实验一 约瑟夫问题求解

1）**内容：**

约瑟夫(Joseph)问题的一种描述是:编号为1,2,..., n 的n 个人按顺时针方向围坐一圈， 每人持有一个密码(正整数)。一开始选任一个正整数作为报数上限值m， 从第一个人开始按顺时针方向自1开始顺序报数，报到m时停止报数。报m的人出列，将它的密码作为新的m值，再从下个人开始新一轮报数，如此反复，直到剩下最后一人则为获胜者。试设计一个程序求出出列顺序。

2）**要求：**

利用单向循环链表存储结构模拟此过程， 按照出列的顺序印出各人的编号。

3）**测试数据：**

n=7，7 个人的密码依次为：3，1，7，2，4，8，4 。m的初值为20，则正确的出列顺序应为6，1，4，7，2，3，5。

4）**输入输出：**

输入数据：建立输入处理输入数据，输入n输入以及每个人的密码；m的初值。

输出形式：建立一个输出函数，输出正确的序列。

实验二 停车场问题

1）**内容：**

设停车场是一个可停放 n 辆汽车的狭长通道，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，最先到达的在最北端），若停车场内已经停满 n辆车，那么后来的车只能在场外等候，一旦有车开走，则等候在第一位的车即可开入（这是一个队列设长度为m）；当停车场内某辆车需要开出，则在它之后的车辆必须给它让道，当这辆车驶出停车场后，其他车辆按序入栈。每辆车按时间收费。

2）**要求：**

以栈模拟停车场，以队列模拟车场外的便道，按照从终端读入数据的序列进行模拟管理。每一组输入数据包括三个数据：汽车的“到达”（'A'表示）或“离去”（'D'表示）信息，汽车标识（牌照号）以及到达或离去的时刻。对每一组输入数据进行操作后的输出信息为：若是车辆到达，则输出汽车在停车场内或者便道上的停车位置；若是车辆离去，则输出汽车在停车场停留的时间和应缴纳的费用（便道上不收

费）。栈以顺序结构实现，队列以链表结构实现。

3）**测试数据：**

设 n＝3，m=4,停车价格为p=2。输入数据为：

('A',101,5),('A',102,10),('D',101,15),('A',103,20),('A',104,25),('A',105,30),('D',102,35),('D',104,40),('E',0,0)。其中'A'表示到达，'D'表示离开，'E'表示结束。时间为相对分钟数。

4）**输入输出：**

输入数据：程序接受5个命令，分别是:到达('A',车牌号,时间)；离去('D',车牌号,时间)；停车场(P,0,0)显示停车场的车；候车场(W,0,0)显示候车场的车;退出(E,0,0)退出程序。

输出数据：对于车辆到达，要输出汽车在停车场内或者便道上的停车位置；对于车辆离去，则输出汽车在停车场停留的时间和应缴纳的费用（便道上不收费）。

实验三 关键词检索

1）**内容：**

实现类似Unix下grep命令的程序。在一个文件中查找某个关键词，并把出现该关键词的行及行号显示出来。

2）**要求：**

使用C语言的字符串存储结构来实现字符串的操作,编写函数index实现在一个传中查找子串的功能。然后从文件中每次读入一行，作为一个主串看待，然后查找是否存在待查找的关键词（子串），如果有则显示该行内容及行号，否则继续处理下一行。

3）**测试数据：**

任意一个文本文件，文件中任意一词语作为关键词。

4）**输入输出：**

输入数据：屏幕输入或命令行给出文本文件名、关键词。

输出数据：屏幕输出文本文件中出现关键词的行及行号。

实验四 huffman编解码

1）**内容：**

利用 Huffman 编码进行通信可以大大提高信道的利用率，缩短信息传输时间，降低传输成本。但是，这要求在发送端通过一个编码系统对待传数据进行预先编码，在接收端进行解码。对于双工信道（即可以双向传输信息的信道），每端都需要一个完整的编/解码系统。

2）**要求：**

一个完整的huffman编解码系统应该具有以下功能：

初始化（**I**nitialization）。从终端读入字符集大小n，以及n个字符和n个权值，建立Huffman 树，并将它存入hfmTree 中。

编码（**E**ncoding）。利用已经建好的Huffman树（如果不在内存，则应从文件hfmTree中读取），对文件ToBeTran中的正文进行编码，然后将结果存入文件CodeFile中。

