

**课程设计报告书**

|  |  |  |  |
| --- | --- | --- | --- |
| **课堂表现及考勤30%** | **系统成绩50%** | **报告成绩20%** | **总成绩** |
|  |  |  |  |

**课 程 名： 数据结构课程设计**

**专业班级：**

**学 号：**

**姓 名：**

**联系电话：**

**指导老师：**

**时 间：**

**计算机科学与工程学院**

1. **课程设计概述**

本次数据结构课程设计共完成三个题目：1-（1）一元稀疏多项式计算器、2-（12）背包问题的求解、3-（22）全国交通咨询模拟。

使用语言：Java语言

编译环境：openJDk-1.8

**二、课程设计题目一**

**【实验内容】**

1-（1）一元稀疏多项式计算器

**【问题描述】**

设计一个简易的一元稀疏多项式计算器。

**【需求分析】**

经过分析，本系统需完成的主要功能如下：

1. 通过图形化界面点击按钮输入或键盘输入多项式
2. 通过输入限制确保符合一元稀疏多项式的格式
3. 通过按钮组选择输出结果：多项式计算结果、在某点的值、多项式的导数(扩展功能)
4. 多项式的计算功能
5. 多项式的求值功能
6. 多项式的求导功能

**【概要设计】**

-=ADT=-

{

void do\_bracket\_event (String ss); //括号输入操作

void do\_mark\_event (String ss); //运算符输入操作

void do\_backspace\_event (String ss); //删除操作

void do\_number\_event (String ss); //数值输入操作

