《数据结构课程设计》

课程名称：数据结构

题　　目：中医院导航系统

学　　院：计算机与信息科学学院

专业年级：计算机科学与技术(3+2)班

小组组长：吴炳燃

小组成员：杨若曦 罗正豪 周敬鑫 周勋海

指导教师：张万里　职称：讲师

2022 年 1 月 6 日

目录

中医院导航系统

一、 设计背景与问题描述

1.1 设计背景

随着计算机性能的不断提高，计算机技术的应用已经涉足到了各种行业，使用计算机进行医院信息管理，已经成为现代化医院运营过程中不可或缺的基础设施与技术支撑环境。医院信息系统简称 HIS（Hospital Information System），是指利用计算机和网络通信技术与设备，为医院所属各部门提供病人诊疗提供便利。

1.2 问题描述

建设数字化医院是医院管理信息系统发展的必然，也是医院现代化管理和高效运行的需要。随着医疗体制改革的不断深入，医疗市场的竞争愈加激烈，因此，改变医院管理模式，提升医院形象，增强医院核心竞争力变得迫在眉睫。从实践过程看，医院信息化管理可以带来许多好处。本系统能帮助用户查询医院的相关科室信息，也可以帮助用户查询各个科室之间可行的路线及其最短路径，通过该系统可以使用户快速了解医院的相关信息，以便于快速进行就医。

二、需求分析

2.1 任务目标

搞清楚医院系统在 C 程序中利用源代码是如何实现的,用到哪些数据结构,以及它的一些设计思路和科学的设计流程。

2.2 功能分析

根据程序模块划分分析，程序应该实现以下功能：

1. 进入医院系统程序

运行系统后，进入系统主页面，用户可根据需要选择相关功能进行体验。

2. 医院地点介绍

此功能可以列出医院个地点的详细信息供用户查阅，各地点有相应编号，以便用户使用后续功能。

3. 查看路线

用户可自行选择地点查询其路线，输入地点相应编号后，可以给出该地点能到达的全部路线。

4. 查询地点间最短路径

用户可自行选择两个地点查询其最短路径，输入地点相应编号后，系统会给出到达目的地的最短路径。

5. 地点信息查询

若用户对某个地点感兴趣，可输入其编号，系统会自行给出相应地点的相关信息。

6. 更改图信息

若发觉地点相关信息并不全面或是需要进行更改，或是想要更改图的相关信息，可选择此项进行更改。

7. 查询两地间可行路径

用户可自行选择两个地点查询其全部可行路径，输入相应地点编号后，系统会给出全部可行路径。

8. 打印邻接矩阵

可给出该系统的邻接矩阵。

9. 退出

选择该项后即可退出此系统。

2.3 小组分工

框架设计：由吴炳燃进行构思，同时参考其他人的意见。

系统设计：由周敬鑫进行框架的编写。同时参考其他人的意见。

程序设计：由周勋海进行各项功能的编辑。同时参考其他人的意见。

程序调试：由杨若曦进行程序的调试与调整。同时参考其他人的意见。

文档制作，由罗正豪完成文档的编辑工作。同时参考其他人的意见。

三、开发环境与设计思路

3.1 设计思路

用 C 进行编写，使用了邻接矩阵和邻接表，迪杰斯特拉求单源最短路算法，深度优先和广度优先计算距离，还有包括整个框架的构思，图等等的运用，最终构成了整个程序。

3.2 开发环境

系统软件：Windows 操作系统

软件环境：Dev c++

四、总体设计

4.1 医院平面图

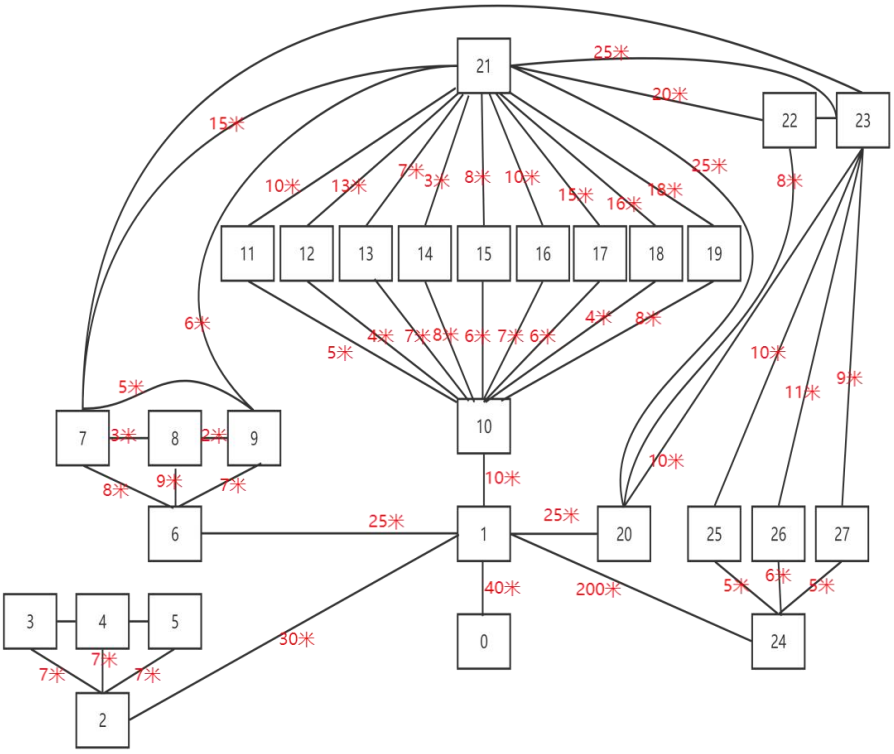

图 1 医院平面图

0:大门;1:门诊大厅;2:住院部;3:重症监护室;4:产科室;5:普通室;6:急诊部;7:手术室;8:B 超;9:放射科;10:门诊部;11:内科;12:外科;13:儿科;14:妇科;15:眼科;16:耳鼻喉科;17:口腔科;18:皮肤科;19:中医科;20:行政楼;21:大夫办公室;22:主任办公室;23:院长办公室;24:后勤室;25:食堂;26:药房;27:病案室。

