|  |
| --- |
| 《算法分析与设计》  **课 程 设 计 报 告**    学院（系）： 软件工程系  班 级： 116030802  学生姓名： 代利文 学号 11603080209  指导教师： 刘祥    **时间： 从2018年12月17日 到 2018年 12月27日** |

目录

[题目一 二项式公式计算 1](#_Toc534142880)

[1. 问题描述 1](#_Toc534142881)

[2. 题所用的算法设计方法及基本思路 1](#_Toc534142882)

[3. 数据结构描述 1](#_Toc534142883)

[4. 算法描述 1](#_Toc534142884)

[5. 算法的时间空间复杂度分析 2](#_Toc534142885)

[6. 算法实例 2](#_Toc534142886)

[题目二 简单分形树 3](#_Toc534142887)

[1. 问题描述 3](#_Toc534142888)

[2. 解决问题所用的算法设计方法及基本思路 3](#_Toc534142889)

[3. 采用的数据结构描述 3](#_Toc534142890)

[4. 算法描述 4](#_Toc534142891)

[5. 算法的时间空间复杂度分析 4](#_Toc534142892)

[6. 算法实例 4](#_Toc534142893)

[题目三 三水杯倒水问题 5](#_Toc534142894)

[1. 问题描述 5](#_Toc534142895)

[2. 解决问题所用的算法设计方法及基本思路 5](#_Toc534142896)

[3. 采用的数据结构描述 5](#_Toc534142897)

[4. 算法描述 5](#_Toc534142898)

[5. 算法的时间空间复杂度分析 6](#_Toc534142899)

[6. 算法实例 6](#_Toc534142900)

[题目四 九宫格问题 8](#_Toc534142901)

[1. 问题描述 8](#_Toc534142902)

[2. 解决问题所用的算法设计方法及基本思路 8](#_Toc534142903)

[3. 采用的数据结构描述 8](#_Toc534142904)

[4. 算法描述 8](#_Toc534142905)

[5. 算法的时间空间复杂度分析 9](#_Toc534142906)

[6. 算法实例 9](#_Toc534142907)

[题目五 租船问题 11](#_Toc534142908)

[1. 问题描述 11](#_Toc534142909)

[2. 解决问题所用的算法设计方法及基本思路 11](#_Toc534142910)

[3. 采用的数据结构描述 11](#_Toc534142911)

[4. 算法描述 11](#_Toc534142912)

[5. 算法的时间空间复杂度分析 12](#_Toc534142913)

[6. 算法实例 12](#_Toc534142914)

[题目六 基因序列比较 14](#_Toc534142915)

[1. 问题描述 14](#_Toc534142916)

[2. 解决问题所用的算法设计方法及基本思路 14](#_Toc534142917)

[3. 采用的数据结构描述 15](#_Toc534142918)

[4. 算法描述 15](#_Toc534142919)

[5. 算法的时间空间复杂度分析 15](#_Toc534142920)

[6. 算法实例 15](#_Toc534142921)

[题目七 数独问题 16](#_Toc534142922)

[1. 问题描述 16](#_Toc534142923)

[2. 解决问题所用的算法设计方法及基本思路 17](#_Toc534142924)

[3. 采用的数据结构描述 17](#_Toc534142925)

[4. 算法描述 17](#_Toc534142926)

[5. 算法的时间空间复杂度分析 18](#_Toc534142927)

[6. 算法实例 18](#_Toc534142928)

[题目八 动态演示快速排序、归并排序 20](#_Toc534142929)

[1. 问题描述 20](#_Toc534142930)

[2. 解决问题所用的算法设计方法及基本思路 20](#_Toc534142931)

[3. 采用的数据结构描述 20](#_Toc534142932)

[4. 算法描述 20](#_Toc534142933)

[5. 算法的时间空间复杂度分析 21](#_Toc534142934)

[6. 算法实例 22](#_Toc534142935)

[课程设计总结 27](#_Toc534142936)

# 题目一 二项式公式计算

## 问题描述

完成二项式公式计算，即C:\Users\Administrator\AppData\Roaming\Tencent\Users\450326779\QQ\WinTemp\RichOle\P5CD`$%X_[)7D$AC~(Z9OMV.png. 公式解释：为了从n个不同元素中抓取k个元素(C:\Users\Administrator\AppData\Roaming\Tencent\Users\450326779\QQ\WinTemp\RichOle\EAFZBI{OS%{}SV`TAMJ00%B.png)，可以这样考虑，如果第一个元素一定在结果中，那么就需要从剩下的n-1个元素中抓取k-1个元素(C:\Users\Administrator\AppData\Roaming\Tencent\Users\450326779\QQ\WinTemp\RichOle\S)P[T0{WJ7G$B0]IRLA8`DL.png)；如果第一个元素不在结果中，就需要从剩下的n-1个元素中抓取k个元素(C:\Users\Administrator\AppData\Roaming\Tencent\Users\450326779\QQ\WinTemp\RichOle\[Y1A4WS(93V350MA2(VQDWJ.png)。

要求分别采用以下方法计算，并进行三种方法所需时间的经验分析。

a. 直接采用递归算法

b. 采用备忘录方法

c. 采用迭代算法

要有性能数据。最好是C:\Users\Administrator\AppData\Roaming\Tencent\Users\450326779\QQ\WinTemp\RichOle\0G]W(0LSGKF]0YYYWL4A((E.png的曲线或者曲面图。迭代算法要和递归思路相关。另外，Excel其实可以帮你你完成函数拟合（回归），就是帮你猜时间复杂度。

