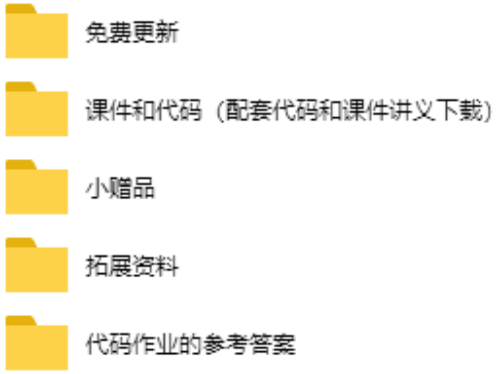
温馨提示

（1）视频中提到的附件可在**售后群的群文件**中下载。

包括**讲义、代码、优秀的作业、我视频中推荐的资料**等。



（2）关注我的**微信公众号《数学建模学习交流》**，后台发送**“软件”**两个字，可获得常见的建模软件下载方法；发送**“数据”**两个字，可获得建模数据的获取方法；发送**“画图”**两个字，可获得数学建模中常见的画图方法。另外，也可以看看公众号的历史文章，里面发布的都是对大家有帮助的技巧。

（3）**购买更多优质精选的数学建模资料**，可关注我的微信公众号《数学建模学习交流》，在后台发送**“买”**这个字即可进入店铺进行购买。

（4）视频价格不贵，但价值很高。单人购买观看只需要**58元**，和另外两名队友一起购买人均仅需**46元**，视频本身也是下载到本地观看的，所以请大家**不要侵犯知识产权**，对视频或者资料进行二次销售。

数学建模学习交流

本节可配合第八讲观看

Floyd算法（弗洛伊德算法）

Floyd-Warshall算法（英语：Floyd-Warshall algorithm或简写为Floyd algorithm），中文亦称弗洛伊德算法，是解决**任意两点间**的最短路径的一种算法，可以正确处理无向图或有向图（可以有负权重，但不可存在负权回路）的最短路径问题。

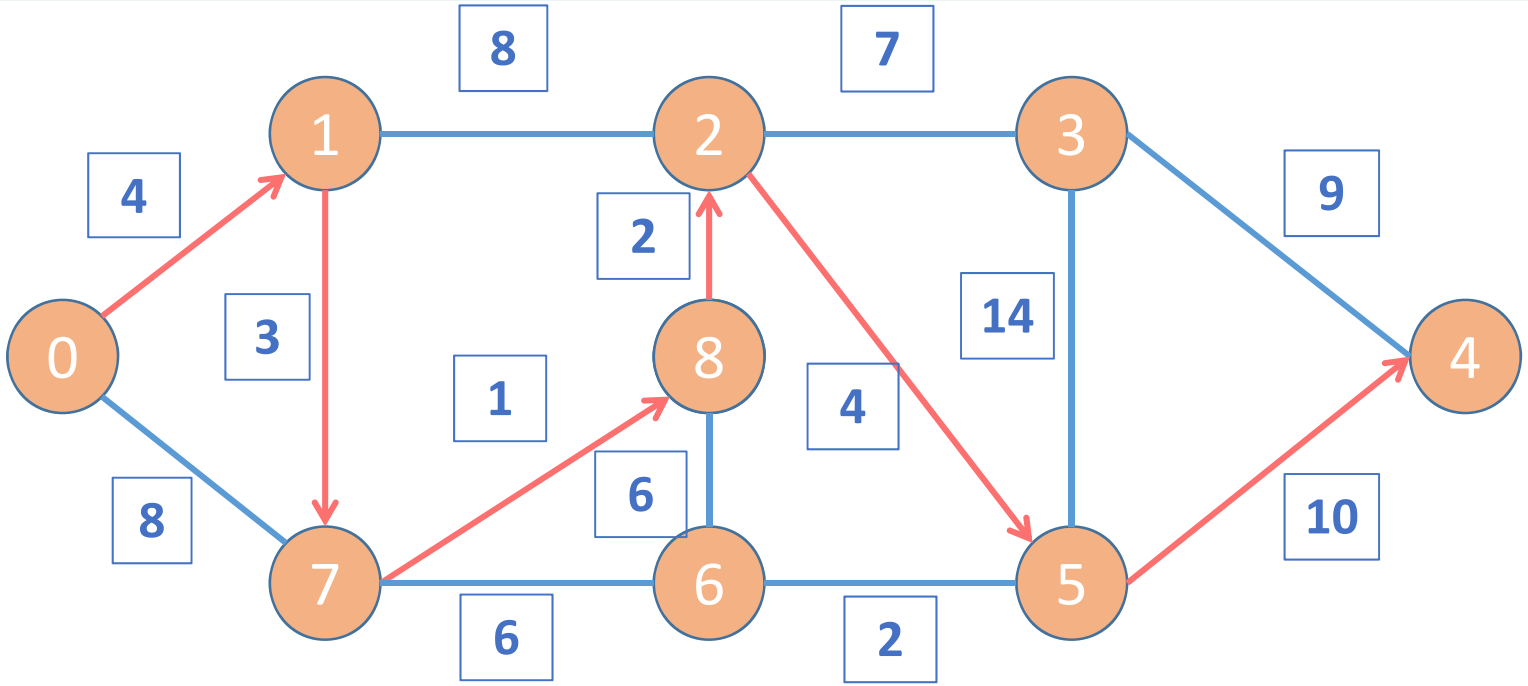
Floyd算法与迪杰斯特拉算法或贝尔曼福特算法相比，能够一次性的求出任意两点之间的最短路径，后两种算法运行一次只能计算出给定的起点和终点之间的最短路径。

