重庆师范大学

《数据结构课程设计》

、

课程名称：数据结构

题　　目：动物园景点导游系统设计

学　　院：计算机与信息科学学院

专业年级：计算机科学与技术 19 级

小组组长：连先柔

小组成员：王珊　黄滟

指导教师：张万里　职称：讲师

2021 年 **12** 月 **10** 日

目录

景区景点导游系统

一、设计目的

随着我国休假制度的完善，人们拥有了更长更多的假期，而假期外出旅游成为了越来越多的人们度过假期的第一选择。在这样的背景条件下，各大旅游景区更是成为了热门中的热门，这也造成了在旅游高峰期，部分旅游景点人流量过大导致旅客迷路等问题，从而影响到游客体验的问题。不过从根本上说，这并不主要是因为游客量过大，往往是因为景区的服务不够全面细致，管理不够科学，效率不高所造成的，例如景区内部的地标不够详细或者是不够完整都可能会影响到游客游玩时的顺畅性，本项目就是在这样的背景下提出的，旨在做出一个能够方便游客的导航系统。这里我们就以出去的动物园为原型进行设计。

二、问题描述

在系统中能够为旅客提供重庆动物园的一个平面图，能够让旅客根据提示对图中任意景点的相关信息查询，为他们提供图中任意景点的问路查询以及图中多个景点的最佳访问路线。

三、基本要求

（1）显示出动物园的平面图及所含景点。

（2）为来访客人提供图中任意景点相关信息的查询。

（3）为来访客人提供图中任意景点的问路查询，即查询任意两个景点之间的一条最短的简单路径。

（4）提供图中任意景点问路查询，即求任意两个景点之间的所有路径。

（5）提供校园图中多个景点的最佳访问路线查询，即求途经这多个景点的最佳路径。

四、概要设计

1、平面设计图

根据百度百科获得动物园的平面图，根据平面图抽象出各个景点以及其大体距离和位置关系。

图 1 动物园平面图

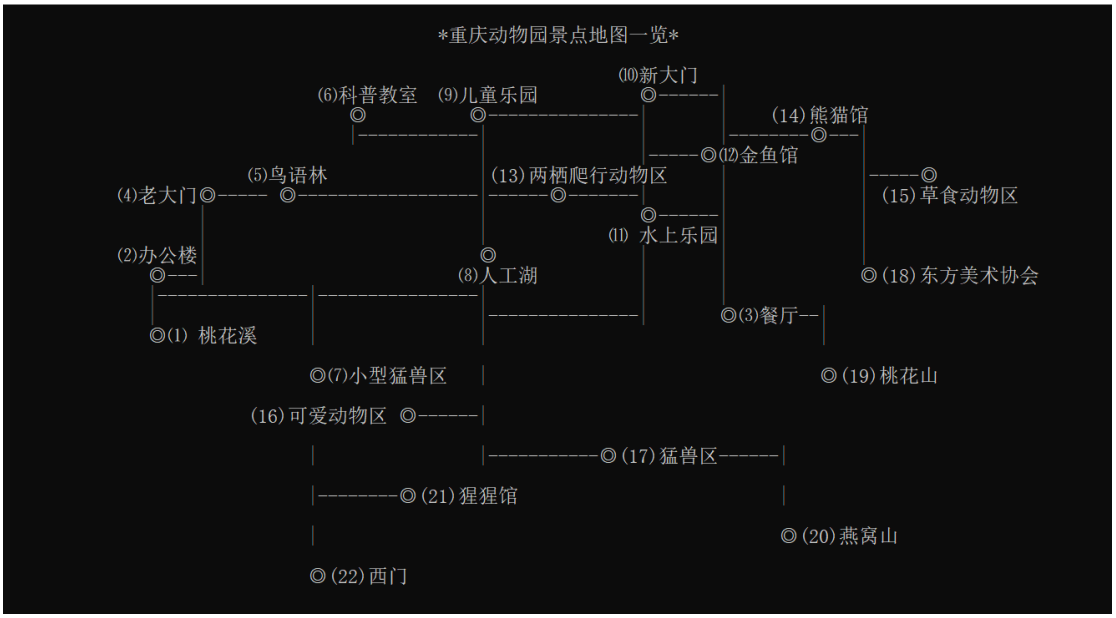


图 2 实验平面图

2、算法思路

用 C++进行编写，使用了邻接矩阵和邻接表，弗洛伊德算法，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、栈、队列等等的运用，最终构成了整个程序。

景点图的结构体：

```
struct vertex///景点信息结构体
{
    int num;///景点编号
    char name[20];///景点名称
    char info[300];///景点介绍
};
struct maps
{
    int n;///点数
    int m;///边数
    vertex v[M];
    int edgs[M][M];///邻接矩阵
```

} ; ///景点图的结构体

①获得景点信息函数：void Creat_vertex()，将各个景点对应的信息存入景点信息结构体数组中。

②生成图和邻接矩阵函数： void Creat_maps()，将获得的景点信息构成邻接矩阵来存图。

③查询景点信息函数：void Search_info()，通过选择菜单，查询任意一个景点的信息。

④多源最短路弗洛伊德算法：void Floyd()，通过弗洛伊德算法实现任意一点到 n-1 点最短路的建立。

⑤打印最短路径函数：void Floyd_print(int s, int e)，回溯弗洛伊德算法打印从 s 点到 e 点的最短路径经过的点，并统计所走的总路程。