## 题所用的算法设计方法及基本思路

利用3个方式来实现。但是3种方式的总体思路是一样的。都是分为两种情况，一种是取元素时抓到了第一个元素，那么就在N-1个中进行K-1的迭代或者是递归。第二种是抓去元素时未抓到第一个那么就在N-1个中进行K的迭代和递归。

## 数据结构描述

未使用数组结构

## 算法描述

算法名称：Binonial

输入：正整数k，正整数n

输出：二项式公式计算结果

算法实现细节

r = int[n + 1][k + 1];

for i <- 1 to n do

for j <- 0 to k do

if i=j || j=0

r[i][j] = 0;

else

r[i][j] = r[i - 1][j - 1] + r[i- 1][j];

return r[n][k];

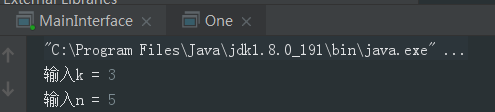
## 算法的时间空间复杂度分析

时间复杂度：O(nlogn)

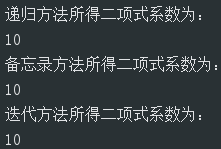
空间复杂度：O(n²)

## 算法实例

输入的数据

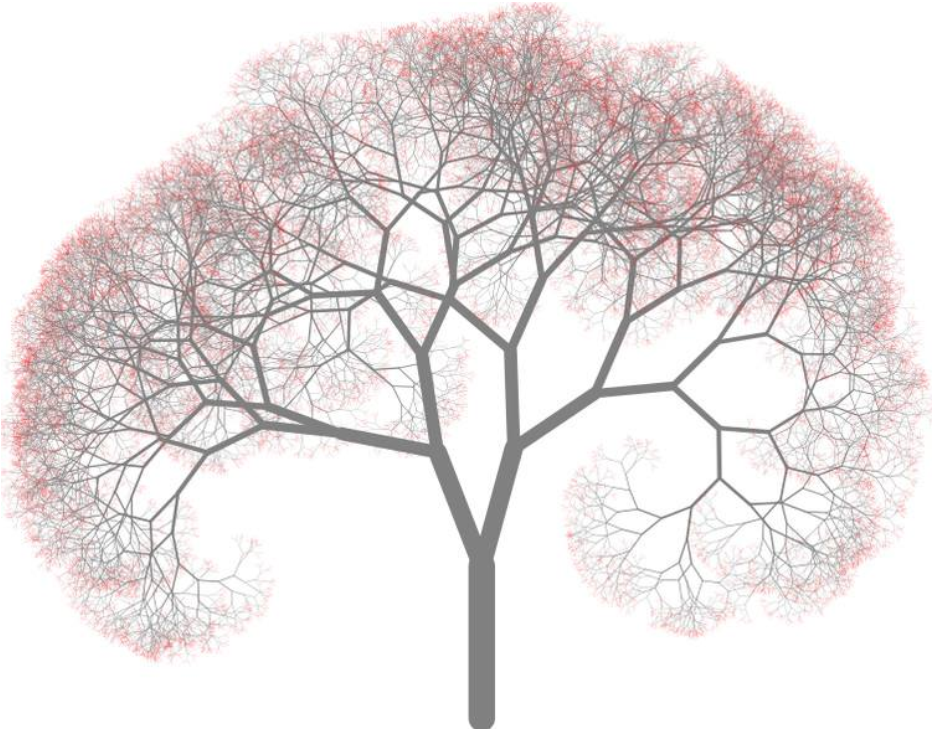


输出的数据



# 题目二 简单分形树

## 问题描述



先垂直绘制一根线段，然后在线段顶端向右一定倾角绘制一根线段，长度分别为原线段的k倍. 再同样的，在线段左侧以固定倾角绘制一根线段，如此反复，直至线段长度小于某个较小的值。其中，线条颜色以及长度，夹角（例如产生某个范围的随机数）都可以自行进行微调。

## 解决问题所用的算法设计方法及基本思路

这题就是用一直递归方式，向下输出分支，设计一个合理的出口形成一个分支树。左右支分别分别就是一次递归。那么两次递归所放的位置在输出前还是输出后，都可以完成分形树的标准。但是所放位置的不同，就有点类似于先序遍历，中序遍历和后序遍历一样。

而最后界面实现利用的javaFX实现界面的输出

## 采用的数据结构描述

未使用数据结构

## 算法描述

算法名称：Branch

输入：正整数n 出口整数

输出：递归输出分形树

算法实现细节

if n < 10

return

Branch(n)

Branch(n)

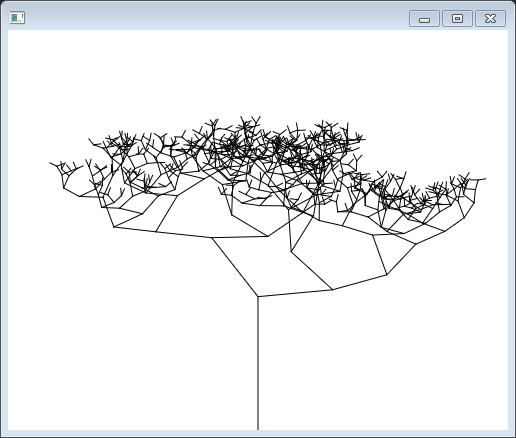
## 算法的时间空间复杂度分析

时间复杂度：O(nlogn)

