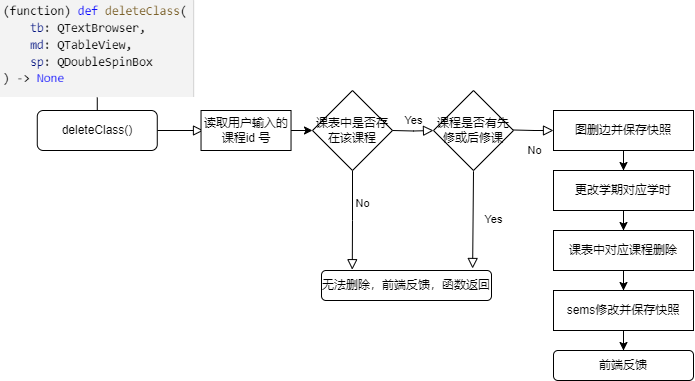


图3.7 deleteClass流程图

虽然从流程图来看，本函数基本上是线性流程，但是结合实际课设应用，考虑到课设程序要给用户到来良好的体验和使用，并完成题目中的要求，需要想到删课后，课表和Ui界面该如何发生变化。首先需要进行是否删边的判断，能删边后需要在图和sems拓扑子集中删除对应数据；为了服务于课表的保存和读取，图和拓扑子集的快照需要保存；删除课程之后，课表中的课程需要进行删除，变为空白单元格，但是考虑到课表的美观，还需要将空白单元格下方的课程顺序上调，填补空位；课程删除后，对应的学期总课时统计单元格的数值，也需要进行改变；数据提示区应当对用户删除课程的成功与否进行反馈，删除失败需要说明失败的类型。

值得一提的是，Pyqt5中，单元格内存储的是字典而不是字符串：{0:字符串}，因此在进行单元格内容提取时，需要进行多重处理。

3.3.2 人工换课 exchangeCourse()

人工换课函数流程图如下图3.8所示

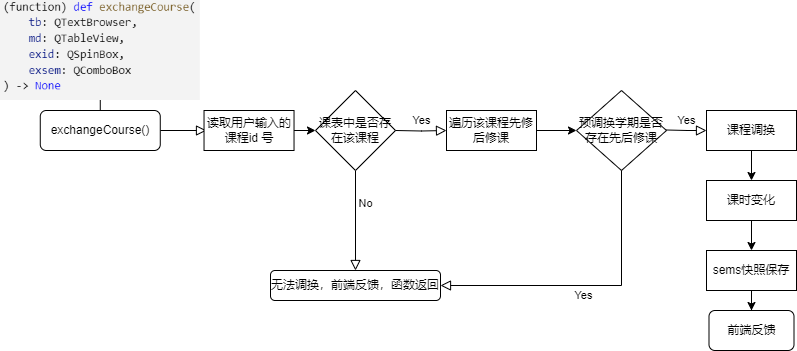


图3.8 exchangeCourse流程图

调换课程需要考虑到调换后是否破坏了先后修课的关系，否则这会导致课表编排变得不合理，不能起到辅助排课的初衷与目的。总的来看，该函数流程结构与人工删课相似，但是内部某些逻辑却不同，不过这些逻辑最终都是为了服务于前后端正确的交互。

3.3.3 人工添课 addCourse()

人工添课函数流程图如下图3.9所示

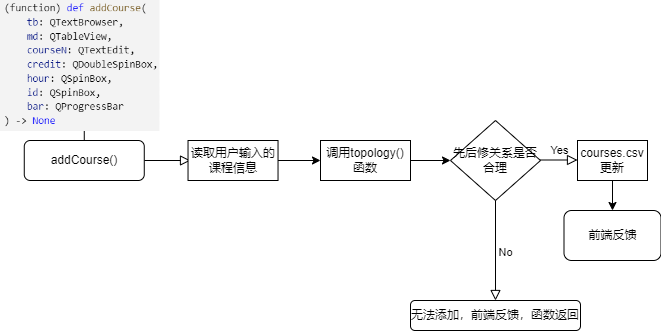


图3.9 addCourse流程图

课程将会添加在course.csv表格中，当用户再次生成排课方案时，新添加的课将会按照用户指定的先后修关系出现在课表中。如果用户输入的课程名称不存在，则视为新加的课，id自动按照courses.csv延续添加；如果以存在，视为对已有课程进行先后修关系的添加，在调用拓扑排序判断无环后才能添加完毕。

1. 测试
   1. 正常运行测试

4.1.1 课表生成测试

测试用例：在Ui界面多次点击“生成方案！”按钮来多次生成排课方案，观察随机排课算法的效果，判断其是否合理。

测试结果：如下图4.1,4.2所示，可以看出必修课的分布较为稳定，说明算法保证了必修课安排在八个学期课表中；对比两次课表可以看出，专业选修课的安排大不相同（重点关注第七第八学期），说明算法保证了非必修课的随机性安排，使得用户可选的课表丰富多样。如图4.3所示，该图为图4.2的总课时，可以看出算法保障了每学期的总课时不会十分接近280，符合设计要求。

