

《数据结构》实验报告4

班级: 10012006

姓名: 夏卓

学号: 2020303245

E-mail: 2769223717@qq.com

日期: 2022.5.21

4.1 求赋权图中一个结点到所有结点的最短路径的长度

实验内容:

实验4.1: 求赋权图中一个结点到所有结点的最短路径的长度

Time Limit: 3000ms, Memory Limit: 10000KB, Accepted: 0, Total Submissions: 0

VIDEO1

Description

给一个赋权图（无向图），求0号结点到其余所有结点的最短路径的长度。

Input

先输入一个小于等于100的正整数n，然后输入赋权图的邻接矩阵（10000表示无穷大,并且任意一条简单路径的长度都小于10000）

Output

按结点编号的顺序输出0号结点所有结点的最短路径的长度。

Sample Input

```
6
0?1?4?10000?10000?10000
1?0?2?7?5?10000
4?2?0?10000?1?10000
10000?7?10000?0?3?2
10000?5?1?3?0?6
10000?10000?10000?2?6?0
```

Sample Output

```
0
1
3
7
4
9
```

一、需求分析:

1. 输入:

先输入一个小于等于100的正整数n，然后输入赋权图的邻接矩阵（10000表示无穷大,并且任意一条简单路径的长度都小于10000）

2. 输出:

按结点编号的顺序输出0号结点到所有结点的最短路径的长度。

3. 程序所能达到的功能：

给一个赋权图（无向图），求0号结点到其余所有结点的最短路径的长度。

二、概要设计：

核心思想：

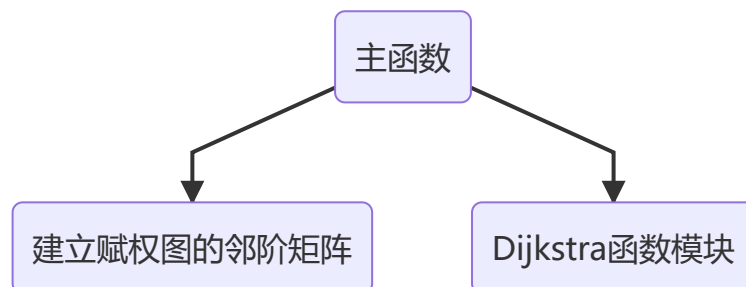
- 用迪杰斯特拉算法求0号结点到其他各结点的最短距离
- 具体步骤包括
 - （1）首先初始化集合，将0号结点加入集合中，并用0号结点到其他结点的距离更新最短路
 - （2）求出集合外结点中距离0号结点最近的结点，将该节点加入集合中
 - （3）用新加入的结点更新其余结点到0号结点的距离。
 - （4）重复（2）、(3)步骤，直到所有结点都被加入到集合之中

程序框架：

本程序包含三个模块：

1. 主程序模块；
2. 建立赋权图的邻阶矩阵模块；
3. Dijkstra函数模块

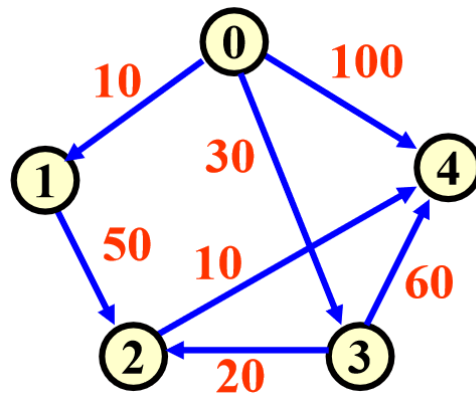
模块调用图：



三、详细设计：

算法实现图：

以以下赋权图为例：



则更新过程中各数组的更新变化如下：

	V1			V2			V3			V4		
	S[1]	d[1]	p[1]	S[2]	d[2]	p[2]	S[3]	d[3]	p[3]	S[4]	d[4]	p[4]
0	0	10	0	0	∞	0	0	30	0	0	100	0
1	1	10	0	0	60	1	0	30	0	0	100	0
3	1	10	0	0	50	3	1	30	0	0	90	3
2	1	10	0	1	50	3	1	30	0	0	60	2
4	1	10	0	1	50	3	1	30	0	1	60	2