4.2 软件框架图

本系统由一个导航系统（主系统）组成，其中导航系统具体功能如图 2 所示：

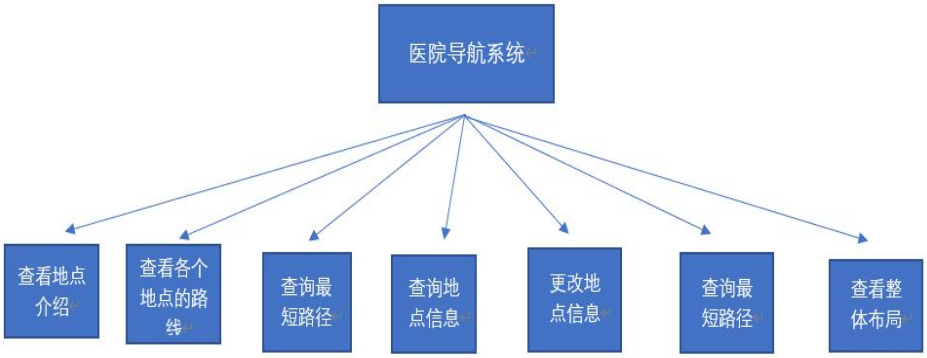

图 2 导航系统功能模块

本系统用法流程图如图 3 所示。

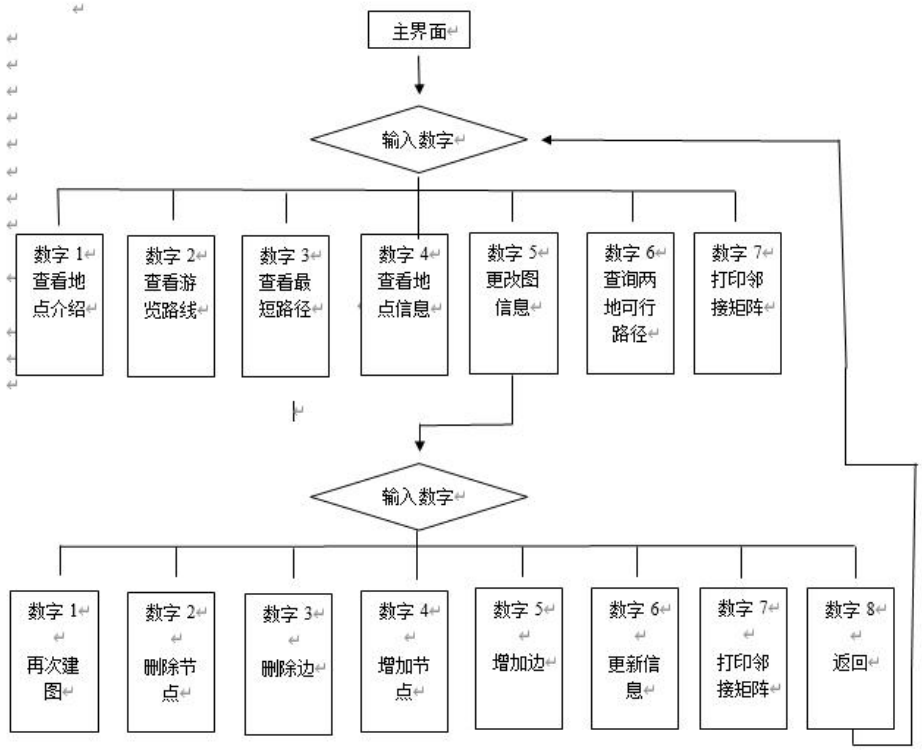

图 3 用法流程图

4.3 主界面设计

为了实现医院系统各功能的管理，首先设计一个含有多个菜单项的主控菜单子程序以链接系统的各项子功能，方便用户使用本系统。本系统主控菜单运行界面如图 6 所示：

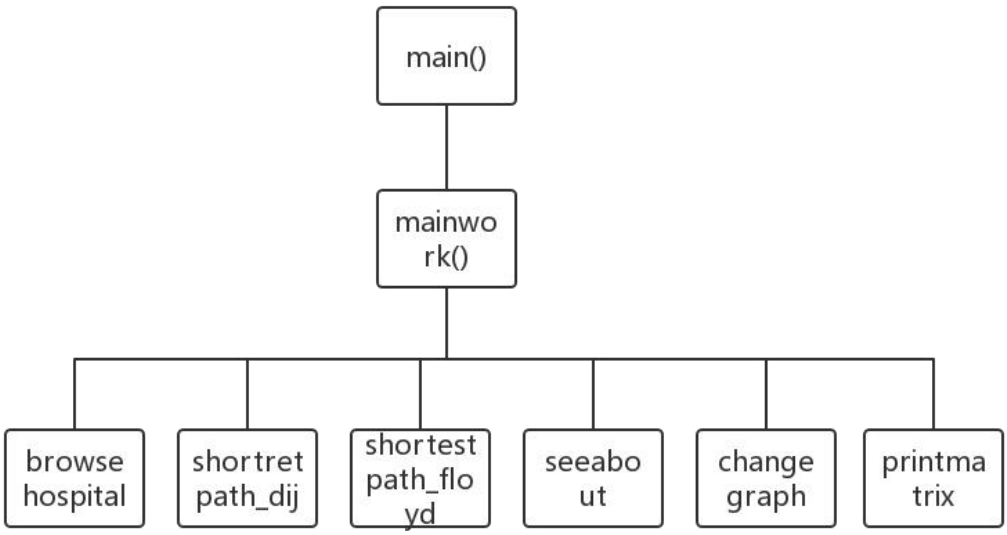

图 4 软件框架

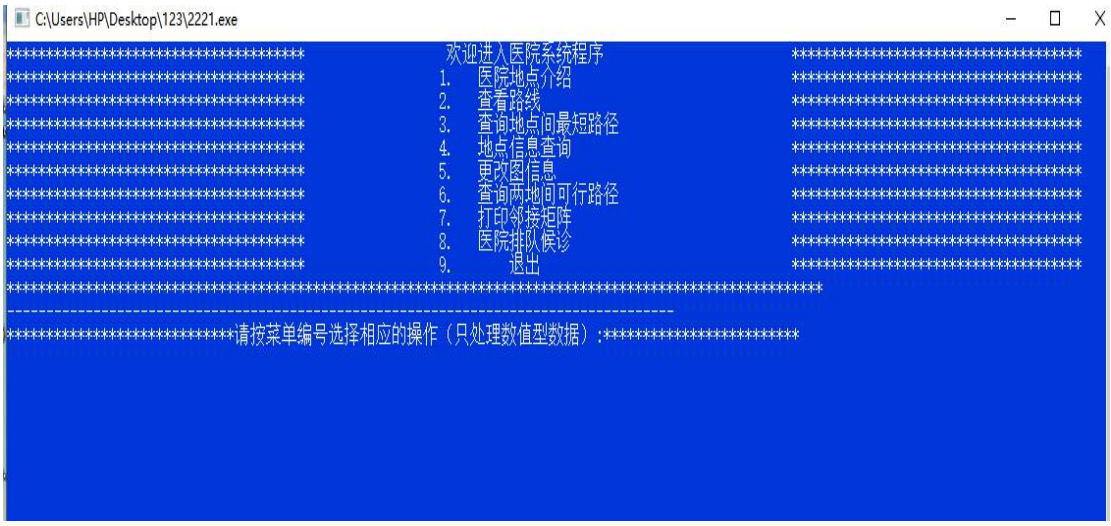

图 5 主系统界面

4.4 存储结构设计

本系统采用图结构类型存储抽象医院图的信息。其中各科室间的邻接关系用

图的邻接矩阵类型来存储；科室信息用结构数组存储，其中每个数组元素是一个结构变量，包括科室编号、科室名称及科室介绍三个分量；图的顶点个数及边的个数由分量 vexnum、arcnum 表示，它们是整型数据。此外，本系统还设置了三个全局变量：visited[]数组用于存储顶点是否被访问标志；d[]数组用于存放边上的权值或存储查找路径顶点的编号；hospital 是一个图结构的全局变量。

五、详细设计

5.1 系统子程序及功能简介

本系统共设置 18 个子程序，各子程序的函数名及功能说明如下。

（1）maraph initgraph()　　//图的初始化

（2）Int locatevex（mgraph c,int v）　　//查找景点在图中的序号

```
for(i=0;i<c.vexnum ;i++)
    if(v==c.vexs[i].position)
        return i;        //找到，返回顶点序号 i

return -1;              //否则，返回-1
```