当然，Floyd算法计算的时间也要高于后两种算法，其算法核心的步骤由三层循环构成，**现在请大家先看完下面这个视频的后半段，时长五分钟，看完之后再来看后面的内容。**

视频地址：https://www.bilibili.com/video/av54668527

数学建模学习交流

最短路径应满足的结论



从上图中不难得到以下两个结论：

（1）如果某个节点（例如点8）位于从起点0到终点4的最短路径上，那么：

从0到4的最短路径的距离 = 从0到8的最短路径的距离 +从8到4的最短路径的距离

（2）如果某个节点（例如点3）不在从起点0到终点4的最短路径上，那么：

从0到4的最短路径的距离 ≤ 从0到3的最短路径的距离 +从3到4的最短路径的距离

（注：这里写≤号是因为我们最终求出来的最短路径的走法可能不唯一）

数学建模学习交流

清风"数学建模算法讲解"全套视频，单人购买观看仅需58元

Floyd算法（弗洛伊德算法）

从上面观察到的两个结论中，我们不难提炼出下面这个思想：
假设现在有一个起点A和终点B，那么对于其他任意的中间点M：

$$D(A,B) \le D(A,M) + D(M,B)$$

这里，D(X,Y)表示X和Y两点之间的最短距离。

因此，Floyd算法实际上核心在于一个三层循环，下面给出**伪代码**：

```
1 let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
2 for each vertex v
3    dist[v][v] ← 0
4 for each edge (u,v)
5    dist[u][v] ← w(u,v)  // the weight of the edge (u,v)
6 for k from 1 to |V|
7    for i from 1 to |V|
8       for j from 1 to |V|
9          if dist[i][j] > dist[i][k] + dist[k][j]
10            dist[i][j] ← dist[i][k] + dist[k][j]
11  end if
```

(引用来源：维基百科)