空间复杂度:O(nlogn)

## 算法实例

数据的输出



# 题目三 三水杯倒水问题

## 问题描述

西蒙.丹尼斯.泊松是著名的法国数学家和物理学家。据说在他遇到某个古老的谜题之后，就开始对数学感兴趣了，这个谜题是这样的：给定一个装满水的8品脱壶以及两个容量分别为5品脱和3品脱的空壶，如何通过完全灌满或者到空这些壶从而使得某个壶精确地装有4品脱的水？用广度优先查找来求解这个谜题。要求在输出结果中包含广度优先的遍历过程（结点的遍历顺序）。

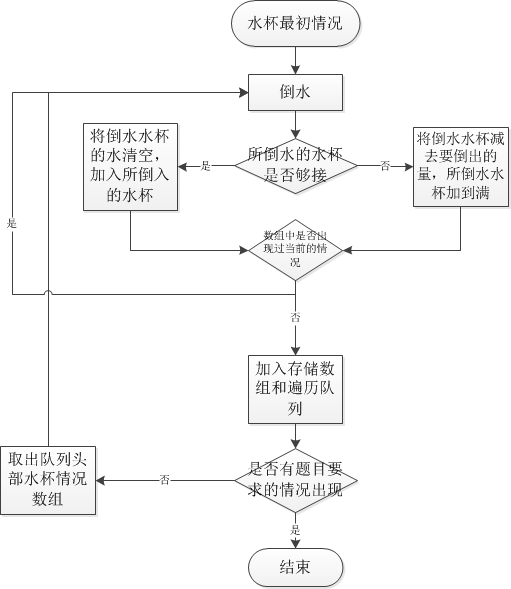
## 解决问题所用的算法设计方法及基本思路

本题根据提示使用广度优先遍历，对每次倒水的所有方式先拿出来，存储到队列当中。对队列中的倒水后的情况取出，又得到所有的当前情况下的所有倒水操作，进行存储。最后得到题目要求的数，立即终止当前树的遍历回到之前的节点。

## 采用的数据结构描述

利用队列先进先出的特征，进行广度优先遍历。

## 算法描述



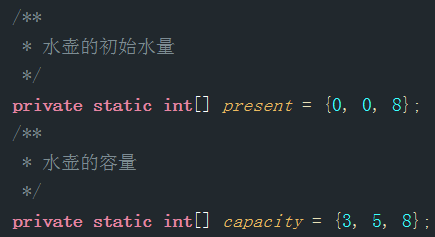
## 算法的时间空间复杂度分析

时间复杂度：O(n²)

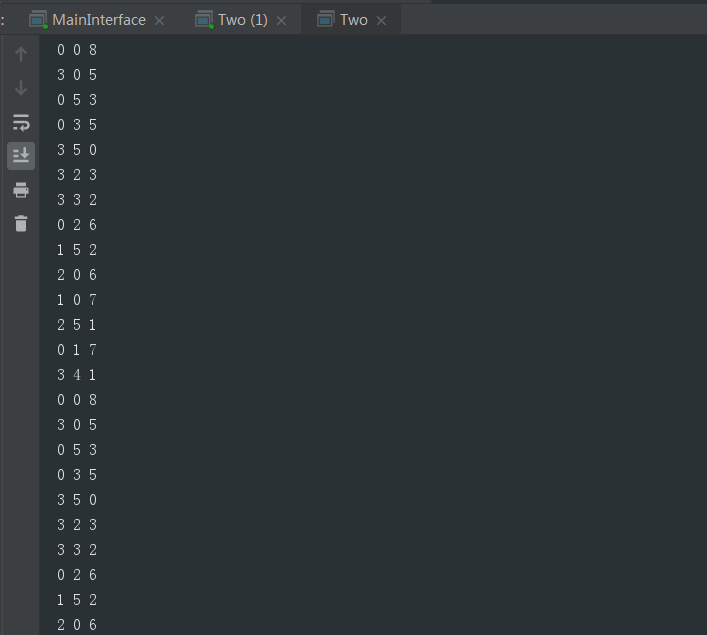
空间复杂度：O(n²)

