重庆师范大学

课程设计报告

课程名称：数据结构与算法

题目名称：算法解析系统

学生学院：计算机与信息科学学院

专业班级：计算机科学与技术 2020 级 3，4 班

小组组长：张诗雨

小组成员：张诗雨　袁雪琴　杨昌松　杨鸿强

指导老师：唐万梅　　职称　教 授

2020 年 6 月 17 日

目 录

算法教学系统

1 设计目的

《数据结构课程设计》是面向计算机科学与技术专业的一门综合运用课。数据结构课程平时的实验题目，往往局限于当时所讲授章节的知识运用，所解决的问题规模小，与实际有差距；数据结构课程设计题目则着眼于整个数据结构课程所学知识的综合运用和软件开发整个流程的综合训练，所解决的问题也更接近实际。

本课程设计针对实际问题，归纳和抽象描述实际问题的数据及其逻辑关系，选择合适的存储结构在计算机内部表示这些数据及其逻辑关系，进而设计出整个软件架构和具体算法来解决实际问题。

本课程设计涉及到程序的编码、调试与测试，撰写实验报告。实验报告要求全面反映设计流程，充分描述主要设计环节，分析测试结果，分析主要算法的时间和空间效率，提出改进设想。

通过上述综合训练，能极大的扩展大家的综合运用数据结构知识分析问题、解决问题的能力，培养大家软件开发和软件调试能力，培养软件文档和科学论文的写作能力。

本次的课程设计提供了在实践中应用算法的思想，感受算法思想，加深相关理解与认识的机会；在实践让学生通过自身编写和调试程序，真切体会到任何粗枝大叶和思维不缜密都会带来 bug 和隐患，从而培养其工匠精神，树立起科技报国的家国情怀和使命担当。

本小组的算法解析系统的设计目的，是让算法学习简单化，减少算法学习过程中的困难，通过本系统提供的解析，能够更加清晰明了的，懂得在某种算法过程中，它各种数据的变化形式，给使用者一个清晰明确的反馈，增进算法学习的积极性。

给的算法学习带来便利，通过这种学习方式，让用户思维更加严密，给用户代码能力带来提高，提升用户代码质量与效率。

2 问题描述

算法学习是十分抽象的，且多数计算过程复杂、逻辑性要求很高，稍不细心就会导致计算出错，得到错误的结果或者逻辑混乱思考很久也无法找到正确的解题思路。在本系统的辅助下，用户能够更快的理解算法，让使用者的思路跟着系统的计算阶段前进，

将算法按阶段进行的剖析，每个阶段都能在用户的脑海中清晰呈现，每个阶段的作用及其结果，和对后续阶段计算的影响。

为解决算法学习中遇到的种种困难，本次课程设计挑选了如下几个算法进行了深度的剖析，将每个算法的计算过程输出到屏幕上，让算法的运行过程清晰明了：

（1）矩阵的运算

（2）最长公共子序列

（3）最大子段和

（4）最优二叉搜索树

（5）打家劫舍问题

（6）归并排序

（7）全排列算法

在用户正确输入相应的输入后，系统便会进行计算，给出相应的解题过程，把该算法的计算过程呈现在用户面前，为用户的算法学习提供便利。

3 工作分配

本项目设计的分工如下：

系统设计：张诗雨， 袁雪琴， 杨鸿强， 杨昌松

程序设计：张诗雨， 袁雪琴， 杨鸿强， 杨昌松

文档制作：杨鸿强， 杨昌松

PPT 制作：张诗雨， 袁雪琴

需求分析：张诗雨， 袁雪琴， 杨鸿强， 杨昌松

视频录制：杨昌松

4 基本功能

本系统通过将算法分阶段解析，把算法的计算过程呈现在用户面前，为用户的算法学习提供帮助。

本系统提供了五个算法的解析：

（1）矩阵计算

对两个 n 维矩阵进行加、减、乘、求逆等的运算，并给出计算结果。

（2）最长公共子序列

给定两个由 n 个字符组成的线性表，求这两个线性表公共子序列的最长的长度。

（3）最大子段和

给定 n 个整数组成的线性表，求出这个线性表连续子段节点和的最大值。

（4）最优二叉搜索树

对于一个给定的概率集合，构造一棵期望搜索代价最小的二叉搜索树，称为最优二叉搜索树。对本问题来说，穷举并检查所有可能的二叉搜索树不是一个高效的算法，需要指数时间。所以，这里使用动态规划方法求解此问题。