void do\_ decimalPoint\_event (String ss); //小数点输入操作

void do\_ equals\_event (String ss); //运算输出操作

void do\_ clear\_event (String ss); //清空重置操作

Boolean resultPreOperate(); //运算输出前置操作

void derivation(); //导数运算

void value(); //数值运算

void Polynomial(); //多项式运算

}

**【存储结构】**

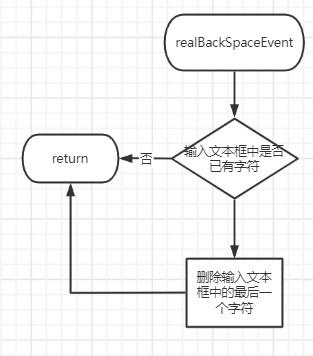
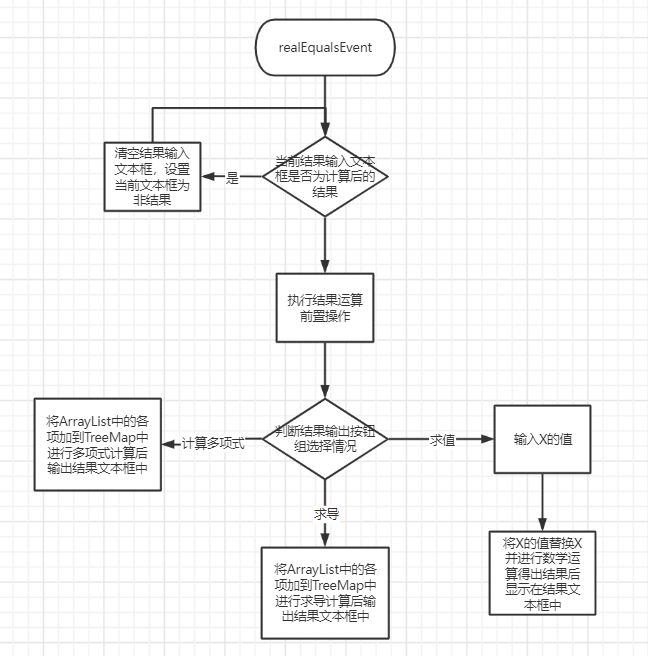
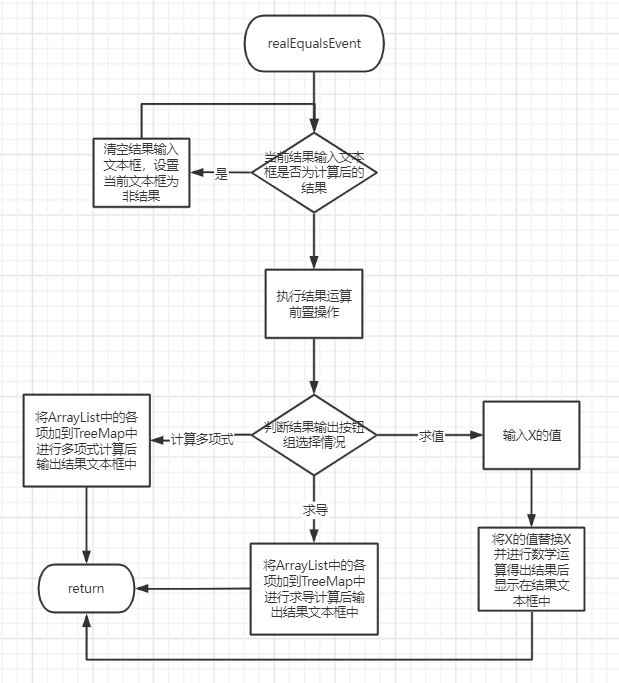
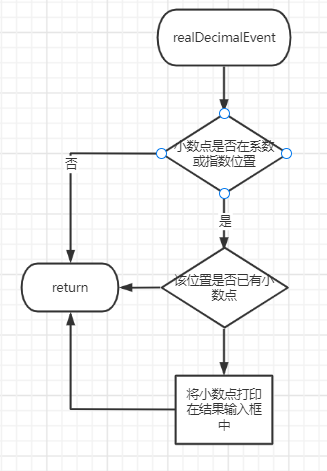
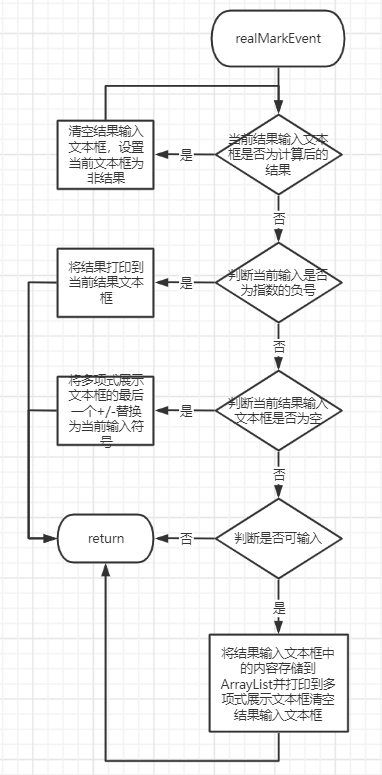
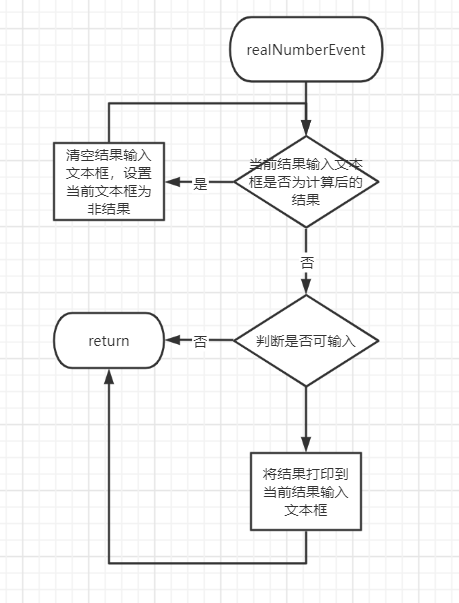
通过ArrayList<ArrayList<String>>和TreeMap<Double, Double>存储

ArrayList<ArrayList<String>>外层存储整个多项计算式中的每个一级多项式

内层ArrayList存储每个一级多项式中的每项

TreeMap进行运算过程，将每一项的指数作为Key，相同指数的系数加和作为对应的Value

**【算法流程图】**



**【算法时间复杂度分析】**

本程序主要以操作treeMap为主，故时间复杂度为O(logn)

