

## 一、实验目的：

掌握图的存储方法和最短路径算法。

## 二、实验要求：

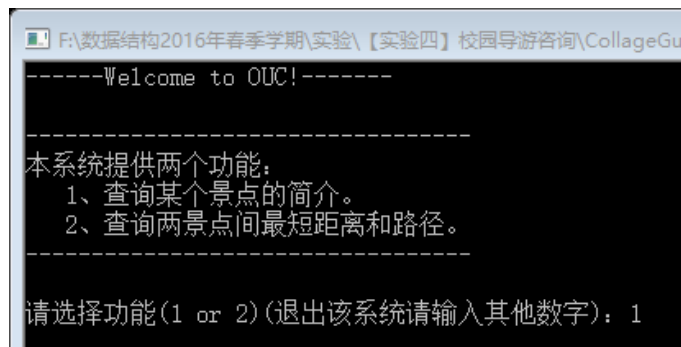
- 1、设计所在学校的校园平面图，所含景点不少于 10 个。以图中顶点表示校内各景点，存放景点名称、代号、简介等信息；以边表示路径，存放路径长度等相关信息。
- 2、为来访客人提供图中任意景点相关信息的查询。
- 3、为来访客人提供图中任意景点的纹路查询，即查询任意两个景点之间的一条最短的简单路径。

## 三、实验内容和实验步骤：

### 1、需求分析：

(1) 输入形式：

- a) 功能二选一，输入功能序号进入该功能，输入其他数字结束程序。

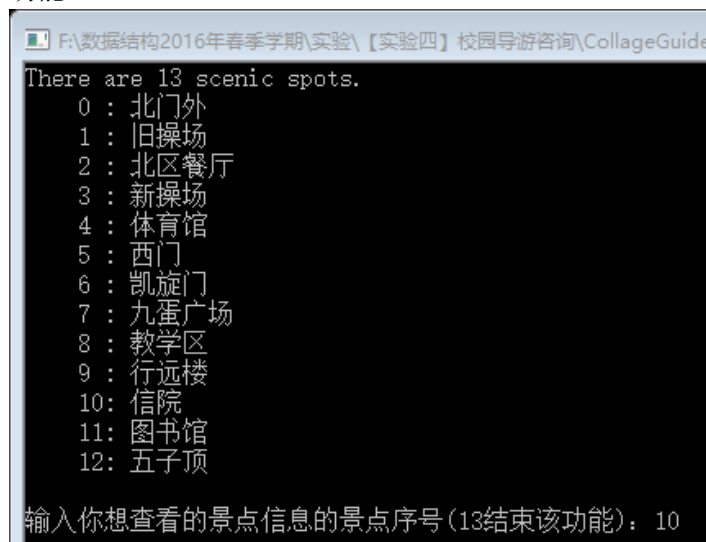


```
F:\数据结构2016年春季学期\实验\【实验四】校园导游咨询\CollageGui
-----Welcome to OUC!-----

本系统提供两个功能：
1、查询某个景点的简介。
2、查询两景点间最短距离和路径。

请选择功能(1 or 2) (退出该系统请输入其他数字)：1
```

- b) 查询景点功能：根据景点对应序号，输入需要查询的序号，规定数字退出该功能。



```
F:\数据结构2016年春季学期\实验\【实验四】校园导游咨询\CollageGuide.
There are 13 scenic spots.
0 : 北门外
1 : 旧操场
2 : 北区餐厅
3 : 新操场
4 : 体育馆
5 : 西门
6 : 凯旋门
7 : 九蛋广场
8 : 教学区
9 : 行远楼
10: 信院
11: 图书馆
12: 五子顶

输入你想查看的景点信息的景点序号(13结束该功能)：10
```

- c) 查询最短路径功能：输入起始点序号：“Start End”，空格隔开，景点序号范围外的序号视为退出该功能。

```
F:\数据结构2016年春季学期\实验\【实验四】校园导游咨询\CollageGuide.exe
There are 13 scenic spots.
0 : 北门外
1 : 旧操场
2 : 北区餐厅
3 : 新操场
4 : 体育馆
5 : 西门
6 : 凯旋门
7 : 九蛋广场
8 : 教学区
9 : 行远楼
10: 信院
11: 图书馆
12: 五子顶

请输入起点和目的地的序号：(不在序号范围则退出该功能)0 10
```

(2) 输出形式:

- a) 初次进入系统，显示欢迎词（如上图所示）。
- b) 每次进入功能，都会列出所有景点及其对应序号（如上图所示）。
- c) 查询景点功能：输出格式“序号：景点名 景点简介”，并询问是否继续使用  
该功能。

```
输入你想查看的景点信息的景点序号(13结束该功能): 9
9 : 行远楼 海大行远楼，一跳解千愁

继续查看景点信息请输入序号，(13结束该功能):
```

- d) 查询最短路径功能：输出最短距离以及行走路径“序号景点 -> 序号景点  
-> ……”并并询问是否继续使用该功能。

```
请输入起点和目的地的序号：(不在序号范围则退出该功能)0 9
最短距离是：28
最短路径是：0北门外 -> 2北区餐厅 -> 1旧操场 -> 4体育馆 -> 5西门 -> 9行远楼

继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能:
```

- e) 为了美观每次退出功能或进入功能都会清屏。（详细见输入形式的介绍和下  
图）

```
F:\数据结构2016年春季学期\实验\【实验四】校园导游咨询\CollageGuide.exe
-----
本系统提供两个功能：
1、查询某个景点的简介。
2、查询两景点间最短距离和路径。
-----
继续使用请输入功能序号(1 or 2)，退出则输入其他数字:
```

- f) 退出程序输出结束词：“Thanks! Good Bye!”

```

请选择功能(1 or 2)(退出该系统请输入其他数字): 3
Thanks! Good Bye!

```

### (3) 实现功能:

- 用户输入功能序号, 提供查询景点或查询路径功能。
- 查询景点信息: 根据用户输入数字 (景点序号), 显示对应的景点信息。
- 查询最短路径: 根据用户输入起始点数字, 显示两点间一条最短的简单路径。

## 2、概要设计:

### (1) 数据结构:

- 图结点 (顶点): 包含序号、名字、基本介绍。
- 图采用邻接矩阵的存储结构, 二维数组下标表示结点序号, 值为两点之间的弧的权值, 限定整型数据。若主对角线元素为 0, 无弧则为最大值 (在此默认 30000)。数组大小固定, 行和列大小均为最大顶点个数, 使用宏定义。

```

#define MAX_VERTEX_NUM 13//最大顶点个数, 在此为景点个数
#define INFINITY 30000//默认最大值

typedef struct{//顶点
    int num;//景点序号
    char *name;//景点名字
    char *info;//景点简介
}VertexType;

typedef struct{//图
    VertexType vexs[MAX_VERTEX_NUM];
    int arcs[MAX_VERTEX_NUM][MAX_VERTEX_NUM]; //数组为权值
    int vexnum, arcnum; //图的当前顶点数和弧数
}MGraph;

```

### (2) 主函数流程:

- 创建校园地图 (无向带权图);
- 提示选择功能, 接受用户需要的功能序号并进入对应的功能;

### (3) 各程序模块之间的调用关系

```

//创建地图
void CreatMGraph(MGraph *G){

//用Dijkstra算法求有向图G的v0顶点到其余顶点v的最短路径P[v]及其带权长度D[v]...
void ShortestPath_DIJ(MGraph *G, int v0, int P[][G->vexnum], int *D){

//查询景点信息
void SeeInfo(MGraph *G){

//查询最短路径
void Search_Path(MGraph *G){