（5）打家劫舍问题

你是一位高级小偷，你在一个房屋的排列类似于一棵二叉树的地方进行盗窃。 如果两个直接相连的房子在同一天晚上被打劫，房屋将自动报警。计算在不触动警报的情况下，你一晚能够盗取的最高金额。

（6）归并排序

①申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列；②设定两个指针，最初位置分别为两个已经排序序列的起始位置；③比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置；④重复步骤③直到某一指针达到序列尾；⑤将另一序列剩下的所有元素直接复制到合并序列尾。

（7）全排列算法

从 n 个不同元素中任取 m（m≤n）个元素，按照一定的顺序排列起来，叫做从 n 个不同元素中取出 m 个元素的一个排列。

5 算法思路

5.1 总体设计

利用 C++ 面向对象的思想进行编写，每个人的代码封装成类，将系统所解析的算法：矩阵运算，最长公共子序列，最大字段和，最优二叉生成树，抢劫问题，归并排序，全排列算法等七个不同算法，进行封装，由每个成员分别对该类进行设计，当用户打开该程序，由主程序进入，自行选择需要学习的算法，选择算法后，会该算法会提供一个案例输入，提示用户应该输入怎样的数据，然后将会在用户输入数据后，会展示该组数据在该算法中的各个阶段的变化，并将该过程直观明了的显示给用户，让用户理解不同的算法中，数据变化的不同，带了的不同的运行效率。通过这种学习方式，给用户代码