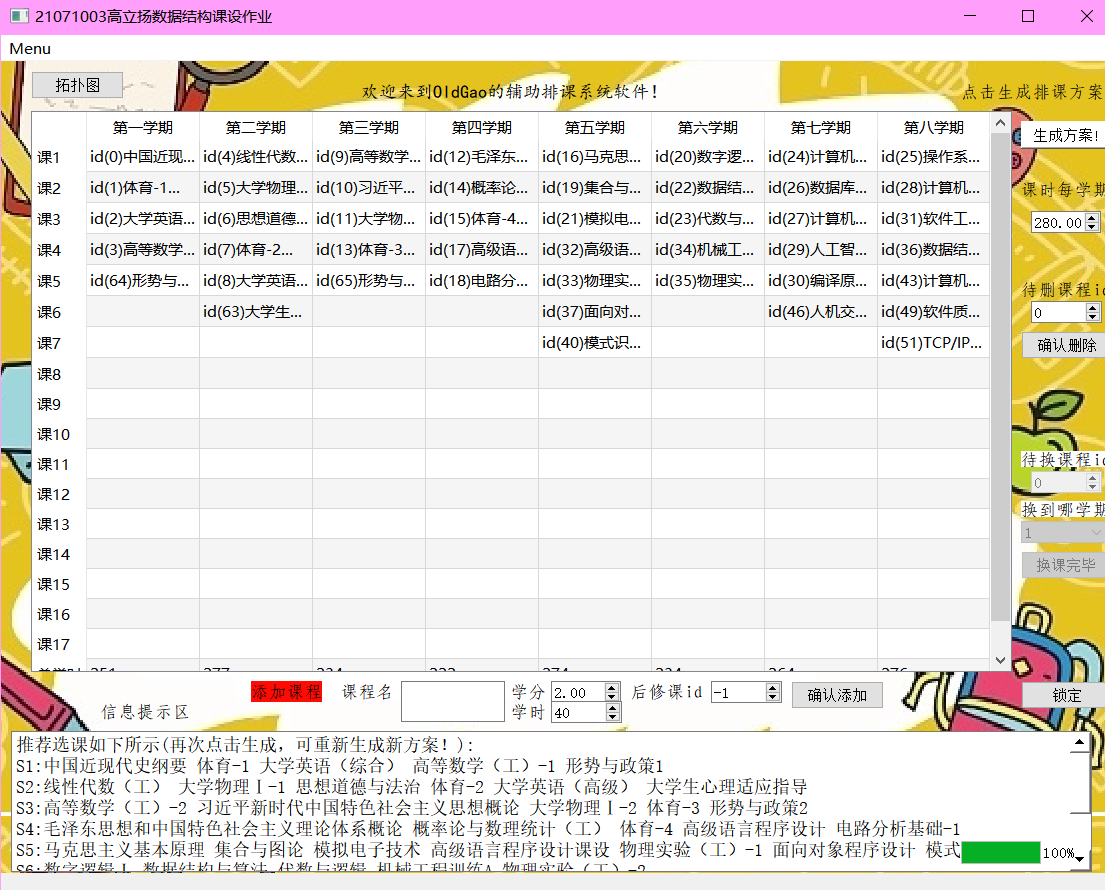


图4.1 课表测试1

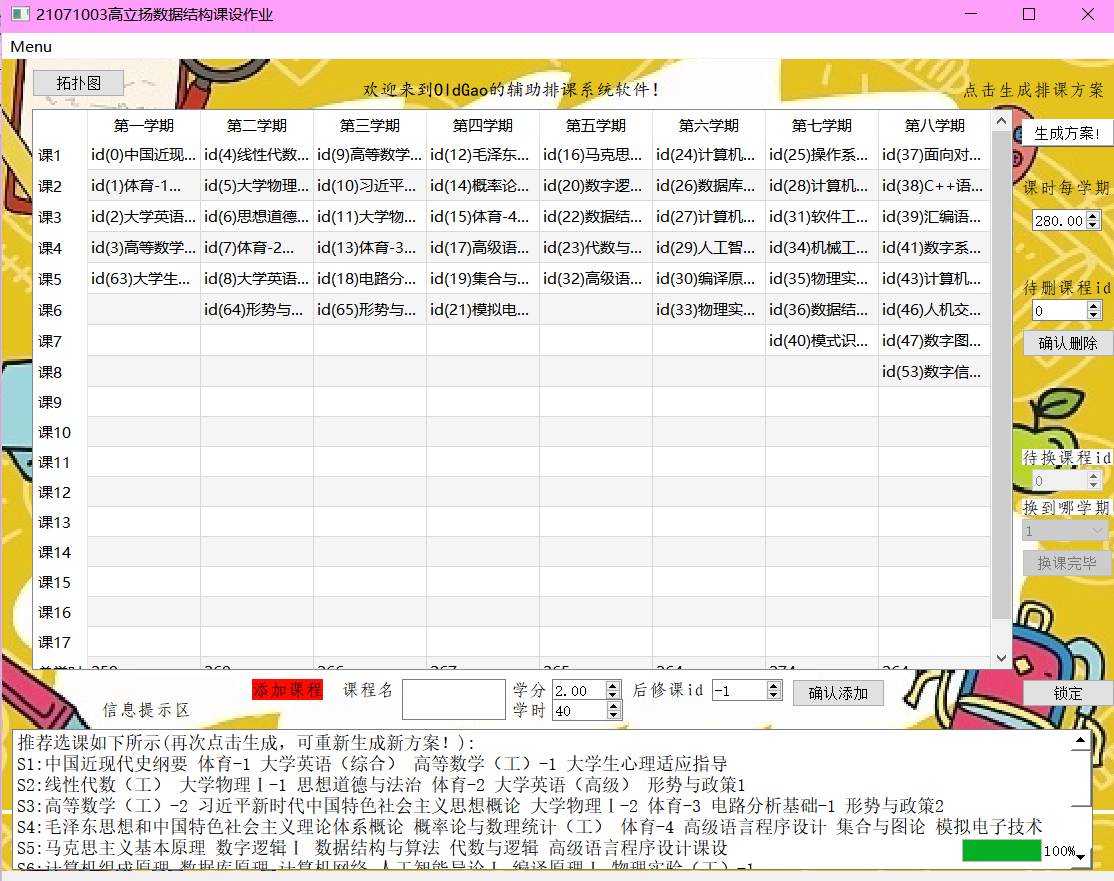


图4.2 课表测试2



图4.3 上图4.2的课时

4.1.2 拓扑图绘制测试

测试用例：基于图4.2的课表，进行拓扑图绘制，并测试图片保存功能效果。

测试结果：拓扑图很快就绘制出来，如图4.4，它反映出了课表中所有课程的拓扑关系，以便用户在后续进行删课和换课时能考虑到先修后修课的课程（否则会影响拓扑顺序和课程知识安排），也能减少用户在添课时构成环的可能（虽然程序会判断环并告知用户）。如图4.5，用户可以保存拓扑图。如图4.6，保存成功。如图4.7，绘制成功后Ui也会进行反馈，提示用户手动保存拓扑图。