```

其中 Search\_Path 函数调用 ShortestPath\_DIJ 函数, 通过传参 P、D 数组获取最短路径并对最短路径进行按序输出。

### 3、详细设计（主要展示 ShortestPath\_DIJ 函数和 Search\_Path 的函数）

#### (1) ShortestPath\_DIJ 函数

- 用 Dijkstra 算法求图 G 的  $v_0$  顶点到其余顶点  $v$  的最短路径  $P[v]$  及其带权长度  $D[v]$ 。
- 若  $P[v][w]$  为 TRUE，则  $w$  是从  $v_0$  到  $v$  当前求得最短路径上的顶点。
- $final[v]$  为 TRUE 当且仅当  $v \in S$ ，即已经求得  $v_0$  到  $v$  的最短路径。

```
//用Dijkstra算法求有向图G的v0顶点到其余顶点v的最短路径P[v]及其带权长度D[v]。
void ShortestPath_DIJ(MGraph *G, int v0, int P[][G->vexnum], int *D){
    //用Dijkstra算法求有向图G的v0顶点到其余顶点v的最短路径P[v]及其带权长度D[v]。
    //若P[v][w]为TRUE，则w是从v0到v当前求得最短路径上的顶点。
    //final[v]为TRUE当且仅当v∈S，即已经求得从v0到v的最短路径。
    int v, i, j, w, min;
    int final[G->vexnum];

    for(v = 0; v < G->vexnum; ++v){
        final[v] = FALSE;
        D[v] = G->arcs[v0][v];
        for(w = 0; w < G->vexnum; ++w)
            P[v][w] = FALSE; //沿途路径
        if(D[v] < INFINITY){
            P[v][v0] = TRUE;
            P[v][v] = TRUE;
        } //if
    } //for
    D[v0] = 0;
    final[v0] = TRUE; //初始化，v0顶点属于S集
    //开始主循环，每次求得v0到某个+顶点的最短路径，并加入S集
    for(i = 1; i < G->vexnum; ++i){ //其余G.vexnum-1个顶点
        min = INFINITY; //当前所知的v0顶点的最近距离
        for(w = 0; w < G->vexnum; ++w)
            if(!final[w] //w顶点在V-S中
                if(D[w] < min){ //w顶点离v0顶点更近
                    v = w;
                    min = D[w];
                } //if
            } //if

        final[v] = TRUE; //离v0顶点最近的w加入S集
        for(w = 0; w < G->vexnum; ++w) //更新当前最短路径及距离
            if(!final[w] && (min + G->arcs[v][w] < D[w])){ //修改D[w]和P[w]，w∈V-S
                D[w] = min + G->arcs[v][w];
                for(j = 0; j < G->vexnum; j++)
                    P[w][j] = P[v][j];
                P[w][v] = TRUE; //P[w]=P[v]+[w]
                //printf("%d's step is %d\n",w,step[w]);
            } //if
    } //for
} //ShortestPath_DIJ
```

#### (2) Search\_Path 函数

- 通过 D 数组的元素的值判断是否有通路，若为 0 则用户输入的始终点是同一点，显示“原地!”；若为程序定义最大值，则无通路，显示“无法到达!”；若为其他值，则说明两点间有通路。则进入输出路径流程。
- 由于 P 数组只能记录那些点是最短路径上的点，并不能分辨这些点的先后顺序，所以输出最短路径的难点和重点在于如何分辨这些点的先后顺序。
- 思路如下：从起点开始，查询起点到这些最短路径的点是否有弧（通过邻接矩阵），有，则选出权值最小的弧，把与起点相邻的点作为新的起点，重复上述步骤知道与最新的起点相邻的点为终点。

```

//查询最短路径
void Search_Path(MGrpah *G){
    int i, start, end;
    printf("There are %d scenic spots.\n", G->vexnum);
    for(i = 0; i < G->vexnum; i++)
        printf("%-2d: %s\n", G->vexs[i].num, G->vexs[i].name);

    printf("\n请输入起点和目的地的序号: (不在序号范围则退出该功能);");
    scanf("%d %d", &start, &end);

    while(0 <= start && start < G->vexnum && 0 <= end && end < G->vexnum){
        int P[G->vexnum][G->vexnum], D[G->vexnum];
        ShortestPath_DIJ(G, start, P, D);

        if(D[end] == INFINITY)
            printf("无法达到!\n");
        else if(D[end] == 0)
            printf("原地!\n");
        else{
            printf("最短距离是: %d\n", D[end]);
            printf("最短路径是: ");

            int i, j;
            int count = 0; //记录路径的顶点个数
            for(i = 0; i < G->vexnum; i++)
                if(P[end][i])
                    count++;

            int a[count]; //记录路径的顶点序号
            a[0] = start; //保证头尾是用户输入的始终点
            a[count - 1] = end;
            for(i = 0, count = 1; i < G->vexnum; i++)
                if(P[end][i] && i != start && i != end)
                    a[count++] = i;
            count = count - 1;

            //让数组a按路径顺序排序
            int min1, tmp, i1; //i1为中间变量, min1记录最小的权值, tmp用于交换序号, i1记录权值最小的序号
            for(i = 0; i < count - 1; i++){
                min1 = INFINITY;
                for(j = i + 1; j < count; j++){
                    //可能有多条通路, 那么就选权值最小的
                    if(G->arcs[a[i]][a[j]] < min1){
                        min1 = G->arcs[a[i]][a[j]];
                        i1 = j;
                    }
                }
                tmp = a[i1];
                a[i1] = a[i + 1];
                a[i + 1] = tmp;
            }

            for(i = 0; i < count - 1; i++)
                printf("%d%s -> ", a[i], G->vexs[a[i]].name);
            printf("%d%s\n", a[i], G->vexs[a[i]].name);
        }
        printf("\n继续查询请直接输入起点和目的地序号, 不在序号范围则退出该功能:");
        scanf("%d %d", &start, &end);
    }
}
//Search_Path