核心算法的伪代码框架：

```

1 //数据结构(求最短路的邻阶矩阵图)
2 struct Graph
3 {
4     int d[NumVertices][NumVertices];    //存储各点之间的距离
5     int dis[NumVertices];    //存储各点到0号结点的最短路径长度
6     int S[NumVertices];    //维护集合
7 };
8
9 //核心算法
10 void Dijkstra(int n,int v)
11 {
12     for(int i=0;i<n;i++){
13         dis[i]=d[v][i];    //初始化dis数组为各点直接到v号结点的距离
14         S[i]=0;    //集合中最开始没有任何元素
15     }
16     S[v]=1; dis[v]=0;    //将v号结点加入到集合中
17     for(int i=1;i<n;i++){
18         int minn=MAX_NUM;int u=v;
19         //找出集合外距离v号结点最近的点
20         for(int j=0;j<n;j++){
21             if(S[j]==0 && dis[j]<minn){
22                 minn=dis[j];
23                 u=j;
24             }
25         }
26         S[u]=1; //将该点加入到集合之中
27         for(int j=0;j<n;j++)
28             if(S[j]==0 && dis[u]+Edge[j][u]<dis[j])

```

```

29         dis[j]=dis[u]+Edge[j][u];    //用该节点更新集合外的点到v号结点的距
    离
30     }
31 }

```

四、使用说明、测试分析及结果：

1. 说明如何使用你编写的程序

本程序的运行环境为visual studio 2019。

先输入一个小于等于100的正整数n，然后输入赋权图的邻接矩阵（10000表示无穷大,并且要保证任意一条简单路径的长度都小于10000）

程序会按结点编号的顺序输出0号结点到所有结点的最短路径的长度。


2. 测试结果与分析

本程序较好的实现了实验需求，经分析，Dijkstra算法的时间复杂度为 $O(n^2)$ ，其中 n 为结点的数量

3. 调试过程中遇到的问题及解决方法

- 无，注意需要先将源结点加入到集合中，之后只需要循环n-1次即可

4.运行界面

 Microsoft Visual Studio 调试控制台

```

6
0 1 4 10000 10000 10000
1 0 2 7 5 10000
4 2 0 10000 1 10000
10000 7 10000 0 3 2
10000 5 1 3 0 6
10000 10000 10000 2 6 0
0
1
3
7
4
9

```

D:\Code\VS code\homework\求赋权图中一个结点到所有结点的距离.exe (进程 22124) 已退出，代码为 0。
按任意键关闭此窗口. . .

五、实验总结

- 本实验我在编程中用时10分钟
- 在调试中用时2分钟
- 本题主要考察Dijkstra算法，并不难，记住关键步骤即可。主要需要维护集合保证加入集合中的元素是到源节点的最短路。

4.2 用迪杰斯特拉算法求赋权图中的最短路径

实验4.2：用迪杰斯特拉算法求赋权图中的最短路径

Time Limit: 3000ms , Memory Limit: 10000KB , Accepted: 0 , Total Submissions: 0

VIDEO 1

Description

用迪杰斯特拉算法求一点到其余所有结点的最短路径。

Input

先输入一个小于100的正整数n，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），最后输入两个0到n-1的整数表示两个点。

Output

先用迪杰斯特拉算法求给定的第一个点到其余所有结点的最短路径。
然后再输出给定的两个点之间的最短路径（按顺序输出最短路径上的每一个点，每个数据占一行）。

Sample Input

```
4
0?2?10?10000
2?0?7?3
10?7?0?6
10000?3?6?0
0?2
```

Sample Output

```
0
1
2
```

一、需求分析：

1. 输入：

先输入一个小于100的正整数n，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），最后输入两个0到n-1的整数表示两个点。