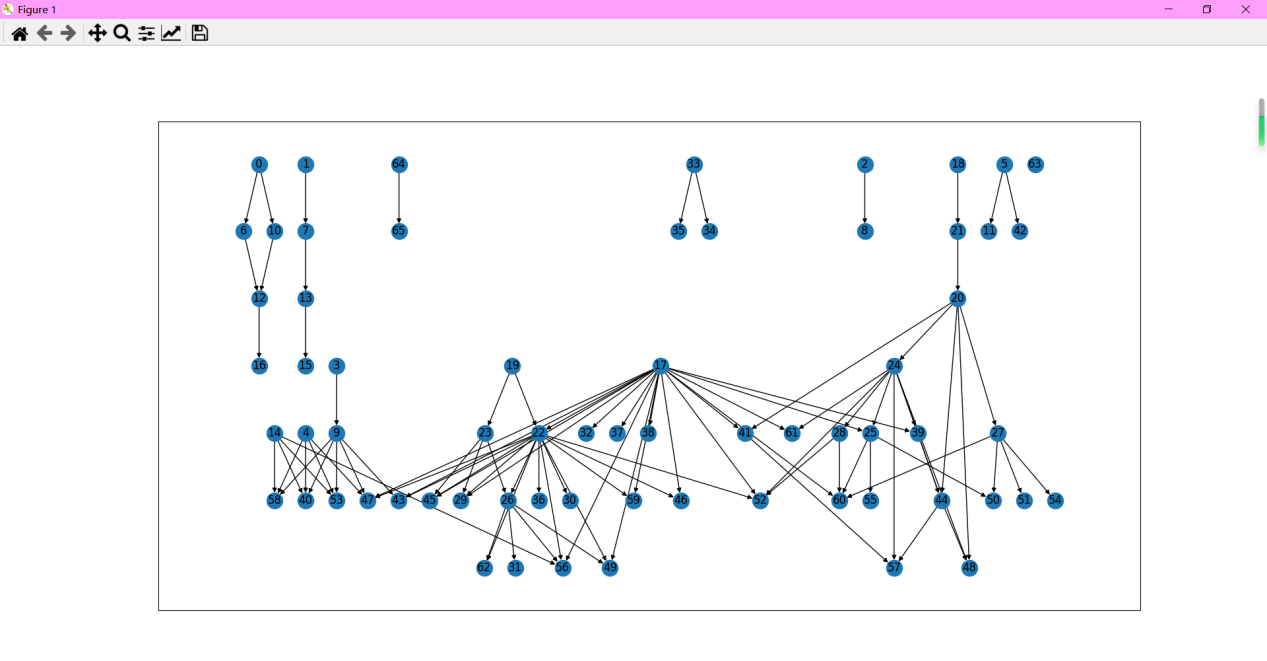


图4.4 拓扑图

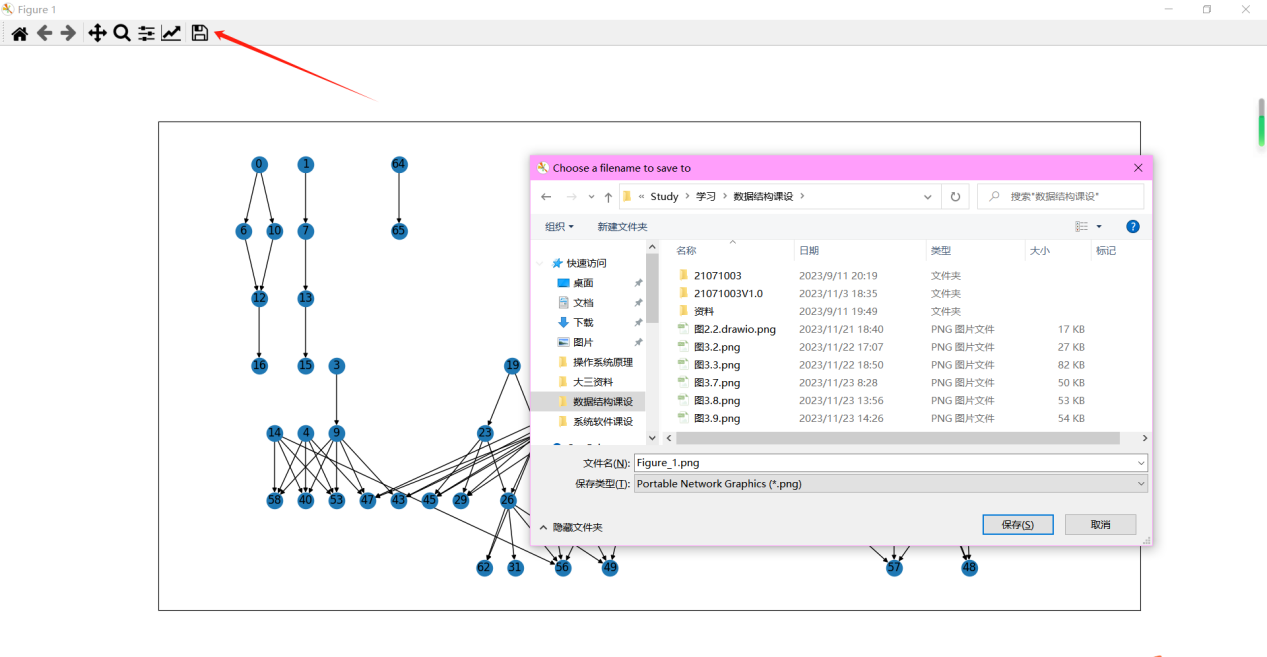


图4.5 保存界面

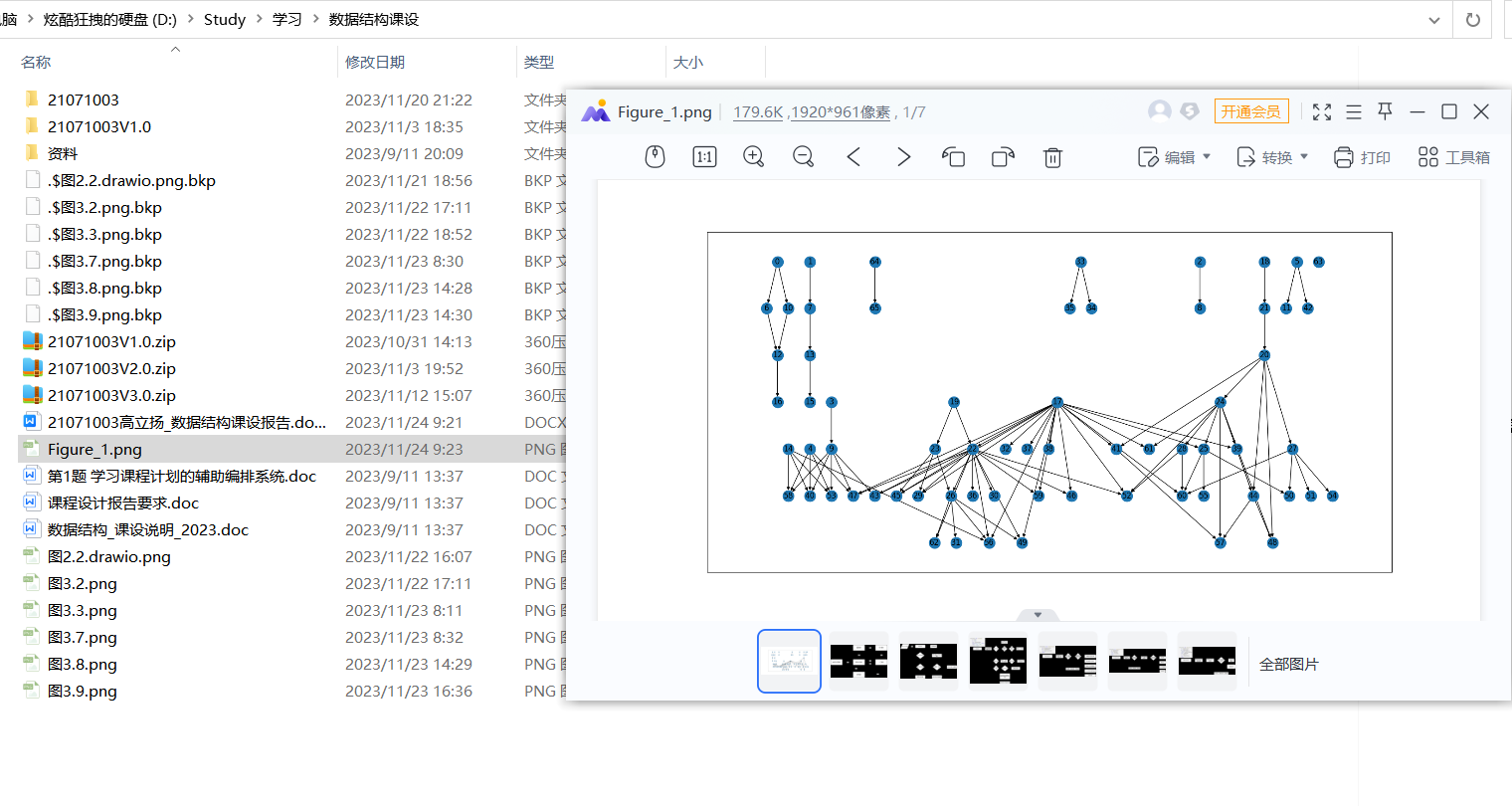


图4.6 保存成功



图4.7 反馈

4.1.3 保存课表测试

测试用例：随机生成一张课表，并进行保存

测试结果：如下图4.8所示，生成课表后，在左上角Menu选择“保存排课方案”，Ui的信息提示区会进行文字提示，提示用户保存完毕，并告知用户保存了哪些文件。如图4.9可见，文件已成功保存。