数学建模学习交流

关注微信公众号"数学建模学习交流"获取更多优质资料

伪代码详解

```
1 let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
```
V是顶点的集合， |V|表示顶点的个数，首先我们初始化最小距离矩阵dist，其中每一个元素都是Inf.
```
2 for each vertex v
3     dist[v][v] ← 0
```
将dist矩阵的主对角线元素变为0.(相同的点的最短距离当然是0喽)
```
4 for each edge (u,v)
5     dist[u][v] ← w(u,v)  // the weight of the edge (u,v)
```
如果u,v两个顶点之间有权重，则用权重更新最短距离矩阵.
（事实上，1-5步就是在生成一个权重邻接矩阵）
```
6 for k from 1 to |V|
```
中间节点k从1- |V| 循环
```
7    for i from 1 to |V|
```
起始节点i从1- |V| 循环
```
8      for j from 1 to |V|
```
终点节点j从1- |V| 循环
```
9     if dist[i][j] > dist[i][k] + dist[k][j]
```
如果i,j两个节点间的最短距离大于i和k的最短距离+k和j的最短距离
```
10   dist[i][j] ← dist[i][k] + dist[k][j]
```
那么我们就令这两个较短的距离之和取代i,j两点之间的最短距离
```
11   end if
```
结束if循环

数学建模学习交流

思考：怎么记录最短路径经过的点？

上一页的伪代码可以实现计算任意两点之间的最短路径的距离，那么：
这个最短路径经过哪些点我们能否通过另一个矩阵记录下来呢？
在视频中：https://www.bilibili.com/video/av54668527，我们确实看到了可通过一个路径矩阵path（视频中用的符号是S）来记录下这个过程。
下面我们就在我们的伪代码中加入这一功能：

```
1 let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
2 for each vertex v
3    dist[v][v] ← 0
4 for each edge (u,v)
5     dist[u][v] ← w(u,v)  // the weight of the edge (u,v)
```

在这里初始化路径矩阵path(里面的每一个元素用终点填充，即path_ij=j，另外，我们可以令主对角线元素为-1)，path是路径矩阵，其元素path_ij表示起点为i，终点为j的两个节点之间的最短路径要经过的节点

```
6 for k from 1 to |V|
7    for i from 1 to |V|
8       for j from 1 to |V|
9          if dist[i][j] > dist[i][k] + dist[k][j]
10           dist[i][j] ← dist[i][k] + dist[k][j]
```

在这个if语句中加入一行：path[i][j] ← path[i][k]

