课程设计题目及要求

一、约瑟夫双向生死游戏（难度系数 0.8）

1 项目简介

约瑟夫双向生死游戏是在约瑟夫生者死者游戏的基础上，正向计数后反向计数，然后再正向计数。具体描述如下：30 个旅客同乘一条船，因为严重超载，加上风高浪大，危险万分；因此船长告诉乘客，只有将全船一半的旅客投入海中，其余人才能幸免遇难。无奈，大家只得同意这种办法，并议定 30 个人围成一圈，由第一个人开始，顺时针依次报数，数到第 9 人，便把他投入大海中，然后从他的下一个人数起，逆时针数到第 5 人，将他投入大海，然后从他逆时针的下一个人数起，顺时针数到第 9 人，再将他投入大海，如此循环，直到剩下 15 个乘客为止。问哪些位置是将被扔下大海的位置。

2 设计思路

本游戏的数学建模如下：假设 n 个旅客排成一个环形，依次顺序编号 1，2，…，n。从某个指定的第 1 号开始，沿环计数，数到第 m 个人就让其出列，然后从第 m+1 个人反向计数到 m-k+1 个人，让其出列，然后从 m-k 个人开始重新正向沿环计数，再数 m 个人后让其出列，然后再反向数 k 个人后让其出列。这个过程一直进行到剩下 q 个旅客为止。

本游戏的要求用户输入的内容包括：

1. 旅客的个数，也就是 n 的值；
2. 正向离开旅客的间隔数，也就是 m 的值；
3. 反向离开旅客的间隔数，也就是 k 的值；
4. 所有旅客的序号作为一组数据要求存放在某种数据结构中。

本游戏要求输出的内容是包括

1. 离开旅客的序号；
2. 剩余旅客的序号；

所以，根据上面的模型分析及输入输出参数分析，可以定义一种数据结构后进行算法实现。

3 数据结构

约瑟夫双向生死游戏如果用单循环链表作为线性存储结构，就只能正向计数结点，反向

计数比较困难，算法较为复杂，而且效率低。用双向循环链表解决这一问题，实现的方法相对要简单得多。

为了不失一般性，将 30 改为一个任意输入的正整数 n，而正向报数上限（原为 9）也为一个任选的正整数 m，正向报数上限（原为 5）也为一个任选的正整数 k，。这样该算法描述如下：

(1) 创建含有 n 个结点的双向循环链表；

(2) 生着与死者的选择：

p 指向链表的第一个结点，初始 i 置为 1；

while(i<=n/2)　//删除一半的结点

｛从 p 指向的结点沿链前进 m-1 步；

删除第 m 个结点（q 所指向的结点）；

p 指向 q 的下一个结点；

输出其位置 q->data;

i 自增 1；

从 p 指向的结点沿链后退 k-1 步；

删除第 k 个结点（q 所指向的结点）；

p 指向 q 的上一个结点；

输出其位置 q->data;

i 自增 1；

}

(3) 输出所有生者的位置。

二、迷宫旅行游戏（难度系统 0.8）

1 项目简介

迷宫只有两个门，一个门叫入口，另一个门叫出口。一个骑士骑马从入口走进迷宫，迷宫中设置很多墙壁，对前进方向形成了多处障碍。骑士需要在迷宫中寻找通路以到达出口。

2 设计思路

迷宫问题的求解过程可以采用回溯法即在一定的约束条件下试探地搜索前进，若前进中受阻，则及时回头纠正错误另择通路继续搜索的方法。从入口出发，按某一方向向前探索，若能走通（未走过的），即某处可达，则到达新点，否则试探下一方向；若所有的方向均没有通路，则沿原路返回前一点，换下一个方向再继续试探，直到所有可能的通路都探索到，或找到一条通路，或无路可走又返回到入口点。

在求解过程中，为了保证在到达某一点后不能向前继续行走（无路）时，能正确返回前一点以便继续从下一个方向向前试探，则需要在试探过程中保存所能够到达的每一点的下标及从该点前进的方向，当找到出口时试探过程就结束了。

为了确保程序能够终止，调整时，必须保证曾被放弃过的填数序列不被再次试验，即要求按某种有序模型生成填数序列。给解的候选者设定一个被检验的顺序，按这个顺序逐一生成候选者并检验。

3 数据结构

迷宫问题是栈应用的一个典型例子。通过前面分析，我们知道在试探过程中为了能够沿着原路逆序回退，就需要一种数据结构来保存试探过程中曾走过的点的下标及从该点前进的方向，在不能继续走下去时就要退回前一点继续试探下一个方向，栈底元素是入口，栈顶元素是回退的第一站，也即后走过的点先退回，先走过的点后退回，与栈的“后进选出，先进后出”特点一致，故在该问题求解的程序中可以采用栈这种数据结构。在迷宫有通路时，栈中保存的点逆序连起来就是一条迷宫的通路，否则栈中没有通路。

三、八皇后问题（难度系统 0.8）

1 项目简介

八皇后问题是一个古老而著名的问题，是回溯算法的典型例题。该问题是十九世纪著名的数学家高斯 1850 年提出：在 8×8 格的国际象棋棋盘上，安放八个皇后，要求没有一个皇后能够“吃掉”任何其他一个皇后，即任意两个皇后都不能处于同一行、同一列或同一条对角线上，求解有多少种摆法。

高斯认为有 76 种方案。1854 年在柏林的象棋杂志上不同的作者发表了 40 种不同的解，后来有人用图论的方法得出结论，有 92 种摆法。

2 设计思路

八皇后在棋盘上分布的各种可能的格局数目非常大，约等于 2^{32} 种，但是，可以将一些明显不满足问题要求的格局排除掉。由于任意两个皇后不能同行，即每一行只能放置一个皇后，因此将第 i 个皇后放置在第 i 行上。这样在放置第 i 个皇后时，只要考虑它与前 i-1 个皇后处于不同列和不同对角线位置上即可。