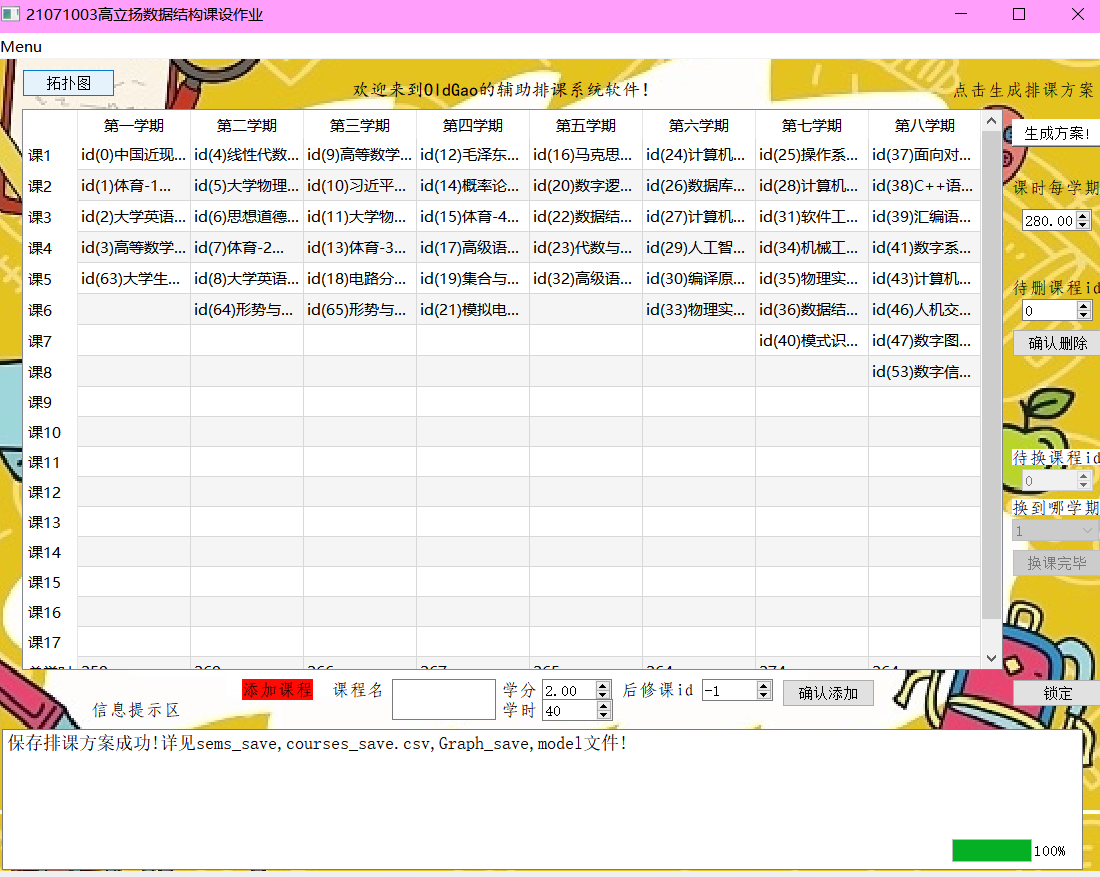


图4.8 保存反馈



图4.9 保存文件

4.1.4 读取课表测试

测试用例：重启程序随机生成课表后，读取4.1.3中保存的课表，并在后续测试中继续使用，这样可以验证是否图、拓扑子集等都会成功保存。

测试结果：如图4.10，此时排课方案刷新。如图4.11，选择Menu中“打开排课方案”后，课表重新加载出来，同图4.8。并且我们还可以看到，信息反馈区同样会反馈读取成功，并且提醒用户修改后保存。读取之后，不再允许用户添加课程或重新生成课表，因为保存&读取的情况，已经是用户对当前算法生成的课表满意的前提下，继续人工精修。用户若还想重新生成的话，重启应用即可。



图4.10 重新生成排课方案

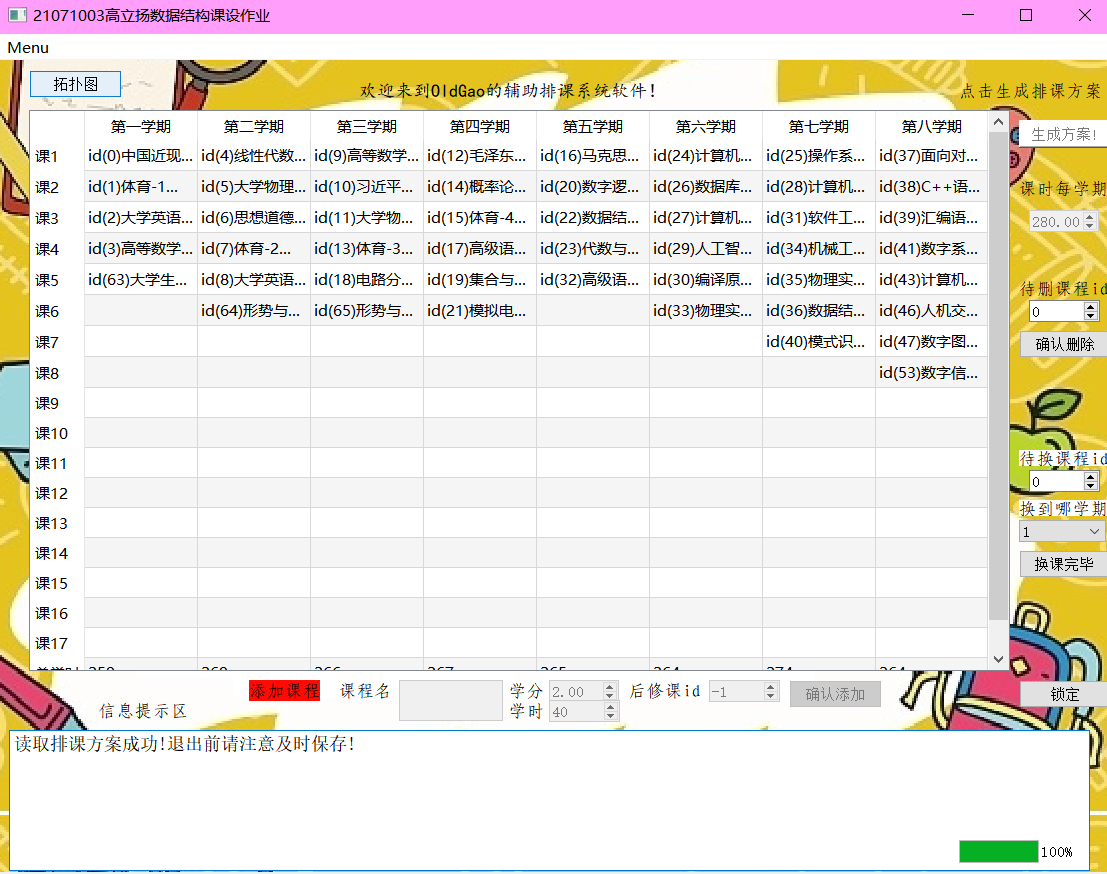


图4.11 读取成功

4.1.5 课时调整测试

测试用例：调整课时为380和500，对比图4.11的课表（280课时）。

测试结果：如图4.12和4.13，可以看出课表密集程度发生改变，如图4.14和图4.15查看课时发现此时课程安排仍然是合理的，课时=380较为贴近真实课程密度。如图4.16，改变课时成功的话，信息提示区会进行反馈。