2. 输出：

先用迪杰斯特拉算法求给定的第一个点到其余所有结点的最短路径。然后再输出给定的两个点之间的最短路径(按顺序输出最短路径上的每一个点，每个数据占一行)。

3. 程序所能达到的功能：

用迪杰斯特拉算法求一点到其余所有结点的最短路径。

二、概要设计：

核心思想：

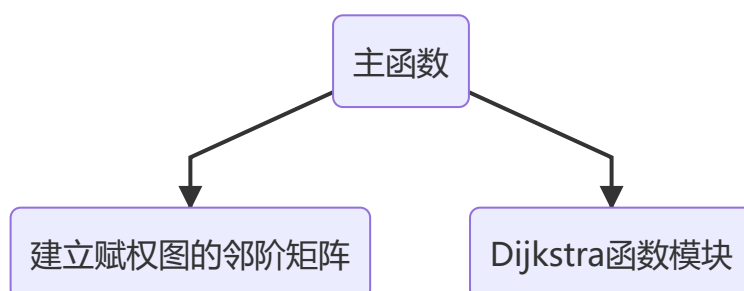
- 用迪杰斯特拉算法求各结点到0号结点的最短距离并记录下最短路径中该节点的前一个结点
- 具体步骤包括
 - (1) 首先初始化集合，将0号结点加入集合中，用0号结点到其他结点的距离更新最短路，并将除0号结点外所有结点到0号结点的路径Path[i]置为0，Path[0]=-1
 - (2) 求出集合外结点中距离0号结点最近的结点，将该节点加入集合中
 - (3) 用新加入的结点更新其余结点到0号结点的距离，若距离被更新，则将被更新结点的路径置为新加入的结点u。
 - (4) 重复 (2)、(3)步骤，直到所有结点都被加入到集合之中

程序框架：

本程序包含三个模块：

1. 主程序模块；
2. 建立赋权图的邻阶矩阵模块；
3. Dijkstra函数模块；

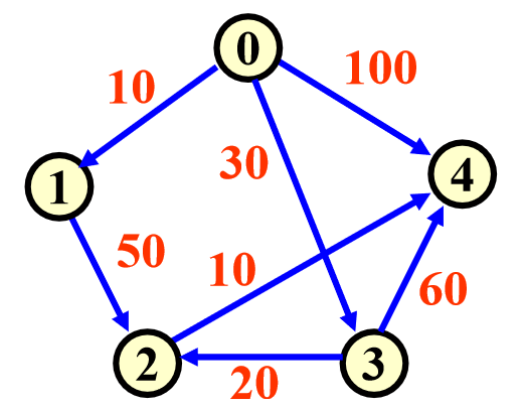
模块调用图：



三、详细设计：

算法实现图：

同样以第一题中的赋权图为例



则更新过程中各数组的更新变化如下：

	V1			V2			V3			V4		
	S[1]	d[1]	p[1]	S[2]	d[2]	p[2]	S[3]	d[3]	p[3]	S[4]	d[4]	p[4]
0	0	10	0	0	∞	0	0	30	0	0	100	0
1	1	10	0	0	60	1	0	30	0	0	100	0
3	1	10	0	0	50	3	1	30	0	0	90	3
2	1	10	0	1	50	3	1	30	0	0	60	2
4	1	10	0	1	50	3	1	30	0	1	60	2

核心算法的伪代码框架：

```
1 //数据结构(求最短路的邻阶矩阵图)
2 struct Graph
3 {
4     int d[NumVertices][NumVertices]; //存储各点之间的距离
5     int dis[NumVertices]; //存储各点到0号结点的最短路径长度
6     int Path[NumVertices]; //存储各节点到源节点的最短路径中该节点的前一个结点
7     int S[Numvertices]; //维护集合
8 };
9
10 //核心算法
11 void Dijkstra(int n,int v)
12 {
13     for(int i=0;i<n;i++){
14         dis[i]=d[v][i]; //初始化dis数组为各点直接到v号结点的距离
15         S[i]=0; //集合中最开始没有任何元素
16         if(i!=v && dis[i]<MAX_NUM) Path[i]=v; //将各结点到源节点的最短路径初始
            化为直接到源节点
17         else Path[i]=-1;
18     }
19     S[v]=1; dis[v]=0; //将v号结点加入到集合中
20     for(int i=1;i<n;i++){
21         int minn=MAX_NUM;int u=v;
22         //找出集合外距离v号结点最近的点
23         for(int j=0;j<n;j++){
24             if(S[j]==0 && dis[j]<minn){
25                 minn=dis[j];
26                 u=j;
27             }
28         }
29         S[u]=1; //将该点加入到集合之中
30         for(int j=0;j<n;j++)
31             if(S[j]==0 && dis[u]+Edge[j][u]<dis[j])
32             {
33                 dis[j]=dis[u]+Edge[j][u]; //用该节点更新集合外的点到v号结点的距
                    离
34                 Path[j]=u; //被更新结点的最短路径中该节点的前一个结点为u
35             }
36     }
```

