数模加油站

数学建模模型算法精讲课——

图论最短路径

—— 江北老师

继续跑 带着赤子的骄傲

生命的闪耀不坚持到底怎能看到

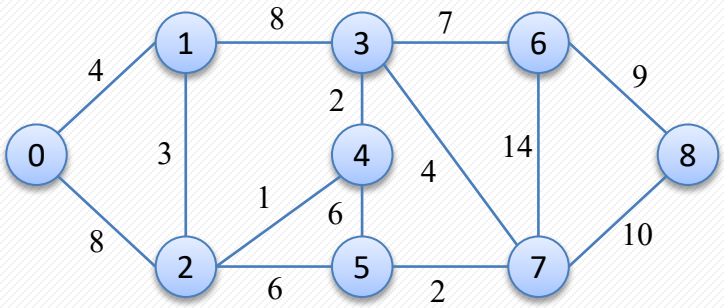
有兴趣的小伙伴可以关注微信公众号或加入建模交流群获取更多免费资料　　公众号：数模加油站　　交流群：709718660

图论最短路径

- 算法引出
- 算法原理
- Matlab代码

图论最短路径——算法引出

➢ KK明星又有问题需要大家帮忙了



KK明星和粉丝谈恋爱后，一直分居两地，KK住在城市0，粉丝女朋友住在城市8，女朋友马上要过生日了，KK想给女朋友一个惊喜，想要以最快的速度赶到女朋友那里，但城市0到城市8的路线比较多，KK在发愁选哪条路线？

- **单源最短路径**：从图中某个顶点出发，到达另一个顶点所经过的边的**权重之和最小**的一条路径
- 图论中的图，边的权重可以是路径距离，也可以是时间、费用等
- 图中边的长度和形状（直线、曲线）与实际无关

➢ 最短路径的求解方法

- 基本性质：**最短路径上的任一子路径也是最短路径**
- 常用算法：
  - ✓ Dijkstra算法（迪克斯特拉算法）

    迪克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。
  - ✓ Floyd算法（弗洛伊德算法）

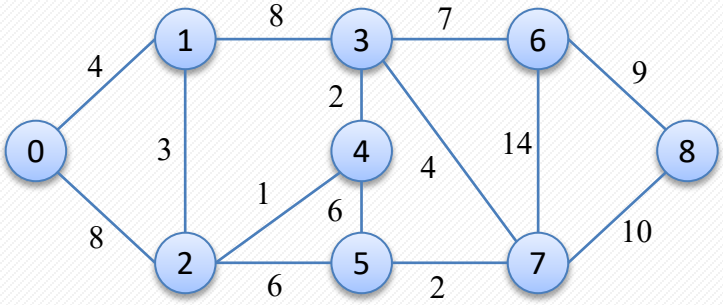
    Floyd算法又称为插点法，是一种利用**动态规划**的思想寻找给定的加权图中多源点之间最短路径的算法
- Matlab函数：
  - ✓ $shortestpath$:

    $[P, D] = shortestpath(G, start, end)$
    G—输入图$(graph \mid digraph$对象$)$
    $start, end$ —其实的节点和目标的节点

Dijkstra算法（迪克斯特拉算法）

迪克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。

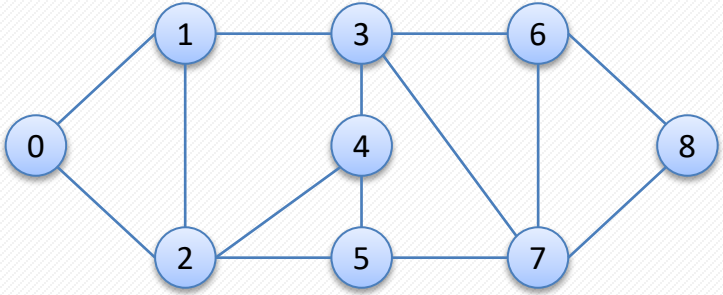
- 贪心算法（greedy algorithm）：是指在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，算法得到的是在某种意义上的局部最优解 。