图4.12 课时=380



图4.13 课时=500



图4.14 课时=380



图4.15 课时=500

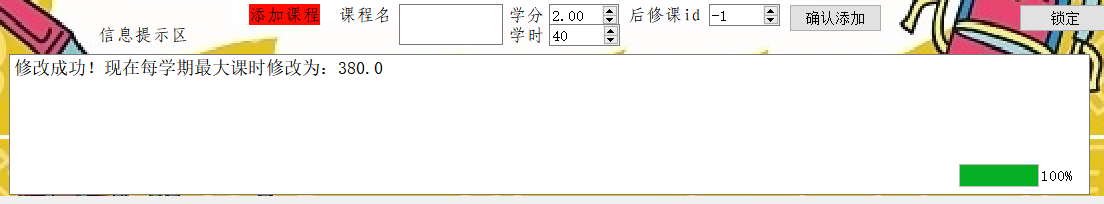


图4.16 Ui反馈

4.1.6 添加课程/新拓扑关系测试

测试用例：本测试重在拓扑图新有向边的添加。首先添加新课程test作为from，在“后修课id”填入作为to的课程的id；接着从课表中的单元格复制课程名称到“课程名”中，然后添加新后修课id。最后添加test2，无后修课。

测试结果：如图4.17，首先进行test课程的添加，将其后修课设置为id64=“形势与政策1”，可见Ui进行了反馈。如图4.18可见courses.csv添加了课程，如图4.19可见Graph文件也添加了新边。如图4.20所示，为了方便查看，设置课时为500，重新生成排课方案之后，test成功加入课表，并且符合拓扑顺序，与图4.21更能看出课程添加、拓扑顺序正确。

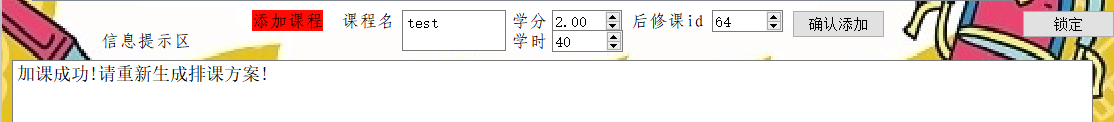


图4.17 test添加-前端

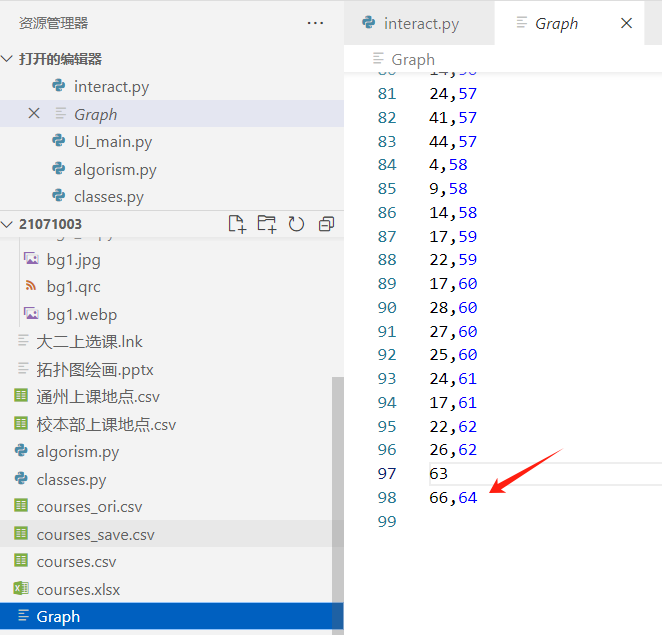
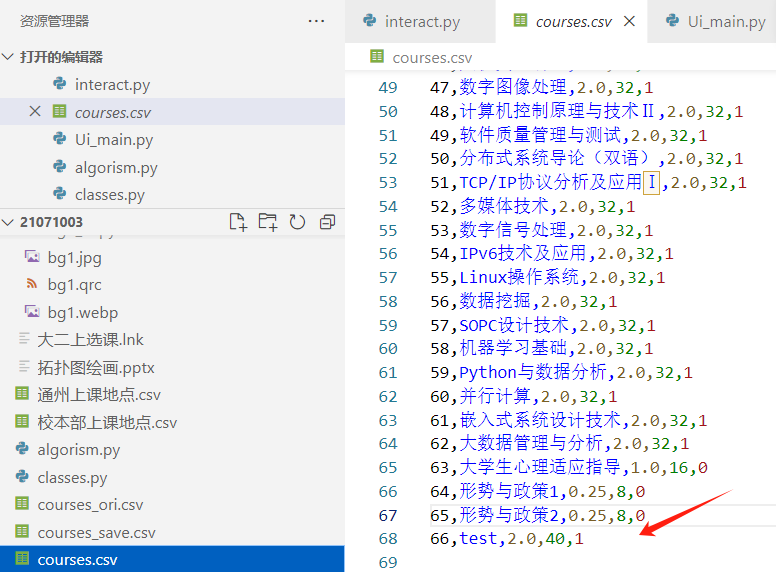