解决该问题采用回溯法。首先将第一个皇后放于第一行第一列，然后依次在下一行上放置下一个皇后，直到八个皇后全放置安全。在放置每一个皇后时，都依次对每一列进行检测，首先检测待第一列是否与已放置的皇后冲突，如不冲突，则将皇后放置在该列，否则，选择该行的下一列进行检没。如整行的八列都冲突，则回到上一行，重新选择位置，依次类推。

3 数据结构

八皇后问题是栈应用的另一个典型例子。通过前面分析，我们知道在对八个皇后的位置进行试探的过程中，可能遇到在某一行上的所有位置都不安全的情况，这时就要退回到上一行，重新摆放在上一行放置的皇后。为了能够对上一行的皇后继续寻找下一个安全位置，就必须记得该皇后目前所在的位置，即需要一种数据结构来保存从第一行开始的每一行上的皇后所在的位置。在放置进行不下去时，已经放置的皇后中后放置的皇后位置先被纠正，先放置的皇后后被纠正，与栈的“后进选出，先进后出”特点一致，故在该问题求解的程序中可以采用栈这种数据结构。在八个皇后都放置安全时，栈中保存的数据就是八个皇后在八行上的列位置。

四、停车场管理（难度系数 1）

1 项目简介

设停车场是一个可以停放 n 辆汽车的南北方向的狭长通道，且只有一个大门可供汽车进出。汽车在停车场内按车辆到达时间的先后顺序，依次由北向南排列（大门在最南端，最先到达的第一辆车停放在车场的最北端），若车场内已停满 n 辆车，那么后来的车只能在门外的便道上等候，一旦有车开走，则排在便道上的第一辆车即可开入；当停车场内某辆车要离开时，在它之后进入的车辆必须先退出车场为它让路，待该辆车开出大门外，其它车辆再按原次序进入车场，每辆停放在车场的车在它离开停车场时必须按它停留的时间长短交纳费用。试为停车场编制按上述要求进行管理的模拟程序。要求程序输出每辆车到达后的停车位置（停车场或便道上），以及某辆车离开停车场时应缴纳的费用和它在停车场内停留的时间。

要求：（1）车辆进入和停留时间要求使用随机函数实现

（2）记录曾经进入过停车场的所有车辆进、出时间及缴费信息

2 设计思路

停车场的管理流程如下：

①当车辆要进入停车场时，检查停车场是否已满，如果未满则车辆进入停车场；如果停车场已满，则车辆进入便道等候。

②当车辆要求出栈时，先让在它之后进入停车场的车辆退出停车场为它让路，再让该车退出停车场，让路的所有车辆再按其原来进入停车场的次序进入停车场。之后，再检查在便道上是否有车等候，有车则让最先等待的那辆车进入停车场。

3 数据结构

由于停车场只有一个大门，当停车场内某辆车要离开时，在它之后进入的车辆必须先退出车场为它让路，先进停车场的后退出，后进车场的先退出，符合栈的“后进先出，先进后出”的操作特点，因此，可以用一个栈来模拟停车场。而当停车场满后，继续来到的其它车辆只能停在便道上，根据便道停车的特点，先排队的车辆先离开便道进入停车场，符合队列的“先进先出，后进后出”的操作特点，因此，可以用一个队列来模拟便道。排在停车场中间的车辆可以提出离开停车场，并且停车场内在要离开的车辆之后到达的车辆都必须先离开停车场为它让路，然后这些车辆依原来到达停车场的次序进入停车场，因此在前面已设的一个栈和一个队列的基础上，还需要有一个地方保存为了让路离开停车场的车辆，由于先退出停车场的后进入停车场，所以很显然保存让路车辆的场地也应该用一个栈来模拟。因此，本题求解过程中需用到两个栈和一个队列。栈以顺序结构实现，队列以链表结构实现。

五、单词检索统计程序（难度系数 0.9）

1 项目简介

给定一个文本文件，要求统计给定单词在文本中出现的总次数，并检索输出某个单词出现在文本中的行号、在该行中出现的次数以及位置。

2 设计思路

本项目的设计要求可以分为三个部分实现：其一，建立一个文本文件，文件名由用户用键盘输入；其二，给定单词计数，输入一个不含空格的单词，统计输出该单词在文本中的出现次数；其三，检索给定单词，输入一个单词，检索并输出该单词所在的行号、该行中出现的次数以及在该行中的相应位置。

1. 建立文件的实现思路是：

(1) 定义一个串变量；

(2) 定义文本文件；

(3) 输入文件名，打开该文件；

(4) 循环读入文本行，写入文本文件，其过程如下：

```
while(不是文件输入结束){
    读入一文本行至串变量；
    串变量写入文件；
    输入是否结束输入标志；
}
```

(5) 关闭文件。

2. 给定单词计数的实现思路是：

该功能需要用到模式匹配算法，逐行扫描文本文件。匹配一个，计数器加 1，直到整个文件扫描结束；然后输出单词出现的次数。

串是非数值处理中的主要对象，如在信息检索、文本编辑、符号处理等许多领域，得到越来越广泛的应用。在串的基本操作中，在主串中查找模式串的模式匹配算法是文本处理中最常用、最重要的操作之一，称为模式匹配或串匹配，就是求子串在主串中首次出现的位置。朴素模式匹配算法的基本思路是将给定字串与主串从第一个字符开始比较，找到首次与子串完全匹配的子串为止，并记住该位置。但为了实现统计子串出现的个数，不仅需要从主串的第一个字符位置开始比较，而且需要从主串的任一位置检索匹配字符串。