（3）Void path（mgraph c，int m,int n,int k）　　//打印序号为（m,n）景点间的长度不度不超过 8 个景点的路径

```
  int t=k+1;             //t 记载路径上下一个中间顶点在 d[]数组中的下标
  if(d[k]==n && k<6)     //d[k]存储路径顶点。若 d[k]是终点 n 且景点个数<=6，则输出该路径
```

（4）Int allpath（mgraph c）　//打印两景点间的景点个数不超过 8 的所有路径，调用（3）

```
m=locatevex(c,i);        //调用(2)，确定该顶点是否存在。若存在，返回该顶点编号
path(c,m,n,0);      //调用(3)。k=0，对应起点 d[0]==m。k 为 d[]数组下标
```

（5）void shortestpath_dij（mgraph c）　//用 Dijkstra 算法，求一个景点到其他景间的最短路径，并打印

以下编号（6）——（12）是图的基本操作。包括重建图，更新信息，删除，增加结点和边等。

（6）int creatgragh（mgraph *c） //建图。以图的邻接矩阵存储图，返回值：1 或-1

for(i=0;i<c->vexnum ;i++) //构造顶点向量(数组)

for(i=1;i<=c->arcnum ;i++) //构造邻接矩阵

（7）int newgraph（mgraph *c）//更新图的部分信息。返回值：1

int changenum; //计数。用于记录要修改的对象的个数

（8）int enarc（mgraph *c） //增加一条边。返回值：1

（9）int envex（mgraph *c） //增加一个结点。返回值：1

for(i=0;i<c->vexnum;i++) //对原邻接矩阵新增加的一行及一列进行初始化

c->arcs [c->vexnum -1][i].adj=Infinity; //最后一行(新增的一行)

c->arcs [i][c->vexnum -1].adj=Infinity; //最后一列(新增的一列)