能力带来提高，提升用户代码质量与效率。

5.2 子项目

5.2.1 矩阵运算

加法：

（1）用户输入矩阵 A、矩阵 B；

（2）首先判断矩阵 A、矩阵 B 的行列是否相同，如果相同才能进行矩阵的加法运算；

（3）提示用户矩阵加法运算的步骤；

（4）将加法运算分解为每一行分别相加，提示用户每行相加的结果；

（5）最终再将答案输出。

减法：

（1）用户输入矩阵 A、矩阵 B；

（2）首先判断矩阵 A、矩阵 B 的行列是否相同，如果相同才能进行矩阵的减法运算；

（3）提示用户矩阵减法运算的步骤；

（4）将减法运算分解为每一行分别相减，提示用户每行相减的结果；

（5）最终再将答案输出。

矩阵相乘：

（1）用户输入矩阵 A、矩阵 B；

（2）首先判断矩阵 A 的列数与矩阵 B 的行数是否相同，如果相同才能进行矩阵的乘法运算；

（3）提示用户矩阵乘法运算的步骤；

（4）将乘法运算分解为矩阵 A 的每一行分别与矩阵 B 的每一列对应的元素，然后求和后按行排列，为提示用户所得结果；

（5）最终再将答案输出。

矩阵的数乘：

（1）用户输入矩阵 A、数字；

（2）没有判断条件；

（3）提示用户矩阵数乘运算的步骤；

（4）将数乘运算分解为矩阵 A 的每一个元素分别与数字相乘，提示用户结果；

（5）最终再将答案输出。

矩阵求逆：

（1）用户输入矩阵 A；

（2）首先判断矩阵 A 是否行列数相同，如果相同才可以求该矩阵的逆；

（3）提示用户矩阵数乘运算的步骤；

（4）首先计算出矩阵的行列式，如果行列式为 0，则提示用户该矩阵不可求逆；

（5）其次根据行列式计算出矩阵的代数余子式；

（6）接着根据代数余子式求出矩阵的伴随矩阵；

（7）最后根据矩阵求逆公式 A^-1=A*/|A|求出该矩阵的逆矩阵；

（8）最终再将答案输出。

5.2.2 最长公共子序列长度

（1）首先分别输入两个序列的长度 m,n 代表节点数，再申请含有对应长度个节点的线性表。

（2）初始化一个二维表 c[m][n]来存储第一个序列的前 i 个元素与第二个序列的前 j 个元素之间最长公共子序列的长度

（3）当 x 中第 i 个元素与 y 中第 j 个元素相等时，c[i][j]=c[i-1][j-1]+1;当.当 x 中第 i 个元素与 y 中第 j 个元素不相等时，c[i][j]=max{c[i-1][j]，c[i][j-1]}

（4）最后 c[m][n]是得到的最大的公共子序列长度。

5.2.3 最大子段和

（1）首先输入 n 代表节点数，再申请含有 n 个节点的线性表。

（2）然后再声明一个 d[n]和 rec[n]数组，rec[n]数组用于存储从 i 往后加，得到的和最大的点的下标；d[n]用于存储从 i 加到 rec[i]的和，初始化 d[n-1]等于最后一个元素的值，rec[n-1]等于 n-1。

（3）从后往前遍历，当 d[i+1]的值大于零时，d[i]=d[i+1]+当前元素的值,rec[i]=rec[i+1]；当 d[i+1]小于零时，d[i]=当前元素值,rec[i]=i。

（4）最后遍历 d[n]中最大的值就是最大字段和。

5.2.4 最优二叉搜索树

使用两个一维数组存储输入的节点概率和空隙节点的概率，再使用两个二维数组储

存计算中间值，最后通过回溯，得到最终的结果，并将树的结构输出到屏幕上。

5.2.5 打家劫舍问题

使用链式二叉树来存储输入的各个目标节点，使用一维数组来存储计算中间值，以动态规划的思路对每个结点进行遍历，同时按照公式进行计算，每个子结构都选选择最优是，得到的最终答案也是最优值。

5.2.6 归并排序

（1） 申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列；

（2）设定两个指针，最初位置分别为两个已经排序序列的起始位置；

（3）比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置；

（4）重复步骤 （3） 直到某一指针达到序列尾；

（5）将另一序列剩下的所有元素直接复制到合并序列尾。

当用户输入的数组进入程序时，到了分割点，程序就会打印分割点的位置，使用户清楚了解该数组是怎么进行元素变换的。

5.2.7 全排列算法

从 n 个不同元素中任取 m（m≤n）个元素，按照一定的顺序排列起来，叫做从 n 个不同元素中取出 m 个元素的一个排列。当 m=n 时所有的排列情况叫全排列。当用户输入字符串，每一次都会取走当前数组中的第一个元素，之后的元素进行全排列，同时程序会打印关键位置的信息。