其实现过程如下：

(1) 输入要检索的文本文件名，打开相应的文件；

(2) 输入要检索统计的单词；

(3) 循环读文本文件，读入一行，将其送入定义好的串中，并求该串的实际长度，调用串匹配函数进行计数。具体描述如下：

while(不是文件结束){

读入一行并到串中；

求出串长度；

模式匹配函数计数；

}

(4) 关闭文件，输出统计结果。

3. 检索单词出现在文本文件中的行号、次数及其位置的实现思路是：

这个设计要求同上一个设计类似，但是要相对复杂一些。其实现过程描述如下：

(1) 输入要检索的文本文件名，打开相应的文件；

(2) 输入要检索统计的单词；

(3) 行计数器置初值 0；

(4) while(不是文件结束){

读入一行到指定串中；

求出串长度；

行单词计数器 0；

调用模式匹配函数匹配单词定位、该行匹配单词计数；

行号计数器加 1；

if(行单词计数器!=0)输出行号、该行有匹配单词的个数以及相应的位置；

}

4. 主控菜单程序的结构

该部分内容如下：

(1) 头文件包含；

(2) 菜单选项包括：

1. 建立文件

2. 单词计数

3. 单词定位

4. 退出程序

(3) 选择 1-4 执行相应的操作，其他字符为非法。

3 数据结构

如果在程序设计语言中，串只是作为输入或输出的常量出现，则只需存储此串的串值，即字符序列即可.但在多数非数值处理的程序中，串也以变量的形式出现。串有三种机内表

示方法：定长顺序存储表示、堆分配存储表示和串的块链存储表示。定长顺序存储表示类似于线性表的顺序存储结构，用一组地址连续的存储单元存储串值的字符序列。在串的定长顺序存储结构中，按照予定义的大小，为每个定义的串变量分配一个固定长度的存储区，则可用定长数组描述。

——串的定长顺序存储表示——

#define MAXSTRLEN 255 //用户可在 255 以内定义最大串长

typedef unsigned char SString [MAXSTRLEN+1]; //0 号单元存放串的长度

串的实际长度可在这预定义长度的范围内随意，超过于定义长度的串值则被舍去，称之为"截断"。对串长有两种表示方法:一是如上述定义描述的那样，以下标为 0 的数组分量存放串的实际长度，如 PASCAL 语言中的串类型采用这种表示方法;二是在串值后面加一个不计入串长的结束标记字符，如在有的C语言中以"\0"表示串值的终结。此时的串长为隐含值，显然不便于进行某些串操作。在这种存储结构表示时实现串求子串操作如下：

求子串 SubStrmg(&Sub,S,pos,len)过程即为复制字符序列的过程，将串 S 中从第 pos 个字符开始长度为 len 的字符序列复制到串 Sub 中。显然，本操作不会有需截断的情况，但有可能产生用户给出的参数不符合操作的初始条件，当参数非法时，返回 ERROR。其算法描述如算法 4.3 所示。

StatusSub String (Sstring&Sub,SstringS,intpos,intlen){

//用 Sub 返回串 s 第 pos 个字符起长度为 len 的子串。其中 1≤pos≤StrLength(S)且 0≤len≤StrLength(S)-pos+1.

if(poss[0] || lenS[0]-pos+1) return ERROR;

Sub[1..len] = S[pos..pos+len-1];

Sub[0] =len;

return OK;

} //End of SubString

由此可见，在顺序存储结构中，实现串操作的原操作为“字符序列的复制”，操作的时间复杂度基于复制的字符序列的长度。

本项目主要实现子串的检索和定位操作，所以用定长顺序存储表示比较简单易行。

六、Internet 网络通路管理(难度系数 1)

1 项目简介

本项目是对 Internet 网络通路管理的简单模拟，以完成建立 Internet 网络通路、修改 Internet 网络通路信息和删除 Internet 网络通路信息等功能。

2 设计思路

本项目的实质是完成对 Internet 网络通路信息的建立、查找、插入、修改、删除等功能，可以首先定义项目的数据结构，然后将每个功能写成一个函数来完成对数据的操作，最后完成主函数以验证各个函数功能并得出运行结果。

3 数据结构

Internet 网络通路中的主机之间关系可以是任意的，任意两个站点之间都可能相关。而在图形结构中，结点之间的关系可以是任意的，图中任意两个数据元素之间都可能相关。所以可以用图形结构来表示 n 个主机以及主机之间可能设置的 Internet 网络通路，其中网的顶点表示网络通路中的主机，边表示两个主机之间的网络通路，赋予边的权值表示相应的距离。

可以用一维数组存储图中顶点的信息，用矩阵表示图中各顶点之间的相邻关系。即用两个数组分别存储数据元素（顶点）的信息和数据元素之间的关系（边或弧）的信息。

假设图中顶点数为 n，网的邻接矩阵的定义为，当 v_i 到 v_j 有弧相邻接时，$a_{i,j}$ 的值应为该弧上的权值，否则为∞。将图的顶点信息存储在一个一维数组中，并将它的邻接矩阵存储在一个二维数组中即构成图的数组表示。

图的数组(邻接矩阵)存储表示

```
const INFINITY = INT_MAX;              // 最大值∞
const MAX_VERTEX_NUM = 20;             // 最大顶点个数
typedefenum {DG, DN, AG, AN} GraphKind;
                                       // 类型标志{有向图,有向网,无向图,无向网}
typedefstructArcCell {                 // 弧的定义
  VRTypeadj;                           // VRType 是顶点关系类型。
                                       // 对无权图，用 1 或 0 表示相邻否；对带权图，则为权值类型。
  InfoType *info;                      // 该弧相关信息的指针
} AdjMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM];
typedefstruct {                        // 图的定义
  VertexTypevexs[MAX_VERTEX_NUM]; // 顶点信息
```

```
    AdjMatrix arcs;                    // 表示顶点之间关系的二维数组
    intvexnum, arcnum;                 // 图的当前顶点数和弧(边)数
    GraphKind kind;                    // 图的种类标志
} MGraph;
```

七、家谱管理（难度系数 1）

1 项目简介

家谱（或称族谱）是一种以表谱形式，记载一个以血缘关系为主体的家族世系繁衍和重要人物事迹的特殊图书体裁。家谱是中国特有的文化遗产，是中华民族的三大文献（国史，地志，族谱）之一，属珍贵的人文资料，对于历史学、民俗学、人口学、社会学和经济学的深入研究，均有其不可替代的独特功能。本项目对家谱管理进行简单的模拟，以实现创建家谱、查看家族成员信息、插入家族成员、修改家族成员信息、输出家谱等功能。