（10）int delvex（mgraph *c） //删除图的一个顶点。返回值：1

for(i=m;i<c->vexnum-1 ;i++)//对顶点信息所在顺序表进行删除 m 点的操作

strcpy(c->vexs[i].name ,c->vexs [i+1].name);

strcpy(c->vexs[i].introduction ,c->vexs [i+1].introduction); //对原邻接矩阵，删除该顶点到其余顶点的邻接关系。分别删除相应的行和列

（11）int delarc（mgraph *c） //删除图的一条边。返回值：1

（12）void printmatrix（mgraph c） //删除图的邻接矩阵

（13）int changegraph（mgraph *c） //图操作的主调函数。返回值：1

（14）void shortestpath_floyd（mgraph c） //用 Floyd 算法求任意两景点间的最短路径，并输出//用 floyd 算法求各对顶点 v 和 w 间的最短路经及其带权长度的 d[v][w]。若 p[v][w][u]==1；则 u 是 v 到 w 的当前求得的最短路经上的顶点

（15）void seeabout（mgraph c） //查询景点的信息

（16）void browsecompus（mgraph c） //显示所有景点信息

（17）void mainwork（） //工作区函数。操作区用户界面

（18）void mian（） //主函数。设定界面的颜色和大小，调用工作区

5.2 弗洛伊德（Floyd）算法简介

在本次课程设计中我们大量使用到了 Floyd 这个算法。

在计算机科学中，Floyd-Warshall 算法是一种在具有正或负边缘权重（但没有负周期）的加权图中找到最短路径的算法。算法的单个执行将找到所有顶点对之间的最短路径的长度（加权）。 虽然它不返回路径本身的细节，但是可以通过对算法的简单修改来重建路径。 该算法的版本也可用于查找关系 R 的传递闭包，或（与 Schulze 投票系统相关）在加权图中所有顶点对之间的最宽路径。

Floyd 算法的基本思想：可以将问题分解:第一、先找出最短的距离，第二、然后在考虑如何找出对应的行进路线。如何找出最短路径呢？这里还是用到动态规划的知识，对于任何一个城市而言，i 到 j 的最短距离不外乎存在经过 i 与 j 之间经过 k 和不经过 k 两种可能，所以可以令 k=1，2，3，...，n(n 是城市的数目)，在检查 d(ij)与 d(ik)+d(kj)的值；在此 d(ik)与 d(kj)分别是目前为止所知道的 i 到 k 与 k 到 j 的最短距离，因此 d(ik)+d(kj)就是 i 到 j 经过 k 的最短距离。所以，若有 d(ij)>d(ik)+d(kj)，就表示从 i 出发经过 k 再到 j 的距离要比原来的 i 到 j 距离短，自然把 i 到 j 的 d(ij)重写为 d(ik)+d(kj)，每当一个 k 查完了，d(ij)就是目前的 i 到 j 的最短距离。重复这一过程，最后当查完所有的 k 时，d(ij)里面存放的就是 i 到 j 之间的最短距离了。

Floyd 可以说是 Warshall 算法的扩展了，三个 for 循环便可以解决一个复杂的问题，应该说是十分经典的。从它的三层循环可以看出，它的复杂度是 n3，除了在第二层 for 中加点判断可以略微提高效率，几乎没有其他办法再减少它的复杂度，下面是我们的 Floyd 算法的流程图：（如图 6）