## 算法实例

数据的输入



数据的输出

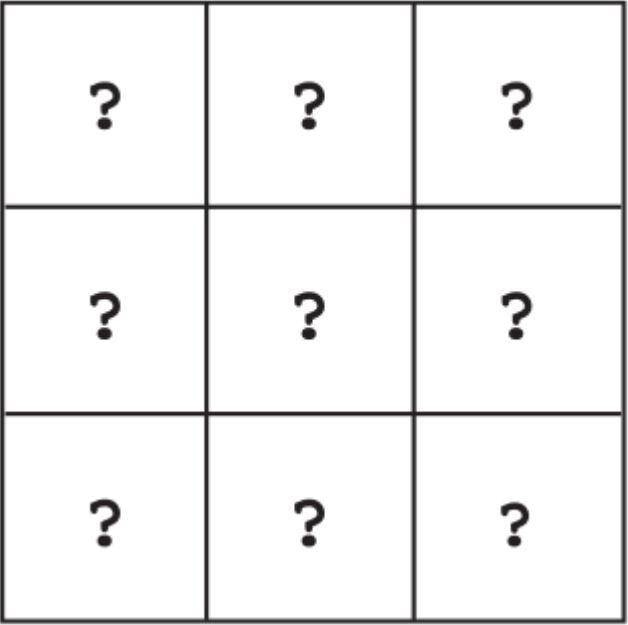


…...（情况未完成截取）

# 题目四 九宫格问题

## 问题描述

画出所有可能的九宫图, 即每行每列，每一斜行的和都相同



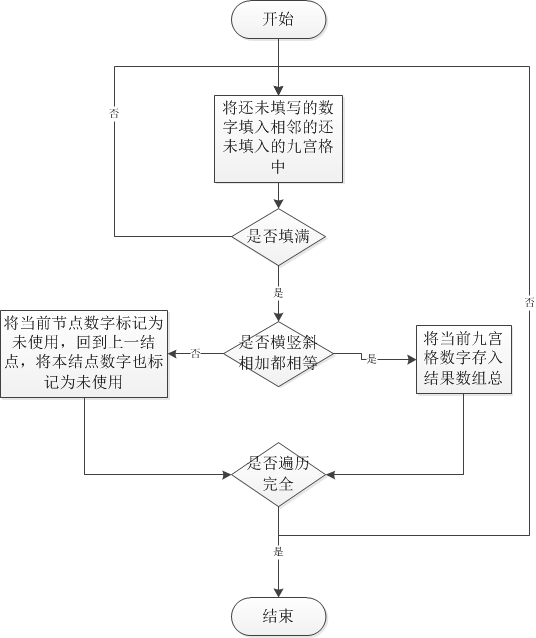
## 解决问题所用的算法设计方法及基本思路

本题使用深度优先遍历，从九宫格从未填开始，依次往下遍历。先填满，看是否符合标准。若符合保存到结果数组中。若不符合那么回到上一结点，更改上一结点的情况，继续向下遍历，直至填满。一直循环，直至将所有情况拿出来，进行核对。

## 采用的数据结构描述

未使用数据结构。

## 算法描述



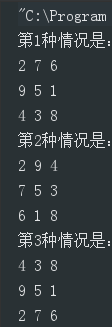
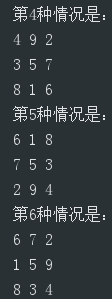
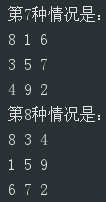
## 算法的时间空间复杂度分析

时间复杂度：O(n²)

空间复杂度：O(n)

## 算法实例

数据的输出

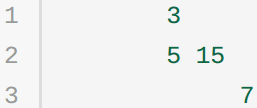
  

# 题目五 租船问题

## 问题描述

长江游艇俱乐部在长江上设置了n个游艇出租站1,2,...,n。游客可在这些游艇出租站租用游艇，并在下游的任何一个游艇出租站归还游艇。游艇出租站i到游艇出租站j之间的租金为r(i,j)，C:\Users\Administrator\AppData\Roaming\Tencent\Users\450326779\QQ\WinTemp\RichOle\1~WPDFVZ$P_NAV0U156SU8H.png。试着设计一个算法，计算处从游艇出租站1到游艇出租站n所需要的最少租金。输入：n和相应规模的r(i,j) 。注意j>i，所以r(i,j)只有n-1行。

例如:



表示n=3,

r(1,2) = 5,

r(1,3) = 15,

r(2,3) = 7

输出：最少租金。

## 解决问题所用的算法设计方法及基本思路

本题的问题可以划分为上一站的最少价值，加上当前的直达价值。这样就符合动态规划的思想。对问题的划分为子问题。而每一个子问题又对当前的问题可以进行影响或是比较。最后确立当前这个问题的最优选择。

## 采用的数据结构描述

未使用数据结构。

## 算法描述

算法名称：Charter

输入：数组a[x][x] 租船每个节点的价格

输出：从出发到最后最便宜的价格

算法实现细节

temp = int[x];

for i <- 0 to x – 1 do

minNum = a[0][i];

for j <- 0 to I do

minNum = min(minNum, temp[j] + a[j][i]);

temp[i] = minNum;

return temp[x - 1];