2 设计思路

本项目的实质是完成对家谱成员信息的建立、查找、插入、修改等功能，可以首先定义家族成员的数据结构，然后将每个功能写成一个函数来完成对数据的操作，最后完成主函数以验证各个函数功能并得出运行结果。

3 数据结构

因为家族中的成员之间存在一个对多个的层次结构关系，所以不能用上面讲的性表来表示。家谱从形状上看像一颗倒长的树，所以用树结构来表示家谱比较适合。树型结构是一类非常重要的非线性数据结构，直观看来，树是以分支关系定义的层次结构。

可以二叉链表作为树的存储结构，链表中的两个链域分别指向该结点的第一个孩子结点和下一个兄弟结点，故该表示法又称二叉树表示法，或孩子兄弟表示法。孩子兄弟表示法是一种链式存储结构，它通过描述每个结点的一个孩子和兄弟信息来反映结点之间的层次关系，其具体的结点结构为：

firstChild	data	nextSibling

其中，firstchild 为指向该结点第一个孩子的指针，nextsibling 为指向该结点下一个兄弟，elem 是数据元素内容，举例如下：

其存储形式定义如下：

```
typedefstructCSLinklist{
    Elemtypedata;
    structCSLinklist *firstchild,*nextsibling;
} CSLinklist,*CSTree;
```

八、表达式求值问题（难度系数 1）

1 项目简介

表达式求值是程序设计语言编译中的一个最基本问题，就是将一个表达式转化为逆波兰表达式并求值。具体要求是以字符序列的形式从终端输入语法正确的、不含变量的整数表达式，并利用给定的优先关系实现对算术四则混合表达式的求值，并演示在求值过程中运算符栈、操作数栈、输入字符和主要操作变化过程。

要把一个表达式翻译成正确求值的一个机器指令序列，或者直接对表达式求值，首先要能正确解释表达式。任何一个表达式都是由操作符（operand）、运算符（operator）和界限符（delimiter）组成，我们成它们为单词。一般的，操作数既可以是常数，也可以是被说明为变量或常量的标识符；运算符可以分为算术运算符、关系运算符和逻辑运算符 3 类；基本界限符有左右括号和表达式结束符等。为了叙述的简洁，我们仅仅讨论简单算术表达式的求值问题。这种表达式只包括加、减、乘、除 4 种运算符。

人们在书写表达式时通常采用的是“中缀”表达形式，也就是将运算符放在两个操作数中间，用这种“中缀”形式表示的表达式称为中缀表达式。但是，这种表达式表示形式对计算机处理来说是不大合适的。对于表达式的表示还有另一种形式，称之为“后缀表达式”，即将运算符紧跟在两个操作数的后面。这种表达式比较适合计算机的处理方式，因此要用计算机来处理、计算表达式的问题，首先要将中缀表达式转化成后缀表达式，又成为逆波兰表达式。

2 设计思路

为了实现表达式求值，可以首先读入原表达式（包括括号）并创建对应二叉树，其次对二叉树进行前序遍历、中序遍历、后续遍历（非递归），并输出逆波兰表达式，最后求解原表达式的值，同时对非法表达式格式能予以判断。

设 str1=（a+b)*((c+d)*e+f*h*g)；str2 是目的串。先用一个堆栈存储 operand，遇到数字就不做处理直接放到目的串中，遇到 operaor 则将其与堆栈顶元素比较优先级，决定是否送到输出串中，这样处理之后能得到一个后续的排列，再建立二叉树，当遇到 ab+这样的建立左右接点为 a 和 b 的根为+的树并返回根接点，当遇到操作符前面只有一个操作数的如 c*这样的情况就建立右子接点为 c 的根接点为*的，左结点为上次操作返回的子树的根接点（这种情况下就是合并树的情况）。

非递归方法进行树的遍历需要用到栈结构。首先调用 stack 的 ADT 中的 creatEmptyStack() 函数来建立一个空栈（注意，栈元素的类型是指向树结点的指针）。

首先将根节点压入栈中。然后进入循环。

如果栈不空，

弹出栈顶元素，

进行访问，

压入右子树（节点指针），

压入左子树（节点指针），

循环。

由于需要同时用到树和栈两种抽象数据类型并且需要进行互动操作，在头文件中需要加入#ifndef#define#endif 宏命令来避免出现重复定义的问题。

3 数据结构

用二叉树的结构来存储表达式，后续遍历二叉树即可得到逆波兰表达式，也可以通过树的合并操作完成表达式的计算。表达式二叉树结点类型如下。

```
typedef    structnodeTag
union{
   intopnd;
  char optr;
}val;
struct    nodeTag *left;
struct    nodeTag *right;
}treeNode;
```

九、图像压缩编码优化

1 项目简介

信息时代，人们对使用计算机获取信息、处理信息的依赖性越来越高。计算机系统面临的是数值、文字、语言、音乐、图形、动画、静图像、电视视频图像等多种媒体。数字化的视频和音频信号的数量之大是惊人的，对于电视画面的分辨率 640×480 的彩色图像，30 帧/s，则一秒钟的数据量为：640×480×24×30＝221　12M，所以播放时，需要 221Mbps 的通信回路。存储时，1 张 CD 可存 640M，则仅可以存放 2m89s 的数据。多媒体信息中，视频信息是一种比较特殊的媒体，数据量极大，信息丰富，并以与时间密切相关的流的形式存在。因此，视频数据的表达、组织、存储和传输都有很大难度。解决的基础在于对视频数据进行压缩。

大数据量的图像信息会给存储器的存储容量，通信干线信道的带宽，以及计算机的处理速度增加极大的压力。单纯靠增加存储器容量，提高信道带宽以及计算机的处理速度等方法来解决这个问题是不现实的，这时就要考虑压缩。压缩的关键在于编码，如果在对数据进行编码时，对于常见的数据，编码器输出较短的码字；而对于少见的数据则用较长的码字表示，就能够实现压缩。