```

#### 4、调试分析:

##### (1) 容错性考虑:

- a. 对无通路或者始终点是同一点的情况的处理: 通过 D 数组的元素的值判断是否有通路, 若为 0 则用户输入的, 显示“原地!”; 若为程序定义最大值, 则无通路,

显示“无法到达!”。

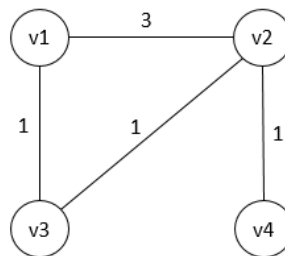
```
if(D[end] == INFINITY)
    printf("无法到达!\n");
else if(D[end] == 0)
    printf("原地!\n");
else{
    printf("最短距离是: %d\n", D[end]);
    printf("最短路径是: ");
```

- b. 用户选择查询景点信息功能时，若输入的序号不在景点范围内则提示重新输入（结束序号除外）。

```
while(n < 0 || n > G->vexnum){
    printf("输入错误，请重新输入(%d结束该功能); ", G->vexnum);
    scanf("%d", &n);
}
```

## (2) 求最短路径顶点的顺序:

- a. 算法：从起点开始，查询起点到这些最短路径的点是否有弧（通过邻接矩阵），有，则选出权值最小的弧，把与起点相邻的点作为新的起点，重复上述步骤知道与最新的起点相邻的点为终点。
- b. 选出权值最小的弧这一点很重要。因为可能与该轮起点相邻的顶点且是最短路径上的顶点不止一个，这时需要判断哪个才是最短路径上与该轮起点直接相邻的。举个例子：



求 v1 到 v4 的最短路径，顺序应为：v1->v3->v2->v4，而数组 P[v4][v1]=P[v4][v2]=P[v4][v3]均为 True。以 v1 为起点时，邻接矩阵上[v1][v2]和[v1][v3]均不为 0 和最大值，若按照顺序检索（从 v2 循环到 v4）遇到可通就停止，那么输出路径就为 v1->v2->v4，这显然是不对的。

- c. 因为是无向图，不排除进入往回走的死循环，例如求出 v1 下一点是 v2 后，发现 v2 符合要求的点又是 v1（假设 v1、v2 均不为起始点），这显然也是不对的。解决方案是，设置一个记录路径点的数组 a，数组大小就是最短路径上点的个数，元素记录的是这些点的序号，一开始先按照点序号的升序记录，在执行求最短路径顶点的顺序过程时，直接从数组 a 中元素获得顶点序号，并对数组 a 通过内部交换的方式进行内部排序，这样就能避免往回走。

## (3) 关于无穷大:

标准库里有对无穷的定义，在 limits.h 库里，无穷大用 INT\_MAX 表示（int 型的无穷大）。因为在 DIJ 算法中会出现无穷大加一，如果此时把无穷大设置为系统的无穷大，则会越界，因此无穷大要自己用宏定义。当然自行定义的无穷大需要大于所有权值之和，否则也不符合条件。

## (4) 逻辑错误

```

while(n != G->vexnum){
    while(n < 0 || n > G->vexnum){
        printf("输入错误, 请重新输入(%d结束该功能): ", G->vexnum);
        scanf("%d", &n);
    }
    if(n != G->vexnum){
        printf("%-2d: %s %s\n", G->vexs[n].num, G->vexs[n].name, G->vexs[n].info);
        printf("\n继续查看景点信息请输入序号, (%d结束该功能): ", G->vexnum);
        scanf("%d", &n);
    }
}
}

```

若省去 if 条件, 则输入一次错误后接着输入退出序号则无法退出且出现越界。

#### (5) 时空复杂度分析

- a. void ShortestPath\_DIJ(MGraph \*G, int v0, int P[][G->vexnum], int \*D)
 