- 我们可以用迪克斯特拉算法来帮助KK找他的女朋友，即从0到8怎么走路程最短

图论最短路径——算法原理

➢ Dijkstra算法（迪克斯特拉算法）



- 基本原理：最短路径上的任一子路径也是最短路径

  如果0到8的最短路径为：0 → 1 → 3 → 7 → 8

  那么0到7的最短路径为：0 → 1 → 3 → 7

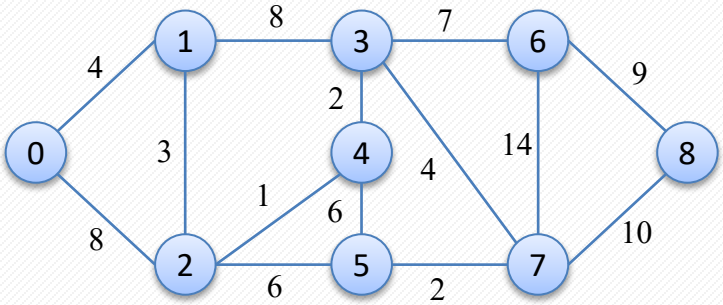
  如果0到7的最短路径为：0 → 2 → 5 → 7

  那么0到8的最短路径应该为：0 → 2 → 5 → 7 → 8

- 所以可以从节点出发，找到邻近节点的最短距离，一步一步向外延伸

图论最短路径——算法原理

➢ Dijkstra算法（迪克斯特拉算法）

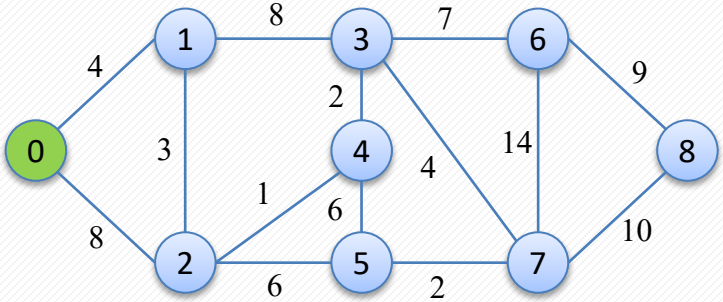


- Visited：把所有顶点分为两个集合，分别为已选顶点（T），和未选定点（F）
- Distance：从顶点出发的最短距离
- Parent：父亲节点（最短路径的上一个节点）

节点	0	1	2	3	4	5	6	7	8
Visited	F	F	F	F	F	F	F	F	F
Distance	inf	inf	inf	inf	inf	inf	inf	inf	inf
Parent	-1	-1	-1	-1	-1	-1	-1	-1	-1

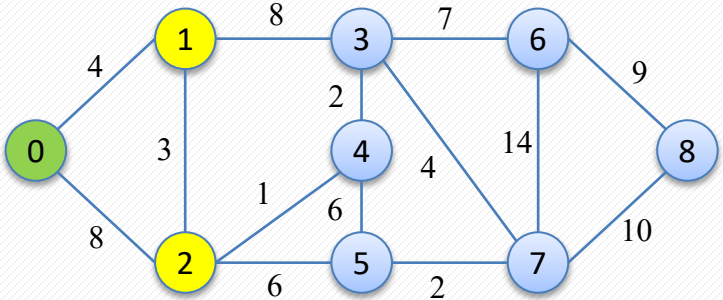
图论最短路径——算法原理

- **Dijkstra算法（迪克斯特拉算法）**



节点	0	1	2	3	4	5	6	7	8
Visited	T	F	F	F	F	F	F	F	F
Distance	0	inf	inf	inf	inf	inf	inf	inf	inf
Parent	/	-1	-1	-1	-1	-1	-1	-1	-1

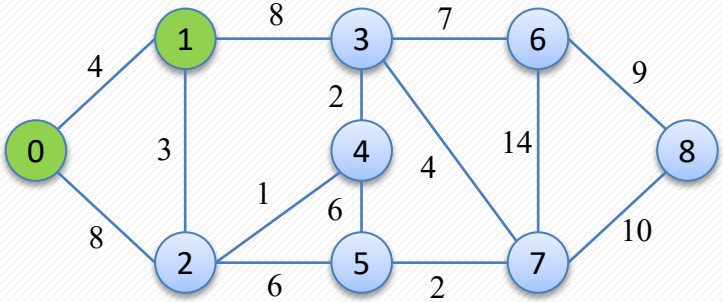
- **Dijkstra算法（迪克斯特拉算法）**



节点	0	1	2	3	4	5	6	7	8
Visited	T	F	F	F	F	F	F	F	F
Distance	0	4	8	inf	inf	inf	inf	inf	inf
Parent	/	0	0	-1	-1	-1	-1	-1	-1