图4.18 课程添加-后端 图4.19 课程添加-后端



图4.20 课表拓扑关系

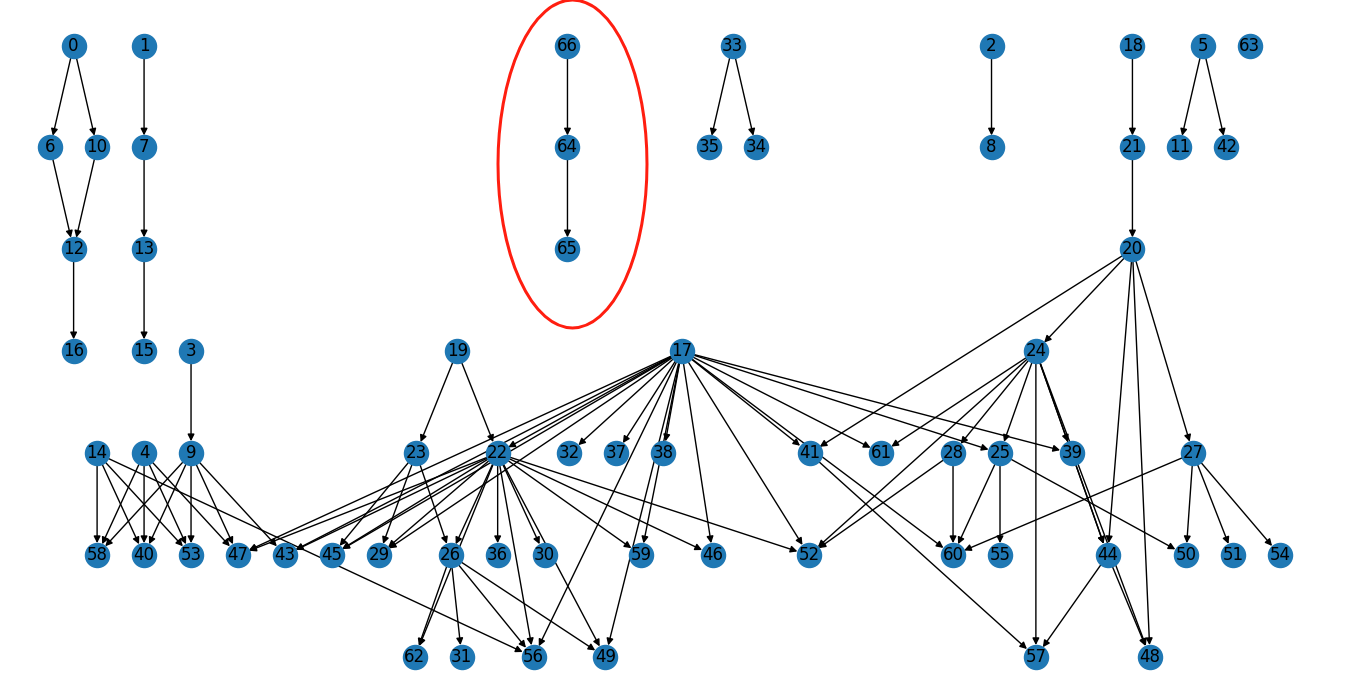


图4.21 拓扑图新关系

接着添加已有课程的新边，以id10=习近平新时代中国特色社会主义思想概论为例，设置其后修课为id6=思想道德与法治。如图4.22，添加成功。如图4.23&4.24，后端也添加成功，并且从重新生成的课表来看，id6永远在id10之后一学期出现，这一点从图4.25可以更清晰反映出。

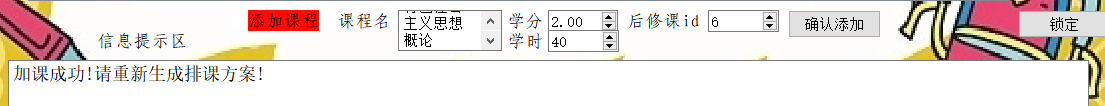


图4.22 添加有向边

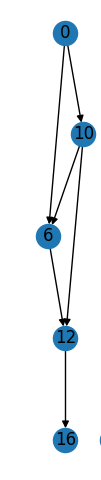
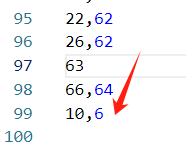


图4.23-4.24 添加成功 图4.25新拓扑

最后添加test2，后修课为无，如图4.26，添加成功，后端文件不再展示。如图4.26，拓扑图中出现了单独的一个点，左边63为心理课，右边67便是test2。

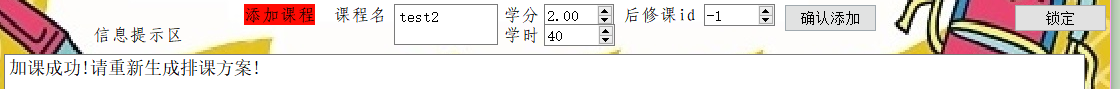


图4.26 单独添加 图4.27 单独点

4.1.7 删除课程测试

测试用例：用之前保存的课表，进行课程删除，观察课表变化

测试结果：读取的课表如图4.11，不再粘贴。如图4.28所示，删除了id0课程，信息提示区有反馈，观察课时也发生了改变，如图4.29-4.30，删除的正是48课时，即id0的课时。