功能要求：已知包含 16 种字符的文件，文件中字符数量>=2000 个，通过哈夫曼编码方案实现对该文件压缩和解压缩。

2 设计思路

假设一个文件中出现了 8 种符号 S0，SQ，S2，S3，S4，S5，S6，S7，那么每种符号要编码，至少需要 3bit。假设编码成 000，001，010，011，100，101，110，111。那么符号序列 S0S1S7S0S1S6S2S2S3S4S5S0S0S1 编码后变成 000001111000001110010010011100101000000001，共用了 42bit。我们发现 S0，S1，S2 这 3 个符号出现的频率比较大，其它符号出现的频率比较小，我们采用这样的编码方案：S0 到 S7 的码辽分别 01，11，101，0000，0001，0010，0011，100，那么上述符号序列变成 011110001110011101101000000010010010111，共用了 39bit。尽管有些码字如 S3，S4，S5，S6 变长了(由 3 位变成 4 位)，但使用频繁的几个码字如 S0，S1 变短了，所以实现了压缩。对于上述的编码可能导致解码出现非单值性：比如说，如果 S0 的码字为 01，S2 的码字为 011，那么当序列中出现 011 时，你不知道是 S0 的码字后面跟了个 1，还是完整的一个 S2 的码字。因此，编码必须保证较短的编码决不能是较长编码的前缀。符合这种要求的编码称之为前缀编码。

3 数据结构

要构造符合这样的二进制编码体系，可以通过二叉树来实现。

1)首先统计出每个符合出现的频率，上例 SO 到 S7 的出现频率分别为 4/14，3/14，2/14，1/14，1/14，1/14，1/14，1/14。

2)从左到右把上述频率按从小到大的顺序排列。

3)每一次选出最小的两个值，作为二叉树的两个叶子节点，将和作为它们的根节点，这两个叶子节点不再参与比较，新的根节点参与比较。

4)重复(3)，直到最后得到和为 1 的根节点。

将形成的二叉树的左节点标 0，右节点标 1。把从最上面的根节点到最下面的叶子节点路径中遇到的 0，1 序列串起来，得到各个符号的编码。

可以看到，符号只能出现在树叶上，任何一个字符的路径都不会是另一字符路径的前缀路径，这样，前缀编码也就构造成功了。

这样一棵二叉树称之为 Huffman 树，常用于最佳判定，它是最优二叉树，是一种带权路径长度最短的二叉树。所谓树的带权路径长度，就是树中所有的叶结点的权值乘上其到根结点的路径长度(若根结点为 0 层，叶结点到根结点的路径长度为叶结点的层数)。树的带权路径长度记为：WPL＝(W1*L1＋W2*L2＋W3*L3＋…＋Wn*Ln)，N 个权值 Wi(i＝1，2，…n)构成一棵有 N 个叶结点的二叉树，相应的叶结点的路径长度为 Li(i＝1，2，…n)。Huffman 树得出的 WPL 值最小。

十、公交线路管理（难度系数 1）

1 项目简介

本项目是对公交车线路信息的简单模拟，以完成建立公交路线信息、修改公交路线信息和删除公交路线信息等功能。

2 设计思路

本项目的实质是完成对公交线路信息的建立、查找、插入、修改、删除等功能，可以首先定义项目的数据结构，然后将每个功能写成一个函数来完成对数据的操作，最后完成主函数以验证各个函数功能并得出运行结果。

3 数据结构

公交站点之间的关系可以是任意的，任意两个站点之间都可能相关。而在图形结构中，结点之间的关系可以是任意的，图中任意两个数据元素之间都可能相关。所以可以用图形结构来表示 n 个公交站点之间以及站点之间可能设置的公交路线，其中网的顶点表示公交站点，边表示两个站点之间的路线，赋予边的权值表示相应的距离。

因为公交路线是有一定的连续关系的，如果想输出从某一个起始点开始到某一终点结束的公交路线，就需要找到从某一顶点开始的第一个邻接点和下一个邻接点。因为在邻接表中容易找到任一顶点的第一个邻接点和下一个邻接点，所以本项目使用了图的邻接表存储结构。邻接表是图的一种链式存储结构。在邻接表中，对图的每一个顶点建立一个单链表，第 i 个单链表中的结点表示依附于顶点 v_i 的边（对有向图是以顶点 v_i 为尾的弧）。每个结点由三个域组成，其中邻接点域(adjvex)指示与顶点 v_i 邻接的点在图中的位置，链域(nextarc)指示下一条边或弧的结点；数据域(info)存储和边或弧相关的信息，如权值等。每个链表上附设一个表头结点，在表头结点中，除了设有链域(firstarc)指向链表中第一个结点之外，还设有存储顶点 v_i 的名或其它有关信息的数据域(data)。这些表头结点通常以顺序结构的形式存储，以便随机访问任意顶点的链表。图的邻接表存储表示结构可形式的说明如下：

```
#define MAX_VERTEX_NUM   20
typedefstructArcNode{     //弧的结构
intadjvex;   //该弧所指向的顶点的位置
structArcNode   *nextarc;   //指向下一条弧的指针
InfoType          *info;   //该弧相关信息的指针
}ArcNode;
typedefstructVNode{       //顶点结构
```

```
VertexType      data;   //顶点信息
ArcNode         *firstarc;   //指向第一条依附该顶点的弧的指针
}VNode,    AdjList[MAX_VERTEX_NUM];
typedef struct {       //图的结构
AdjList     vertices;
int         vexnum,    arcnum;   //图的当前顶点数和弧数
int         kind;   //图的种类标志
}ALGraph;
```

十一、导航最短路径查询(难度系数 0.9)

1 项目简介

设计一个交通咨询系统（城市数量>=10），能让旅客咨询从任一个城市顶点到另一个城市顶点之间的最短路径问题。设计分三个部分，一是建立交通网络图的存储结构；二是解决单源最短路径问题；最后再实现两个城市顶点之间的最短路径问题。

