**南京航空航天大学**

**计算机科学与技术/人工智能学院**

**《数据结构》课程设计报告**

**班 级： 1619303**

**专 业： 软件工程**

**学 号： 061920125**

**姓 名： 郑伟林**

**指导教师： 高航**

**目录**

[一、系统进程统计 4](#_Toc61029942)

[1. 数据结构 4](#_Toc61029943)

[2. 算法思想 4](#_Toc61029944)

[3. 源代码 5](#_Toc61029945)

[4. 测试数据和结果 19](#_Toc61029946)

[5. 小结 20](#_Toc61029947)

[二、迷宫问题 21](#_Toc61029948)

[1. 数据结构 21](#_Toc61029949)

[2. 算法思想 21](#_Toc61029950)

[3. 源代码 21](#_Toc61029951)

[4. 测试数据和结果 26](#_Toc61029952)

[5. 小结 27](#_Toc61029953)

[三、家谱管理系统 27](#_Toc61029954)

[1. 数据结构 27](#_Toc61029955)

[2. 算法思想 28](#_Toc61029956)

[3. 源代码 28](#_Toc61029957)

[4. 测试数据和结果 50](#_Toc61029958)

[5. 小结 52](#_Toc61029959)

[四、Huffman编码与解码 52](#_Toc61029960)

[1. 数据结构 52](#_Toc61029961)

[2. 算法思想 53](#_Toc61029962)

[3. 源代码 53](#_Toc61029963)

[4. 测试数据和结果 60](#_Toc61029964)

[5. 小结 62](#_Toc61029965)

[五、地铁修建 62](#_Toc61029966)

[1. 数据结构 62](#_Toc61029967)

[2. 算法思想 62](#_Toc61029968)

[3. 源代码 63](#_Toc61029969)

[4. 测试数据和结果 64](#_Toc61029970)

[5. 小结 65](#_Toc61029971)

[六、公交线路提示 65](#_Toc61029972)

[1. 数据结构 65](#_Toc61029973)

[2. 算法思想 65](#_Toc61029974)

[3. 源代码 66](#_Toc61029975)

[4. 测试数据和结果 75](#_Toc61029976)

[5. 小结 75](#_Toc61029977)

[七、B-树的应用 75](#_Toc61029978)

[1. 数据结构 75](#_Toc61029979)

[2. 算法思想 76](#_Toc61029980)

[3. 源代码 77](#_Toc61029981)

[4. 测试数据和结果 87](#_Toc61029982)

[5. 小结 88](#_Toc61029983)

[八、排序算法速率比较 89](#_Toc61029984)

[1. 数据结构 89](#_Toc61029985)

[2. 算法思想 89](#_Toc61029986)

[3. 源代码 90](#_Toc61029987)

[4. 测试数据和结果 99](#_Toc61029988)

[5. 小结 100](#_Toc61029989)

[九、数字排序【1】 100](#_Toc61029990)

[1. 数据结构 100](#_Toc61029991)

[2. 算法思想 100](#_Toc61029992)

[3. 源代码 101](#_Toc61029993)

[4. 测试数据和结果 102](#_Toc61029994)

[5. 小结 102](#_Toc61029995)

[十、魔法优惠券【1】 102](#_Toc61029996)

[1. 数据结构 102](#_Toc61029997)

[2. 算法思想 102](#_Toc61029998)

[3. 源代码 103](#_Toc61029999)

[4. 测试数据和结果 105](#_Toc61030000)

[5. 小结 105](#_Toc61030001)

[十一、消除类游戏【2】 105](#_Toc61030002)

[1. 数据结构 105](#_Toc61030003)

[2. 算法思想 105](#_Toc61030004)

[3. 源代码 105](#_Toc61030005)

[4. 测试数据和结果 107](#_Toc61030006)

[5. 小结 108](#_Toc61030007)

[十二、最小二叉树【3】 108](#_Toc61030008)

[1. 数据结构 108](#_Toc61030009)

[2. 算法思想 108](#_Toc61030010)

[3. 源代码 109](#_Toc61030011)

[4. 测试数据和结果 113](#_Toc61030012)

[5. 小结 113](#_Toc61030013)

[十三、课程设计总结 114](#_Toc61030014)

# 系统进程统计

## 数据结构

实现进程存储的包括两个结构：单链表和双链表。其中单链表CurrentProcessList类用以存储当前活动的进程，其包含链表头指针L，而单链表结点由LNode结构体组成，一个LNode中包含结点的数据ProInfo类型的data和指向下一个结点的指针next；双链表EndedProcessList类用以存储已结束的进程，其包含头指针L和尾指针tail，双链表结点由DNode结构体组成，其包含结点数据ProInfo类型的data，指向下一个结点的指针next和指向上一个结点的指针pre。