**【调试分析】**

本程序主要的操作对象是ArrayList与TreeMap，最重要的操作是输入限制和字符串切割，这是本程序中的重点，是此程序可能出现问题的主要原因之一：

【**问题一】**

**现象：**输入计算式在检测加入ArrayList时出现错误。

**原因：**程序期待的输入内容应为aX^b（其中a为系数b为指数）但实际输入过程中会有所省略。例如在指数为1的时候会省略^b，在指数为0的时候（即该项为常数项时）会省略X^b，所以在实际运行过程中应加入相应的分支判断来保证省略写法依旧能代表完整意义加入到存储结构中，且在输出过程中也应该进行相应的输出优化以符合日常用户阅读书写逻辑，例如当系数和指数为整数时不再显示小数点及其后面的0，当系数为1或-1的时候省略掉系数中的数字，当该项为常数项时只显示数字，当该项的指数为1时不显示指数，前面没有其他项的时候加号不再显示，计算后系数为0的项不再显示，当所有项系数均为0的时候最后结果显示为0。

**【问题二】**

**现象：**ArrayList加载到TreeMap时出现错误。

**原因：**输入限制限制不全面。应严格限制输入规范，如加减号不应连续贴合出现，每一项中应只出现一个X，每一项中只能出现一个^，结果输入框中只有在^后面才能添加-，小数点的数量应该严格规范不应在系数或指数部分出现多个小数点导致数值录入失败。

**【问题三】**

**现象：**结果输出指数项为正序。

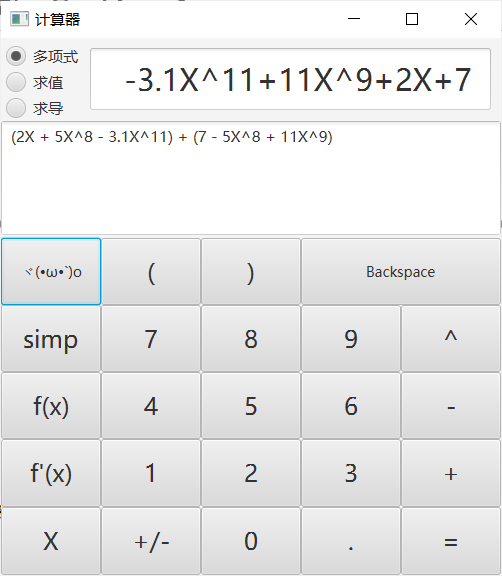
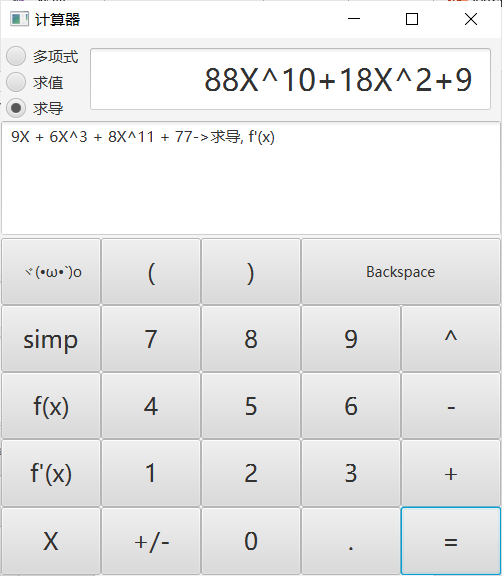
**原因：**比较器自然排序默认从小到大。在实例化TreeMap的同时传入逆序的比较器(o1, o2) -> -Double.compare(o1, o2)。

**【问题四】**

**现象：**结果输出文本框中从右开始会使部分字符顺序出现问题，如 “^” “-”。

**原因：**JavaFx原生问题。设置右对齐替换文本从右开始，尝试使用CSS样式进行控制发现JavaFx的TextField并不能很好的适配CSS的文本右对齐，最终在FXML文件中设置alignment="CENTER\_RIGHT" 和 nodeOrientation="LEFT\_TO\_RIGHT"完成右对齐操作并不出现视觉错误。