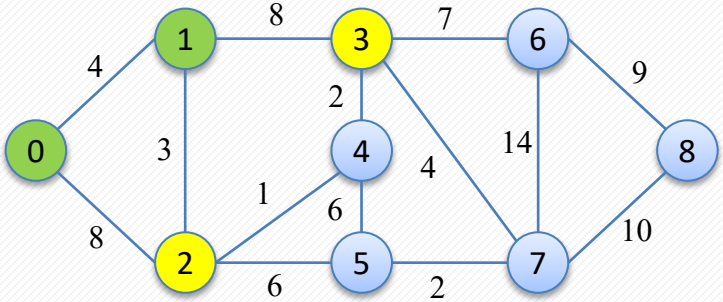
图论最短路径——算法原理

- **Dijkstra算法（迪克斯特拉算法）**



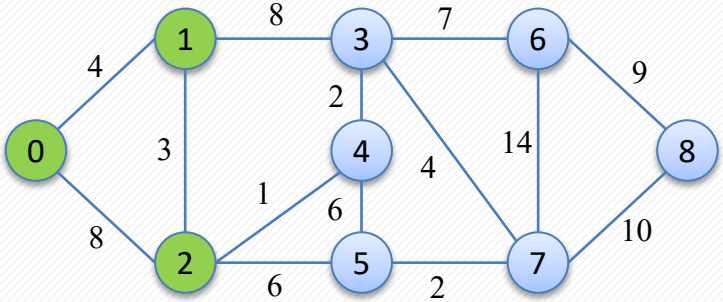
节点	0	1	2	3	4	5	6	7	8
Visited	T	T	F	F	F	F	F	F	F
Distance	0	4	8	inf	inf	inf	inf	inf	inf
Parent	/	0	0	-1	-1	-1	-1	-1	-1

- **Dijkstra算法（迪克斯特拉算法）**



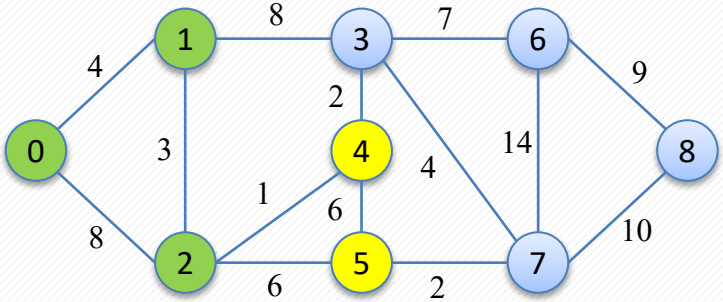
节点	0	1	2	3	4	5	6	7	8
Visited	T	T	F	F	F	F	F	F	F
Distance	0	4	7	12	inf	inf	inf	inf	inf
Parent	/	0	1	1	-1	-1	-1	-1	-1

- **Dijkstra算法（迪克斯特拉算法）**



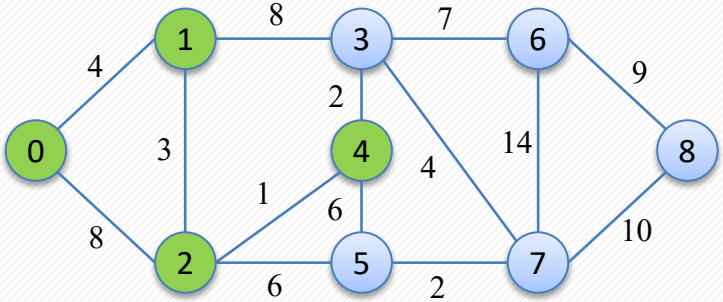
节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	F	F	F	F	F	F
Distance	0	4	7	12	inf	inf	inf	inf	inf
Parent	/	0	1	1	-1	-1	-1	-1	-1