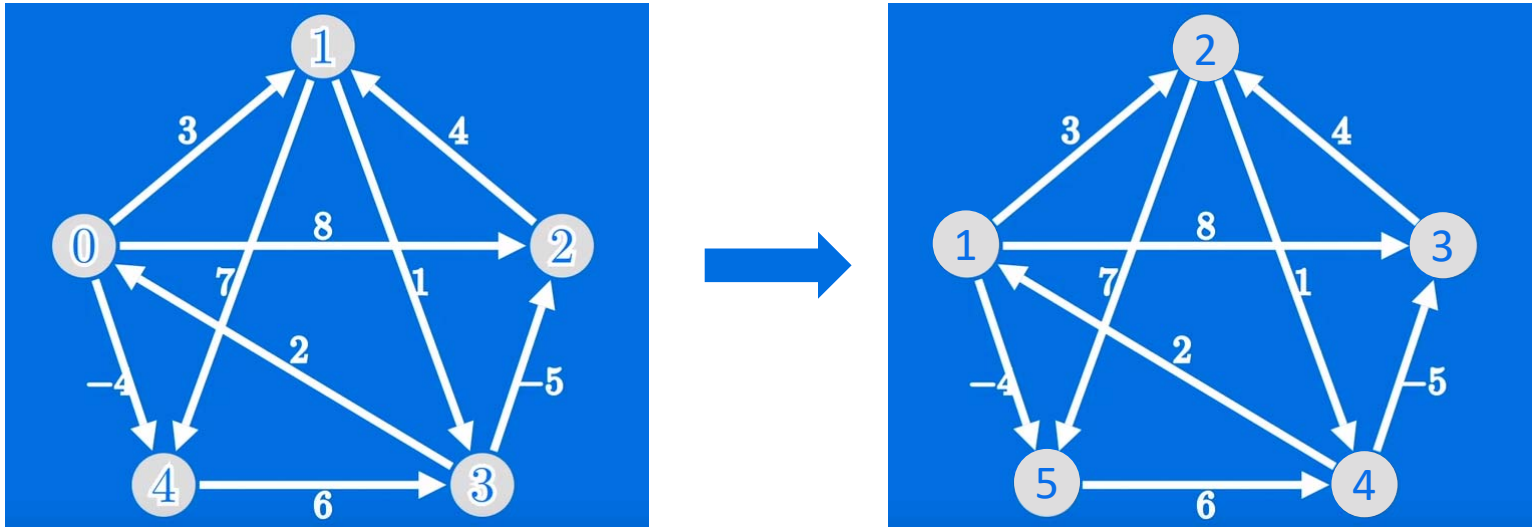
```
11         end if
```

数学建模学习交流

将伪代码转换为Matlab代码

Floyd_algorithm.m

```
%% 该函数用于求解一个权重邻接矩阵任意两个节点之间的最短路径
% 输入:
%       D是权重邻接矩阵
% 输出:
%       dist是最短距离矩阵，其元素dist_ij表示表示i,j两个节点的最短距离
%       path是路径矩阵，其元素path_ij表示起点为i，终点为j的两个节点之间
        的最短路径要经过的节点
```



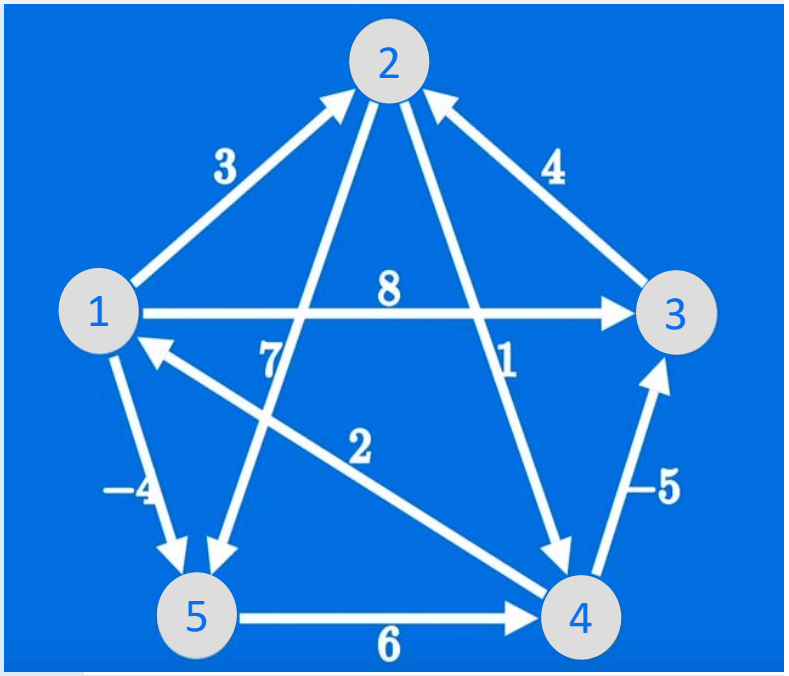
图中节点标号从0开始，在Matlab中下标从1开始，因此我们对图中编号进行变换

数学建模学习交流

运行结果展示

code1.m

```
%% 首先将图转换为权重邻接矩阵D
n = 5;  %一共五个节点
D = ones(n)./zeros(n); % 全部元素初始化为Inf
for i = 1:n
    D(i,i) = 0;  % 主对角线元素为0
end
D(1,2) = 3;
D(1,3) = 8;
D(1,5) = -4;
D(2,5) = 7;
D(2,4) = 1;
D(3,2) = 4;
D(4,3) = -5;
D(5,4) = 6;
D(4,1) = 2;
```




```
%% 调用Floyd_algorithm函数求解
[dist,path] = Floyd_algorithm(D)
```