6 程序实现

本系统功能如下：

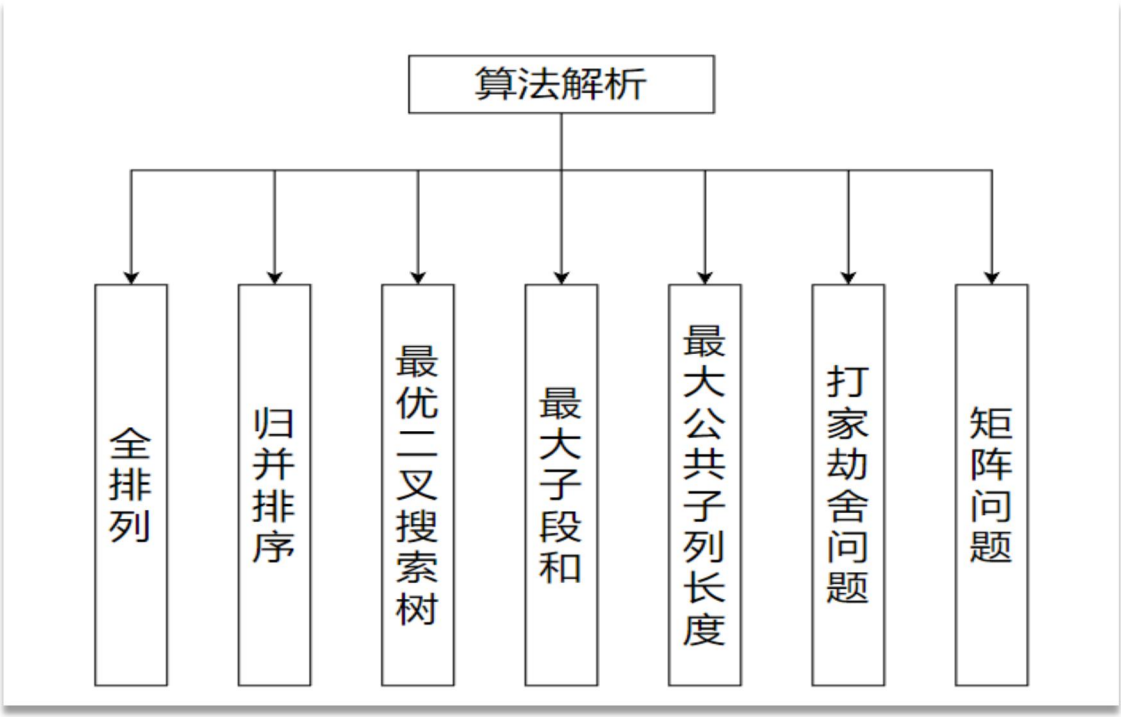

图 6-1 主框架

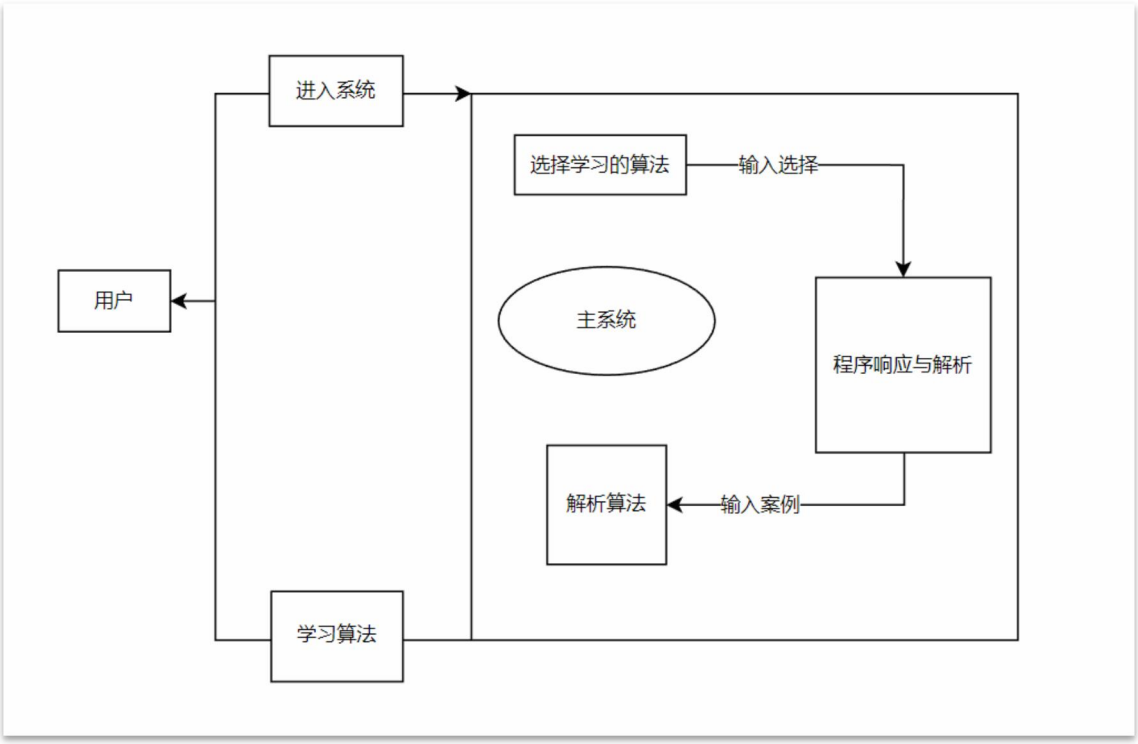

图 6-2 系统操作逻辑图

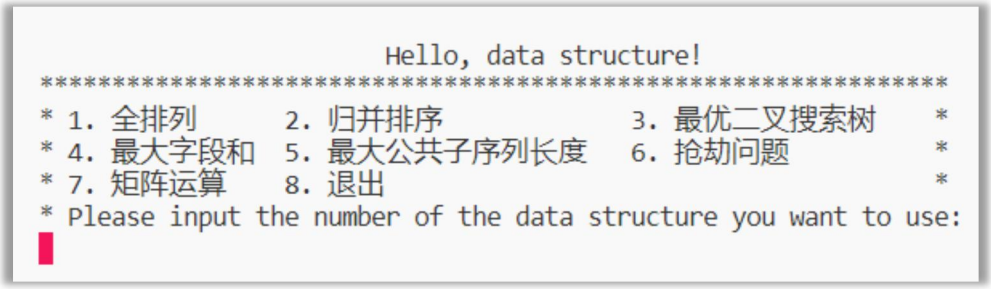

图 6-3 主界面

图 6-4 全排列算法

```
算法简介
1. 申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列
2. 设定两个指针，最初位置分别为两个已经排序序列的起始位置；
3. 比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置；
4. 重复步骤 3 直到某一指针达到序列尾；
5. 将另一序列剩下的所有元素直接复制到合并序列尾。
input the array's length: 4
input the array: 2 3 1 4
归并排序
分割点：1
合并操作
2 3

分割点：3
合并操作
1 4

分割点：2
合并操作
1 2 3 4
```

图 6-5 归并排列

```
请输入顺序表的长度：5
请依次输入该数组的数据：3 2 -4 3 2
首先声明一个d[n]数组用来存储第i个数据加到第rec[i]个数据的和,rec[n]用来记录第i个元素加到第几个元素能得到最大值

初始化d[n-1]就等于最后一个元素的值,rec[n-1]就等于n-1,因为最后一个元素不能再往后加了

从后往前遍历:

当d[4]的值大于零时，d[3]的值就等于当前数据值加上d[4]
同时rec[3]的值就等于rec[4]的值

当d[3]的值大于零时，d[2]的值就等于当前数据值加上d[3]
同时rec[2]的值就等于rec[3]的值

当d[2]的值大于零时，d[1]的值就等于当前数据值加上d[2]
同时rec[1]的值就等于rec[2]的值

当d[1]的值大于零时，d[0]的值就等于当前数据值加上d[1]
同时rec[0]的值就等于rec[1]的值

因为d[i]存储的是从第i个元素开始的最大字段和,所以要求整个序列的最大字段和,只需要遍历d[n]得到最大值就行

最大字段和为：6
最优解为：1,5
```