四、使用说明、测试分析及结果：

1. 说明如何使用你编写的程序

本程序的运行环境为visual studio 2019。

先输入一个小于100的正整数 n ，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），最后输入两个0到 $n-1$ 的整数表示两个点。

程序会输出给定的两个点之间的最短路径(按顺序输出最短路径上的每一个点，每个数据占一行)。

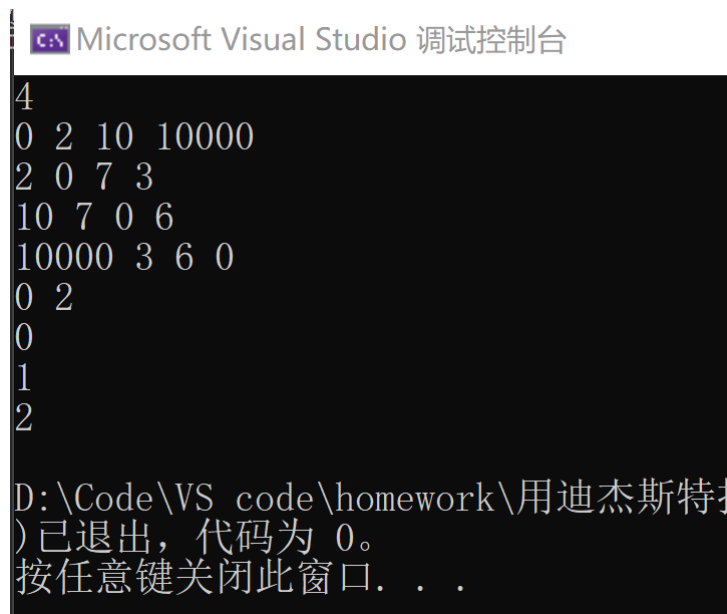
2. 测试结果与分析

本程序较好的实现了实验需求，经分析可知，算法的时间复杂度为 $O(n^2)$ ，其中 n 为结点的数量。

3. 调试过程中遇到的问题及解决方法

- 无，与上一题类似，只需记住Path数组是记录各节点到源节点的最短路径中该节点的前一个结点即可

4.运行界面



```

Microsoft Visual Studio 调试控制台
4
0 2 10 10000
2 0 7 3
10 7 0 6
10000 3 6 0
0 2
0
1
2

D:\Code\VS_code\homework\用迪杰斯特拉
)已退出，代码为 0。
按任意键关闭此窗口. . .
  
```

五、实验总结

- 本实验我在编程中用时10分钟
- 在调试中用时2分钟
- 本题较为简单，与上一题类似，只需记住Path数组是记录各节点到源节点的最短路径中该节点的前一个结点即可

4.3 用弗洛伊德算法求赋权图的两点间的最短路径长度

实验4.3：用弗洛伊德算法求赋权图的两点间的最短路径的长度。

Time Limit: 3000ms , Memory Limit: 10000KB , Accepted: 0 , Total Submissions: 0

VIDEO1

Description

用弗洛伊德算法求任意两点间的最短路径的长度

Input

先输入一个小于100的正整数n，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），之后再输入一个小于100的正整数m，最后的m行每行输入两个不同的0到n-1之间的整数表示两个点。

Output

用弗洛伊德算法求任意两点间的最短路径的长度，并输出这些两个点之间的最短路径的长度。

Sample Input

```
4
0?2?10?10000
2?0?7?3
10?7?0?6
10000?3?6?0
2
0?2
3?0
```

Sample Output

```
9
5
```

一、需求分析：

1. 输入：

先输入一个小于100的正整数n，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），之后再输入一个小于100的正整数m，最后的m行每行输入两个不同的0到n-1之间的整数表示两个点。