dist =

0	1	-3	2	-4
3	0	-4	1	-1
7	4	0	5	3
2	-1	-5	0	-2
8	5	1	6	0

path =

-1	5	5	5	5
4	-1	4	4	4
2	2	-1	2	2
1	3	3	-1	1
4	4	4	4	-1

dist矩阵的含义很好解释，
那么path矩阵呢？

数学建模学习交流

path矩阵怎么看？

从3到1的最短路径为： 3 ---> 2 ---> 4 ---> 1

path =

-1	5	5	5	5
4	-1	4	4	4
2	2	-1	2	2
1	3	3	-1	1
4	4	4	4	-1

Path矩阵

path =

-1	5	5	5	5
4	-1	4	4	4
2	2	-1	2	2
1	3	3	-1	1
4	4	4	4	-1

找path(3,1)=2

path =

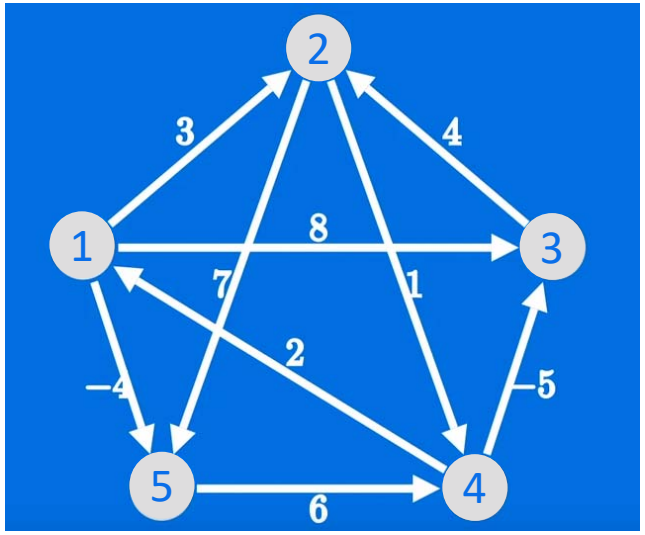
-1	5	5	5	5
4	-1	4	4	4
2	2	-1	2	2
1	3	3	-1	1
4	4	4	4	-1

找path(2,1)=4

path =

-1	5	5	5	5
4	-1	4	4	4
2	2	-1	2	2
1	3	3	-1	1
4	4	4	4	-1

找path(4,1)=1



情况一：如果path(i,j)等于j，则有两种可能：

（1）如果dist(i,j) 为 Inf，则说明从i到j没有路径可以到达；

（2）如果dist(i,j) 不为 Inf，则说明从i到j可直接到达，且为最短路径。

情况二：如果path(i,j)不等于j，等于k：

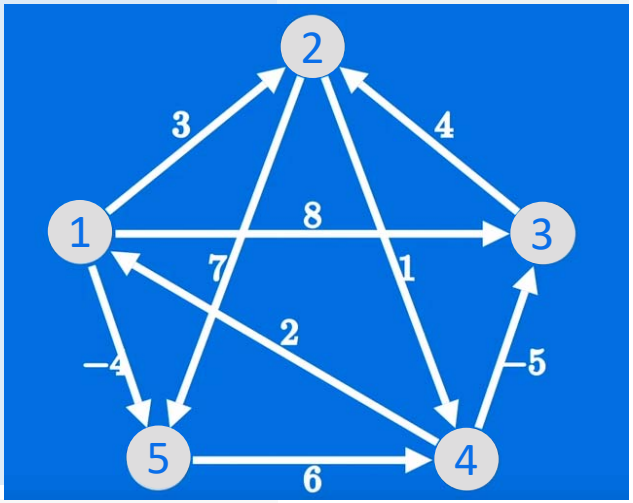
这意味这从i到j的最短路径上要先从i经过k点，接着我们需要判断path(k,j)是否等于j，如果等于j则下一步直接从k点走到j点；否则就重复这个步骤循环下去，直到走到j点结束。