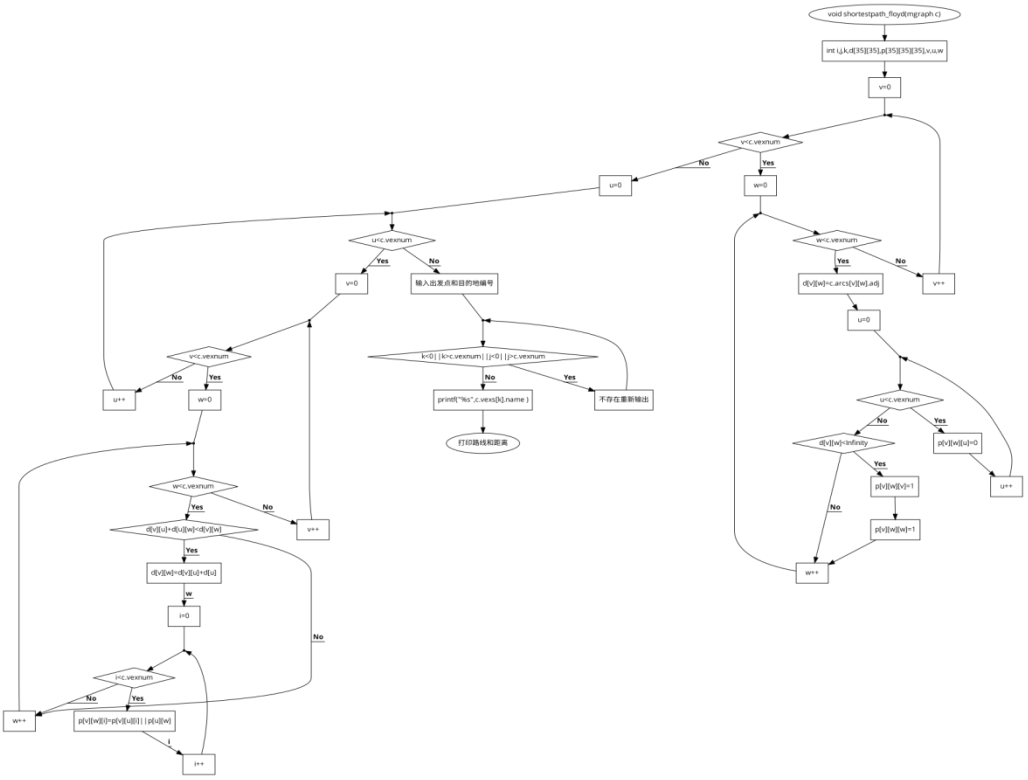

图 6 Floyd 算法的流程图

5.3 迪杰斯特拉算法(Dijkstra)简介

在本次课程设计中我们大量使用到了 Dijkstra 这个算法。

迪杰斯特拉算法(Dijkstra)是由荷兰计算机科学家狄克斯特拉于 1959 年提出的，因此又叫狄克斯特拉算法。是从一个顶点到其余各顶点的最短路径算法，解决的是有权图中最短路径问题。迪杰斯特拉算法主要特点是从起始点开始，采用贪心算法的策略，每次遍历到始点距离最近且未访问过的顶点的邻接节点，直到扩展到终点为止。

通过 Dijkstra 计算图 G 中的最短路径时，需要指定起点 s(即从顶点 s 开始计算)。此外，引进两个集合 S 和 U。S 的作用是记录已求出最短路径的顶点(以及相应的最短路径长度)，而 U 则是记录还未求出最短路径的顶点(以及该顶点到起点 s 的距离)，初始时，S 中只有起点 s；U 中是除 s 之外的顶点，并且 U 中顶点的路径是"起点 s 到该顶点的路径"。然后，从 U 中找出路径最短的顶点，并将其加入到 S 中；接着，更新 U 中的顶点和顶点对应的路径。 然后，再从 U 中找出路径最短的顶点，并将其加入到 S 中；接着，更新 U 中的顶点和顶点对应的路径。 ... 重复该操作，直到遍历完所有顶点。

下面是我们的 Dijkstra 算法的流程图：（如图 7）

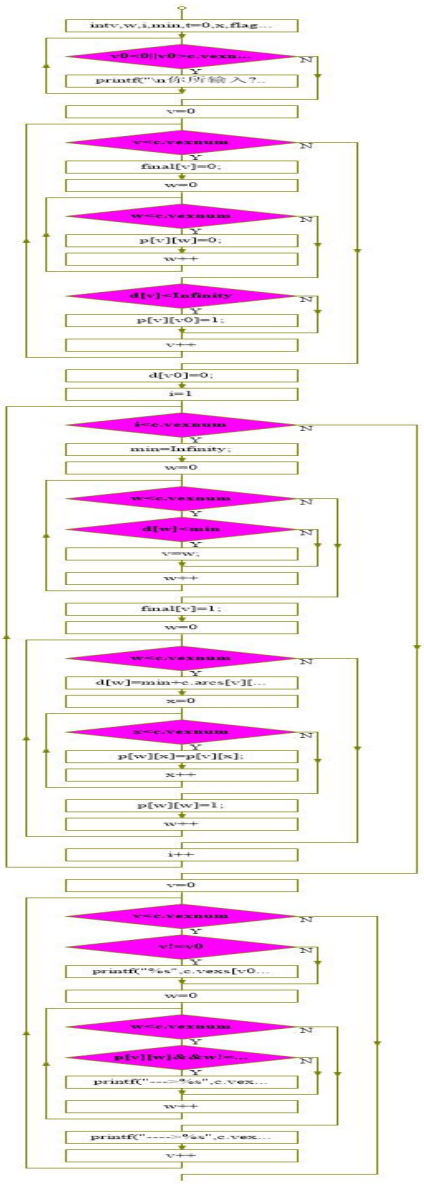

图 7

核心代码：

杰斯特拉算法，求从顶点 v0 到其余顶点的最短路经及其带权长度 d[v]，若 p[v][w]为 1，则 w 是从 v0 到 v 的最短路经上的顶点，final[v]类型用于设置访问标志。vo 为起始景点的编号。

```
int v,w,i,min,t=0,x,flag=1,v0;                  //vo为起始景点的编号
int final[35],d[35],p[35][35];
printf("\n请输入一个起始景点的编号: ");
scanf("%d",&v0);
printf("\n\n");
while(v0<0||v0>c.vexnum)
{
    printf("\n你所输入的景点编号不存在\n");
    printf("请重新输入: ");
    scanf("%d",&v0);
}//while
for(v=0;v<c.vexnum ;v++)
{
    final[v]=0;                                 //初始化各顶点访问标志
    d[v]=c.arcs[v0][v].adj;                     //v0 到各顶点 v 的权值赋值给d[v]
    for(w=0;w<c.vexnum ;w++)                    //初始化p[][]数组，各顶点间的路径全部设置为空路径0
        p[v][w]=0;
    if(d[v]<Infinity)                           //v0 到v 有边相连，修改p[v][v0]的值为1
    {
        p[v][v0]=1;
        p[v][v]=1;                              //各顶点自己到自己要连通
    }
}//for
```