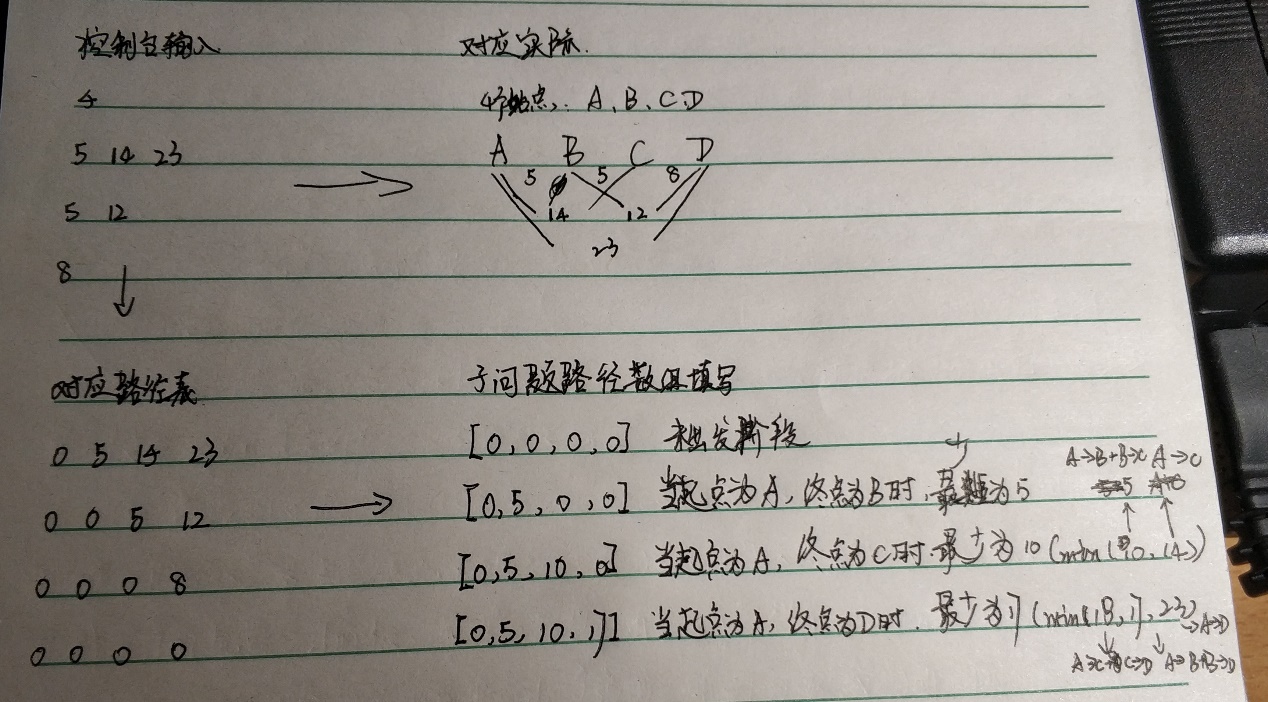
## 算法的时间空间复杂度分析

时间复杂度：O(n²)

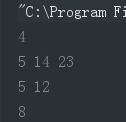
空间复杂度：O(n²)

## 算法实例

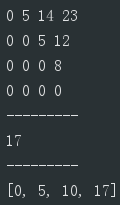
动态规划填表



数据的输入



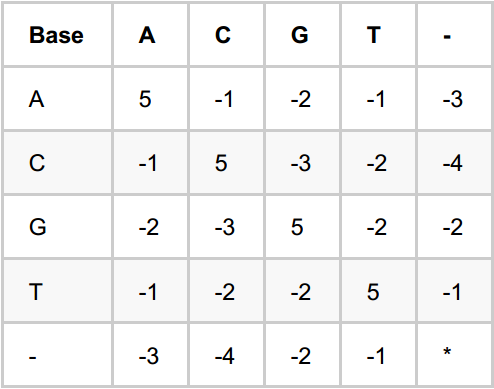
数据的输出



# 题目六 基因序列比较

## 问题描述

人类基因由4种核苷酸，分别用字母ACTG表示。要求编写一个程序，按以下规划比较两个基因序列并确定它们的相似程度。即两给出两个基因序列AGTGATG和GTTAG，它们有多相似呢？测量两个基因的相似度一种方法称为对齐。使用对齐方法可以在基因的适当位置加入空格，让两个基因的长度相等，然后根据基因的分值矩阵计算分数。



比较AGTGATG与GTTAG:

第一种对齐方案为：

首先可以给AGTATG插入一个空格得：AGTAT-G  
GTTAG插入3个空格即得：-GT-TAG

上面的匹配分值为:-3+5+5+(-2)+(-3)+5+(-3)+5=9.  
第二种对齐方案为：

AGTGATG

-GTTA-G

得到的分值为：(-3)+5+5+(-2)+5+(-1)+5=14.

当然还有其它对齐方式，但以上对齐方式是最优的，所以两个基因的相似度就为14。

## 解决问题所用的算法设计方法及基本思路

同样根据题意我们可以将两个基因分为X序列和Y序列。那么我们根据题意的划分基因匹配时无非就是3种情况：第一种XY，既是X序列和Y序列中字符的对应。第二种X-，既是X序列中的字符和空位所对应。第三种-Y，既是空位和Y序列中的字符所对应。

那么当我们将上述的情况就可以从该序列的第一个开始分起。即将一个问题划分了子问题。将从最初的匹配三种情况的最优填入表中，为后面的子问题进行铺垫。最后得到最终问题的答案。

## 采用的数据结构描述

使用Map数据结构。用来获取基因和他对应的数组位置。

## 算法描述

算法名称：geneCompare

输入：字符数组gene1[x] 基因序列1 字符数组gene2[y] 基因序列2

匹配得分数组 sort[5][5]

输出：最大的匹配值

算法实现细节

int[][] temp[x + 1][y + 1]

for i <- 1 to x do

temp[i][0] = temp[i - 1][0] + sort[gene1[i - 1]][4]

for i <- 1 to y do

temp[i][0] = temp[0][i -1] + sort[gene2[i - 1]][4]

for i <- 1 to x do

for j <- 1 to y do

temp[i][j] = max(

temp[i - 1][j - 1] + sort[gene1[i - 1]][ gene2[y - 1]],

temp[i][j - 1] + sort[4][ gene2[y - 1]],

temp[i - 1][j] + sort[gene1[i - 1]][4]

);

return temp[x][y];