数学建模学习交流

打印指定两点间最短路径

print_path.m

```
function [] = print_path(path,dist,i,j)
%% 该函数的作用是打印从i到j经过的最短路径
% 输入:
%       path是使用floyd算法求出来的路径矩阵
%       dist是使用floyd算法求出来的最短距离矩阵
%       i是起始节点的编号
%       j是终点节点的编号
% 输出: 无
```



dist =

0	1	-3	2	-4
3	0	-4	1	-1
7	4	0	5	3
2	-1	-5	0	-2
8	5	1	6	0

path =

-1	5	5	5	5
4	-1	4	4	4
2	2	-1	2	2
1	3	3	-1	1
4	4	4	4	-1

```
>> print_path(path,dist,1,5)
从1到5的最短路径为
1 ---> 5
最短距离为-4
>> print_path(path,dist,1,4)
从1到4的最短路径为
1 ---> 5 ---> 4
最短距离为2
>> print_path(path,dist,3,1)
从3到1的最短路径为
3 ---> 2 ---> 4 ---> 1
最短距离为7
```

数学建模学习交流

代码里一句话的说明

在Floyd_algorithm函数中，我写了这样一句话：

```
for k=1:n    % 中间节点k从1- n 循环
  for i=1:n     % 起始节点i从1- n 循环
    for j=1:n    % 终点节点j从1-n 循环
      if dist(i,j)>dist(i,k)+dist(k,j)  % 如果i,j两个节点间的最短距离大于i和k的最短距离+k和j的最短距离
        dist(i,j)=dist(i,k)+dist(k,j);  % 那么我们就令这两个较短的距离之和取代i,j两点之间的最短距离
        path(i,j)=path(i,k);   % 起点为i，终点为j的两个节点之间的最短路径要经过的节点更新为path(i,k)
      % 注意，上面一行语句不能写成path(i,j) = k; 这是网上很多地方都容易犯的错误，在PPT11页中会告
      end
    end
  end
end
```