成功修改也代表着保存和读取课表的功能是没错的。

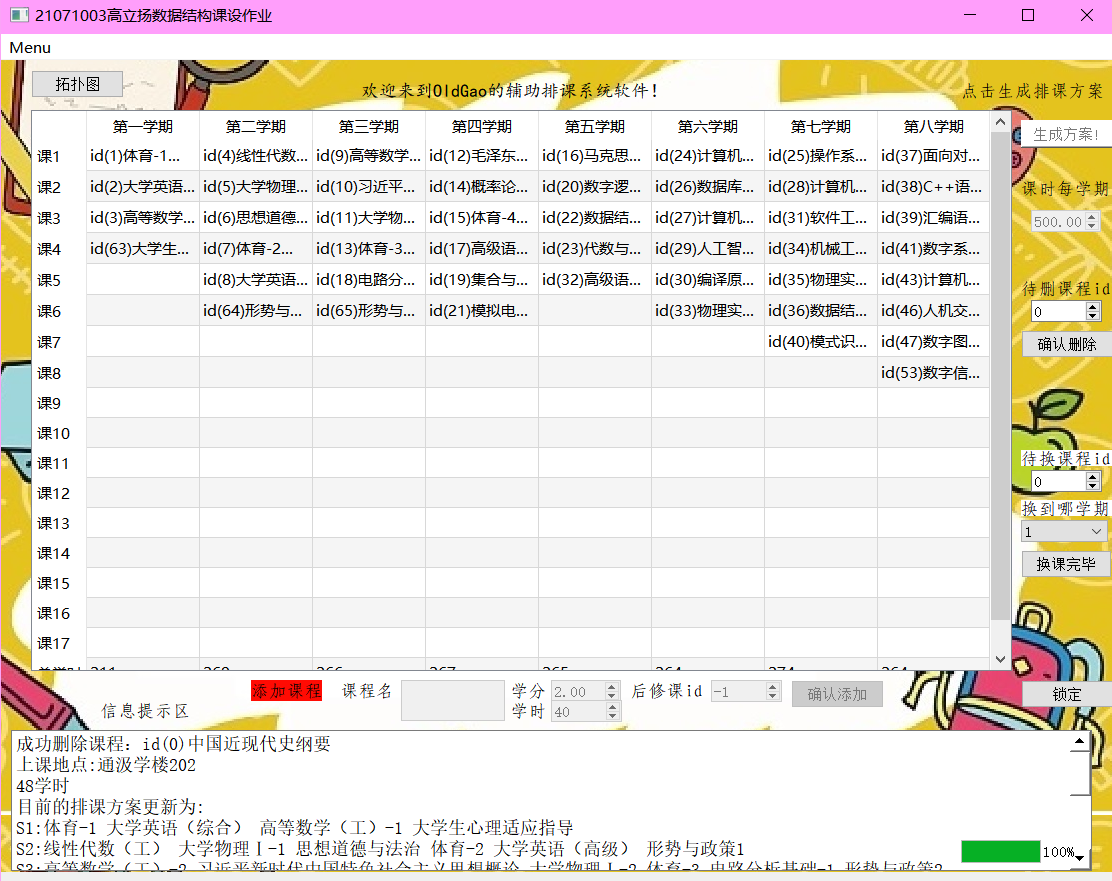


图4.28 删除课程成功



图4.29-4.30 before & after

4.1.8 调换课程测试

测试用例：基于图4.28，继续调换课程。

测试结果：如图4.31，调换成功，课时为280,，信息提示区反馈。如图4.32，调换之前为264课时，差值16课时对应了该课程课时为16。



图4.31 调换成功



图4.32 before

* 1. 错误用例

课表生成、课时调整、拓扑图约束、保存和读取由于存在约束，无错误用例。重点且需要展示的是添课、删课、换课的错误用例。

4.2.1 添加课程错误用例

该测试重点在于后修课id是否正确，以及先后修关系是否确保无环。

用例1：后修课id对应的课程不存在

测试结果1：如图4.33所示，id100的课程不存在，因此Ui信息提示区提示用户不存在该课程，等待用户下一步操作。此错误用例并不会导致程序崩溃。

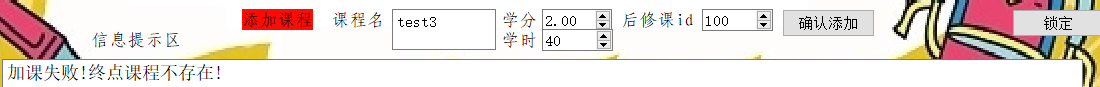


图4.33

用例2：添加6->0构成环

测试结果2：如图4.34所示，试图添加6->0有向边，由于已有0->6，因此添加后会构成环，添加失败。

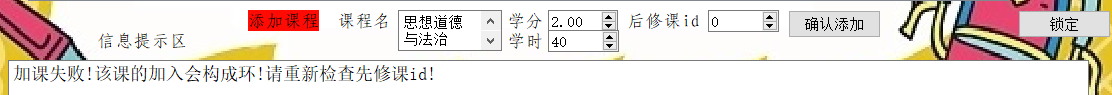


图4.34 成环

由于输入框限制用户只能输入整数，因此不会出现用户输入的内容非数字的情况，也不会出现用户输入的数字小于-1的情况

4.2.2 删课错误用例

该测试重点在于验证程序是否能在删除课程时考虑到不破坏拓扑关系，也需要测试用户错误输入的情况。

用例1：删除课程id100，该课程不存在

测试结果1：如图4.35所示，程序并不会崩溃，且提示删除课程不存在。

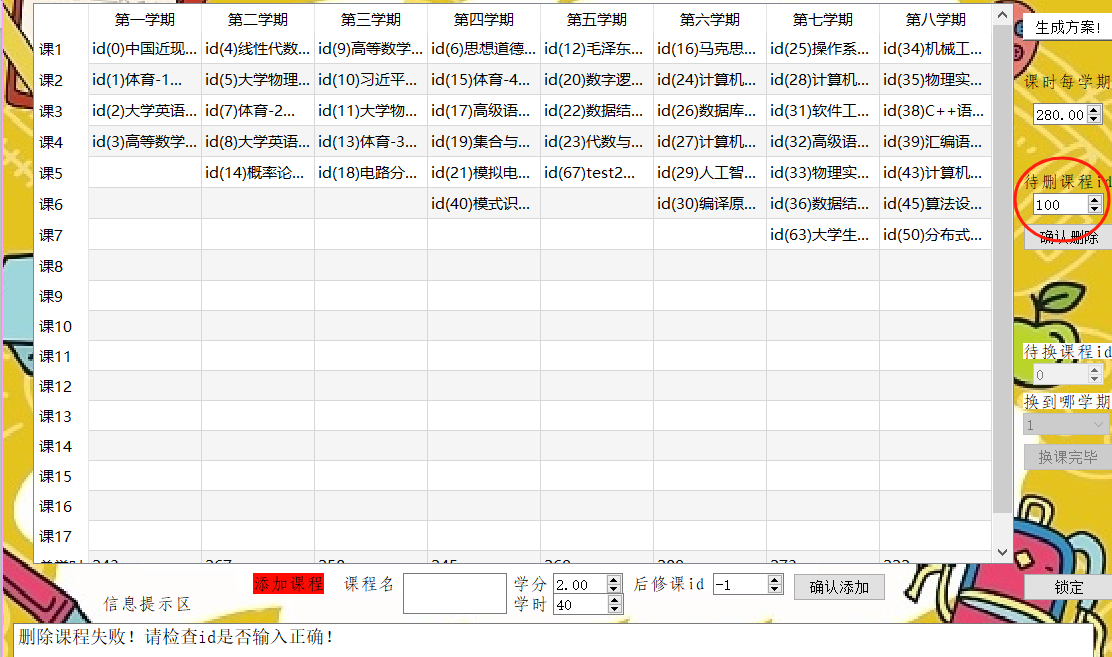


图4.35 不存在课程

用例2：删除id6，该课程先修课为id0，后修课为id12（见上文4.1.6）

测试结果2：如图4.36，信息提示区会提示课程无法删除，因为这会破坏先后修关系，程序并不会崩溃。



图4.36 破坏拓扑关系

输入框限制了用户只能输入非负整数，因此不会出现用户输入id号为负数或非整数的情况。

4.2.3 换课错误用例

该测试重点在于验证调换课程后不破坏先后修关系安排。

用例1：将id6调换到其先修课id0所在的学期；由于判断后修课的算法与判断先修课的一致，因此不再对后修课调换进行验证，同理可得。

测试结果1：首先需要点击信息提示区右上角的“锁定”才能开启课程调换。如图4.37，换课失败，因为id6调换到第一学期会导致它与自己的先修课安排在同一学期。

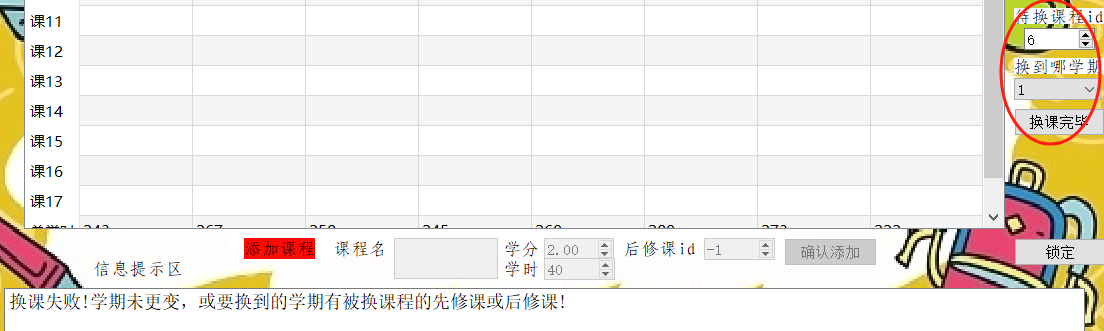


图4.37 调换失败

待换课id输入框限制了用户只能输入非负整数，因此不会出现输入为负数或非数字的情况。而换到哪学期采用下拉框，不会出现输入错误。

* 1. 极端情况

程序内涉及id的输入框，最大值到1000，但是结合实际情况来看，用户添加课程到不了第100门，课表就会因为课程过多而导致很多课程无法安排，这显然也是不符合实际教学需求的，因此课程过多不应当算极端情况。