**【运行结果及分析】**

**三、课程设计题目二**

**【实验内容】**

2-（12）背包问题的求解

**【问题描述】**

求装入哪些物品后恰好背包装满。

**【需求分析】**

经过分析，本系统需完成的主要功能如下：

1. 通过图形化界面输入物品重量组和背包最大承重
2. 通过图形化界面输出背包装入物品组合
3. 通过图形化界面重置内容
4. 使用递归方法解决问题
5. 重置内容

**【概要设计】**

-=ADT=-

{

void recursionSolution (); //使用递归规划解决问题入口

void dfs(int V, int index); //实际递归方法

void output(); //递归的输出函数输出最终结果

void DPSolution(); //使用动态规划解决问题入口

void findAnswer(); //找出件数为1的满足条件的解决办法的入口方法

void find(int start, int[] nums, int count, int tempSum); //找出件数为1的满足条件的解决办法的实际方法

void logAnswer(); //输出最终结果

}

**【存储结构】**

通过ArrayList和数组进行存储

**【算法流程图】**

1. 如果nums序号集合对应的物体总重量大于背包重量直接返回。（因为如果nums序号对应的物品总重量大于背包重量的话，再加其他物品的重量了肯定也大于背包重量）
2. 判断nums序号集合对应的物体总重量是否等于背包重量，等于的话将序号集合加入到解决办法集合中，否则执行第3步
3. 从i = start + 1开始搜索，看i是否在nums序号集合中，如果不在的话，执行下一步操作，否则继续下一次循环。
4. 从第i间开始搜索，找出件数为count+1的满足条件的解决办法。

**【算法时间复杂度分析】**

递归实现时间复杂度位O(2n)

**【调试分析】**

本程序的代码实现皆与递归有关调试难度较大，在图形化界面搭建时发现Javafx的部分原生问题：

【**问题一】**

**现象：**出现栈溢出错误。

**原因：**出口的判断出现问题。在递归问题中递归方法的出口应该清晰明确，避免栈溢出问题的发生。

**【问题二】**

**现象：**Spinner组件数字读取失败。

**原因：**Javafx中存在的原生问题。带微调器的Spinner组件在文本框输入文本后应以回车结束，方可正确放入到文本框中可以读取。此外在fxml文件中还需定义Spinner的最大值与最小值，在Controller中定义Spinner的的id时应加入Integer的泛型。

**【运行结果及分析】**



**四、课程设计题目三**

**【实验内容】**

3-（22）全国交通咨询模拟

**【问题描述】**

处于不同目的的旅客对交通工具有不同的要求。例如，因公出差的旅客希望在旅途中的时间尽可能地短，出门旅游的游客期望旅费尽可能省，而老年旅客则要求中转次数最少。本题目要求编制一个全国城市的交通咨询程序，为旅客提供两种或三种最优决策的交通咨询。具体要求如下:

(1)提供对城市信息进行编辑(如:添加或删除)的功能。

(2)城市之间的交通工具是火车。提供对列车时刻表的管理功能(增加，删除，查询，修改)。

(3)提供两种最优决策:最快到达和最省钱到达。

(4)旅途中耗费的总时间应该包括中转站的等候时间。

(5)咨询以用户和计算机的对话方式进行。由用户输入起始站、终点站、最优决策原则，输出信息:最快需要多长时间才能到达或者最少需要多少旅费才能到达，并详细说明依次于何时乘坐哪一趟列车或哪一次班机到何地。

**【需求分析】**

经过分析，本系统需完成的主要功能如下：

1. 设计城市地图的类
2. 设计站点(有向边)的类
3. 设计实现弗洛伊德算法解决多源最短路径问题
4. 通过文件读取写入修改站点城市班次等信息
5. 设计工具类方法完成时间的计算
6. 设计可视化界面进行交互操作

**【概要设计】**

-=ADT=-

{

boolean setURL(String newURL); //设置操作文件路径

ArrayList<String> getPermulation(int[] arr); //获取全排列的所有可能

BufferedReader getBr(String vehicle); //获取对应的BufferedReader

boolean addInfo(String vehicle,String info); //添加指定信息到指定文件中

boolean deleteInfo(String vehicle,String start,String end); //删除文件中的信息

String transformTime(int costTime); //将分钟（整形）转化为x小时x分钟的字符串形式

StringBuilder getPathInfo(String path,String vehicle,String bestChoice,int costInfo); //获取路径的具体信息，从文件中读取挑选并存入StringBuilder中，并返回StringBuilder