图 6-6 最大子段和

```
请输入第一个序列的长度: 4
请输入第二个序列的长度: 3
请依次输入第一个序列的内容: 3 2 1 4
请依次输入第二个序列的内容: 3 2 4

首先初始化一个二维数组c[m][n]来存储第一个序列的前i个元素与第二个序列的前j个元素之间最长公共子序列的长度

m是第一个序列的长度,n是第二个序列的长度,i代表行数,j代表列数<<endl因为第一个序列中的第1个元素“3”与y中的第1个
元素“3”相等
所以让c[1][1]的值相对于c[0][0]的值加一

因为第一个序列中的第1个元素“3”与y中的第2个元素“2”不相等
所以让c[1][2]的值取c[0][2]和c[1][1]中更大的那个

因为第一个序列中的第1个元素“3”与y中的第3个元素“4”不相等
所以让c[1][3]的值取c[0][3]和c[1][2]中更大的那个

因为第一个序列中的第2个元素“2”与y中的第1个元素“3”不相等
所以让c[2][1]的值取c[1][1]和c[2][0]中更大的那个

因为第一个序列中的第2个元素“2”与y中的第2个元素“2”相等
所以让c[2][2]的值相对于c[1][1]的值加一

因为第一个序列中的第2个元素“2”与y中的第3个元素“4”不相等
所以让c[2][3]的值取c[1][3]和c[2][2]中更大的那个

因为第一个序列中的第3个元素“1”与y中的第1个元素“3”不相等
所以让c[3][1]的值取c[2][1]和c[3][0]中更大的那个

因为第一个序列中的第3个元素“1”与y中的第2个元素“2”不相等
所以让c[3][2]的值取c[2][2]和c[3][1]中更大的那个

因为第一个序列中的第3个元素“1”与y中的第3个元素“4”不相等
所以让c[3][3]的值取c[2][3]和c[3][2]中更大的那个

因为第一个序列中的第4个元素“4”与y中的第1个元素“3”不相等
所以让c[4][1]的值取c[3][1]和c[4][0]中更大的那个

因为第一个序列中的第4个元素“4”与y中的第2个元素“2”不相等
所以让c[4][2]的值取c[3][2]和c[4][1]中更大的那个

因为第一个序列中的第4个元素“4”与y中的第3个元素“4”相等
所以让c[4][3]的值相对于c[3][2]的值加一

c[i][j]:
       1 1 1
       1 2 2
       1 2 2
       1 2 3

因为c[4][3]代表第一个序列中的前4个元素和第二个序列中的前3个元素的最大字段和
所以长子序列长度是: 3
```

图 6-7 最大公共子序列长度

```
示例 7 [3,2,3,-1,3,-1,1]
输入一课完全二叉树
节点数: 7
输入完全二叉树的值:
目标层的节点个数: 4
当前层: 1, 下一层的节点个数: 2
已累计的目标层节点: 4
目标父亲节点在当前节点左孩子上, 当前节点值为: 3
目标父亲节点在当前节点上, 当前节点值为: 2

寻找-1,1的父节点, i: 6
目标父节点在2层
目标层的节点个数: 4
当前层: 1, 下一层的节点个数: 2
已累计的目标层节点: 4
目标父亲节点在当前节点右孩子上, 当前节点值为: 3
目标父亲节点在当前节点上, 当前节点值为: 3

构造完成

开始偷
偷当前的获得的价值: 3
不偷当前的获得的价值: 0

偷当前的获得的价值: 2
不偷当前的获得的价值: 3

偷当前的获得的价值: 1
不偷当前的获得的价值: 0

偷当前的获得的价值: 3
不偷当前的获得的价值: 1

偷当前的获得的价值: 7
不偷当前的获得的价值: 6

最高金额为: 7
7
```

图 6-8 打家劫舍问题

```
****************************************
*                菜单                  *
*           1.矩阵相加                 *
*           2.矩阵相减                 *
*           3.矩阵相乘                 *
*           4.矩阵数乘                 *
*           5.矩阵求逆                 *
*           6.退出                     *
****************************************
请输入您的选择: 1
请输入矩阵1
输入行数与列数:
3 4
输入你的矩阵:
1 2 3 4
1 2 3 4
1 2 3 4
请输入矩阵2
输入行数与列数:
3 4
输入你的矩阵:
4 3 2 1
4 3 2 1
4 3 2 1
第一个矩阵为:
1   2   3   4
1   2   3   4
1   2   3   4
第二个矩阵为:
4   3   2   1
4   3   2   1
4   3   2   1
矩阵相加运算解法如下:
第一个矩阵与第二个矩阵对应的元素值依次相加
两个矩阵的第1行对应的元素相加的结果分别为
          5         5         5         5
两个矩阵的第2行对应的元素相加的结果分别为
          5         5         5         5
两个矩阵的第3行对应的元素相加的结果分别为
          5         5         5         5

结果为:
5   5   5   5
5   5   5   5
5   5   5   5
```