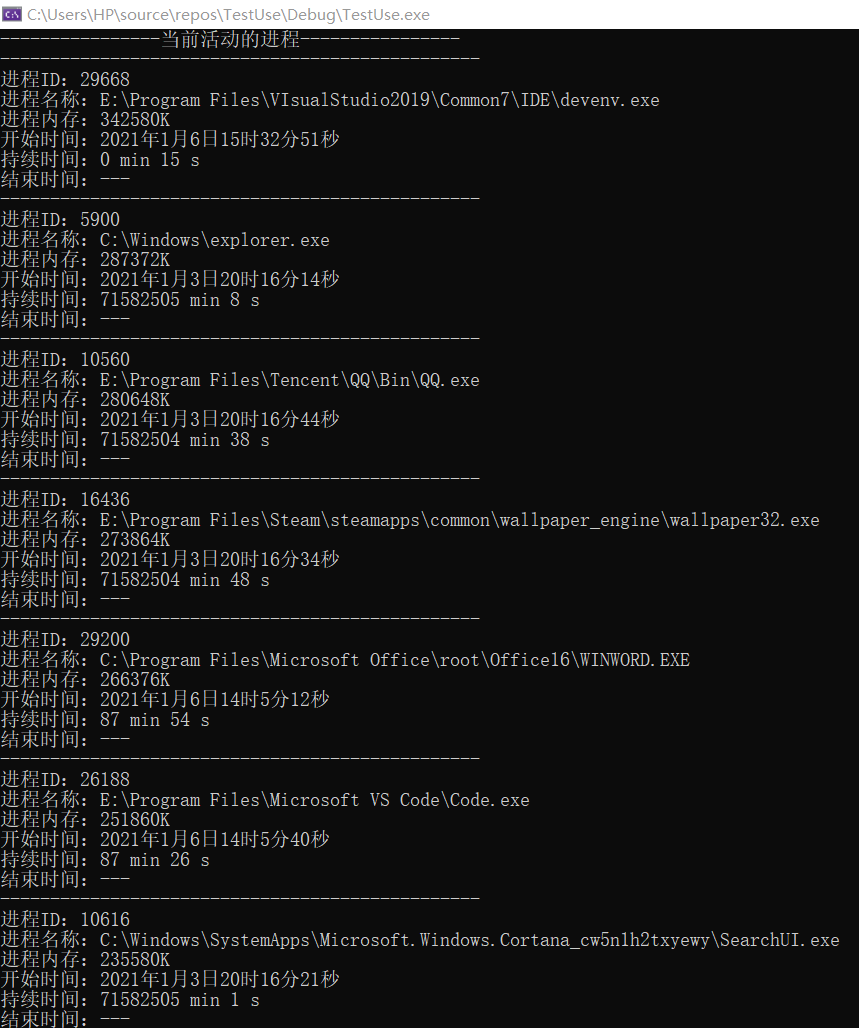
头文件Process\_ZX.h中的结构主要是表示进程信息的ProcessInfo结构体，其包含进程ID DWORD类型的ID变量，进程名称TCHAR类型的Name，进程占用内存Memory，进程开始时间BeginTime，进程持续时间DuraTime和进程结束时间EndTime。