MGraph readInfo(String vehicle); //从指定文件中读取数据，生成一个MGraph对象并返回

void FinishPath(MGraph mg); //完成路径

int addWaitTime(MGraph mg,int v,int u,int w); //添加中转站的等待时间

int getCostTime(String date1,String date2); //传入两个字符类型的时间，计算出两个时间的差值

}

**【存储结构】**

使用邻接矩阵的形式存储图

**【算法流程图】**

设置顶点vi到顶点vk的最短路径已知为Lik，顶点vk到vj的最短路径已知为lkj,顶点vi到vj的路径为li，则vi到vj的最 短路径为: min(Lk+Lkj),Lij), vk的取值为图中所有顶点，则可获得vi到vj的最短路径。

至于vi到vk的最短路径Lik或者vk到vj的最短路径Lkj,是以同样的方式获得。

**【算法时间复杂度分析】**

使用弗洛伊德算法时间复杂度为O(n3)

**【调试分析】**

在有向图的重新构建的时候易出现空指针异常，由于弗洛伊德算法循环量较大，调试时出现问题时需要多点定位寻找错误点，在判断齐之后又出现了以下问题：

【**问题一】**

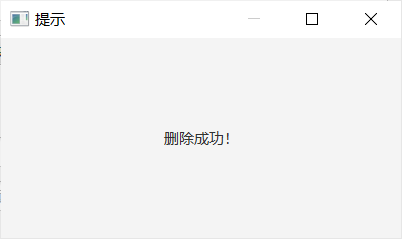