自己到自己的权值设为 0，v0 的访问标志设为 1，v 属于 s 集，对其余 c.vexnum-1 个顶点 w，依次求 v 到 w 的最短路径，在未被访问的顶点中，查找与 v0 最近的顶点 v，v0 到 w (有边)的权值<min。

```
    d[v0]=0;                                                    //自己
    final[v0]=1;                                                //v0
    for(i=1;i<c.vexnum ;i++)                                    //对
    {
        min=Infinity;
        for(w=0;w<c.vexnum ;w++)                                //在
            if(!final[w])
                if(d[w]<min)                                    //v0
                {
                    v=w;
                    min=d[w];
                }//if
        final[v]=1;                                             //
        for(w=0;w<c.vexnum ;w++)                                //
            if(!final[w]&&(min+c.arcs[v][w].adj <d[w]))
            {
                d[w]=min+c.arcs[v][w].adj;
                for(x=0;x<c.vexnum ;x++)                        //
                    p[w][x]=p[v][x];
                p[w][w]=1;
            }//if
    }//for
```

v 的访问标志设置为 1，v 属于 s 集，修改 v0 到其余各顶点 w 的最短路径权值 d[w]，若 w 不属于 s，且 v 到 w 有边相连，修改 v0 到 w 的权值 d[w]，所有 v0 到 v 的最短路径上的顶点 x，都是 v0 到 w 的最短路径上的顶点。

```
        final[v]=1;                                             //
        for(w=0;w<c.vexnum ;w++)                                //
            if(!final[w]&&(min+c.arcs[v][w].adj <d[w]))
            {
                d[w]=min+c.arcs[v][w].adj;
                for(x=0;x<c.vexnum ;x++)                        //
                    p[w][x]=p[v][x];
                p[w][w]=1;
            }//if
```

5.4 模块功能说明

本系统除了要完成图的初始化功能外还设置了 9 个子功能菜单。图的初始化由函数 initgraph 实现。依据读入的图的顶点个数和边的个数，分别初始化图结构中图的顶点向量数组和图的邻接矩阵。9 个子功能的设计描述如下：

（1）医院地点介绍

医院地点介绍由函数 browsehospital 实现。当用户选择该功能，系统即能输出医院全部地点的信息：包括地点编号、地点名称、地点简介。具体页面如图 8 所示。

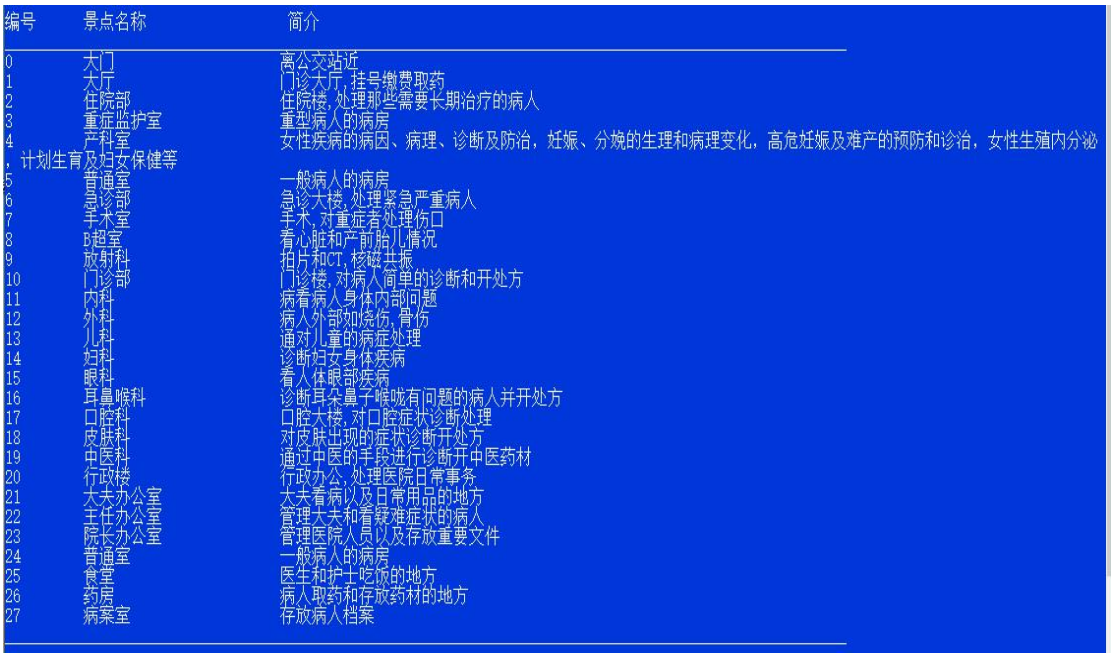

图 8 地点介绍页面

（2）查看浏览路线

查看浏览路线由函数 shortretpath_dij 实现。该功能采用迪杰斯特拉（Dijkstra）算法实现。Dijkstra 算法是用来求任意两个顶点之间的最短路径。在该实验中，我们用邻接矩阵来存储图。在该程序中设置一个二维数组来存储任意两个顶点之间的边的权值。当用户选择该功能，系统能根据用户输入的起始地点编号，求出从该地点到其它地点的最短路径线路及距离。具体页面如图 9 所示。

图 9 路线浏览页面

（3）查询地点间最短路径

查询地点间最短路径由函数 shortretpath_floyd 实现。当用户选择改功能，系统能根据用户输入的起始地点及目的地点编号，查询任意两个地点之间的最短路径线路及距离。迪杰斯特拉算法，求从顶点 v0 到其余顶点的最短路经及其带权长度 d[v]若 p[v][w]为 1，则 w 是从 v0 到 v 的最短路经上的顶点 final[v]类型用于设置访问标志。具体页面如图 10 所示。