⑥打印两点之间所有路径函数：void Dfs_allpath(int s,int e)，通过 DFS 深度优先搜索打印从 s 点到 e 点的所有路径。

⑦提供多景点之间最佳路径函数：void Bestpath_Multispot()，打印多个景点之间的最短路径，并统计所要走的总路程。

3、工作分配

（1）框架设计：由王珊进行构思，同时参考其他人的意见。

（2）系统设计：由黄滟进行框架的编写。

（3）程序设计：由连先柔进行各项功能的编辑。

（4）程序调试：由连先柔、黄滟进行程序的调试与调整。

（5）文档制作：由黄滟、王珊完成文档的编辑工作。

4、程序模块

程序的模块为：

景点信息查询功能：

```
void Search_info()
{
    int i,n;
    printf("重庆动物园的景点有：\n");
    for(i=0; i<22; i++)
    {
```

```
            printf("%d:%s\n",g.v[i].num,g.v[i].name);
        }
        while(1)
        {
            printf("请输入你想要查询的景点编号：\n");
            printf("按 0 退出\n\n");
            scanf("%d",&n);
            getchar();
            if(n==0)
            {
                break;
            }
            else  if(n<0||n>22)
            {
                printf("输入有误，请重新输入!!!  \n\n");
                continue;
            }
            else
            {
                printf("%d:%s\n",g.v[n-1].num,g.v[n-1].name);
                printf("%s\n\n",g.v[n-1].info);
            }
        }
        return  ;
}
```

两景点之间最短路径查询功能：

```
int  main()
{
        int  i,n;
        int  start,ends;
        Creat_vertex();
        Creat_maps();
        Dis_map();
```

```
while(1)
{
    Dis_menu();
    printf("请输入需要操作的命令：\n");
    scanf("%d",&n);
    getchar();
    if(n<1||n>5)
    {
        printf("输入有误，请重新输入!!! \n");
        continue;
    }
    else
    {
        if(n==1)
        {
            Search_info();
        }
        else  if(n==2)
        {
            Dis_map();
            printf("请输入起点的景点：\n");
            scanf("%d",&start);
            printf("请输入终点的景点：\n");
            scanf("%d",&ends);
            Floyd();///弗洛伊德
            printf("从%s 到%s 最短距离是：%d\n",g.v[start-1].name,g.v[end
s-1].name,dist[start-1][ends-1]);
            printf("%s->",g.v[start-1].name);
            Floyd_print(start-1, ends-1);
            printf("%s\n",g.v[ends-1].name);
        }
        else  if(n==3)
```

```
            {
                Dis_map();
                counts=1;///起始路径数为 1
                printf("请输入起点的景点：\n");
                scanf("%d",&start);
                printf("请输入终点的景点：\n");
                scanf("%d",&ends);
                Dfs_allpath(start-1,ends-1);
            }
            else  if(n==4)
            {
                Dis_map();
                Floyd();///弗洛伊德
                Bestpath_Multispot();
            }
            else  if(n==5)
            {
                return  0;
            }
        }

    }
    return  0;
}
```

两景点之间所有路径查询功能：

```
void  Dfs_allpath(int  s,int  e)
{
    int  dis=0;
    int  i,j;
    Stack[top]=s;
    top++;   ///起点入栈
    visited[s]=1;///标记入栈
```

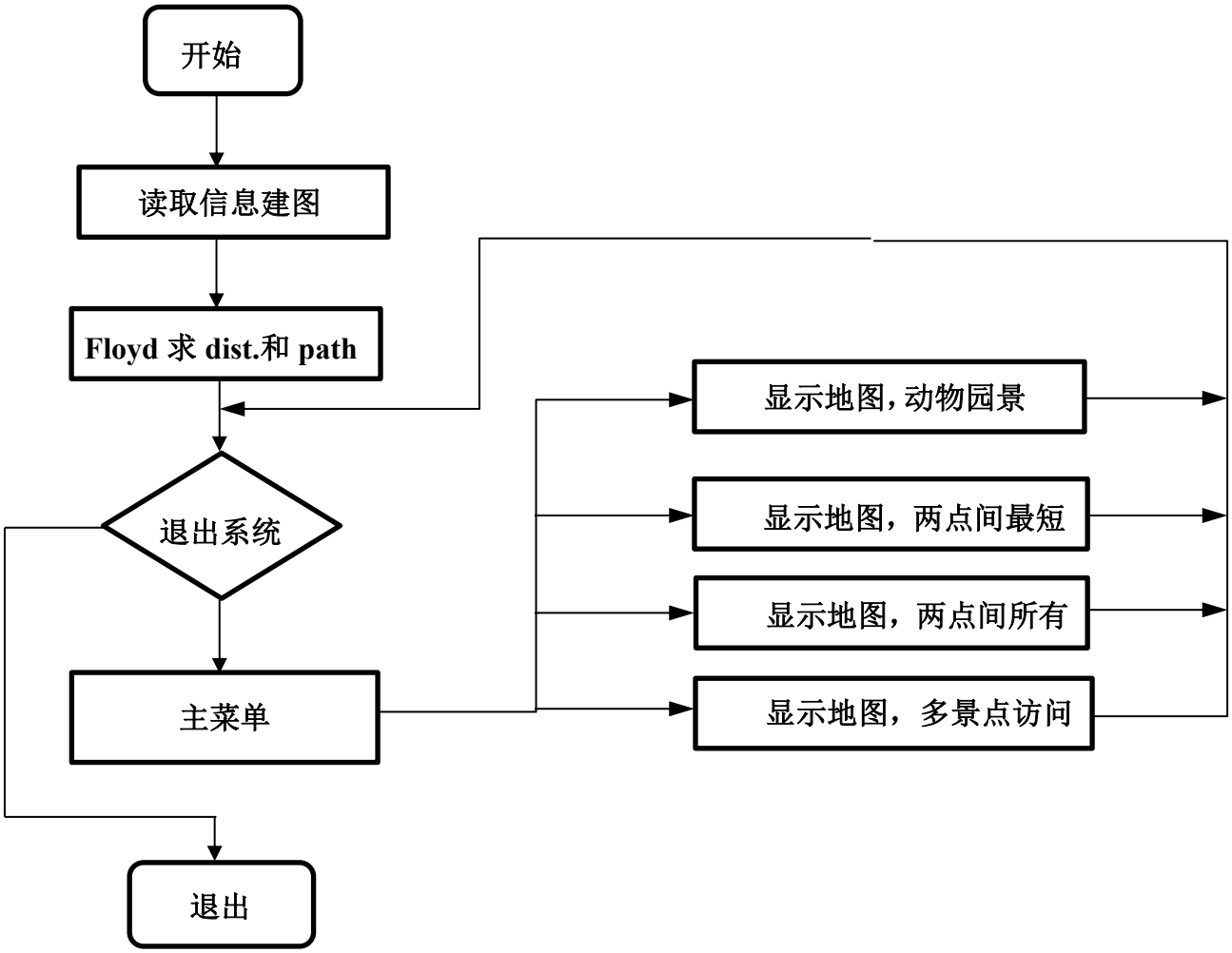
```
    for(i=0;  i<g.n;  i++)
    {
        if(g.edgs[s][i]>0&&g.edgs[s][i]!=INF&&!visited[i])
        {
            ///表明两点可达且未被访问
            if(i==e)///DFS 到了终点，打印路径
            {
                printf("第%d 条路:",counts++);
                for(j=0;  j<top;  j++)
                {
                    printf("%s->",g.v[Stack[j]].name);
                    if(j<top-1)///统计路径长度
                    {
                        dis=dis+g.edgs[Stack[j]][Stack[j+1]];
                    }
                }
                dis=dis+g.edgs[Stack[top-1]][e];
                printf("%s\n",g.v[e].name);///打印终点
                printf("总长度是：%dm\n\n",dis);
            }
            else///不是终点接着 DFS
            {
                Dfs_allpath(i,e);
                top--;///支路全被访问一遍,顶点出栈
                visited[i]=0;///出栈点标记为已出栈，允许下次访问
            }
        }
    }
}
```