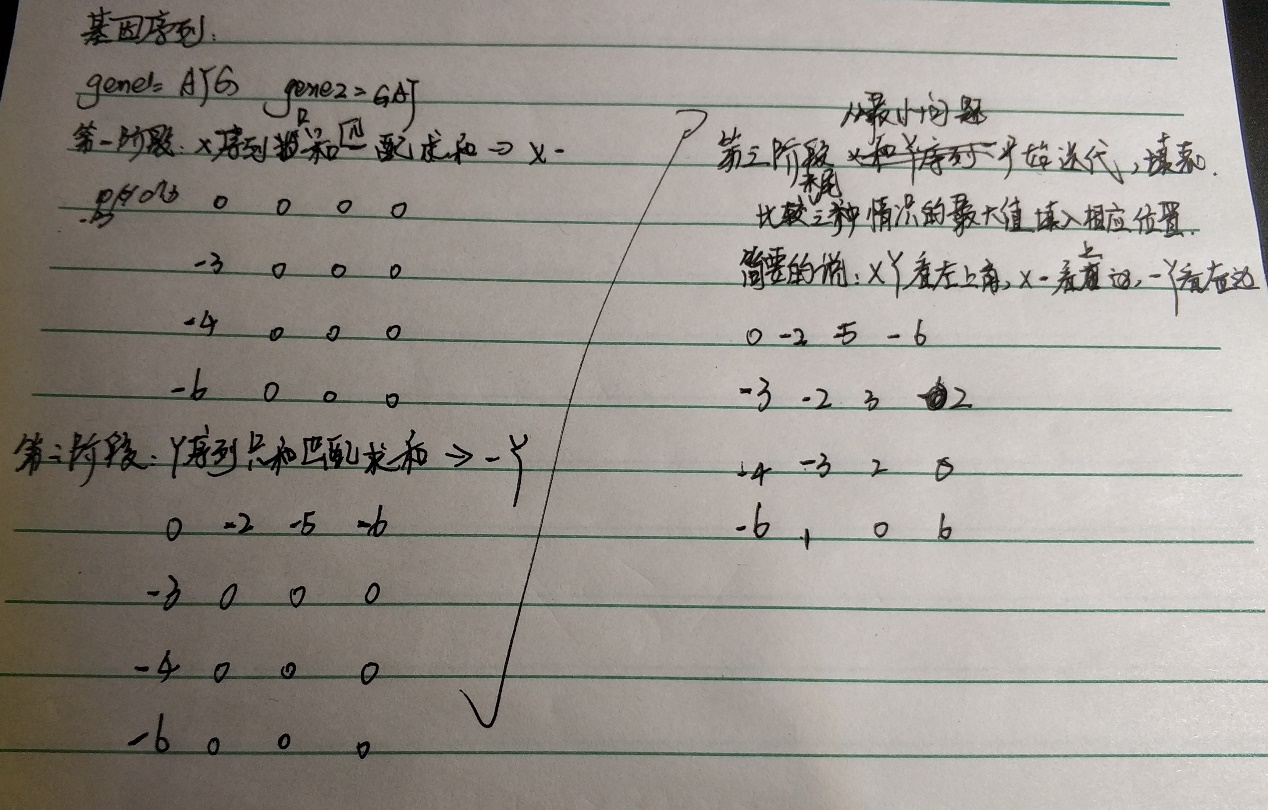
## 算法的时间空间复杂度分析

时间复杂度：O(n²)

空间复杂度：O(n²)

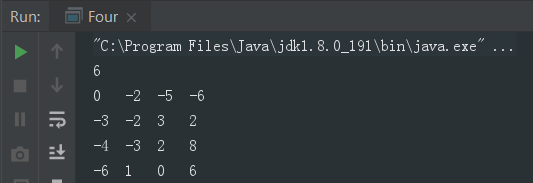
## 算法实例

动态规划填表

数据的输入

C:\Users\Administrator\AppData\Roaming\Tencent\Users\450326779\QQ\WinTemp\RichOle\9PTHW61SXC}ZX`0G~P2J}HM.png

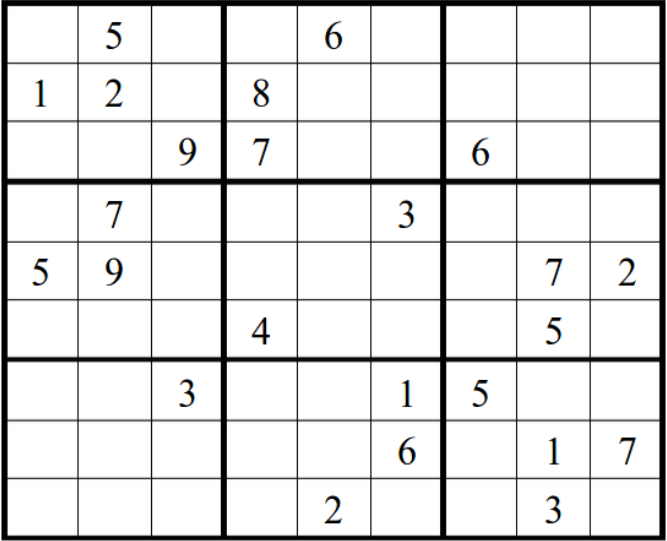
数据的输出



# 题目七 数独问题

## 问题描述

建议大家下一个数独的手机app了解规则和培养感觉。  
这里要求大家利用计算机完成数组游戏的填写。  
输入：一份填有部分数字的数独表格  
输出：完成后的表格（或者宣告无解）