2. 输出：

用弗洛伊德算法求任意两点间的最短路径的长度，并输出这些两个点之间的最短路径的长度。

3. 程序所能达到的功能：

用弗洛伊德算法求任意两点间的最短路径的长度

二、概要设计：

核心思想：

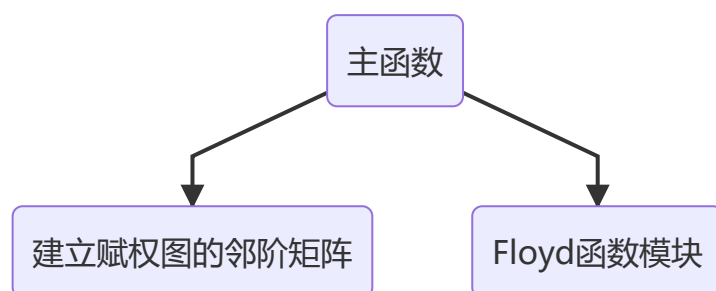
- 利用 *Floyd* 算法求各节点之间的最短路径长度
- 具体步骤包括：
 - 循环遍历三次各个结点，
 - 第一重循环遍历的是中间结点，
 - 第二重循环遍历的是源点，
 - 第三重循环遍历的是终点，
 - 用中间结点更新源点到终点的距离

程序框架：

本程序包含三个模块：

1. 主程序模块；
2. 建立赋权图的邻阶矩阵模块；
3. Floyd函数模块

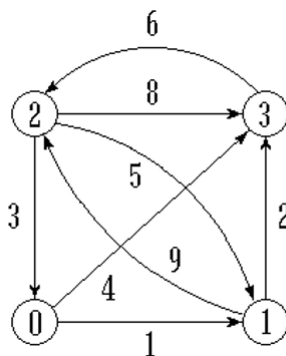
模块调用图：



三、详细设计：

算法实现图：

以以下赋权图为例



赋权图的邻接矩阵如下

$$\begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{pmatrix} 0 & 1 & \infty & 4 \\ \infty & 0 & 9 & 2 \\ 3 & 5 & 0 & 8 \\ \infty & \infty & 6 & 0 \end{pmatrix} & \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} \end{matrix}$$

Floyd算法的实现图如下

	$A^{(-1)}$				$A^{(0)}$				$A^{(1)}$				$A^{(2)}$				$A^{(3)}$			
	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
0	0	1	∞	4	0	1	∞	4	0	1	10	3	0	1	10	3	0	1	9	3
1	∞	0	9	2	∞	0	9	2	∞	0	9	2	12	0	9	2	11	0	8	2
2	3	5	0	8	3	4	0	7	3	4	0	6	3	4	0	6	3	4	0	6
3	∞	∞	6	0	∞	∞	6	0	∞	∞	6	0	9	10	6	0	9	10	6	0

核心算法的伪代码框架

```

1 //数据结构(求各结点间最短路径长度的邻阶矩阵图)
2 struct Graph
3 {
4     int d[NumVertices][NumVertices];    //存储各点之间的距离
5 };
6
7 //核心算法
8 void Floyd(int n){
9     for (int i = 0; i < n; i++)
10         for (int j = 0; j < n; j++)
11             cin >> d[i][j];
12
13     for (int k = 0; k < n; k++) //第一重循环遍历的是中间结点
14         for (int i = 0; i < n; i++) //第二重循环遍历的是源点
15             for (int j = 0; j < n; j++) //第一重循环遍历的是终点
16                 d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
17 }

```

四、使用说明、测试分析及结果：

1. 说明如何使用你编写的程序

本程序的运行环境为visual studio 2019。

先输入一个小于100的正整数n，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），之后再输入一个小于100的正整数m，最后的m行每行输入两个不同的0到n-1之间的整数表示两个点。

程序会输出这两个点之间的最短路径的长度。


2. 测试结果与分析

本程序较好的完成了实验需求，经分析，算法的时间复杂度为 $O(n^3)$ ，其中 n 为结点的数量。

3. 调试过程中遇到的问题及解决方法

- 无，注意第一重循环是遍历的中间结点即可。

4.运行界面

 Microsoft Visual Studio 调试控制台

```
4
0 2 10 10000
2 0 7 3
10 7 0 6
10000 3 6 0
2
0 2
3 0
9
5
D:\Code\VS_code\homework\用弗洛伊德算法
长度.exe (进程 25528) 已退出，代码为 0。
按任意键关闭此窗口. . .
```

五、实验总结

- 本实验我在编程中花费6分钟
- 在调试中用时1分钟
- 主要需要注意遍历的顺序。
第一重循环遍历的是中间结点，
第二重循环遍历的是源点，
第三重循环遍历的是终点，
用中间结点更新源点到终点的距离