图 10 最短路径页面

（4）地点信息查询

地点信息查询由函数 seeabout 实现。该功能根据用户输入的地点编号输出该地点的相关信息。如地点编号、名称等。具体页面如图 11 所示

图 11 地点信息页面

（5）更改图的信息

更改图的信息功能由主调函数 changegraph 及若干子函数完成，可以实现图的若干基本操作。如增加新的地点、删除边、更新信息等。具体页面如图 12 所示。

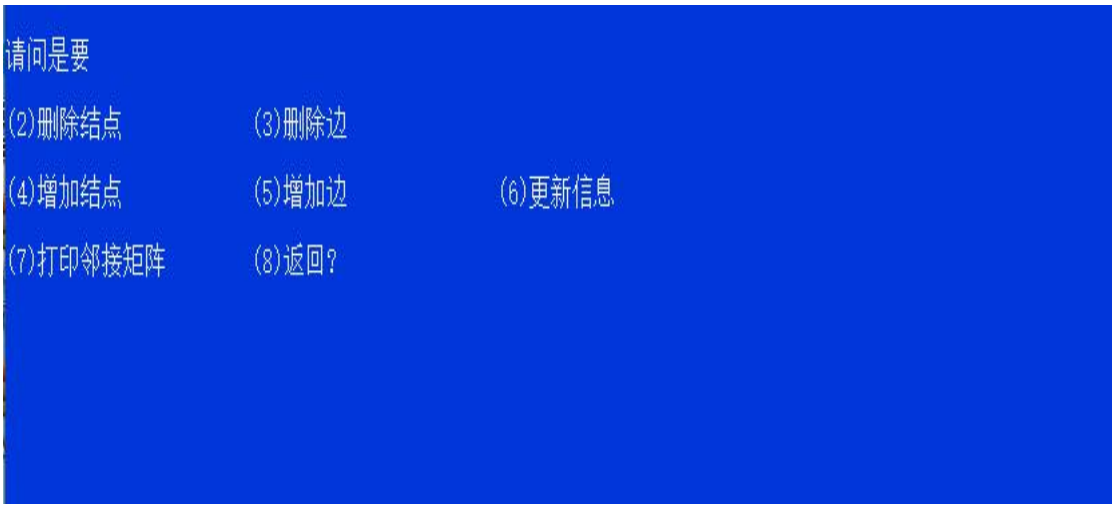

图 12 更改图页面

增加一个结点：对原邻接矩阵新增加的一行及一列进行初始化

```
int envex(mgraph *c)
{
    int i;
    printf("请输入你要增加结点的信息：");
    printf("\n编号：");
    scanf("%d",&c->vexs[c->vexnum ].position );
    printf("名称：");
    scanf("%s",c->vexs[c->vexnum ].name );
    printf("简介：");
    scanf("%s",c->vexs[c->vexnum ].introduction) ;
    c->vexnum ++;
    for(i=0;i<c->vexnum;i++)                    //对原邻接矩阵新增加的一行及一列进行初始化
    {
        c->arcs [c->vexnum -1][i].adj=Infinity;          //最后一行(新增的一行)
        c->arcs [i][c->vexnum -1].adj=Infinity;          //最后一列(新增的一列)
    }
    return 1;
```

删除图的一个顶点：

对顶点信息所在顺序表进行删除 m 点的操作

```
for(i=m;i<c->vexnum-1 ;i++)//对顶点信息所在顺序表进行删除m 点的操作
{
    strcpy(c->vexs[i].name ,c->vexs [i+1].name );
    strcpy(c->vexs[i].introduction ,c->vexs [i+1].introduction );
}
```

对原邻接矩阵，删除该顶点到其余顶点的邻接关系。分别删除相应的行和列

```
for(i=m;i<c->vexnum-1 ;i++)                 //行
    for(j=0;j<c->vexnum ;j++)               //列
        c->arcs [i][j]=c->arcs [i+1][j];    //二维数组，从第m+1行开始依次往前移一行。即删除第m 行
for(i=m;i<c->vexnum-1 ;i++)
    for(j=0;j<c->vexnum ;j++)
        c->arcs [j][i]=c->arcs [j][i+1];    //二维数组，从第m+1列开始依次往前移一列。即删除第m 列
c->vexnum --;
return 1;
```

（6）查询地点间可行路径

查询地点间所有可行路径由函数 allpath 和函数 path 实现，其中 path 函数是直接递归函数。由于是无向网，如果网中的边数很多，任意两个地点之间的所有路径也会有很多，但很多路径是无实际意义的（有近路，为什么要走远路呢）。所以本算法在求得的两地点间所有可行路径中，限制只输出路径长度不超过 8 个地点的路线。具体页面如图 13 所示。