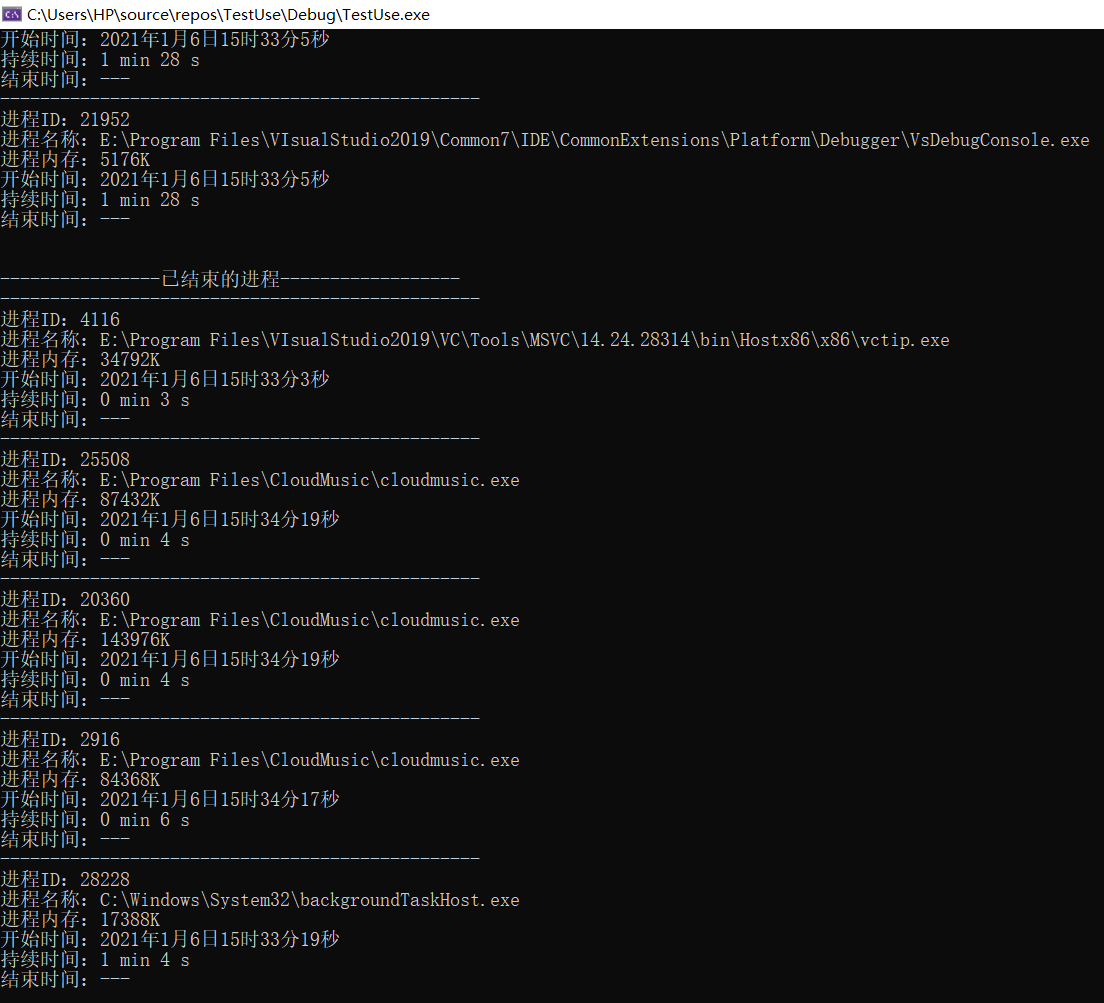
## 算法思想

在主程序中，先利用头文件Process\_ZX.h中的GetCurrentProcesses函数获取当前系统的活动程序，在调用FreshList函数用当前获取的信息L更新活动进程列表CurList和结束进程列表DelList。

FreshList函数的主要思想为：如果新的进程列表中有进程位于结束进程列表D中，则将其从D中删去；在从原活动进程列表C中找到不在新列表L中的进程，将其放入D中，以达到更新D列表，最后将L覆盖C即可。

## 测试数据和结果





## 小结

链表操作中，排序操作时间复杂度为O（n²），其它均为O（n）。刷新函数FreshList的时间复杂度为O（n²）。

优化方面可以在双链表的排序上，可以选用更快的排序算法。

# 迷宫问题

## 数据结构

地图存储采用二维数组map[N][N]记录，1为不可走，0为可走。求解过程运用栈结构来实现非递归深搜，栈元素类型为Point结构体，其包含某个点的坐标i与j。栈类中包含栈底指针pBase和栈顶指针pTop以及栈分配空间stacksize。

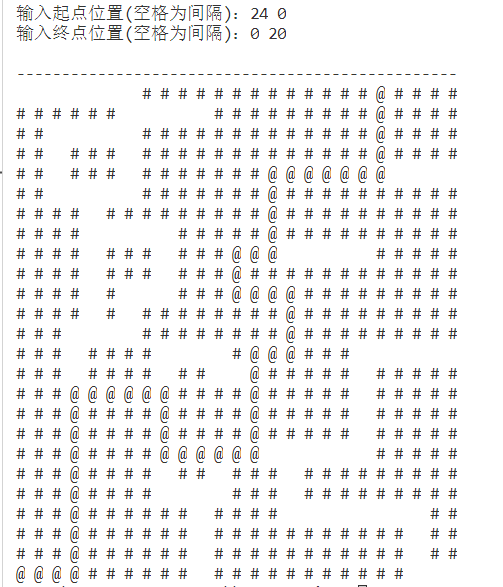
## 算法思想

核心算法FindWay函数来寻找位置a到位置b的一条迷宫通路。其主要思想是利用栈操作实现深度优先搜索。具体步骤：先将起点位置a入栈并标记其位置已走过，然后开始当栈不为空时循环：获取栈顶元素cur为当前位置，尝试走cur四个方向，如果可走且没走过，则将其入栈并标记走过，如果四个方向位置都不可走，则判断是否为终点，是的话输出路线，否则将栈顶元素出栈。

## 4. 测试数据和结果



迷宫文件采用群文件里的，运行时命名为Maze.txt



## 5. 小结

非递归的深度优先搜索时间复杂度为O（n²）。

# 家谱管理系统

## 1. 数据结构

存储家谱的数据结构为多叉树。建立家谱树类FamilyTree，其包含数根结点指针root以及家谱的各类操作。树结点BiTNode结构体包含成员数据Member类型的data，结点代数generation，父亲指针parent，孩子数childnum和孩子指针数组child。Member结构体中包含字符串型姓名name和地址address，date型生日birth和忌日death，布尔型婚否merry和健在否alive。