多景点间访问最佳路线查询功能：

```
void  Bestpath_Multispot()
```

```
{
    int  vNum[M]=  {0};
    int  i,j,dis;
    j=1;
    dis=0;///统计全程总长
    printf("请输入你要游览的第%d 个景点的编号（输入-1 结束输入）：",j);
    scanf("%d",&vNum[j-1]);
    while(vNum[j-1]!=-1&&j<13)
    {
        printf("请输入你要游览的第%d 个景点编号：",++j);
        scanf("%d",  &vNum[j-1]);
        if(vNum[j-1]==-1)
        {
            break;
        }
    }
    printf("\n 这是最佳访问路径：");
    for(i=0;  vNum[i]>0&&vNum[i+1]>0;  i++)
    {
        printf("%s->",g.v[vNum[i]-1].name);///输出路径上的起点
        Floyd_print(vNum[i]-1,vNum[i+1]-1);///利用佛洛依德算法
        dis+=dist[vNum[i]-1][vNum[i+1]-1];
    }
    printf("%s\n\n",g.v[vNum[j-2]-1].name);///*输出路径上的终点*/
    printf("全程总长为：%dm\n\n",dis);
}
```

5、总体设计图



五、主程序

程序 1. 邻接表的存储结构创建：

```
struct  vertex///景点信息结构体
{
    int  num;///景点编号
    char  name[22];///景点名称
    char  info[300];///景点介绍
};
struct  maps
{
    int  n;///点数
    int  m;///边数
```

```
        vertex  v[M];
        int  edgs[M][M];///邻接矩阵
} g; ///景点图的结构体
```