最短路径问题的提出随着计算机的普及以及地理信息科学的发展，GIS 因其强大的功能得到日益广泛和深入的应用。网络分析作为 GIS 最主要的功能之一，在电子导航、交通旅游、城市规划以及电力、通讯等各种管网、管线的布局设计中发挥了重要的作用。而网络分析中最基本和关键的问题是最短路径问题。

最短路径不仅仅指一般地理意义上的距离最短，还可以引申到其他的度量，如时间、费用、线路容量等。相应地，最短路径问题就成为最快路径问题、最低费用问题等。由于最短路径问题在实际中常用于汽车导航系统以及各种应急系统等（110 报警、119 火警以及医疗救护系统），这些系统一般要求计算出到出事地点的最佳路线的时间一般在 1s-3s，在行车过程中还需要实时计算出车辆前方的行驶路线，这就决定了最短路径问题的实现应该是高效率的[36]。最优路径问题不仅包括最短路径问题，还有可能涉及到最少时间问题、最少收费（存在收费公路）问题、或者是几个问题的综合，这时将必须考虑道路级别、道路流量、道路穿越代价（如红灯平均等待时间）等诸多因素。但是必须指出的是，一般来说最优路径在距离上应该是最短的，但最短路径在行驶时间和能源消耗的意义上未必是最优的。其实，无论是距离最短、时间最快还是费用最低，它们的核心算法都是最短路径算法。

2 设计思路

单源最短路径算法的主要代表之一是 Dijkstra（迪杰斯特拉）算法。该算法是目前多数系统解决最短路径问题采用的理论基础，在每一步都选择局部最优解，以期望产生一个全局最优解[40]。

Dijksira 算法的基本思路是：对于图 G=(V,E)，V 是包含 n 个顶点的顶点集，E 是包含 m 条弧的弧集，(v, w)是 E 中从 v 到 w 的弧，c(v, w)是弧(v, w)的非负权值，设 s 为 V 中的顶点，t 为 V 中可由 s 到达的顶点，则求解从 s 至 t 的具有最小弧权值和的最短路径搜索过程如下：

(1) 将 v 中的顶点分为 3 类：已标记点、未标记点、已扫描点。将 s 初始化为已标记点，其它顶点为未标记点。为每个顶点 v 都建立一个权值 d 和后向顶点指针 p，并将 d 初始化如下：d(v)=0，v=s；d(v)=∞，v≠s。

(2) 重复进行扫描操作：从所有已标记点中选择一个具有最小权值的顶点 v 并将其设为已扫描点，然后检测每个以 v 为顶点的弧(v, w)，若满足 d(v) + c(v, w) < d(w) 则将顶点 v 设

为已标记点，并令 d(w) = d(v) + c(v, w), p(w) = v。

(3) 若终点 t 被设为已扫描点，则搜索结束。由 t 开始遍历后向顶点指针 P 直至起点 s，即获得最短路径解。

3 数据结构

（1）定义一个数组 min_dist，它的每个数组元素 min_dist[i]表示当前所找到的从始点 vi 到每个终点 vj 的最短路径的长度。它的初态为：若从 vi 到 vj 有边，则 min_dist[j]为边的权值；否则置 min_dist[i]为∞。定义一个数组 path，其元素 path[k]（0≤k≤n-1）用以记录 vi 到 vk 最短路径中 vk 的直接前驱结点序号，如果 vi 到 vk 存在边，则 path[k]初值为 i。定义一个数组 W，存储任意两点之间边的权值。

（2）查找 min(min_dist[j]，j∈V-S)，设 min_dist[k]最小，将 k 加入 S 中。修改对于 V-S 中的任一点 vj，min_dist[j]=min(min_dist[k]+w[k][j], min_dist[j]) 且 path[j]=k。

（3）重复上一步，直到 V-S 为空。

在算法设计时，用一个 tag 数组来记录某个顶点是否已计算过最短距离，如果 tag[k]=0，则 vk∈V-S，否则 vk∈S。初始值除 tag[i]=1 以外，所有值均为 0。

十二、电网建设造价计算

1 项目简介

假设一个城市有 n 个小区，要实现 n 个小区之间的电网都能够相互接通，构造这个城市 n 个小区之间的电网，使总工程造价最低。请分别用克鲁斯卡尔和普里姆两种算法分别设计一个能满足要求的造价方案。

2 设计思路

在每个小区之间都可以设置一条电网线路，相应的都要付出一点经济代价。n 个小区之间最多可以有 n(n-1)/2 条线路，选择其中的 n-1 条使总的耗费最少。可以用连通网来表示 n 个城市之间以及 n 个城市之间可能设置的电网线路，其中网的顶点表示小区，边表示两个小区之间的线路，赋予边的权值表示相应的代价。对于 n 个顶点的连通网可以建立许多不同的生成树，每一颗生成树都可以是一个电路网。现在，我们要选择总耗费最少的生成树，就是构造连通网的最小代价生成树的问题，一颗生成树的代价就是树上各边的代价之和。

设 G=(V, E)是具有 n 个顶点的网络，T=(U, TE)为 G 的最小生成树，U 是 T 的顶点集合，TE 是 T 的边集合。Prim 算法的基本思想是：首先从集合 V 中任取一顶点（例如去顶点 v0）放入集合 U 中，这时 U={ v0}，TE=NULL。然后找出所有一个顶点在集合 U 里，另一个顶点在集合 V-U 里的边，使权(u, v)(u∈U, v∈V-U)最小，将该边放入 TE，并将顶点 v 加入集合 U。重复上诉操作直到 U=V 为止。这时 TE 中有 n-1 条边，T=(U, TE)就是 G 的一颗最小生成树。

3 数据结构

假设图采用邻接矩阵表示法表示，用一对顶点的下标（在顶点表中的下标）表示一条边，定义如下：

```
typedef struct{
intstart_vex, stop vex;      //边的起点和终点
AdjType weight;          //边的权
}Edge;
```