大家可以把这一句改成错误的写法，然后运行code1.m文件，得到的结果如下所示：

dist =

0	1	-3	2	-4
3	0	-4	1	-1
7	4	0	5	3
2	-1	-5	0	-2
8	5	1	6	0

path =

-1	5	5	5	5
4	-1	4	4	4
4	2	-1	2	4
1	3	3	-1	1
4	4	4	4	-1

dist矩阵和原来的相同
path矩阵的部分元素和原来不同

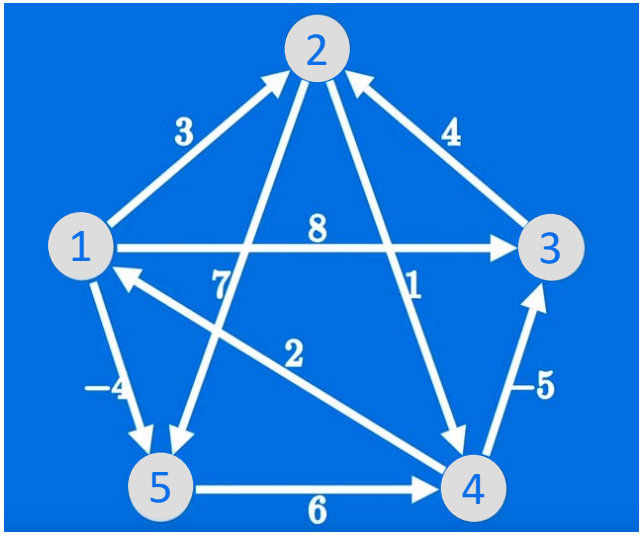
数学建模学习交流

为什么path不同？

从3到1的最短路径为： 3 ---> 2 ---> 4 ---> 1

```
path =

    -1     5     5     5     5
     4    -1     4     4     4
     2     2    -1     2     2
     1     3     3    -1     1
     4     4     4     4    -1
```

```
if dist(i,j)>dist(i,k)+dist(k,j)
    dist(i,j)=dist(i,k)+dist(k,j);
    path(i,j) = path(i,k); %正确的
end
```




```
path =

    -1     5     5     5     5
     4    -1     4     4     4
     4     2    -1     2     4
     1     3     3    -1     1
     4     4     4     4    -1
```

```
if dist(i,j)>dist(i,k)+dist(k,j)
    dist(i,j)=dist(i,k)+dist(k,j);
    path(i,j) = k; %错误的
end
```

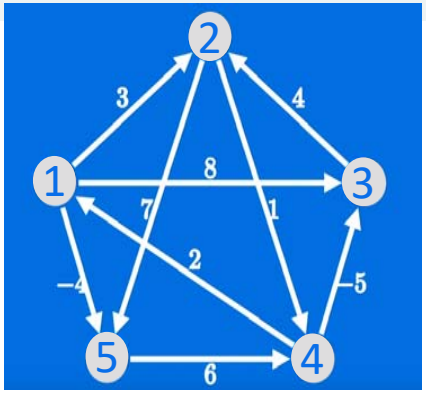
如果按照下面这个错误代码来解释path矩阵，那么注意到path(3,1)=4，这意味着要从起点3到达终点1的最短路径上，必须先由起点3到达中间点4，但是我们如果直接从图中来看的话，从3直接到4的是不可能的，距离为Inf，只能先从3到2，再从2到4。这说明，下面这个代码得到的结果不能揭示中间具体每一步的走法，所以我们正确的代码要改为上面这个。

数学建模学习交流

打印任意两点间最短路径

print_all_path.m

```
function [] = print_all_path(D)
%% 该函数的作用是求解一个权重邻接矩阵任意两个节点之间的最短路径,并打印所有的结果出来
% 输入:
%       D是权重邻接矩阵
% 输出：无
[dist,path] = Floyd_algorithm(D);  % 调用之前的Floyd_algorithm函数
n = size(D,1);
if n == 1
   warning('请输入至少两阶以上的权重邻接矩阵')  % 在屏幕中提示警告信息
   return;  % 不运行下面的语句，直接退出函数
end
for i = 1:n
   for j = 1:n
      if i ~= j  % 不等号用~=表示
         print_path(path,dist,i,j);  % 调用之前的print_path函数
         disp('----------------------------------------')
         disp(' ')
      end
   end
end
end
```