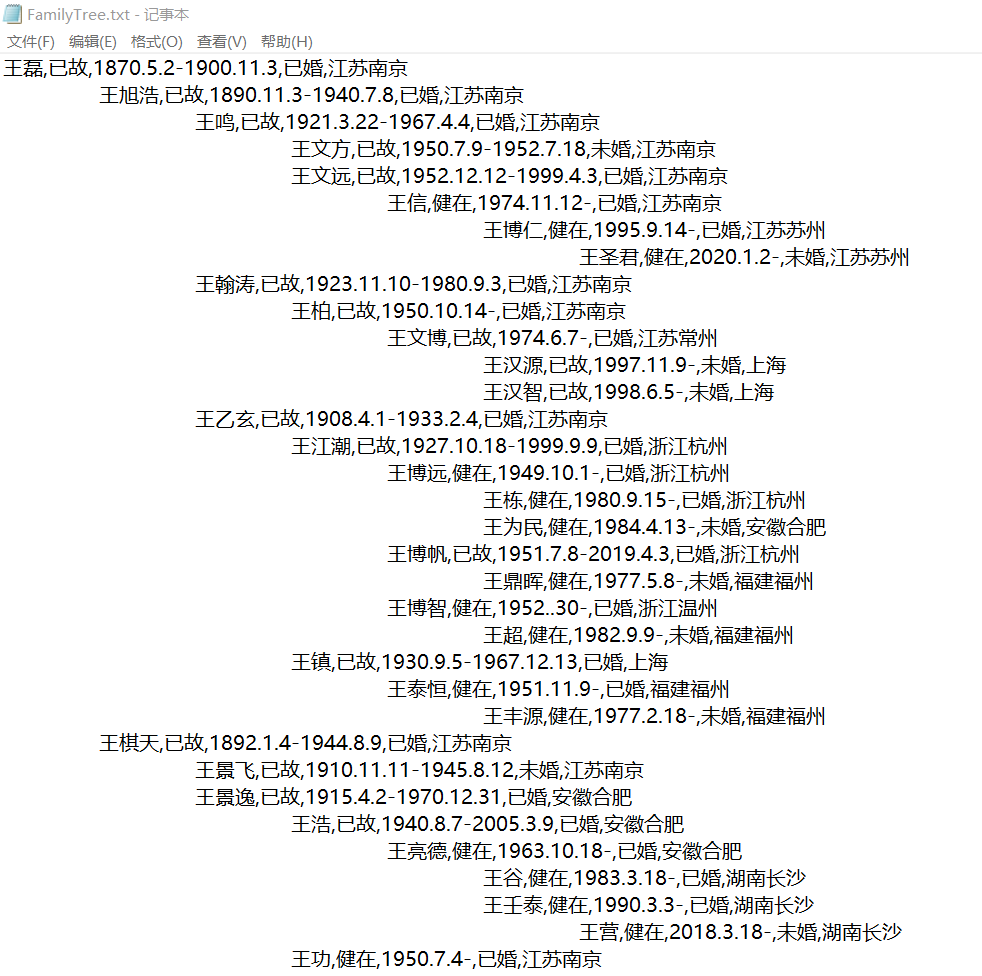
家谱文件数据存储时，一个人信息在一行里，而信息前的制表符个数表示父子关系

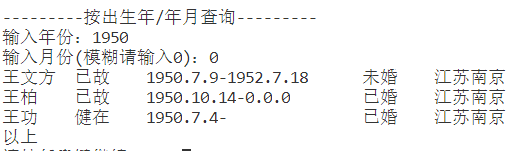
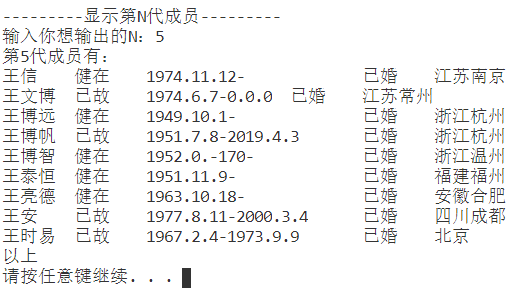
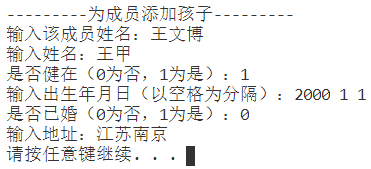
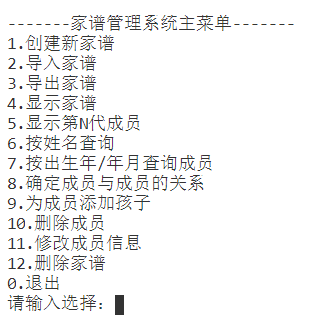
## 2. 算法思想

利用储存时的制表符以确定父子关系，以此往树中添加成员。基于以上的数据结构设置，所有的查询类功能以及输出导出功能均可通过深度优先搜索以递归的方法实现。添加孩子、修改信息可利用定位函数定位到后直接添加或修改，删除成员也是定位到后调用释放函数Relase通过类似后序遍历来删除。确定两人关系，因为是双向树，所以直接匹配两者长辈即可。