在构造最小生成树的过程中定义一个类型为 Edge 的数组 mst: Edge mst[n-1]；其中，n 为网络中顶点的个数，算法结束时，mst 中存放求出的最小生成树的 n-1 条边。

可以用带权的无向图（即无向网）表示这 n 个小区之间的电网连接，其中顶点表示小区，权值表示城市之间电网建设的造价，构造一个无向网的最小生成树即是满足要求的最低电网连接造价方案。

十三、软件工程进度规划（难度系数 0.8）

1 项目简介

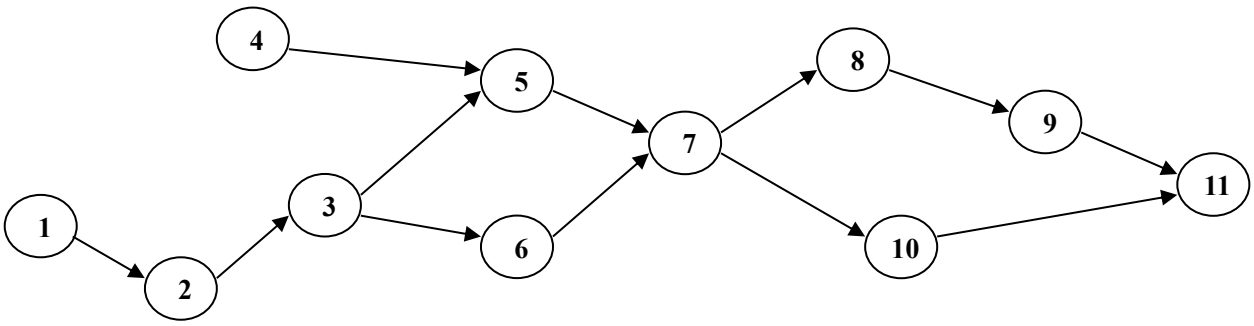
设计一个软件，需要进行用户需求分析、系统需求确认、系统概要设计、设计用例场景、系统的详细设计、数据库详细设计、编码、单元测试、集成测试、系统测试、维护等活动。用户需求分析需要在系统需求确认之前完成，系统的详细设计必须在系统的概要设计、设计系统用例和设计用例场景之前完成。

如表所示，是一系列活动之间的关系。

表系统活动之间的关系

活动代码	活动名称	先需活动
A1	用户需求分析	无
A2	系统需求确认	A1
A3	系统概要设计	A2
A4	设计用例场景	无
A5	系统的详细设计	A3, A4
A6	数据库详细设计	A3
A7	编码	A5, A6
A8	单元测试	A7
A9	集成测试	A8
A10	系统测试	A7
A11	维护	A11

图所示是设计一个软件的 AOV 网示意图。



图软件设计流程的 AOV 网

请设计算法判断该软件设计流程是否有回路，若无请给出该软件设计 AOV 网的拓扑序列。

2 设计思路

拓扑排序(topological sort)是求解网络问题所需的主要算法。管理技术如计划评审技术 PERT(Performance Evaluation And Review Technique)和关键路径法 CPM(Critical Path Method)都应用这一算法。通常，软件开发、施工过程、生产流程、程序流程等都可作为一个工程。一个工程可分成若干子工程，子工程常称为活动(activity)。因此要完成整个工程，必须完成所有的活动。活动的执行常常伴随着某些先决条件，一些活动必须先于另一些活动被完成。

AOV 网络代表的领先关系应当是一种拟序关系，它具有传递性(transitive)和反自反性(irreflexive)。如果这种关系不是反自反的，就意味着要求一个活动必须在它自己开始之前就完成。这显然是荒谬的，这类工程是不可实施的。如果给定了一个 AOV 网络，我们所关心的事情之一，是要确定由此网络的各边所规定的领先关系是否是反自反的，也就是说，该 AOV 网络中是否包含任何有向回路。一般地，它应当是一个有向无环图(DAG)。

一个拓扑序列(topological order)是 AOV 网络中顶点的线性序列，使得对图中任意两个顶点 i 和 j，i 是 j 的前驱结点，则在线性序列中 i 先于 j。

拓扑排序算法可描述如下：

(1) 在图中选择一个入度为零的顶点，并输出之；

(2) 从图中删除该顶点及其所有出边(以该顶点为尾的有向边)；

(3) 重复(1)和(2)，直到所有顶点都已列出，或者直到剩下的图中再也没有入度为零的顶点为止。后者表示图中包含有向回路。

3 数据结构

拓扑排序可以在不同的存储结构上实现，与遍历运算相似，邻接表方法在这里更有效。拓扑排序算法包括两个基本操作：(1)决定一个顶点是否入度为零；(2)删除一个顶点的所有出边。如果我们对每个顶点的直接前驱予以计数，使用一个数组 InDgree 保存每个顶点的入度，即 InDgree[i]为顶点 i 的入度，则基本操作(1)很容易实现。而基本操作(2)在使用邻接表表示时，一般会比邻接矩阵更有效。在邻接矩阵的情况下，必须处理与该顶点有关的整行元素(n 个)，而邻接表只需处理在邻接矩阵中非零的那些顶点。

十四、银行业务活动的模拟

1 项目简介

假设某银行有 4 个窗口对外接待客户，从早晨银行开门起不断有客户进入银行，由于每个窗口在某个时刻只能接待一个客户。因此在客户人数众多时需要在每个窗口前顺次排队，对于刚进入银行的客户。如果某个窗口的业务员正空闲，则可上前办理业务。反之，若每个窗口均有客户所占，他便会排在人数最少的队伍后面。编制一个程序模拟银行的这种业务活动并随时统计当前客户在银行的平均逗留时间。