程序 2. 邻接矩阵的存储结构：

```
    void  Creat_vertex()
    {
        g.v[0].num=1;
        strcpy(g.v[0].name,"桃花溪");
        strcpy(g.v[0].info,"桃花流水窅然去，别有天地非人间");
        g.v[1].num=2;
        strcpy(g.v[1].name,"办公楼");
        strcpy(g.v[1].info,"单位行政管理人员，业务技术人员等办公的业务用房");
        g.v[2].num=3;
        strcpy(g.v[2].name,"餐饮区");
        strcpy(g.v[2].info,"美食者不必是饕餮客");
        g.v[3].num=4;
        strcpy(g.v[3].name,"老大门");
        strcpy(g.v[3].info,"动物园建园最早修建的一个大门");
        g.v[4].num=5;
        strcpy(g.v[4].name,"鸟语林");
        strcpy(g.v[4].info,"其中有来自美洲、欧洲、非洲、东南亚的珍稀鸟类动物 30 余种，有鹦鹉家族中个头最大的紫蓝金刚鹦鹉、美丽的火烈鸟、黑天鹅等。");
        g.v[5].num=6;
        strcpy(g.v[5].name,"科普教室");
        strcpy(g.v[5].info,"天文地理奥妙多,奇思妙想乐趣多");
        g.v[6].num=7;
        strcpy(g.v[6].name,"小型猛禽区");
        strcpy(g.v[6].info,"从凶猛的猛禽，到害羞的小动物。");
```

```
g.v[7].num=8;
strcpy(g.v[7].name,"人工湖");
strcpy(g.v[7].info,"湖水碧绿碧绿的，像一片宽宽的草海。");
g.v[8].num=9;
strcpy(g.v[8].name,"儿童乐园");
strcpy(g.v[8].info,"让孩子流连忘返 的乐园!!! ");
g.v[9].num=10;
strcpy(g.v[9].name,"新大门");
strcpy(g.v[9].info,"动物园里的第二个大门。");
g.v[10].num=11;
strcpy(g.v[10].name,"水上乐园");
strcpy(g.v[10].info,"让游客在这里畅意戏水的乐园。");
g.v[11].num=12;
strcpy(g.v[11].name,"金鱼馆");
strcpy(g.v[11].info,"这里有金鱼长廊，展出金鱼和热带鱼共 30 多种近 1000 条。");
g.v[12].num=13;
strcpy(g.v[12].name,"两栖爬行动物区");
strcpy(g.v[12].info,"重庆动物园两栖爬行馆规模在西南地区排第一，饲养爬行动物 55 种共 400 多只。");
g.v[13].num=14;
strcpy(g.v[13].name,"熊猫馆");
strcpy(g.v[13].info,"重庆动物园大熊猫数量达 18 只，拥有数量在中国各地动物园中排名第一。");
g.v[14].num=15;
strcpy(g.v[14].name,"草食动物区");
strcpy(g.v[14].info,"食草系动物萌到不行。");
g.v[15].num=16;
strcpy(g.v[15].name,"可爱动物区");
strcpy(g.v[15].info,"区内饲养着羊驼、浣熊、豪猪、香猪、小兔、小
```

```
羊等动物");
        g.v[16].num=17;
        strcpy(g.v[16].name,"猛兽区");
        strcpy(g.v[16].info,"见证一下丛林霸王的存在。");
        g.v[17].num=18;
        strcpy(g.v[17].name,"东方美术协会");
        strcpy(g.v[17].info,"美的见证。");
        g.v[18].num=19;
        strcpy(g.v[18].name,"桃花山");
        strcpy(g.v[18].info,"漫山遍野的浪漫。");
        g.v[19].num=20;
        strcpy(g.v[19].name,"燕窝山");
        strcpy(g.v[19].info,"曲径通幽的休闲区。");
        g.v[20].num=21;
        strcpy(g.v[20].name,"猩猩馆");
        strcpy(g.v[20].info,"重庆动物园猩猩馆占地面积 2360 平米，饲养着黑
猩猩和红猩猩等");
        g.v[21].num=22;
        strcpy(g.v[21].name,"西门");
        strcpy(g.v[21].info,"动物园里第三个大门。");
    }
    void  Creat_maps()
    {
        int  i,j;
        g.n=22;///22 个景点
        g.m=33;///33 条双向路径
        for(i=0;  i<g.n;  i++)  ///初始化邻接矩阵
        {
            for(j=0;  j<g.n;  j++)
            {
                g.edgs[i][j]=INF;
```

```
    }
}
g.edgs[0][1]=g.edgs[1][0]=70;///写入边的信息
g.edgs[0][6]=g.edgs[6][0]=460;
g.edgs[0][7]=g.edgs[7][0]=690;
g.edgs[1][3]=g.edgs[3][1]=175;
g.edgs[3][4]=g.edgs[4][3]=100;
g.edgs[6][7]=g.edgs[7][6]=440;
g.edgs[6][20]=g.edgs[20][6]=425;
g.edgs[6][21]=g.edgs[21][6]=280;
g.edgs[4][12]=g.edgs[12][4]=420;
g.edgs[7][8]=g.edgs[8][7]=310;
g.edgs[7][15]=g.edgs[15][7]=320;
g.edgs[7][16]=g.edgs[16][7]=520;
g.edgs[0][21]=g.edgs[21][0]=650;
g.edgs[0][20]=g.edgs[20][0]=775;
g.edgs[20][21]=g.edgs[21][20]=325;
g.edgs[8][5]=g.edgs[5][8]=255;
g.edgs[8][9]=g.edgs[9][8]=300;
g.edgs[12][10]=g.edgs[10][12]=195;
g.edgs[12][11]=g.edgs[11][12]=340;
g.edgs[12][9]=g.edgs[9][12]=345;
g.edgs[16][19]=g.edgs[19][16]=325;
g.edgs[7][10]=g.edgs[10][7]=425;
g.edgs[10][11]=g.edgs[11][10]=180;
g.edgs[10][9]=g.edgs[9][10]=160;
g.edgs[9][11]=g.edgs[11][9]=180;
g.edgs[9][13]=g.edgs[13][9]=305;
g.edgs[13][14]=g.edgs[14][13]=175;
g.edgs[13][17]=g.edgs[17][13]=250;
g.edgs[14][17]=g.edgs[17][14]=190;
g.edgs[4][8]=g.edgs[8][4]=440;
g.edgs[4][7]=g.edgs[7][4]=460;
```

```
        g.edgs[10][2]=g.edgs[2][10]=225;
        g.edgs[2][18]=g.edgs[18][2]=175;
        g.edgs[9][2]=g.edgs[2][9]=385;
        g.edgs[2][13]=g.edgs[13][2]=350;

}
```

程序 3. 弗洛伊德最短路径算法：

```
int  main()
{
    int  i,n;
    int  start,ends;
    Creat_vertex();
    Creat_maps();
    Dis_map();
    while(1)
    {
        Dis_menu();
        printf("请输入需要操作的命令：\n");
        scanf("%d",&n);
        getchar();
        if(n<1||n>5)
        {
            printf("输入有误，请重新输入!!! \n");
            continue;
        }
        else
        {
            if(n==1)
            {
                Search_info();
            }
            else  if(n==2)
```

```
{
    Dis_map();
    printf("请输入起点的景点：\n");
    scanf("%d",&start);
    printf("请输入终点的景点：\n");
    scanf("%d",&ends);
    Floyd();///弗洛伊德
    printf("从%s 到%s 最短距离是：%d\n",g.v[start-1].name,
g.v[ends-1].name,dist[start-1][ends-1]);
    printf("%s->",g.v[start-1].name);
    Floyd_print(start-1, ends-1);
    printf("%s\n",g.v[ends-1].name);
}
else if(n==3)
{
    Dis_map();
    counts=1;///起始路径数为 1
    printf("请输入起点的景点：\n");
    scanf("%d",&start);
    printf("请输入终点的景点：\n");
    scanf("%d",&ends);
    Dfs_allpath(start-1,ends-1);
}
else if(n==4)
{
    Dis_map();
    Floyd();///弗洛伊德
    Bestpath_Multispot();
}
else if(n==5)
{
    return 0;
```

```
                }
            }

        }
        return 0;
}
```