## 4. 测试数据和结果

原始家谱的数据：





## 5. 小结

所有基于深搜的算法时间复杂度均为O（n）。

该程序的优化可在查询第N代所有成员，可以改为广搜，这样就减少无用功，提高效率。

# Huffman编码与解码

## 1. 数据结构

存储结构是霍夫曼树（即二叉树），建立霍夫曼树类Huffman，包含霍夫曼树数组HT，结点数量n，字符的编码数组HC。树结点定义为结构体HTNode，其包含结点权重weight，父亲指针、左右孩子指针（以数组下标为指针）。

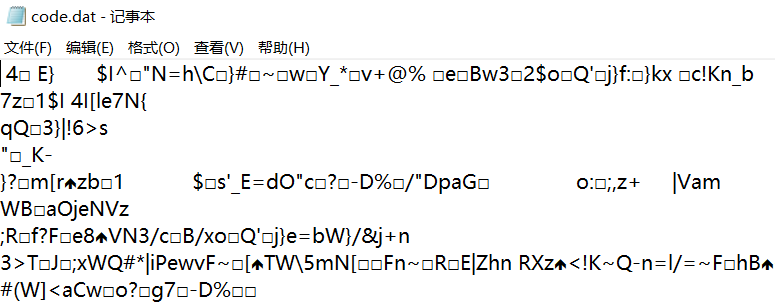
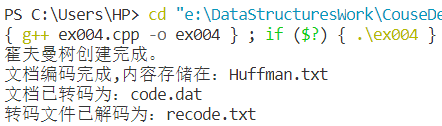
## 2. 算法思想

编码算法：先统计各个字符出现次数，生成树结点的权重到霍夫曼树数组HT中。再每次从数组中选取两个无双亲最小的权重结点，生成他们的父结点，如此重复直到只剩一个结点，即根结点。接下来进行编码，从底部每一个叶子结点根据父亲指针向上走，判断当前编0还是1，最后将生成的01字符串倒置后放入HC数组中。

转码算法：逐个从source.txt中读取字符，转为对应01编码，存在bin数组的第1到7个单元中（0号空着），满了后将bin数组对应二进制转为十进制再转为ASCII字符写入文件，并清空bin数组，如此重复。

解码算法：从code.dat文件中逐个读取字符，将其转为ASCII对应十进制数，然后将其转为二进制从高位一位一位存到temp数组中，存一位对比一次temp数组对应01字符串是否为预设编码，是的话转为对应字符写入recode.txt，并清空temp继续上述操作。

## 4. 测试数据和结果





## 5. 小结

统计权重和霍夫曼树创建及编码解码算法时间复杂度均为O（n）

# 地铁修建

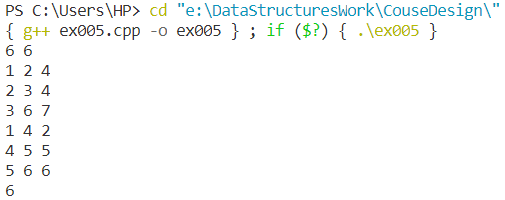
## 1. 数据结构

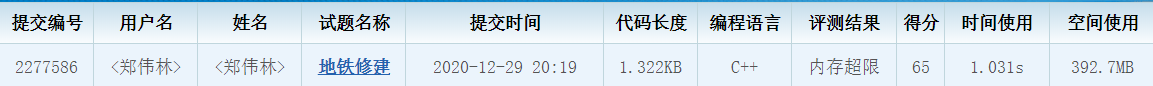
采用邻接矩阵存储地图。

## 2. 算法思想

核心算法为利用Prim算法，从1号点开始生成一棵最小生成树，然后记录这颗最小生成树中最大的一条边，当生成树包括最后一号点时输出最大边。

## 4. 测试数据和结果





## 5. 小结

普通Prim算法时间复杂度为O（n²）。因此可以对Prim进行优化，可以利用并查集来减少搜索点时的时间。

# 公交线路提示

## 1. 数据结构

建立地图类Map，包含站点数组sta、公交数组bus，以及对应的总站点数allstanum、总公交数allbugnum；站点结构为Station结构体，其包含站点名称name，经过的公交车编号数组pass，及经过公交车线路数passnum；公交车结构为Bus结构体，包含公交车编号num（即几路车），经过的站点编号route数组，及经过的站点数stanum；最后建立一个Path结构体，用以记录搜索过程中的路径，包含当前站点所属公交线bus\_num，前一站点编号pre。