功能要求：

1).实现数据的输入；

2).各个窗口数据的访问和输出；

3)当前窗口的人员分布情况动态显示。

界面要求：有合理的提示，每个功能可以设立菜单，根据提示，可以完成相关的功能要求。

2 设计思路

银行客户办理业务的流程如下：

① 进入银行之后，首先在四个窗口中选择一个排队等候人数最少的窗口进行排队，该窗口的第一位客户办理完业务之后离开队伍。

② 每位客户接受服务的时长各不相同，因此每个队伍队长和排队时长都不固定。

③ 统计所有客户在银行内的平均逗留时间，就需要在每个客户入队开始计时直至其出队，然后计算所有客户消耗的平均时间。

④ 客户信息可以提前存放于某文件内，这样每内需要新的客户时，直接从文件中依次读取即可。

3 数据结构

四个窗口可以用四个队列进行模拟，队列可以选择链式存储结构，进入银行即入队操作，业队办理完毕后离开银行即出队，耗时指的时入队与出队时间差。因此首先要建立四个队列代表四个窗口，100 个客户信息事先存入到某文件中，根据操作菜单手工操作从文件中依次读取客户信息，即进入银行，进入时选择队长最小的进入，最同记录该客户的入队时间。客户信息除了银行账号，姓名，性别之外，还应该包括进入银行时间和离开银行时间。

队列中除了记录客户之外，还应该能够统计曾经进入过的客户的数量及其滞留的时间。

十五、进销存管理系统

1 项目简介

假设某超市需要一小型管理系统，用于处理进货、销售、盘查库存、统计营业额等功能，以便实时掌握库存信息，及时进货或调整货品。

2 设计思路

进销存管理系统主要包括进货、销售、盘货、统计等功能，对应的操作主要有查找、修改、添加、删除等功能。设计该系统，那么首先存将设计销售商品意味着如果该商品存在且数量满足销售的要求则修改该商品库存数量，进货意味着如果该商品存在就修改增加其数量，如果原来库存中本没有该商品刚需要先在库存在添加上该商品并修改其库存数量，盘货时则需要统计输出每一种商品目前的库存情况，并对于库存少于一定数量（阈值）的商品单独输出预警信息，统计销售额则是要将近期内（可以按订单数量统计，也可以按照日期统计）销售情况，为方便统计在每销售一笔时，应该记录该订单的总价，因此还应该建立另外这个链表用于存放订单信息，查找商品信息可以用链式存储结构实现，

3 数据结构

假设超市中商品种类更新较少，因此商品信息可以用顺序存储结构实现，商品信息中包含编号（主码）、名称、分类、单价、数量等，商品编号可以根据分类等信息自动生成，如001 为首的表示食品类，002 为首的表示日用品类，以此类推，名称与分类均可以是字符串类型，单价为浮点型数据，数量为整型。订单信息几乎在一直添加中，因此订单可以使用链式存储结构，订单信息包括：序号、销售日期、销售时间、总价等，订单序号从 1 开始自动编号，销售日期和时间可以从系统中获取。

十六、活期储蓄账目管理系统

1 项目简介

模拟银行活期储蓄的流程，要求实现储户开户、销户、存入、支出等活动。要求是快速地找到储户的账户，实现存款、取款，还要能比较简单、迅速地实现插入和删除，以实现开户和销户的需要。

主要功能：

（1） 实现储户开户。开户需要填写登记卡，记录相关信息，设置开户余额 0。
（2）实现储户销户。
（3）向某帐户存款。
（4）从某帐户取款。
（5）排序显示所有账户信息，根据帐号进行排序。
（6）查询某帐户余额。
（7）查询某帐户交易记录。
所有账户及其信息存储至文件，程序运行时从从文件中读入

2 设计思路

3 数据结构

3 数据结构

主要功能：

（1） 实现储户开户。开户需要填写登记卡，记录相关信息，设置开户余额 0。

（8）实现储户销户。

（9）向某帐户存款。

（10）从某帐户取款。

（11）排序显示所有账户信息，根据帐号进行排序。

（12）查询某帐户余额。

（13）查询某帐户交易记录。

所有账户及其信息存储至文件，程序运行时从从文件中读入

题目二： 简单的职工管理系统

1. 问题描述

对单位的职工进行管理， 包括插入、 删除、 查找、 排序等功能。

2. 具体功能

职工对象包括姓名、 性别、 出生年月、 工作年月、 学历、 职务、 住址、 电话等信息。

（1） 新增一名职工： 将新增职工对象按姓名以字典方式职工管理文件中。

（2） 删除一名职工： 从职工管理文件中删除一名职工对象。

（3） 查询： 从职工管理文件中查询符合某些条件的职工。

（4） 修改： 检索某个职工对象， 对其某些属性进行修改。

（5） 排序： 按某种需要对职工对象文件进行排序。

3. 要求：

职工对象数不必很多， 便于一次读入内存， 所有操作不经过内外存交换。

（1） 由键盘输 入职工对象， 以文件方式保存。 程序执行时先将文件读入内存。

（2） 对职工对象中的"姓名"按字典顺序进行排序。

（3） 对排序后的职工对象进行增、 删、 查询、 修改、 排序等操作。

4. 选做内容

将职工对象按散列法存储， 并设计解决冲突的方法。 在此基础上实现增、 删、 查询、 修改、 排序等操作。

题目一 同学录管理系统

设计目的：

1.掌握线性表的数据存储。

2.掌握线性表的基本操作。

3.掌握查找的基本算法。

具体功能：

1.对同学基本信息进行录入，应能分类录入不同阶段的同学信息，同学信息包括同学姓名、 身份证号、 联系电话、 qq 号码、 爱好、 联系地址、 邮箱等信息；

2.对同学信息进行查询， 可按同学姓名、 地址、 年龄段等多种方式进行查询；

3.对同学信息进行排序， 可按同学姓名、 年龄、 地址、 qq 号等多种方式查询； ；

4.对同学信息进行修改；

5.对同学信息进行删除；

6.可以对同学的信息进行不同方式的展示。

设计要求：

1.要求用文件实现对信息的保存；

2.选用链表和顺序表均可；