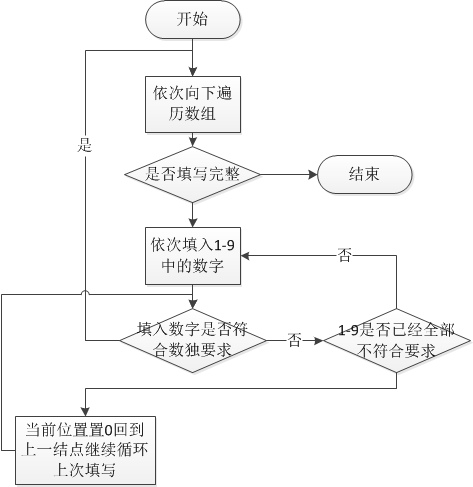
## 解决问题所用的算法设计方法及基本思路

本题的问题和之前的九宫格问题有所不同。九宫格是先填完再检查而数独问题则是利用检查当前节点是否符合数独要求。若符合则继续填下一个点。若不符合就当前点设为0，即未填写。然后回溯到之前来进行重新填写。当数组全部填写完整那么就代表完成了当前的算法。

## 采用的数据结构描述

未使用数据结构。

## 算法描述



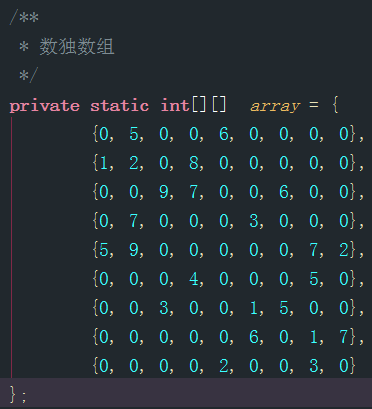
## 算法的时间空间复杂度分析

时间复杂度：O(n²)

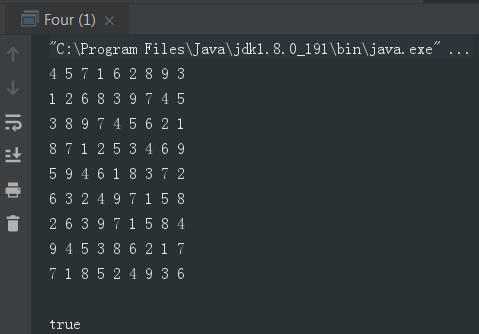
空间复杂度：O(n²)

## 算法实例

数据的输入



数据的输出：



# 题目八 动态演示快速排序、归并排序

## 问题描述

设计一个演示系统动态演示快速排序、归并排序的执行过程。

## 解决问题所用的算法设计方法及基本思路

本题利用JAVAFX将快速排序和归并排序开始直至完成前的每一步进行打印。对于快速排序中关键的数进行颜色的标注。对应归并排序中分组进行下划线的指明。因为时间和能力的有限。未实现动画版。而是静态的打印，没有动画。

快速排序：采用分治法（Divide and Conquer）的一个非常典型的应用。每次交换是跳跃式的。每次排序的时候设置一个基准点，将小于等于基准点的数全部放到基准点的左边，将大于等于基准点的数全部放到基准点的右边。这样在每次交换的时候就不会像冒泡排序一样每次只能在相邻的数之间进行交换，交换的距离就大的多了。因此总的比较和交换次数就少了，速度自然就提高了。当然在最坏的情况下，仍可能是相邻的两个数进行了交换。

归并排序：也是采用分治法（Divide and Conquer）的一个非常典型的应用。将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。若将两个有序表合并成一个有序表，称为二路[归并](https://baike.baidu.com/item/%E5%BD%92%E5%B9%B6/253741)。

## 采用的数据结构描述

未使用数据结构。

## 算法描述

算法名称：quickSort

输入：数组A 排序数组 lo当前数组左下标 hi当前数组右下标

输出：排序完成数组

算法实现细节

quicksort(A, lo, hi)

if lo < hi

p = partition(A, lo, hi)

quicksort(A, lo, p - 1)

quicksort(A, p + 1, hi)

partition(A, lo, hi)

pivot = A[hi]

i = lo

for j = lo to hi - 1

if A[j] <= pivot

swap A[i] with A[j]

i = i + 1

swap A[i] with A[hi]

return i

算法名称：mergeSort

输入：数组A 排序数组 p当前数组左下标 r当前数组右下标

输出：排序完成数组

算法实现细节

merge(A,p,q,r):

n1 <- q-p+1

n2 <- r-q

create array L[0,n1] and R[0,n2]

for i <- 0 to n1-1

do L[i] <- A[p+i]

for j <- 0 to n2-1

do R[j] <- A[q+j+1]

L[n1] <- +∞

R[n2] <- +∞

i <- 0

j <- 0

for k i <- p to r

do if L[i] <= R[j]

then A[k] <- L[i]

i <- i+1

else A[k] <- R[j]

j <- j+1

mergeSort(A,p,r)：

if p<r

then q <- [(p+r)/2]

mergeSort(A,p,q)

mergeSort(A,q+1,r)

merge(A,p,q,r)

## 算法的时间空间复杂度分析

时间复杂度：O(NlogN)

空间复杂度：O(N)

## 算法实例

快速排序

1.选取一个关键字(key)作为枢轴，一般取整组记录的第一个数或者最后一个，这里采用选取序列最后一个数为枢轴。

2.设置两个变量left = 0;right = N - 1;