程序 4. 深度优先遍历算法：

```
    void Floyd_print(int s, int e)
    {
        if(path[s][e]==-1||path[s][e]==e||path[s][e]==s)///递归终止条件
        {
            return;
        }
        else
        {
            Floyd_print(s,path[s][e]);///将中间点作为终点继续打印路径
            printf("%s->",g.v[path[s][e]].name);///打印中间景点名字
            Floyd_print(path[s][e],e);///将中间点作为起点继续打印路径
        }
    }

    void Dfs_allpath(int s,int e)
    {
        int dis=0;
        int i,j;
        Stack[top]=s;
        top++;   ///起点入栈
        visited[s]=1;///标记入栈
        for(i=0; i<g.n; i++)
        {
            if(g.edgs[s][i]>0&&g.edgs[s][i]!=INF&&!visited[i])
```

```
            {
                ///表明两点可达且未被访问
                if(i==e)///DFS 到了终点，打印路径
                {
                    printf("第%d 条路:",counts++);
                    for(j=0; j<top; j++)
                    {
                        printf("%s->",g.v[Stack[j]].name);
                        if(j<top-1)///统计路径长度
                        {
                            dis=dis+g.edgs[Stack[j]][Stack[j+1]];
                        }
                    }
                    dis=dis+g.edgs[Stack[top-1]][e];
                    printf("%s\n",g.v[e].name);///打印终点
                    printf("总长度是：%dm\n\n",dis);
                }
                else///不是终点接着 DFS
                {
                    Dfs_allpath(i,e);
                    top--;///支路全被访问一遍,顶点出栈
                    visited[i]=0;///出栈点标记为已出栈，允许下次访问
                }
            }
        }
    }
    void  Bestpath_Multispot()
    {
        int  vNum[M]= {0};
        int  i,j,dis;
        j=1;
        dis=0;///统计全程总长
```

```
        printf("请输入你要游览的第%d 个景点的编号（输入-1 结束输入）：",
j);
        scanf("%d",&vNum[j-1]);
        while(vNum[j-1]!=-1&&j<13)
        {
            printf("请输入你要游览的第%d 个景点编号：",++j);
            scanf("%d",  &vNum[j-1]);
            if(vNum[j-1]==-1)
            {
                break;
            }
        }
        printf("\n 这是最佳访问路径：");
        for(i=0;  vNum[i]>0&&vNum[i+1]>0;  i++)
        {
            printf("%s->",g.v[vNum[i]-1].name);///输出路径上的起点
            Floyd_print(vNum[i]-1,vNum[i+1]-1);///利用佛洛依德算法
            dis+=dist[vNum[i]-1][vNum[i+1]-1];
        }
        printf("%s\n\n",g.v[vNum[j-2]-1].name);///*输出路径上的终点*/
        printf("全程总长为：%dm\n\n",dis);
}
```