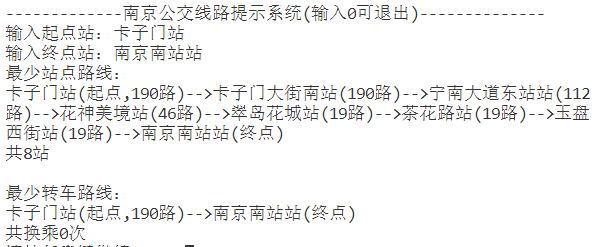
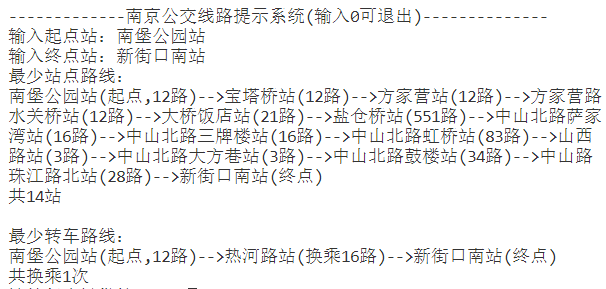
## 2. 算法思想

核心搜索算法为广度优先搜索。

搜索最少站点路线：广搜由队列实现。先将起点入队，标记为访问过，然后当队不为空时循环：队头出队，先判断是否为终点，是的话输出路线，否则将出队站所涉及的所有公交车中其前后两个未被访问过的站点入队，并标记访问过。

搜索最少转站路线：同样为广搜，只不过将入队条件改为出队站所涉及的所有公交车线路中的未访问过的站点。

## 4. 测试数据和结果



## 5. 小结

广搜时间复杂度为O（（n）

# B-树的应用

## 1. 数据结构

B-树是一棵M阶树，即它结点最多M个孩子。

存储结构为BTree类，其包含B-树根结点指针root。结点为BTNode结构体，包含结点关键字个数keynum，关键字数组key，父结点指针parent，孩子结点指针数组ptr。

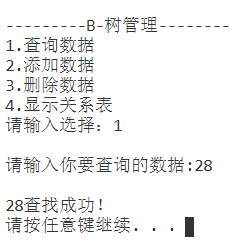
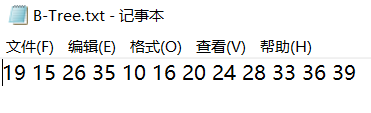
## 2. 算法思想

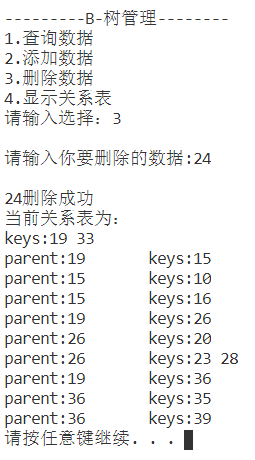
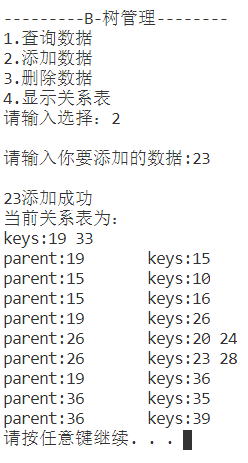
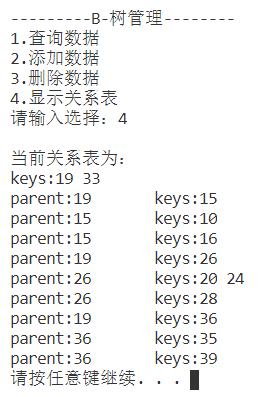
查询算法：首先将key与根结点中的ki比较，如果key=ki，则查找成功；如果key<ki，则沿着p0指向的子树继续查找；如果ki<key<ki+1，则沿着pi指向的子树继续查找；如果key>kn，则沿着pn指向的子树继续查找；如果此时pi指向空，则查找不成功。

插入算法：第一步，先查找插入的关键字key在B-树底层结点中合适的位置。第二部，插入：如果插入该关键字后当前分支结点关键字个数n<m-1则直接插入到合适位置；如果插入后n=m-1，则需进行分裂。分裂步骤为将key插入该结点且排好序后，取中间位置[m/2]（向上取整）右侧所有关键字放入新建结点中，左侧不动，中间关键字和指向新结点的指针放入原结点的父结点中。如果父结点插入后n=m-1，则继续向上分裂，直到满足n<m-1或到根结点为止。

删除算法：第一步，先查找关键字所处结点q和其位置i。第二步，如果q是底层的结点，则直接删除该最底层分支节点中的该关键字；如果q不是底层结点，则找到q右支中最小的关键字（该关键字必在底层结点），将其覆盖q中的第i个关键字即所要删除的关键字，再将q右支中所找到的最小关键字删除。第三步，删除底层分支结点中的关键字：如果删除后关键字个数不少于[m/2]-1(向上取整)，则直接删除；如果删除后少于[m/2]-1，先向左（或右）兄弟借，如果可以借（即其关键字数大于[m/2]-1），则将左（右）兄弟中最大（小）关键字上移到父结点，再把父结点中大于（小于）上移关键字的关键字下移到q中；如果左右兄弟都借不了了，则需合并结点，即将删除ki后的q与其左（无左时选右）兄弟以及父结点中分割二者的关键字合并为一个结点，如果此操作使得父结点关键字也少于[m/2]-1则继续上诉第三步操作。