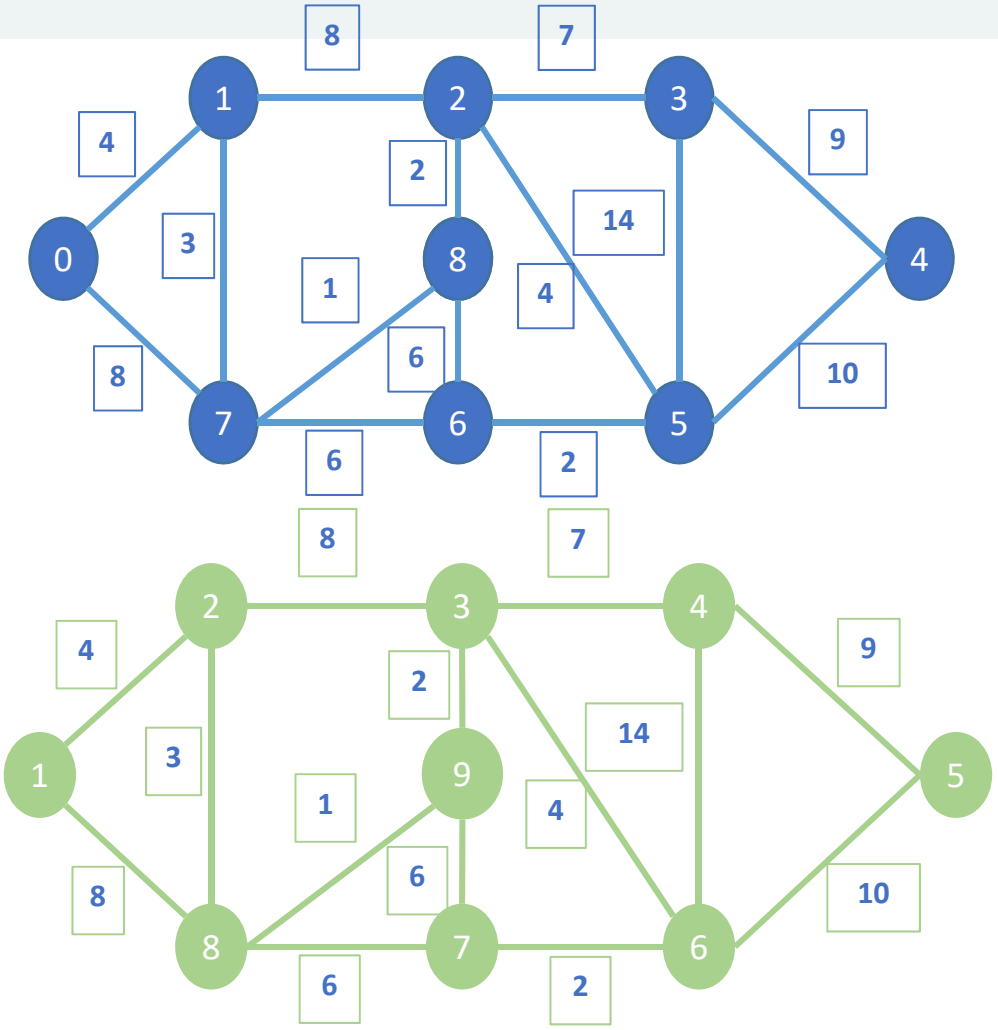


部分结果展示

```
从1到2的最短路径为
1 ---> 5 ---> 4 ---> 3 ---> 2
最短距离为1
----------------------------------

从1到3的最短路径为
1 ---> 5 ---> 4 ---> 3
最短距离为-3
----------------------------------

从1到4的最短路径为
1 ---> 5 ---> 4
最短距离为2
----------------------------------

从1到5的最短路径为
1 ---> 5
最短距离为-4
----------------------------------
```

数学建模学习交流

思考题：求出任意两点间的最短路径



图中节点标号从0开始，在Matlab中下标从1开始，因此我们对上图编号都加1
(另一个变换方法：直接将上图的0改为9哦)

数学建模学习交流

思考题参考答案

code2.m

```
% 参考答案
%% 首先将图转换为权重邻接矩阵D
n = 9;  %一共五个节点
D = zeros(n);  % 全部元素初始化为0, 等会你们就知道为什么这样设置啦
% 因为是无向图，所以权重邻接矩阵是一个对称矩阵
D(1,2) = 4; D(1,8) = 8;
D(2,8) = 3; D(2,3) = 8;
D(8,9) = 1; D(8,7) = 6;
D(9,7) = 6; D(9,3) = 2;
D(7,6) = 2; D(3,4) = 7;
D(3,6) = 4; D(6,4) = 14;
D(4,5) = 9; D(6,5) = 10;
D = D+D';   % 这个操作可以得到对称矩阵的另一半
for i = 1:n
    for j = 1:n
        if (i ~= j) && (D(i,j) == 0)
            D(i,j) = Inf;   % 将非主对角线上的0元素全部变为Inf
        end
    end
end
%% 调用Floyd_algorithm函数求解
[dist,path] = Floyd_algorithm(D)
```

数学建模学习交流