调换课程存在先后修课的严格限制，因此不会出现极端情况。

课表生成、拓扑图、课时调整、保存和读取不存在极端数据输入。

最后，对于删除课程来说，依次删除所有课程为极端情况，值得验证，并且在删除过程中可以穿插课程调换。

用例：读取4.1.3保存的课表，进行课程删除，删除约一半的课程后进行课程调换，最后删除全部课程。

测试结果：如图4.38所示，此时已经删除了一半的课程，进行课程调换。调换结果如图4.39所示，并未出现错误。如图4.40所示，课程全部删除且程序未出错。



图4.38 删除一半课程



图4.39 随机调换课程

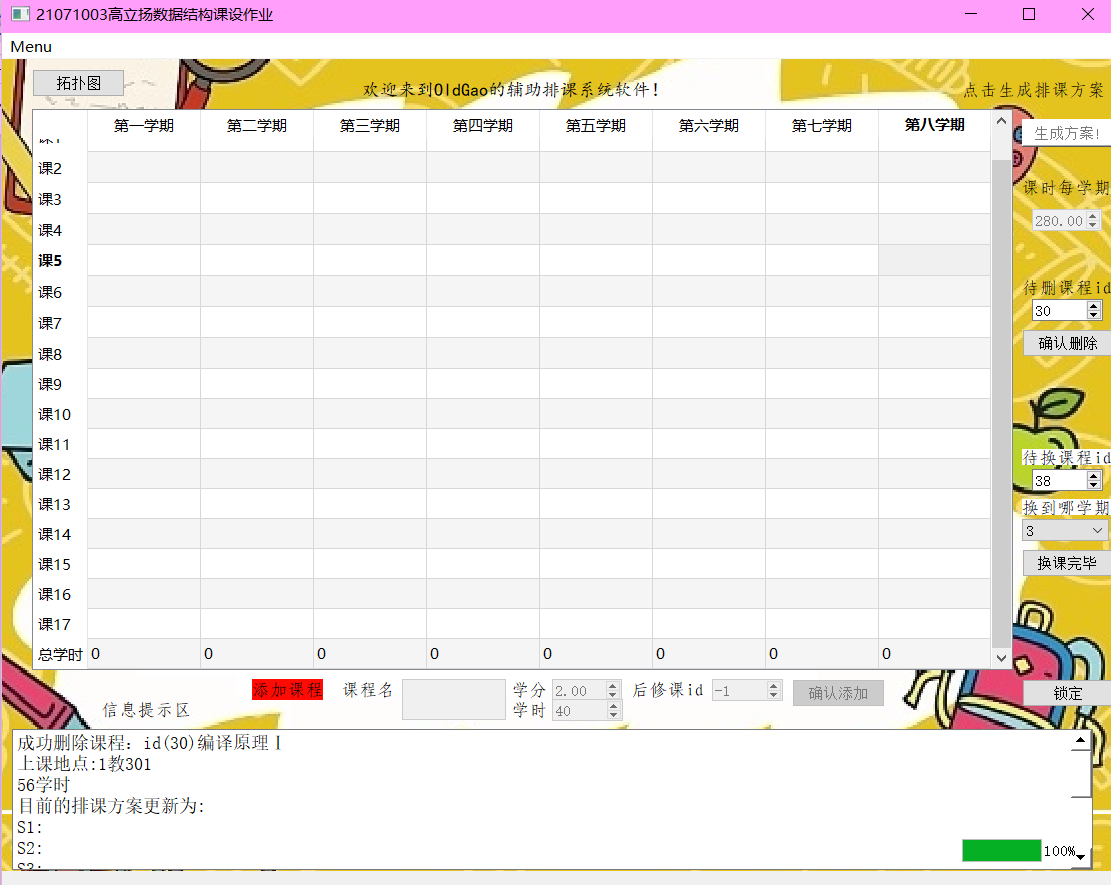


图4.40 全部删除成功

1. 总结与提高
   1. 体会与收获

通过本次课设，我收获颇多：

①第一次使用Python语言完成了课设，体会到了其方便性；

②第一次不参考网上资料，完全自主进行核心算法编写和调试；

③体验了快速开发。由于本学期任务繁多，因此相较于大一的C语言课设，本次课设我采用了较短的时间进行开发；

④学习了Qt框架的使用，体验了槽函数和Qt Designer为交互所带来的的便捷；

⑤在编写前端算法时，因为开发初期没有做详细的规划和设计构思，导致需要写很多冗余代码来弥补因此产生的bug，到现在来看，部分算法在阅读起来效果欠佳，且冗余代码会带来一定的时间开销。但是在写冗余代码修bug的时候，我仍采用了剪枝等方法，将损失降到最小，这也启发着我以后在进行程序开发之前，先要尽可能考虑周到，写文档来规范化设计等；

⑥应当多与同学进行交流，在开发过程中，有一次我和同学展示程序功能时，发现了意料之外的bug，后来在观看其他同学的程序时，同学的程序也发生了一些他们没有想到的bug，这说明在开发时，应当多沟通多交流，自己一个人总有想不到的意外情况；

⑦写报告是一件十分辛苦的事情，由于本学期我选了不少课，系统软件课设选择了难度最高的pintos，平时也在和中科院软件所沟通学习，时间非常紧，从报告中一些文件的日期就可以看到，我用了六天中一切零碎的时间来写报告。虽然时间紧，但是我仍愿意静下心来好好写我的报告，我认为这不仅是对自己课设程序的尊重，也是对个人耐性的锻炼。日后读研我肯定会走向科研，面对未来的磨炼，我认为在本科阶段应当耐下心来，做好每一件事。

* 1. 问题与解决

在开发过程中，我遇到了不少问题，在前端槽函数编写遇到的问题占大多数，基本可以总结为如下几类：Python语法遗忘、不会Qt语法、特例考虑不周到、后台数据更新错误。针对这几类问题，我的解决办法分别是：

在test.ipynb文件内进行测试，通过print重温一些方法的功能；

查Qt文档，对照调试；

特例考虑不周到会导致交互时发生错误，程序崩溃，于是我就根据错误追根溯源，通过print方法打印数据，并观察后台数据，得出结论并添加对特例的处理代码；

后台数据更新错误，也会导致程序崩溃，或数据显示不正确的问题，因此仍然用到了print法，并在Ui界面操作时，时刻观察出错的后台数据，最后根据数据变化发现问题，并解决问题。

* 1. 自我评估

按照课设的要求和我个人的完成情况，如果满分10分，我对我自我评价为9分。我认为我的程序已经考虑到了十分多的特殊情况，并且针对不同的特殊情况都会有不同的信息反馈，有良好的交互性，并且完成了一些课设要求之外的内容，比如删除课程功能。

缺点肯定是有的，我不能逃避自己的缺点，我认为我的课设最大的缺陷就是前端算法中，因为设计初期考虑不周到导致需要用冗余代码来修bug。

具体情况是，交互时需要用到图、课表信息，而后端创建这两个数据结构的算法都是封装好的，无法调出来，因此我需要用全局变量去接收。因为前端算法需要对图和课表信息进行修改更新，比如topology算法，为了不影响删课和调课的使用，我需要深拷贝上述两个数据结构。但是拷贝之后，无法用原数据结构的成员去搜索拷贝数据结构的成员，因此就要通过一遍遍的遍历，来通过两数据结构内成员的特征进行对比寻找到目标。

尽管我设置了循环剪枝，遍历过程仍带来了巨大的无用时间开销。幸好对于该课设来说，数据量不算大，因此带来的开销不明显，在使用过程中几乎没有卡顿。

这都是因为缺少前端开发经验，开发之前缺少规划导致的，因此自我评估里我为我自己扣一分扣在这里，在以后的开发中，我会吸取本次课设带给我的教训！