## 4. 测试数据和结果





## 5. 小结

三个算法时间复杂度均为O（n²）。

# 排序算法速率比较

## 1. 数据结构

采用二维数组存储十组每组50000个的数据。

## 2. 算法思想

插入排序：将一个记录插入到已经排好队一组有序表的合适位置，如此重复，直到所有记录完成插入。

希尔排序：先将整个待排记录分成若干个子序列，分别进行直接插入排序，待整个序列中记录基本有序（即子序列均呈有序趋势），这个过程可换着分法排若干次，再对全体记录进行直接插入排序。

冒泡排序：从第一个数据开始，左右两个逐一比较交换，往后到最后可将最大（小）排到最后一个记录，如次重复。

快速排序：取一个基准数为枢轴（以第一个记录为例），将大于基准数的放置到枢轴之后，小于的放之前，再递归快速排序前半部分和后半部分。如此重复。

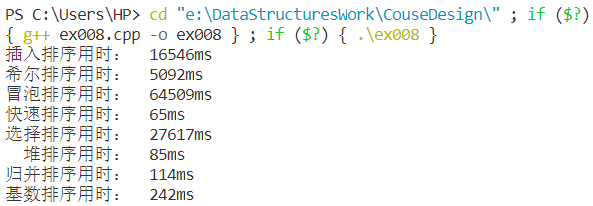
选择排序：每一趟选择剩余记录中最小（大）的数交换到这段序列第一个记录，如此继续对剩下记录排序。

堆排序：先将所有记录建立成一颗完成二叉树即堆，经过调整使其达成大顶堆，再将堆顶记录与最后一个元素交换，使得最大记录存到末尾，再将堆进行调整，重复上述操作。

归并排序：设初始序列有n个记录，可将其看成n个有序的子序列，每个序列长度为1，然后两两归并，得到n/2个长度为2的有序子序列集，如此重复最后得到1个长度为n的有序序列。

基数排序：将记录按照各位的数值分配到编号0~9的十个队列中，再将数据从这十个队列中按0~9顺序取出，如此对十位百位……完成上诉操作，完成排序。

## 4. 测试数据和结果



## 5. 小结

时间复杂度：

插入排序：O（n²） 希尔排序：O（n1.5）

冒泡排序：O（n²） 快速排序：O（nlog2n）

选择排序：O（n²） 堆排序：O（nlog2n）

归并排序：O（nlog2n） 基数排序：O（dn）

# 数字排序【1】

## 1. 数据结构

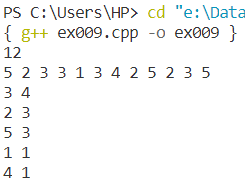
建立一个一维数组a，类型为自建的Data结构体，Data中包含元素值value以及计数count。

## 2. 算法思想

一个个数据输入，例如输入到temp中，则a[temp]中的count+1，然后再对a数组按count由大到小排序即可。

## 4. 测试数据和结果





## 5. 小结

时间复杂度为排序的O（n²）。可换用效率更好且稳定的排序方法。

# 魔法优惠券【1】

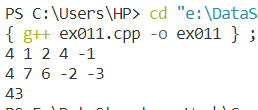
## 1. 数据结构

建立队列来解决。

## 2. 算法思想

建立四个队列对应正优惠券、负优惠券、正商品和负商品。将其从大到小排序。然后将正优惠券和正商品队列一个个出队相乘直到一个队列为空，负优惠券和负商品队列同样操作。

## 4. 测试数据和结果



## 5. 小结

时间复杂度为排序的O（nlogn）。

# 消除类游戏【2】

## 1. 数据结构

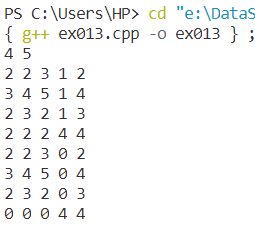
建立二维数组map来存储数据，其为自建的Point结构体，包含该位置的值value和是否可以被消除status。

## 2. 算法思想

在读入数据后将所有位置设为不可被消除（即status为0），然后逐行判断有无可消除的序列，将其状态status改为1，再逐列判断一次，最后将状态为1的消除。

## 4. 测试数据和结果





## 5. 小结

时间复杂度为O（n²）。

# 最小二叉树【3】

## 1. 数据结构

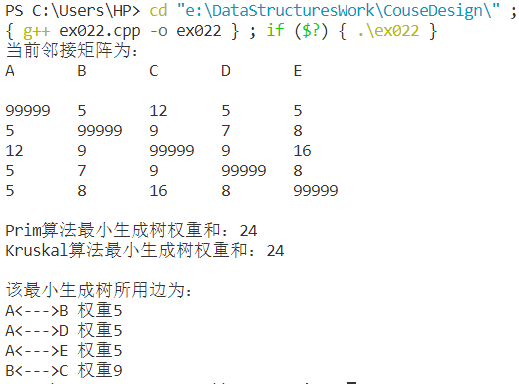
Prim算法用的是邻接矩阵的结构，Kruskal算法直接用边集输入即可（边集为edge型数组edg，edge结构体包含边起点i，边终点j和边权重w），还用到并查集以便判断回路。