//用 Dijkstra 算法求无向图 G 的 v0 顶点到其余顶点 v 的最短路径 P[v]及其带权长度 D[v]。

时间复杂度  $O(vexnum^3)$
- b. void Search\_Path(MGraph \*G) //查询最短路径
 

时间复杂度总的  $O(count^2 + 2*vexnum)$

  1. 计算最短路径顶点的个数: 需要遍历 P[v]:  $O(vexnum)$
  2. 为存储最短路径顶点的数组 a 初始化也需要遍历 P[v]:  $O(vexnum)$
  3.  $O(count^2)$  部分是而且因为是数组内部通过元素交换实现排序, 实质是一简单的选择排序, 所以进行的移动操作很多。

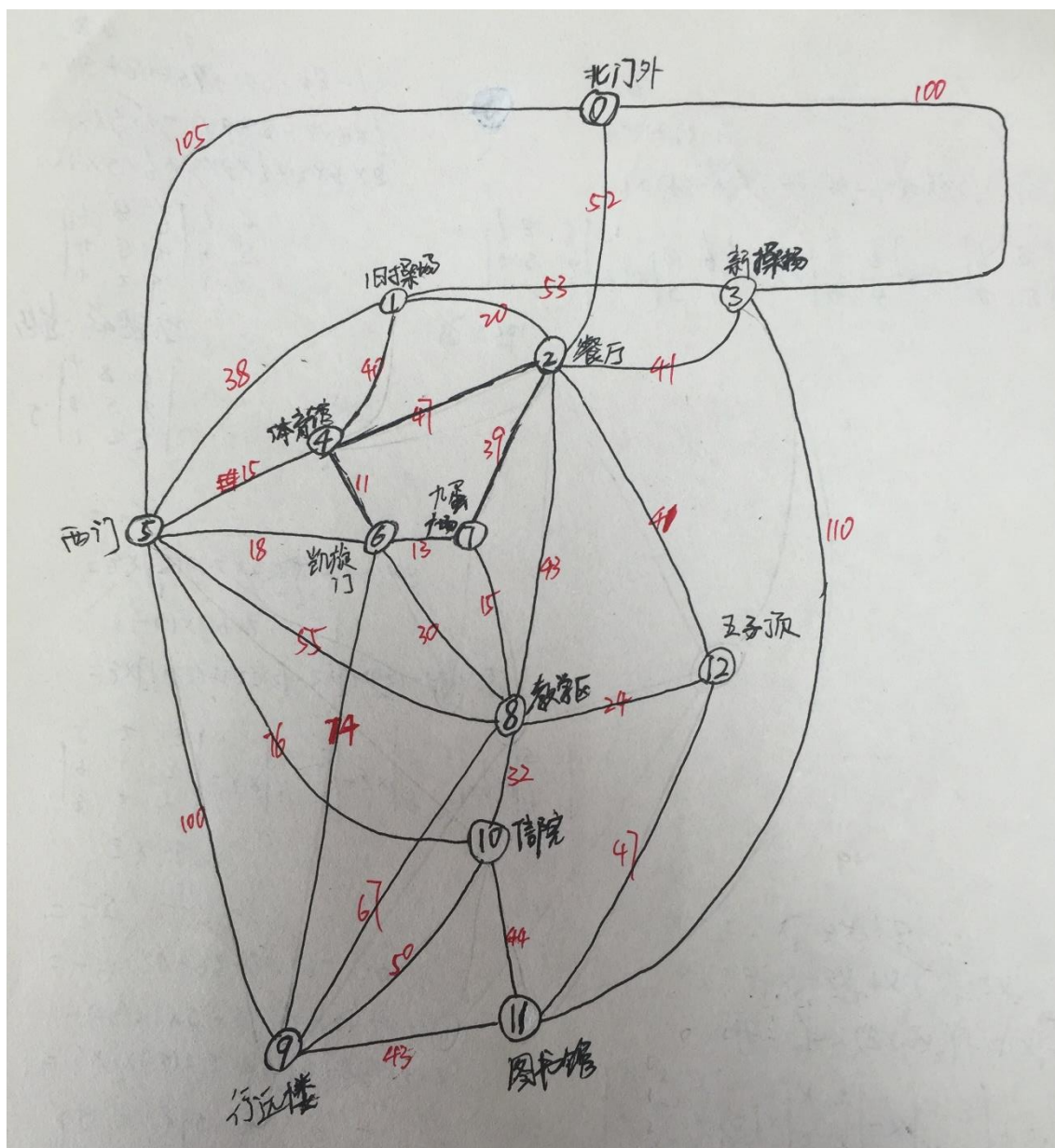
空间复杂度:

  1. 二维数组 P[G->vexnum][G->vexnum] (int)
  2. 一位数组 D[G->vexnum] (int)、a[count] (int)
  3. 此函数是程序中占用空间比较多的, 数组本来就大, 而且非动态开辟。
- c. 由于景点个数是初始设定, 并非由用户决定, 而且不能修改, 所以而且采用 Dijkstra 算法求最短路径, 要求用邻接矩阵的结构存储图, 同时为了算法简便, 没有对邻接矩阵进行压缩存储处理。因而改程序对内存空间的消耗比较大。

## 四、实验结果:

说明程序中的无向带权图及其邻接矩阵 (用表格形式给出) 下图所示





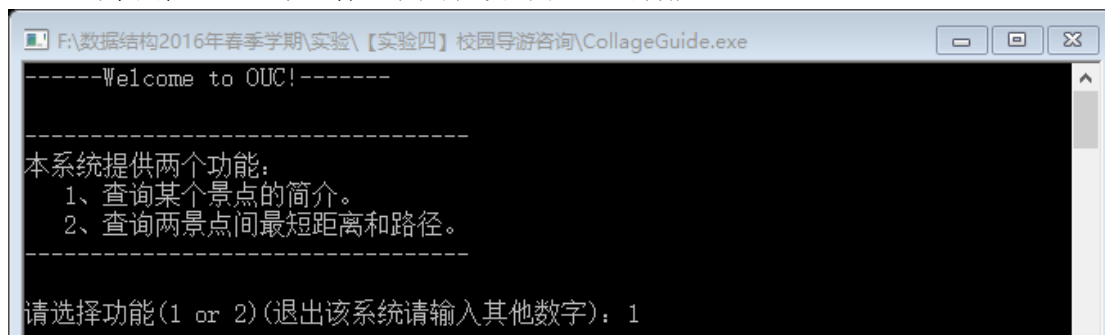
(带数字的圈为景点序号，红色为该弧的权值)



	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	-	52	100	-	105	-	-	-	-	-	-	-
1	-	0	20	53	40	38	-	-	-	-	-	-	-
2	52	20	0	41	47	-	-	39	43	-	-	-	41
3	100	53	41	0	-	-	-	-	-	-	-	110	-
4	-	40	47	-	0	15	11	-	-	-	-	-	-
5	105	38	-	-	15	0	18	-	55	100	76	-	-
6	-	-	-	-	11	18	0	13	30	74	-	-	-
7	-	-	39	-	-	-	13	0	15	-	-	-	-
8	-	-	43	-	-	55	30	15	0	67	32	-	24
9	-	-	-	-	-	100	74	-	67	0	50	43	-
10	-	-	-	-	-	76	-	-	32	50	0	44	-
11	-	-	-	110	-	-	-	-	-	43	44	0	47
12	-	-	41	-	-	-	-	-	24	-	-	47	0

（为了方便查看，采用表格形式列出、“-”表示无穷大）

#### 1、查询景点信息（正常查看、不在序号范围、退出功能）



```
F:\数据结构2016年春季学期\实验\【实验四】校园导游咨询\CollageGuide.exe
There are 13 scenic spots.
0 : 北门外
1 : 旧操场
2 : 北区餐厅
3 : 新操场
4 : 体育馆
5 : 西门
6 : 凯旋门
7 : 九蛋广场
8 : 教学区
9 : 行远楼
10: 信院
11: 图书馆
12: 五子顶

输入你想查看的景点信息的景点序号(13结束该功能): 0
0 : 北门外 撸串好地方

继续查看景点信息请输入序号, (13结束该功能): 9
9 : 行远楼 海大行远楼, 一跳解千愁

继续查看景点信息请输入序号, (13结束该功能): 15
输入错误, 请重新输入(13结束该功能): -1
输入错误, 请重新输入(13结束该功能): 13
```