4.4 用弗洛伊德算法求赋权图中任意两点间的最短路径

实验4.4：用弗洛伊德算法求赋权图中任意两点间的最短路径。

Time Limit: 3000ms , Memory Limit: 10000KB , Accepted: 0 , Total Submissions: 0

VIDEO1

Description

用弗洛伊德算法求任意两点间的最短路径，并输出指定的m对结点间的最短路径。

Input

先输入一个小于100的正整数n，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），之后再输入一个小于100的正整数m，最后的m行每行输入两个不同的0到n-1之间的整数表示两个点。

Output

用弗洛伊德算法求任意两点间的最短路径，并输出这些两个点之间的最短路径。

Sample Input

```
4
0 2 10 10000
2 0 7 3
10 7 0 6
10000 3 6 0
2
0 2
3 0
```

Sample Output

```
0
1
2
3
1
0
```

一、需求分析：

1. 输入：

先输入一个小于100的正整数n，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），之后再输入一个小于100的正整数m，最后的m行每行输入两个不同的0到n-1之间的整数表示两个点。

2. 输出：

用弗洛伊德算法求任意两点间的最短路径，并输出这些两个点之间的最短路径。

3. 程序所能达到的功能：

用弗洛伊德算法求任意两点间的最短路径，并输出指定的m对结点间的最短路径。

二、概要设计：

核心思想：

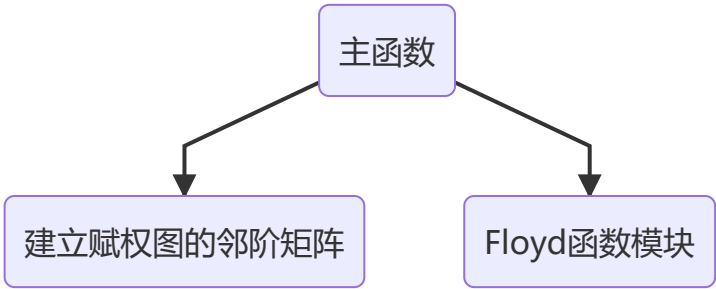
- 利用Floyd算法求各节点之间的最短路径
- 具体步骤包括：
 - 循环遍历三次各个结点，

第一重循环遍历的是中间结点，
第二重循环遍历的是源点，
第三重循环遍历的是终点，
用中间结点更新源点到终点的距离，若更新成功，则终点到源点的前一个结点为中间结点

程序框架：

- 本程序包含三个模块：
- 1. 主程序模块；
 - 2. 建立赋权图的邻阶矩阵模块；
 - 3. Floyd函数模块

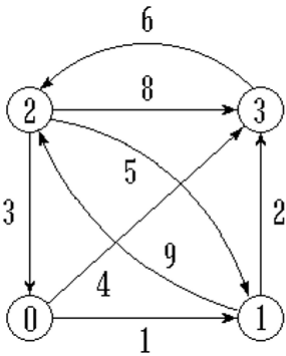
模块调用图：



三、详细设计：

算法实现图：

同样以以下赋权图为例



赋权图的邻接矩阵如下

	0	1	2	3	
0	0	1	∞	4	0
1	∞	0	9	2	1
2	3	5	0	8	2
3	∞	∞	6	0	3

Floyd算法的实现图如下

四、使用说明、测试分析及结果：

1. 说明如何使用你编写的程序

本程序的运行环境为visual studio 2019。

先输入一个小于100的正整数 n ，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），之后再输入一个小于100的正整数 m ，最后的 m 行每行输入两个不同的0到 $n-1$ 之间的整数表示两个点。

程序会输出这些两个点之间的最短路径。

2. 测试结果与分析

本程序较好的完成了实验需求，经分析，该算法的时间复杂度为 $O(n^3)$ ，其中 n 为结点的数量。

3. 调试过程中遇到的问题及解决方法

- 无，只需注意Path中存储的是终点到源点的最短路径中的前一个结点即可

4.运行界面



```
Microsoft Visual Studio 调试控制台
4
0 2 10 10000
2 0 7 3
10 7 0 6
10000 3 6 0
2
0 2
0
1
2
3 0
3
1
0

D:\Code\VS code\homework\用弗洛伊德
.exe (进程 16760) 已退出，代码为 0。
```

五、实验总结

- 本实验我在编程中花费8分钟
- 在调试中花费1分钟
- 注意Path中存储的是终点到源点的最短路径中的前一个结点即可