## 2. 算法思想

Prim算法：先从一个点出发，找到距离当前最近的点加入形成一棵子树，再继续找距离当前子树最近的点并加入，直到所有点都再子树中，该树即为最小生成树。

Kruskal算法：先对边集从小到大排序。再对并查集初始化。然后遍历边集，如果两条边可以相连（即连接后的子树不会有回路），则进行合并，直到子树中包含n-1条边，此时子树为最小生成树。

## 4. 测试数据和结果



## 5. 小结

Prim算法时间复杂度为O（n），Kruskal算法时间复杂度为O（nlogn）。

Prim算法可以用堆优化。

# 课程设计总结

**代码行数**

1.系统进程统计   355行  
2.迷宫问题   181行  
3.家谱管理系统    951行  
4.Huffman编码与解码    299行  
5.地铁修建    67行  
6.公交线路提示    372行  
7.B-树的应用    416行  
8.排序算法速率比较   416行  
9.数字排序【1】   46行  
11.魔法优惠券【1】   89行  
13.消除类游戏【2】   77行  
22.最小二叉树【3】   172行  
 共计3445行

**心得体会**

**终于结束了漫长的课程设计。虽然老师早早布置，但由于各种原因还是一拖再拖，最后只有十天才开始动手。于是乎开始了全天候编程模式，每天除了吃饭就是编程，朝九晚十一，紧赶慢赶终于将必做题做完并做了四题选做题。这算是人生到目前为止编程强度最大的一段时间，无论是工作量还是难度都是较大的一个水平。**

**回顾所做的课程设计，也发现不少当时赶工时没注意到问题，进行了稍加改进。印象最深的还是B-树的构建，花费了好几天不断改bug，最后才能够正常的运行并操作B-树，插入算法和删除算法中需要考虑到各种情况以及结构化设计，其中我的解决方法是将插入中的查询为第一部分，再调用插入关键字函数进行插入，而删除较为复杂分为三步，第一步查询，如果不是叶子结点则从右支寻找最大值来覆盖再删除右支最大值，如果是直接调用删除关键字函数，第二步删除关键字，第三步进行需要合并的操作。整个过程中情况稍为复杂，所以该程序标记了最多的中文注释，以便调试和之后的修改作为帮助，因此养成了编程中多做注释的习惯，也深深感受到良好的代码注释对调试和维护的重要性。而说到复杂，另外一题家谱是几个程序中行数最多的一个，将近一千行的代码量，而这里面最复杂的却是从文件导入数据这个过程，需要自己设置数据文件格式，并且考验字符串处理的操作，有一小小个错误都会使所读数据出现偏移而造成重大错误。这个过程中我经历了各种各样的报错，不断地在网上检索资料询问如何解决，最后总发现使字符串处理出错导致的一系列问题。同样的，在Huffman编码中，字符串的处理仍是很重要的一个部分，巧妙地利用ASCII码将八位二进制的01字符串变成一个字节输出，这样才能真正将空间压缩下来。因此还需多加练习字符串处理部分，csp考试中的第三题也往往会考到这部分。而公交线路这题也有字符串处理相关，不过在家谱锻炼之后，在这里的出错减少了许多。这题关键的部分在于数据结构的搭建，如果直接用邻接矩阵搭建一个图，由于其有五千多点站点，这个图将十分巨大，再利用dijkastra算法求解最短路径将十分不易。在他人的提示下，我将车和站点分开存储，但又互相包含相关信息，即车次中包含经过站点，站点中包含经过车次，这样将两者关联起来，再利用广度优先搜索即可很快速的求出答案。因此一个准确的数据结构配合合适的算法对问题求解来说将会有巨大的帮助。**

**由于时间原因没能做到选做题十二分，但在选做题中，有不少挺重要也挺有趣的题目，在寒假时间可以尝试尝试。尤其是系统设计类的题目，通过自己构思出一个系统并将其编写出来，这是一件十分有成就感的事情，同时还能锻炼自己的程序设计和数据结构设计能力。而CSP相关的题目，我还需要学习STL容器的运用，听说这个东西掌握之后做题将十分方便。**

**最后的最后，感谢高航老师的教学和指导。让我系统地学习了数据结构这门程序设计中最为重要的学科之一，并且对各个数据结构都有了较深的印象，能够在编程中较为清晰的分析和搭建。还有感谢帮助过我的同学们，百忙之中也为我解惑。**