## 2、查询最短路径（正常情况，从 v1 到 v2，从 v2 到 v1）

```
F:\数据结构2016年春季学期\实验\【实验四】校园导游咨询\CollageGuide.exe
-----
本系统提供两个功能：
1、查询某个景点的简介。
2、查询两景点间最短距离和路径。
-----
继续使用请输入功能序号(1 or 2)，退出则输入其他数字：2
```

```
F:\数据结构2016年春季学期\实验\【实验四】校园导游咨询\CollageGuide.exe
There are 13 scenic spots.
0 : 北门外
1 : 旧操场
2 : 北区餐厅
3 : 新操场
4 : 体育馆
5 : 西门
6 : 凯旋门
7 : 九蛋广场
8 : 教学区
9 : 行远楼
10: 信院
11: 图书馆
12: 五子顶

请输入起点和目的地的序号：(不在序号范围则退出该功能)0 3
最短距离是：93
最短路径是：0北门外 -> 2北区餐厅 -> 3新操场

继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能：3 0
最短距离是：93
最短路径是：3新操场 -> 2北区餐厅 -> 0北门外
```

```
继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能：0 9
最短距离是：162
最短路径是：0北门外 -> 2北区餐厅 -> 8教学区 -> 9行远楼
```

```
继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能：9 0
最短距离是：162
最短路径是：9行远楼 -> 8教学区 -> 2北区餐厅 -> 0北门外
```

### 3、查询最短路径（从 v1 到 v2 只需一步）

```
继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能：0 5
最短距离是：105
最短路径是：0北门外 -> 5西门
```

```
继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能：2 0
最短距离是：52
最短路径是：2北区餐厅 -> 0北门外
```

### 4、查询最短路径（原地，始终点一致的情况）

```
继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能：0 0
原地！
```

```
继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能：8 8
原地！
```

### 5、查询最短路径（无解情况，为了方便展示，将把图中序号为 0 的点变成孤立点）

### 5、退出查询最短路径功能和程序（始终点只要有一个不在序号范围内即可）

```
继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能：9 0  
无法达到！
```

```
继续查询请直接输入起点和目的地序号，不在序号范围则退出该功能：18 0
```

F:\数据结构2016年春季学期\实验\【实验四】校园导游咨询\CollageGuide.exe

```
-----  
本系统提供两个功能：
```

- 1、查询某个景点的简介。
- 2、查询两景点间最短距离和路径。

```
-----  
继续使用请输入功能序号(1 or 2)，退出则输入其他数字：9
```

```
Thanks! Good Bye!
```

```
Process returned 0 (0x0)   execution time : 150.805 s
```

```
Press any key to continue.
```

## 五、实验总结：

- 1、通过实验进一步掌握了 Dijkstra 算法。除了用 Dijkstra 算法求最短路径外，还可以用 Floyd 算法求解，总得执行时间是都是  $O(n^3)$ 。后者形式上稍微简单点，我认为前者更易于理解。
- 2、在还原最短路径的时候实际上可以用另外一种方法：那就是把 Dijkstra 算法中的二维数组  $P$  用于存放路径顺序而不仅仅指示该点是否为最短路径上的点，这样需要解决的问题是为更新最短路径时要注意记录的顺序是否正确复制。
- 3、Dijkstra 算法书上有伪代码算法描述，只要理解后稍加修改就能完成 Dijkstra 算法的功能，实验难点是还原路径，这个需要借助 Dijkstra 算法中的  $P$  和  $D$  数组，并对其进行处理。实际实验的时候使用过不少方法，包括上述第二点说的，但是发现更新最短路径时序号并非简单的复制即可，多少会有整体偏移的情况。最终选择的方法虽然易于理解但是实现起来仍有一定的难度。对存放最短路径的数组  $a$  排序，原理和简单插入排序相同，但是，寻找插入位置时条件有点混乱，对数组  $a$  进行插入排序，排序条件却是以  $a$  中元素作为顶点序号，分别合适地放入邻接矩阵和数组  $P$  中，再用选择排序的思想挑选出权值最小的弧对应的顶点，之后找出该顶点在数组  $a$  中对应的下边，然后进行交换，其实实现起来很麻烦的，由于采用的编译环境在不建立工程工程的情况下是无法进行调试的，当时调试的时候不停加入输出提示来观察数组  $a$  排序是否正确。
- 4、特殊例子的考虑并非简单，需要大量的测试才可能发现，就例如找出权值最大的弧这一点。还有之所以先初始化数组  $a$  再排序而不是选择使用插入排序来初始化  $a$  就是因为当时发现从序号大的到序号小的地方输出的路径和从序号小的到序号大的路径顺序是一样的，因为循环时习惯从小到大开始循环。
- 5、学会了清屏操作，在 `window.h` 库中，语句 `system("cls")`，即能清屏，能使界面更加美观。