六、运行截图

（1）主菜单

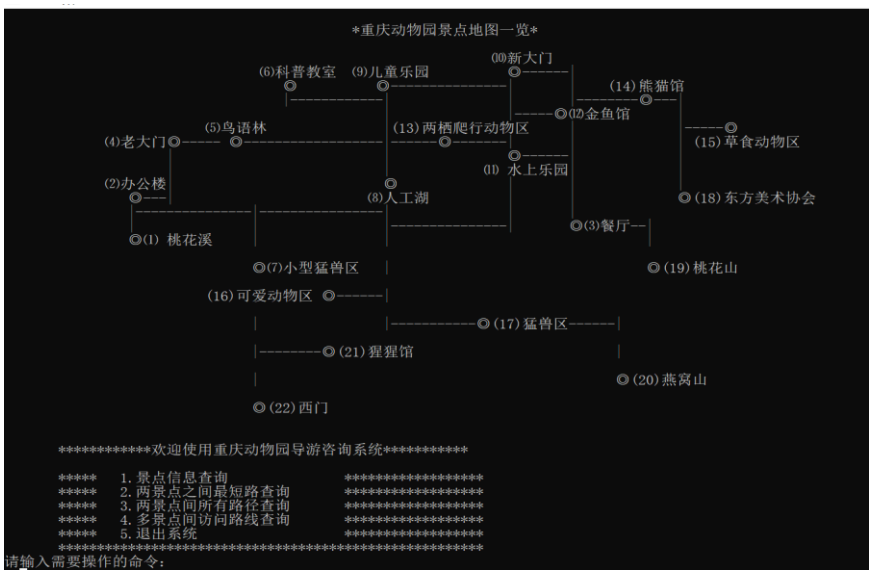


图 3 主菜单

（2） 景点信息查询

图 4 景点信息查询

（3）两景点之间最短路径查询

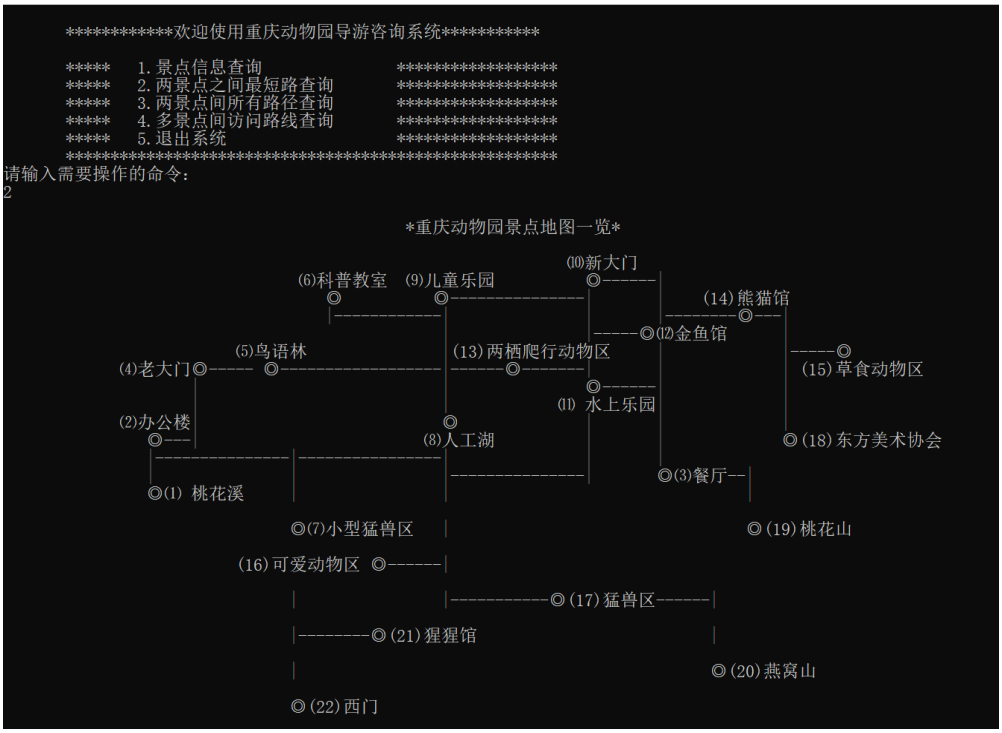


图 5 选择

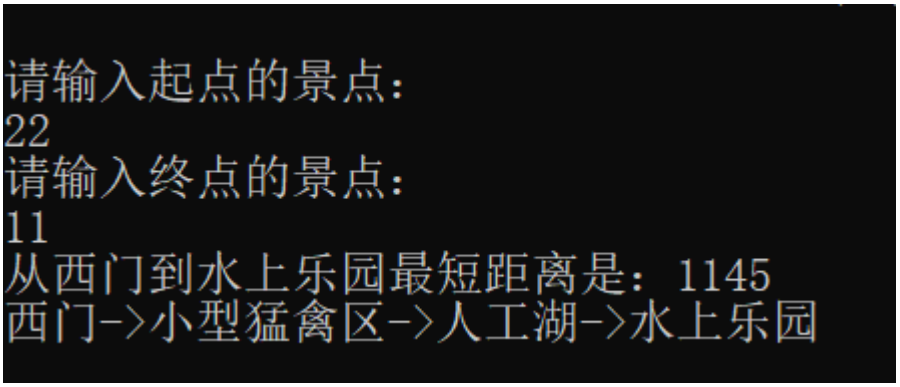


图 6 两景点最短路径

（4）两景点之间的所有路径查询

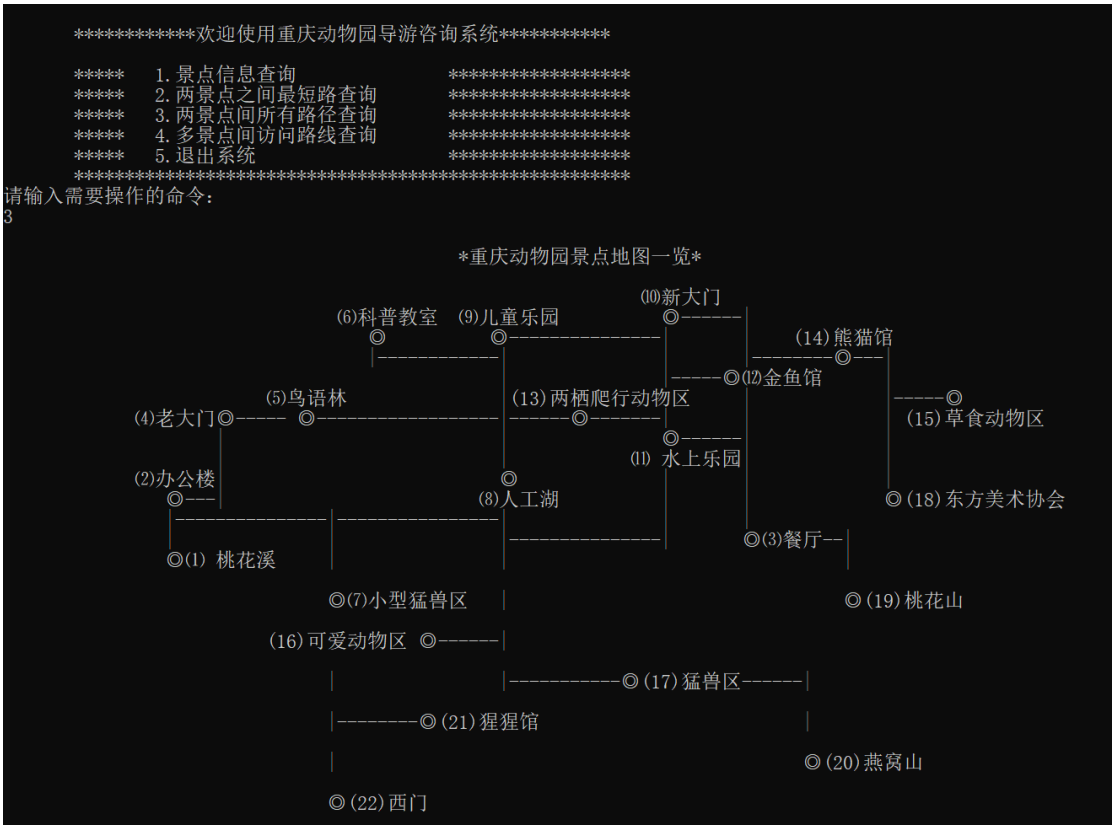


图 7 选择

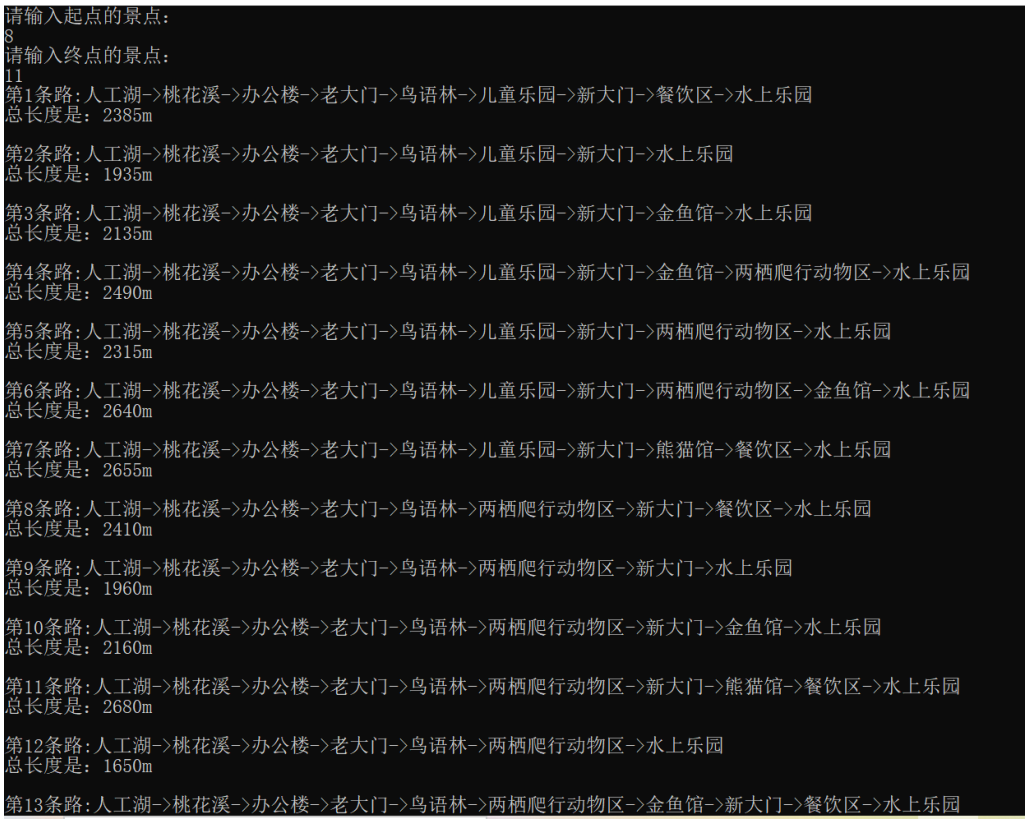


图 8 开始路径查询

```
第117条路:人工湖->儿童乐园->鸟语林->两栖爬行动物区->水上乐园
总长度是: 1365m

第118条路:人工湖->儿童乐园->鸟语林->两栖爬行动物区->金鱼馆->新大门->餐饮区->水上乐园
总长度是: 2300m

第119条路:人工湖->儿童乐园->鸟语林->两栖爬行动物区->金鱼馆->新大门->水上乐园
总长度是: 1850m

第120条路:人工湖->儿童乐园->鸟语林->两栖爬行动物区->金鱼馆->新大门->熊猫馆->餐饮区->水上乐园
总长度是: 2570m

第121条路:人工湖->儿童乐园->鸟语林->两栖爬行动物区->金鱼馆->水上乐园
总长度是: 1690m

第122条路:人工湖->儿童乐园->新大门->餐饮区->水上乐园
总长度是: 1220m

第123条路:人工湖->儿童乐园->新大门->水上乐园
总长度是: 770m

第124条路:人工湖->儿童乐园->新大门->金鱼馆->水上乐园
总长度是: 970m

第125条路:人工湖->儿童乐园->新大门->金鱼馆->两栖爬行动物区->水上乐园
总长度是: 1325m

第126条路:人工湖->儿童乐园->新大门->两栖爬行动物区->水上乐园
总长度是: 1150m

第127条路:人工湖->儿童乐园->新大门->两栖爬行动物区->金鱼馆->水上乐园
总长度是: 1475m

第128条路:人工湖->儿童乐园->新大门->熊猫馆->餐饮区->水上乐园
总长度是: 1490m

第129条路:人工湖->水上乐园
总长度是: 425m
```