图 6-9 矩阵相加

```
请输入您的选择: 2
请输入矩阵1
输入行数与列数:
3 4
输入你的矩阵:
4 3 2 1
4 3 2 1
4 3 2 1
请输入矩阵2
输入行数与列数:
3 4
输入你的矩阵:
1 2 2 1
1 2 2 1
1 2 2 1
第一个矩阵为:
4  3  2  1
4  3  2  1
4  3  2  1
第二个矩阵为:
1  2  2  1
1  2  2  1
1  2  2  1
矩阵相减运算解法如下:
第一个矩阵与第二个矩阵对应的元素值依次相减
两个矩阵的第1行对应的元素相减的结果分别为
        3       1       0       0
两个矩阵的第2行对应的元素相减的结果分别为
        3       1       0       0
两个矩阵的第3行对应的元素相减的结果分别为
        3       1       0       0

结果为:
3  1  0  0
3  1  0  0
3  1  0  0
```

图 6-10 矩阵相减

```
请输入您的选择：3
请输入矩阵1
输入行数与列数：
3 4
输入你的矩阵：
1 2 3 4
1 2 3 4
1 2 3 4
请输入矩阵2
输入行数与列数：
4 3
输入你的矩阵：
1 2 3
4 3 1
3 2 1
1 2 3
第一个矩阵为：
1  2  3  4
1  2  3  4
1  2  3  4
第二个矩阵为：
1  2  3
4  3  1
3  2  1
1  2  3
矩阵相乘运算解法如下：
第一个矩阵的第1行乘第二个矩阵的每一列对应的元素，然后求和后按行排列为：
        22      22      20
第一个矩阵的第2行乘第二个矩阵的每一列对应的元素，然后求和后按行排列为：
        22      22      20
第一个矩阵的第3行乘第二个矩阵的每一列对应的元素，然后求和后按行排列为：
        22      22      20
结果为：
22  22  20
22  22  20
22  22  20
```

图 6-11 矩阵相乘

```
请输入您的选择：4
请输入矩阵
输入行数与列数：
2 3
输入你的矩阵：
1 2 3
2 1 3
请输入数值
2
矩阵为：
1   2   3
2   1   3
数值为：
2
数乘运算解法如下：
矩阵的每个值依次乘这个数值
数据2与第1行的数相乘结果分别为
        2       4       6
数据2与第2行的数相乘结果分别为
        4       2       6
结果为：
2   4   6
4   2   6
```

图片 6-12 矩阵数乘

```
请输入您的选择：5
请输入矩阵的行数：2 3
请依次输入第1行的元素：1 3 2
请依次输入第2行的元素：原始矩阵：
   3         1
   3         2
1、求矩阵的逆首先我们要做的是求矩阵的行列式
顺序求和，主对角线元素相乘之和依次是：
         6
主对角线元素相乘之和是6
逆序相减，减去次对角线元素乘积依次是：
         3
主对角线元素相乘之和是3
因此，矩阵的行列式为：6-3=3
去除元素A(0,0)得到的新矩阵为：
   2
现已经求出矩阵的行列式和矩阵的伴随矩阵。
开始根据公式A^(-1)=A*/|A|计算矩阵的逆：
去除元素A(0,1)得到的新矩阵为：
   3
现已经求出矩阵的行列式和矩阵的伴随矩阵。
开始根据公式A^(-1)=A*/|A|计算矩阵的逆：
去除元素A(1,0)得到的新矩阵为：
   1
现已经求出矩阵的行列式和矩阵的伴随矩阵。
开始根据公式A^(-1)=A*/|A|计算矩阵的逆：
去除元素A(1,1)得到的新矩阵为：
   3
现已经求出矩阵的行列式和矩阵的伴随矩阵。
开始根据公式A^(-1)=A*/|A|计算矩阵的逆：
矩阵的逆矩阵为：
0.666667 -0.333333
  -1         1
```