- **Dijkstra算法（迪克斯特拉算法）**



节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	F	F	F	F	F	F
Distance	0	4	7	12	8	13	inf	inf	inf
Parent	/	0	1	1	2	2	-1	-1	-1

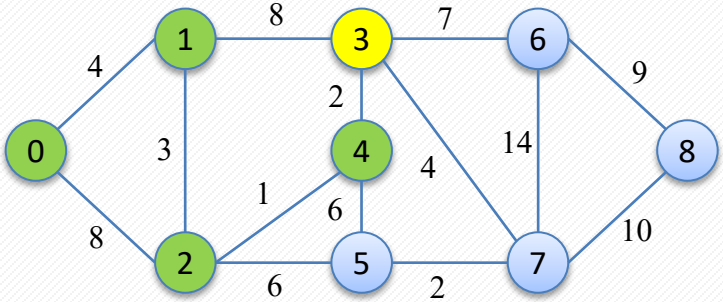
- **Dijkstra算法（迪克斯特拉算法）**



节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	F	T	F	F	F	F
Distance	0	4	7	12	8	13	inf	inf	inf
Parent	/	0	1	1	2	2	-1	-1	-1

图论最短路径——算法原理

- **Dijkstra算法（迪克斯特拉算法）**

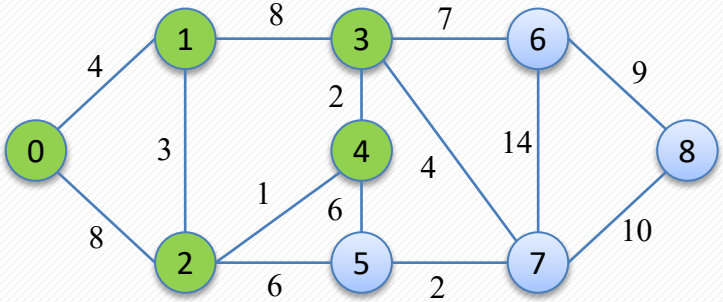


节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	F	T	F	F	F	F
Distance	0	4	7	10	8	13	inf	inf	inf
Parent	/	0	1	4	2	2	-1	-1	-1

图论最短路径——算法原理

MATH 数模加油站

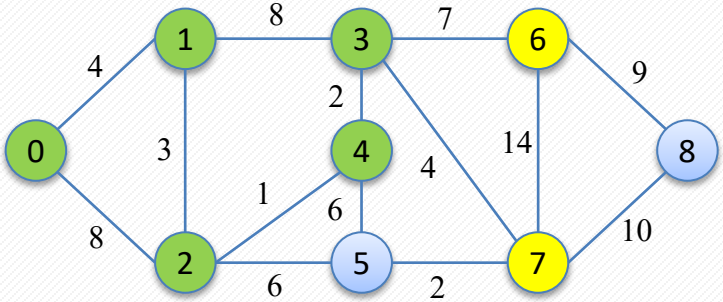
- **Dijkstra算法（迪克斯特拉算法）**



节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	T	T	F	F	F	F
Distance	0	4	7	10	8	13	inf	inf	inf
Parent	/	0	1	4	2	2	-1	-1	-1

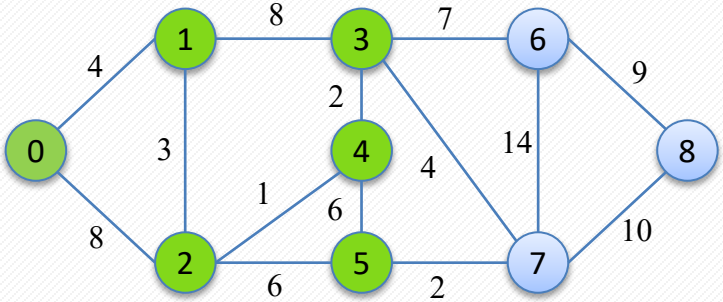
图论最短路径——算法原理

- **Dijkstra算法（迪克斯特拉算法）**



节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	T	T	F	F	F	F
Distance	0	4	7	10	8	13	17	14	inf
Parent	/	0	1	4	2	2	3	3	-1