图 9 结束路径查询

（5）多景点间访问路线查询

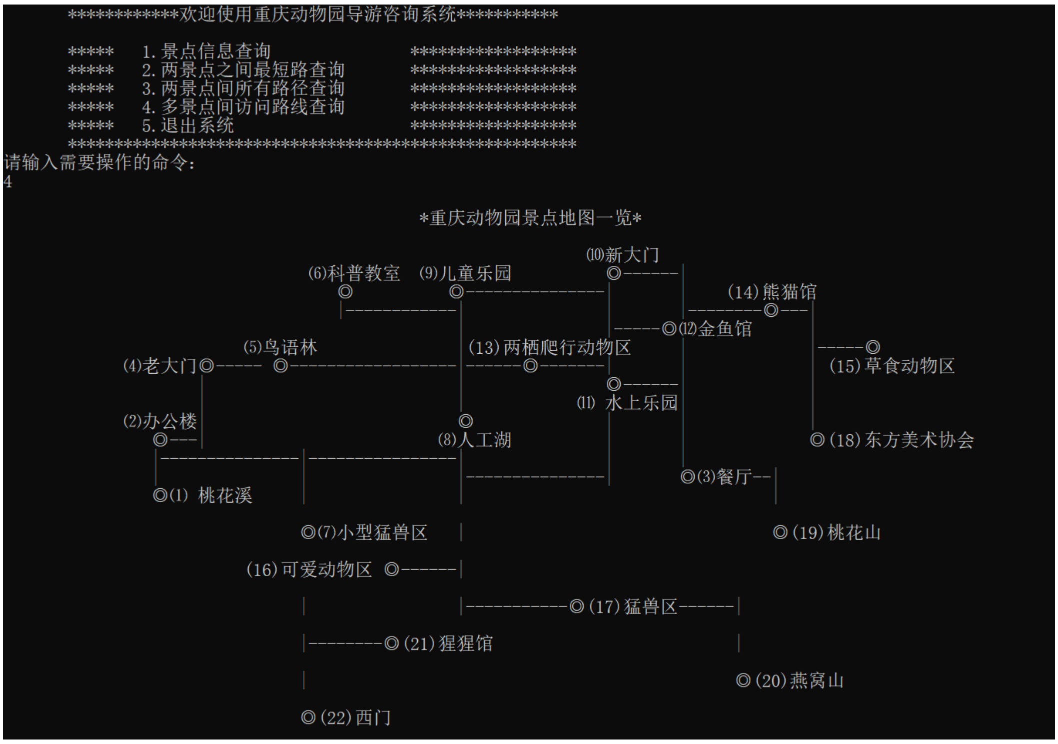


图 10 选择

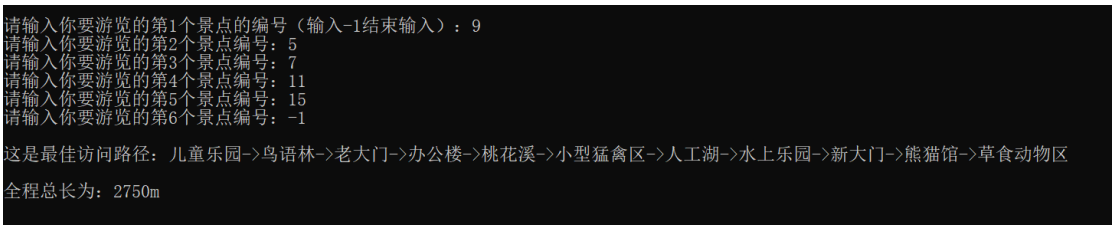


图 11 可访问路线

（6）退出系统

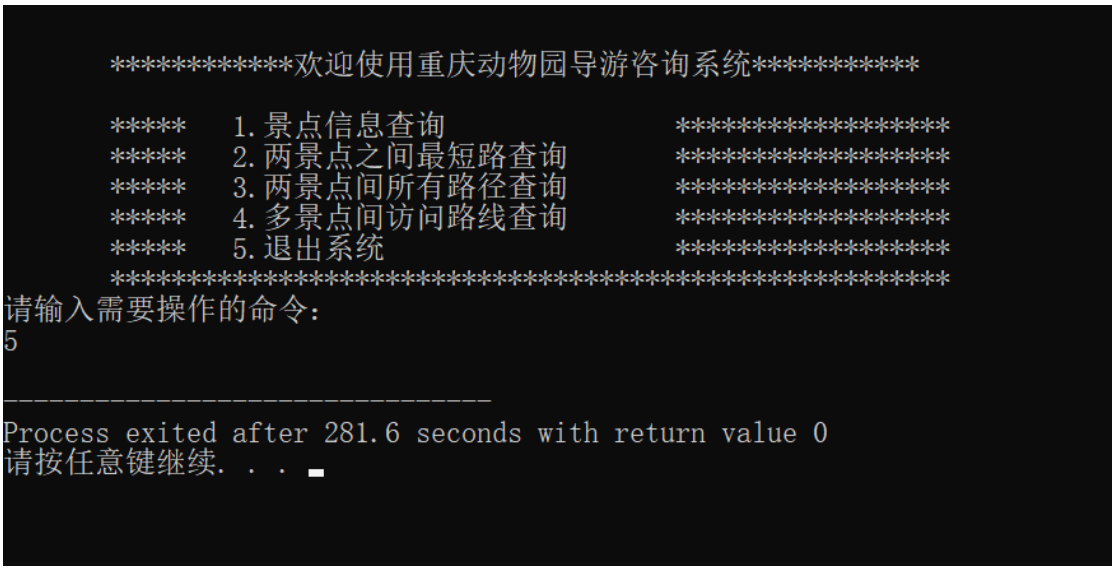


图 12 退出选择

七、总结

通过一学期的数据结构课程的学习，在进行课程学习时只是单纯的学习了原理和算法，没有注重代码的具体实现，我们这种学习方式是错误的。不管我们再学习什么课程时，都应该注重理论和实践的结合。在用代码写一个具体的东西的时候，必须自己亲身操作一遍，遇到错误找到问题并进行思考，期间不断寻求更加快捷高效的方式来改进。还有不管用的是面向对象的语言还是面向过程的语言,在考虑问题的大方向上是一致的,要根据具体语言的特点来对应进行变换.将代码的功能尽量细分,按照功能来区分。类比于面向借口的思想.在完成整个系统的框架之后，按功能来一步步实现。