图片 6-13 矩阵求逆

```
0.15 0.1 0.05 0.1 0.2
请输入空隙节点的概率:
0.05 0.1 0.05 0.05 0.05 0.1
开始计算最优二叉树...
初始化 w 和 e 数组:
w:
0       0       0       0       0       0
0.05    0       0       0       0       0
0       0.1     0       0       0       0
0       0       0.05    0       0       0
0       0       0       0.05    0       0
0       0       0       0       0.1     0

e:
0       0       0       0       0       0
0.05    0       0       0       0       0
0       0.1     0       0       0       0
0       0       0.05    0       0       0
0       0       0       0.05    0       0
0       0       0       0       0.1     0
开始填表:
w[1][1]: 0.4
e[1][1]: 0.65

w[2][2]: 0.25
e[2][2]: 0.4

w[3][3]: 0.2
e[3][3]: 0.35

w[4][4]: 0.2
e[4][4]: 0.3

w[5][5]: 0.35
e[5][5]: 0.5

w[1][2]: 0.6
e[1][2]: 1.2

w[2][3]: 0.35
e[2][3]: 0.75

w[3][4]: 0.35
e[3][4]: 0.75

w[4][5]: 0.5
e[4][5]: 0.9

w[1][3]: 0.7
e[1][3]: 1.65
```

图片 6-14 最优二叉搜索树 (a)

```
w[2][4]: 0.5
e[2][4]: 1.2

w[3][5]: 0.65
e[3][5]: 1.5

w[1][4]: 0.85
e[1][4]: 2.25

w[2][5]: 0.8
e[2][5]: 2.05

w[1][5]: 1.15
e[1][5]: 3.25

最优二叉树的根节点为：各子树的根：
1 1 1 1 3
0 2 2 3 4
0 0 3 3 4
0 0 0 4 5
0 0 0 0 5

最优二叉树结构：
最优二叉树的结果为：
k3是根
最优二叉树的结果为：
k1是k3的左孩子
最优二叉树的结果为：
d0是k1的左孩子
最优二叉树的结果为：
k2是k1的右孩子
最优二叉树的结果为：
d1是k2的左孩子
最优二叉树的结果为：
d2是k2的右孩子
最优二叉树的结果为：
k5是k3的右孩子
最优二叉树的结果为：
k4是k5的左孩子
最优二叉树的结果为：
d3是k4的左孩子
最优二叉树的结果为：
d4是k4的右孩子
最优二叉树的结果为：
d5是k5的右孩子
```

图片 6-14 最优二叉搜索树 (b)

7 总结

通过本次课程设计的总结如下：

(1) 没有精美的前端展示页面，没有各种精美富有细节的动效，给用户的反馈和感知不强，整个程序设计比较死板和生硬，不具备面向大众性，同时，我们对于每种算法的解析不够彻底，不能反映更加微小的细节。

(2) 本次课程设计，经过一段时间的实践以及和小组成员的磨合，使得我们对 C++ 语言有了更进一步的认识和了解，通过团队的协作开发，使得我们认识到，完成一个项目，不仅仅是需要个人的技术和能力，还需要和团队之间磨合交流，才能实现所设想的需求，同时在程序编写的时候，我们也提升了很多自身的代码能力，对于复杂的数据处理更加的得心应手。

参考文献：

1. 严蔚敏，吴伟民，《数据结构题集》（C 语言版）， 北京：清华大学出版社，1999.

2. 范策等，《算法与数据结构》，北京：机械工业出版社，2004.

3. 许绪松等，《数据结构与算法导论》，北京：电子工业出版社，2001.

4. Stephen Prata 等，《C++ Prime Plus (第六版)》，北京：人民邮电出版社，2012.

5. Gilbert Strang 等，《线性代数》，北京：清华大学出版社，2019.