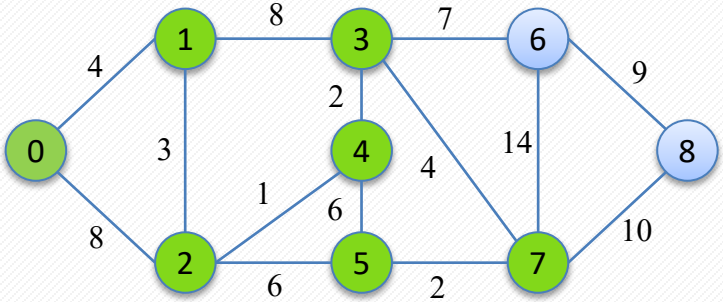
- Dijkstra算法（迪克斯特拉算法）



节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	T	T	T	F	F	F
Distance	0	4	7	10	8	13	17	14	inf
Parent	/	0	1	4	2	2	3	3	-1

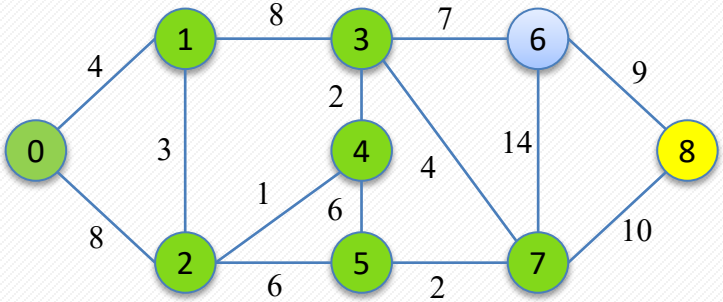
图论最短路径——算法原理

- **Dijkstra算法（迪克斯特拉算法）**



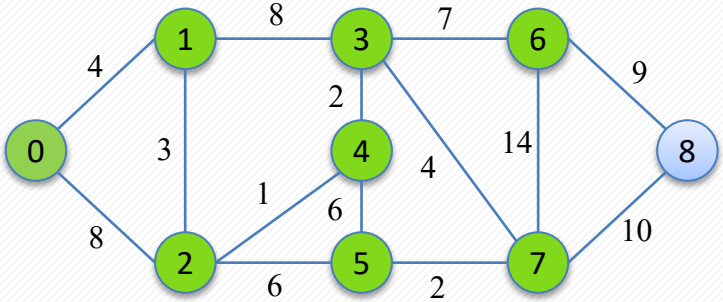
节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	T	T	T	F	T	F
Distance	0	4	7	10	8	13	17	14	inf
Parent	/	0	1	4	2	2	3	3	-1

- **Dijkstra算法（迪克斯特拉算法）**



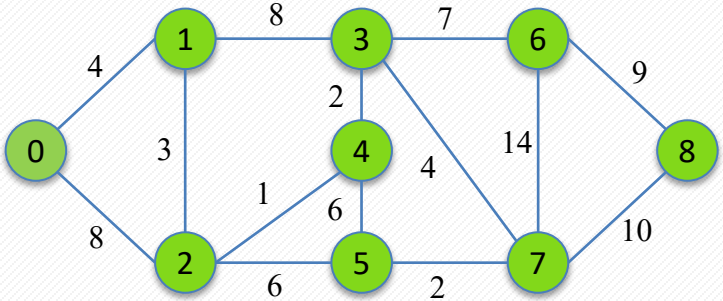
节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	T	T	T	F	T	F
Distance	0	4	7	10	8	13	17	14	24
Parent	/	0	1	4	2	2	3	3	7

- **Dijkstra算法（迪克斯特拉算法）**



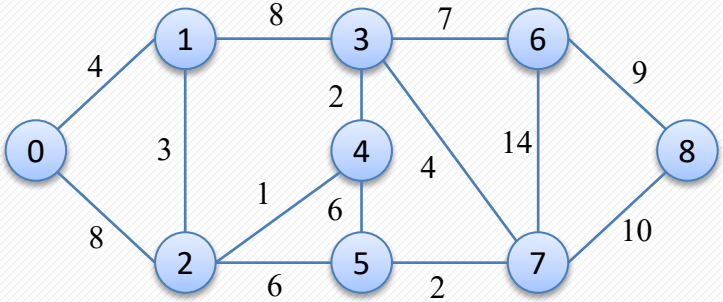
节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	T	T	T	T	T	F
Distance	0	4	7	10	8	13	17	14	24
Parent	/	0	1	4	2	2	3	3	7