解码（**D**ecoding）。利用已经建立好的Huffman树将文件CodeFile中的代码进行解码，结果存入TextFile中。

打印代码文件（Print）。将文件CodeFile以紧凑的格式显示在终端上，每行 50 个代码。同时将此字符形式的编码文件写入文件CodePrint中。

打印Huffman树（**T**ree Printing）。将已经在内存中的Huffman树以直观的形式（树或者凹入的形式）显示在终端上，同时将此字符形式的Huffman 树写入文件TreePrint中。

3）**测试数据：**

用下表给出的字符集和频度的实际统计数据建立Huffman树，并对以下报文进行编码和译码：“THIS PROGRAM IS MY FAVORITE”。

字符		A	B	C	D	E	F	G	H	I	J	K	L	M
频度	186	64	13	22	32	103	21	15	47	57	1	5	32	20
字符	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
频度	57	63	15	1	48	51	80	23	8	18	1	16	1	

4）**输入输出：**

- 字符集大小 n，n个字符和 n个权值均从终端读入，初始化后的huffman树存储在hfmTree文件中，待编码文件为ToBeTran,编码结果以文本的方式存储在文件

CodeFile中，解码文件存在TextFile中,打印的编码和赫夫曼树分别存储在CodePrint和TreePrint文件中。

☞ 用户界面可以设计为“菜单”方式：显示上述功能符号，再加上一个退出功能“Q”，表示退出（quit）。用户键入一个选择功能符，此功能执行完毕后再显示此菜单，直至某次用户选择了 Q为止。

实验五 管道铺设施工的最佳方案

1）**内容：**

需要在某个城市n个居民小区之间铺设煤气管道，则在这n个居民小区之间只需要铺设n-1条管道即可。假设任意两个小区之间都可以铺设管道，但由于地理环境不同，所需要的费用也不尽相同。选择最优的方案能使总投资尽可能小，这个问题即为求无向网的最小生成树。