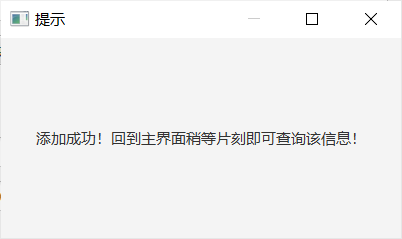
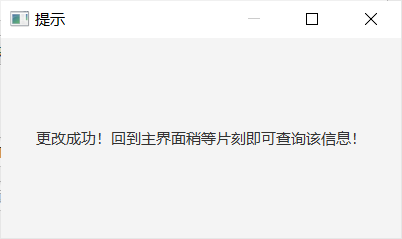
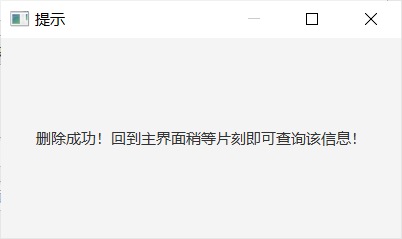
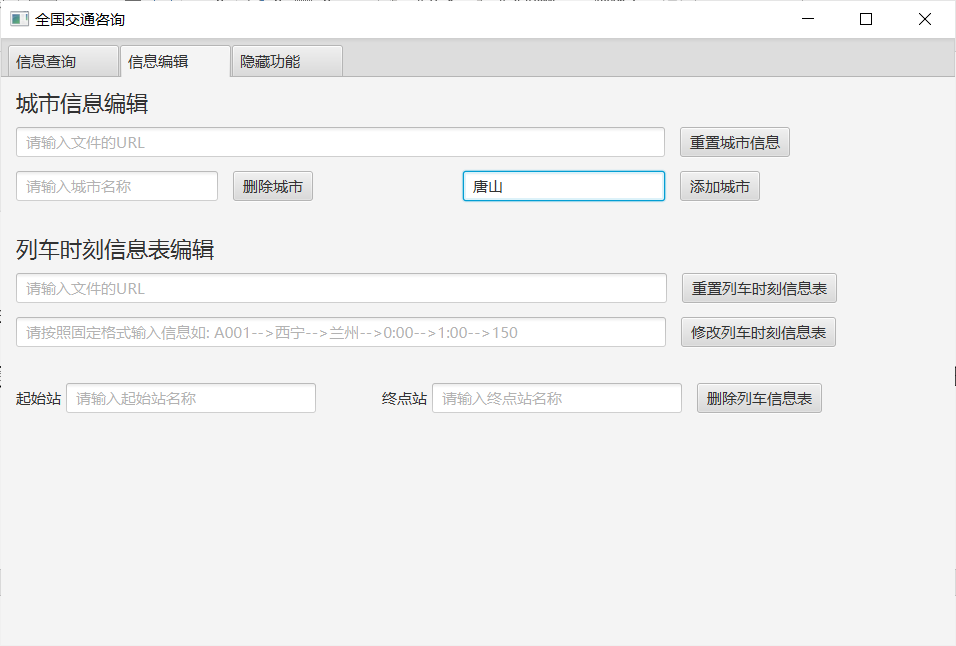
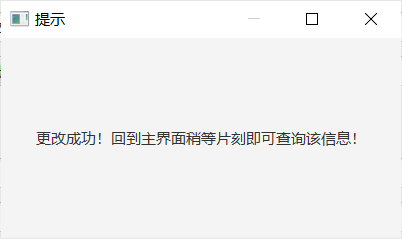
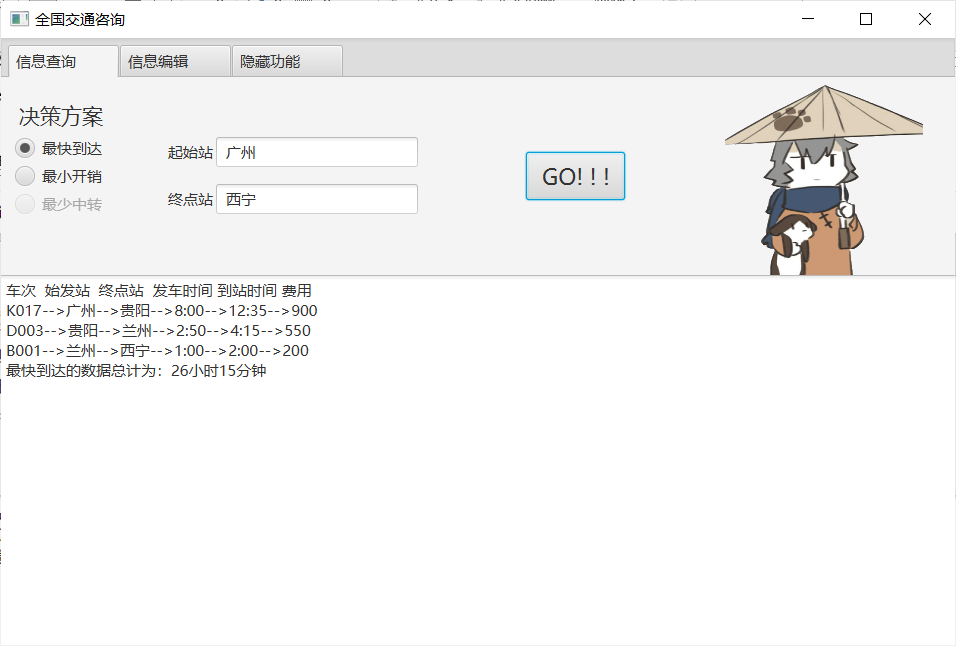
**现象：**时间输出出现错误。

**原因：**加入额外的时间解析类。例如在12:00 🡪11:00中间的时间应该为23个小时而非-1小时或1小时，当前后时间差为负数时应+24小时而进行计算。并在优化体验中计算的单位为小时和分钟，应注意分钟为60进制。

**【问题二】**

**现象：**ImageView组件图片无法加载。

**原因：**路径文件夹应加载在out文件夹下。Fxml的路径设置和Java默认的路径设置不尽相同，Java中的相对路径默认实现是从工程文件的根目录开始寻址，而JavaFx是从当前文件夹通过相对路径寻址。

**【运行结果及分析】**

**三、参考文献**

梁勇著，戴开宇译，《Java语言程序设计与数据结构》（进阶篇）（原书第11版），机械工业出版社，2018