- Dijkstra算法（迪克斯特拉算法）



节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	T	T	T	T	T	T
Distance	0	4	7	10	8	13	17	14	24
Parent	/	0	1	4	2	2	3	3	7

- **Dijkstra算法（迪克斯特拉算法）**



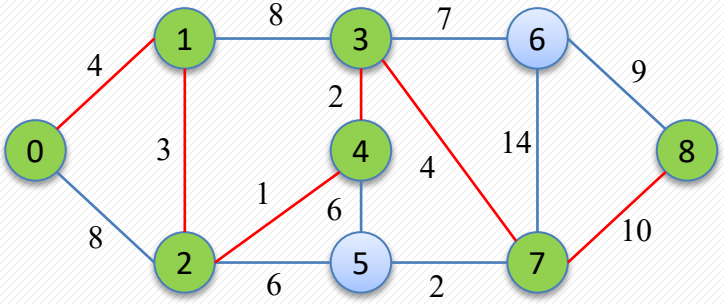
节点	0	1	2	3	4	5	6	7	8
Visited	T	T	T	T	T	T	T	T	T
Distance	0	4	7	10	8	13	17	14	24
Parent	/	0	1	4	2	2	3	3	7

- 那么0到8的最短距离就是24，路径为 0 → 1 → 2 → 4 → 3 → 7 → 8

图论最短路径——算法原理

➢ Floyd算法（弗洛伊德算法）

Floyd算法又称为插点法，是一种利用**动态规划的思想**寻找给定的加权图中多源点之间最短路径的算法



- 如果某个节点位于从起点到终点的最短距离路径上:
  - ✓ 节点0→节点8 = 节点0→节点4 + 节点4→节点8
  - ✓ 节点0→节点8 = 节点0→节点7 + 节点7→节点8
- 如果某个节点不位于从起点到终点的最短距离路径上:
  - ✓ 节点0→节点8 < 节点0→节点5 + 节点5→节点8
  - ✓ 节点0→节点8 < 节点0→节点6 + 节点6→节点8

- Python代码:

```
for k in range (self.V):
    for i in range (self.V):
        for j in range (self.V):
            if  self.D[i][k] + self.D[k][j] < self.D[i][j]:
                self.D[i][j] = self.D[i][k] + self.D[k][j]
                self.S[i][j] =  self.S[i][k]
```

图论最短路径——matlab代码

KK找女朋友代码

```
% 定义图的边和权重
s = [9 9 1 1 3 3 3 2 2 5 5 7 7 8]; % 起始节点编号
t = [1 2 2 3 4 6 7 4 5 4 7 6 8 6]; % 终止节点编号
w = [4 8 3 8 2 7 4 1 6 6 2 14 10 9]; % 边的权重
% 创建一个图形对象 G
G = graph(s,t,w);

% 绘制图形 G，并将边的权重添加到图形上
% G.Edges.Weight 表示图形对象 G 中所有边的权重值，'EdgeLabel' 表示在图形上显示这些权重值
plot(G, 'EdgeLabel', G.Edges.Weight, 'linewidth', 2)
% 隐藏图形的坐标轴
set( gca, 'XTick', [], 'YTick', [] );

% shortestpath 函数计算从节点 9 到节点 8 的最短路径和路径长度，并将路径和路径长度分别存储在 P 和 d 中
[P,d] = shortestpath(G, 9, 8)
% 在图形 G 中高亮显示最短路径

% highlight 函数高亮图形对象 myplot 中的路径 P，'EdgeColor', 'r' 表示将路径颜色设置为红色。
myplot = plot(G, 'EdgeLabel', G.Edges.Weight, 'linewidth', 2);
highlight(myplot, P, 'EdgeColor', 'r')

% 计算图形 G 中任意两点之间的最短路径矩阵
D = distances(G)
% 输出 1 到 2 的最短路径
D(1,2)
% 输出 9 到 8 的最短路径
D(9,8)
% 找出图形 G 中距离节点 2 不超过 10 的所有节点
[nodeIDs,dist] = nearest(G, 2, 10)
```

欢迎关注数模加油站

THANKS