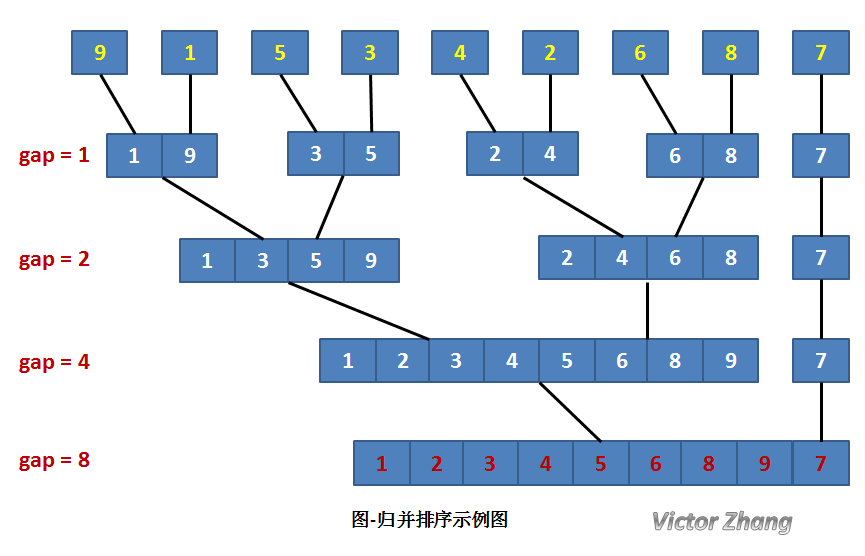
3.从left一直向后走，直到找到一个大于key的值，right从后至前，直至找到一个小于key的值，然后交换这两个数。

4.重复第三步，一直往后找，直到left和right相遇，这时将key放置left的位置即可。



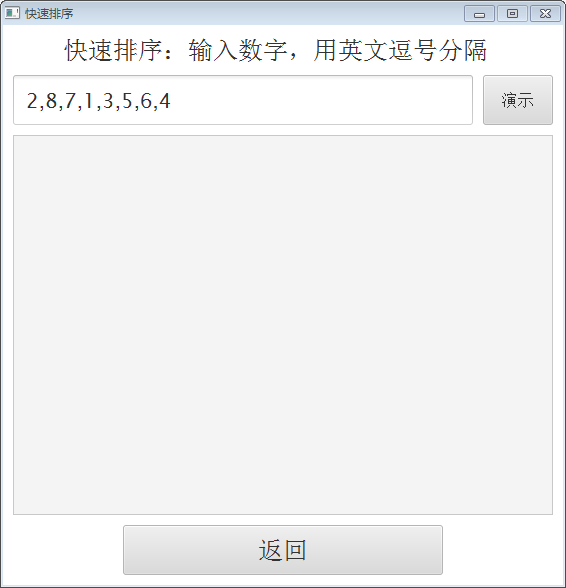
归并排序

利用递归和分而治之的技术 将数据序列划分成为越来越小的半子表，再对半子表排序，最后再用递归步骤将排好序的半子表合并成为越来越大的有序序列。



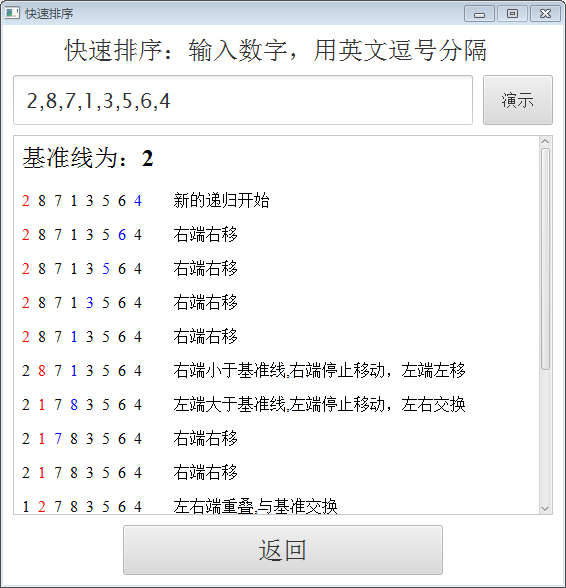
数据的输入

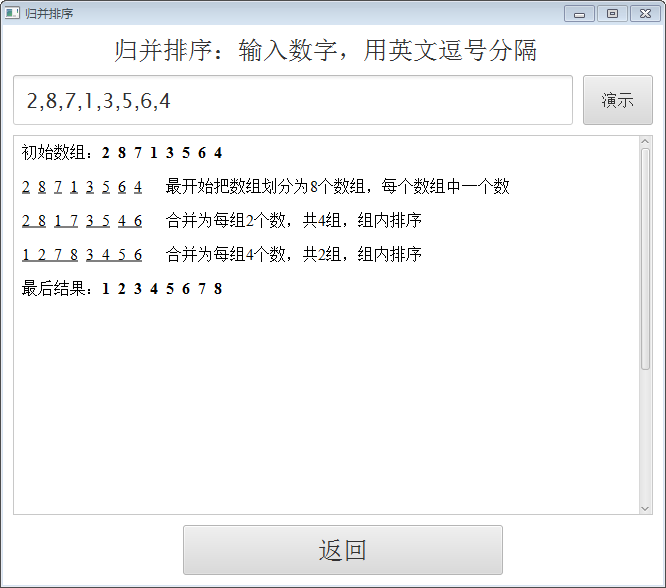






数据的输出





# 课程设计总结

这次课程设计的是对前面的学习过程的一个总结。因为我在前面的课程学习中并不是很好，所以这次课程设计总的来说有很多的东西是比较模糊的。再加上这次课程设计的时间并不长。最后能够完成8道题感觉还是学到了很多东西。补全了很多不太懂的东西。对前面的东西有了一个系统的认识。对动态规划和分支界限都有了比较清晰的认识。