2）**要求：**

在可能假设的m条管道中，选取n-1条管道，使得既能连通n个小区，又能使总投资最小。每条管道的费用以网中该边的权值形式给出，网的存储采用邻接表的结构。

3）**测试数据：**

使用下图给出的无线网数据作为程序的输入，求出最佳铺设方案。右侧是给出的参考解。

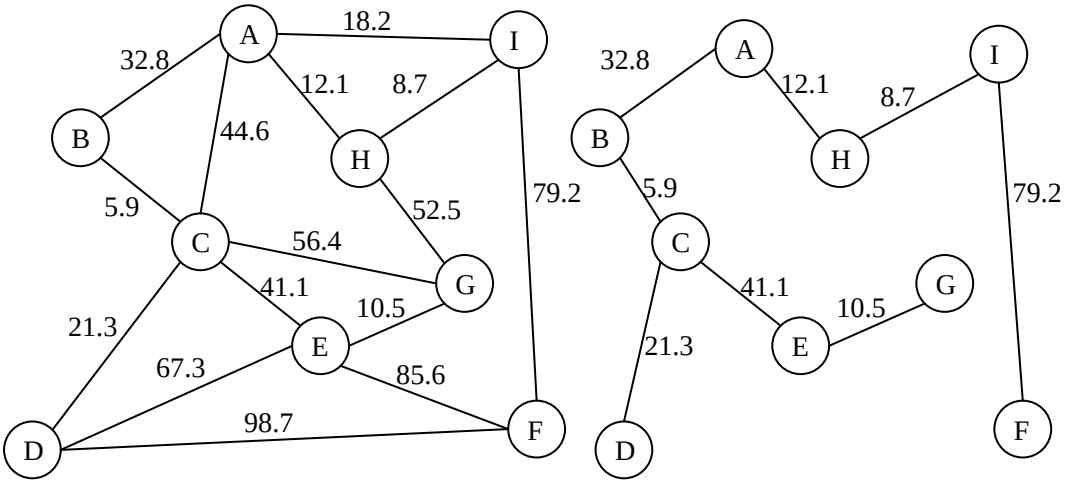


图3.2 小区煤气管道铺设网及其参考解

4）**输入输出：**

参考示例中图的创建方式，从键盘或文件读入上图中的无向网，以顶点对(i,j)的形式输出最小生成树的边。

实验六　利用哈希表统计两源程序的相似性

1）**内容：**

对于两个 C 语言的源程序清单，用哈希表的方法分别统计两程序中使用C语言关键字的情况，并最终按定量的计算结果，得出两份源程序的相似性。

2）**要求与提示：**

C 语言关键字的哈希表可以自建，也可以采用下面的哈希函数昨晚参考：

Hash(key)=(key第一个字符序号*100+key最后一个字符序号)%41

表长m取43。此题的工作主要是扫描给定的源程序，累计在每个源程序中C语言关键字出现的频度。为保证查找效率，建议自建哈希表的平均查找长度不大于2。 扫描两个源程序所统计的所有关键字不同频度， 可以得到两个向量。如下面简单的例子所示：

关键字	void	int		for	char		if	else		while
程序1中关键字频度	4	3		4	3		7	0		2
程序2中关键字频度	4	2		5	4		5	2		1
哈希地址	0	1	2	3	4	5	6	7	8	9

根据程序1和程序2中关键字出现的频度，可提取到两个程序的特征向量X1和X2，其中

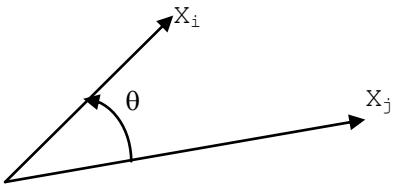
X1=(4 3 0 4 3 0 7 0 0 2)^T

X2=(4 2 0 5 4 0 5 2 0 1)^T

一般情况下，可以通过计算向量Xi和Xj的相似值来判断对应两个程序的相似性，相似值的判别函数计算公式为：

$$S(X_i, X_j) = \frac{X_i^T X_j}{|X_i| \cdot |X_j|} \qquad (3\text{-}1)$$

其中，$|X_i|=\sqrt{X^T\cdot X_i}$。$S(X_i,X_j)$的值介于[0,1]之间，也称广义余弦，即$S(X_i,X_j)=COS\theta$。$X_i=X_j$ 时，显见$S(X_i,X_j)=1$，$\theta=0$；X_iX_j 差别很大时，$S(X_i,X_j)$



接近0，θ接近π/2。如X1=(1 0)^T,X2=(0 1)^T,则$S(X_i,X_j)=0$，$\theta=\pi/2$。可以用下面的二维的图示来直观地表示向量的相似程度：

图3.3 向量相似度示意图

有些情况下，还需要做进一步的考虑，如下图所示：

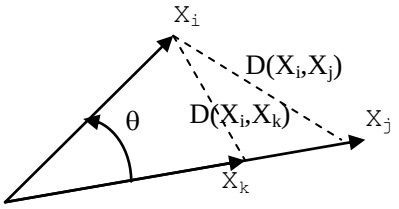


图3.4 向量几何距离

从图中看出， 尽管$S(X_i,X_j)$和$S(X_i,X_k)$的值是一样的，但直观上X_i与X_k更相似。因此当S值接近1 的时候，为避免误判相似性（可能是夹角很小，模值很大的向量），应当再次计算之间的“几何距离” $D(X_i,X_k)$。其计算公式为：

$$D(X_i,X_k)=|X_i-X_k|=\sqrt{(X_i-X_k)^T(X_i-X_k)} \qquad (3\text{-}2)$$

最后的相似性判别计算可分两步完成：

第一步 用式（3-1）计算S，把接近 1的保留，抛弃接近0的情况（把不相似的排除）；

第二步 对保留下来的特征向量，再用式（3-2）计算D，如D值也比较小，说明两者对应的程序确实可能相似（慎重肯定相似的）。

S和D的值达到什么门限才能决定取舍？需要积累经验，选择合适的阈值。

3） **测试数据：**

做几个编译和运行都无误的C程序，程序之间有相近的和差别大的，用上述方法求S，并对比差异程度。

4）**输入输出：**

输入为若干个C源程序，输出为程序间的相似度以及向量的几何距离。