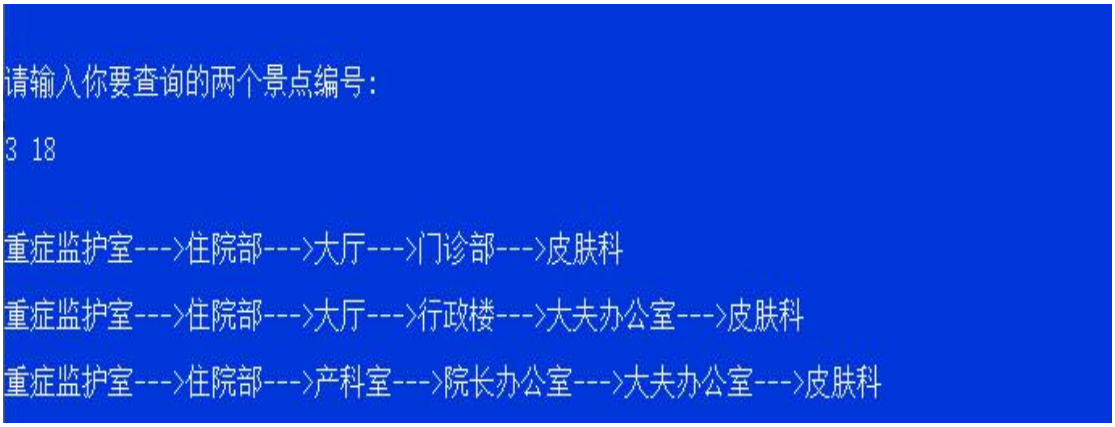

图 13 查询可行地点页面

（7）打印邻接矩阵

打印邻接矩阵即输出图的邻接矩阵的值，由函数 printmatrix 实现。具体页面如图 14 所示。

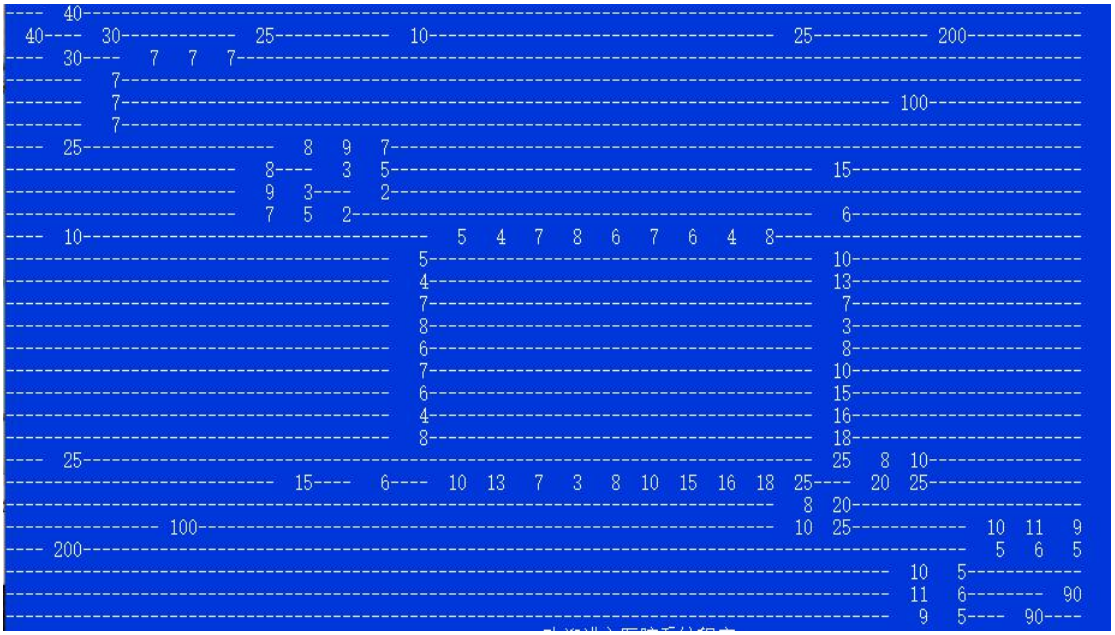

图 14 打印邻接矩阵页面

（8）退出

退出即退出医院系统，由函数 exit（0）实现。

六、展望

我们在医院的导航功能方面已经做得比较的完善了，但是现在面临了一个巨大的问题，也就是一个系统仅仅就只有导航相关的功能，在目前，科技水平高速发展的当今设计毫无竞争力，所以我们一致决定在后续我们增加挂号，等等其他的与医院系统相关的功能。

所以我们会在之后的时间里改进此程序，提升这个程序各个方面的能力，从而也大大的提升我们自身的能力。

七、心得体会

在数据结构课程设计这门课程中，我们通过小组组合，做出了该门课程的课设报告。我们组做的是医院系统，在初步了解一般医院系统结构后，上网查找相关知识，运用所学，通过各个组员在这一星期的不断尝试，最终完成了我们的任务。这次课设，带给了我很多东西，望在以后的学习过程中，尽力和同学互帮互助，最终使对方都有收获，达到共赢。

参 考 文 献

[1] 胡学钢. 数据结构算法设计指导[M]. 清华大学出版社, 1999.
[2] 耿国华数据结构一用 C 语言描述.北京:高等教育出版社，2011.
[3]王红梅，胡明，王涛. 数据结构(C++版)． 北京:清华大学出版社，2011.
[4]潘金贵. 现代计算机常用数据结构和算法[M]. 南京大学出版社, 1994.
[5]王晓东. 数据结构与算法设计[M]. 机械工业出版社, 2012.
[6]谭浩强 .C 程序设计［Z］.北京：清华大学出版社，2001.
[7]严蔚民，吴伟民. 数据结构（C 语言版). 北京：清华大学出版社，20011.
[8]薛万鹏.C 语言程序设计教程［Z］.北京：机械工业出版社，2000.
[9]严蔚敏，陈文博数据结构及应用算法教程(修订版).北京:清华大学出版社，2011