图论

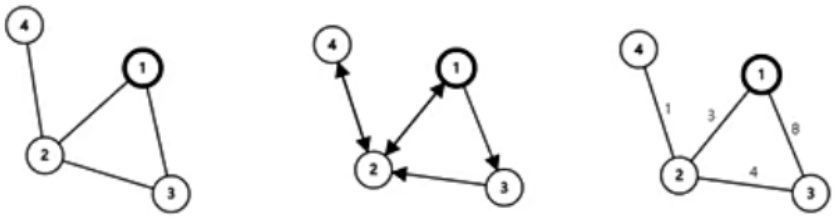
1.基本概念

图论中的图（Graph）是由若干给定的点及连接两点的线所构成的图形，这种图形通常用来描述某些事物之间的某种特定关系，用点代表事物，用连接两点的线表示相应两个事物间具有这种关系。

一个图可以用数学语言描述为G(V(G),E(G))。V(vertex)指的是图的顶点集，E(edge)指的是图的边集。

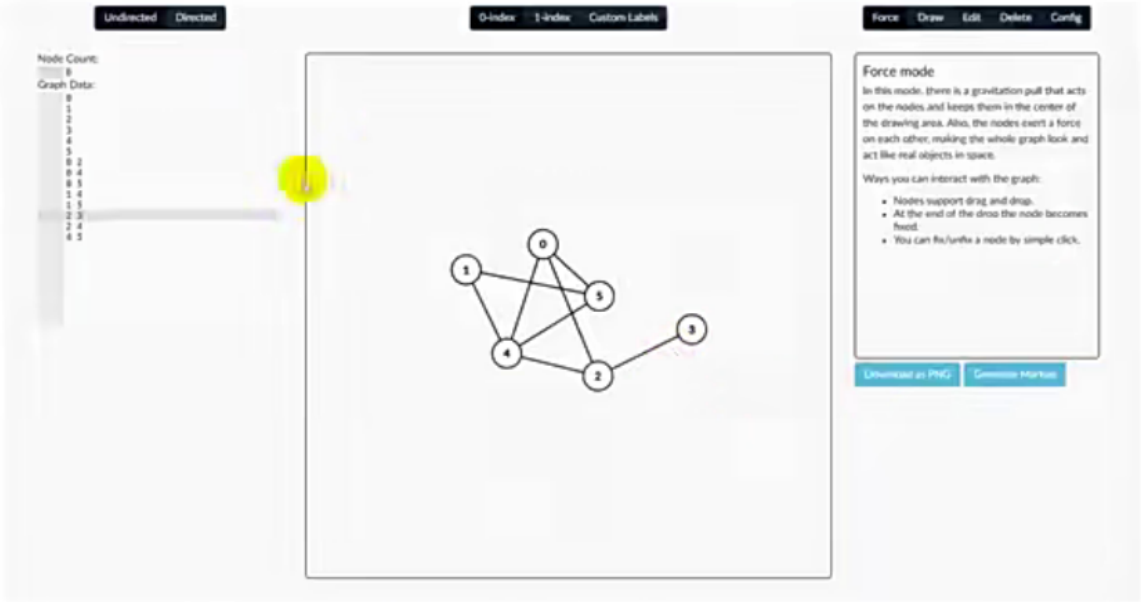
根据边是否有方向，可将图分为有向图和无向图。

另外，有些图的边上还可能有权值，这样的图称为有权图。

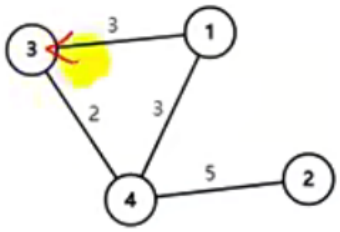


在线做图

https://csacademy.com/app/graph_editor/



无向图的权重邻接矩阵



带权重的四个节点的无向图

$$D = \begin{bmatrix} 0 & Inf & 3 & 3 \\ Inf & 0 & Inf & 5 \\ 3 & Inf & 0 & 2 \\ 3 & 5 & 2 & 0 \end{bmatrix}$$

无向图对应的权重邻接矩阵

	1	2	3	4
1	0	Inf	3	3
2	Inf	0	Inf	5
3	3	Inf	0	2
4	3	5	2	0

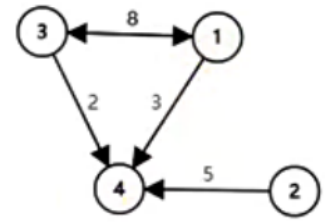
结论：

（1）无向图对应的权重邻接矩阵D是一个对称矩阵；

（2）其主对角线上元素为0.

（3）D_{ij}表示第i个节点到第j个节点的权重。

有向图的权重邻接矩阵



带权重的四个节点的有向图

$$D = \begin{bmatrix} 0 & Inf & 8 & 3 \\ Inf & 0 & Inf & 5 \\ 8 & Inf & 0 & 2 \\ Inf & Inf & Inf & 0 \end{bmatrix}$$

有向图对应的权重邻接矩阵

	1	2	3	4
1	0	Inf	8	3
2	Inf	0	Inf	5
3	8	Inf	0	2
4	Inf	Inf	Inf	0

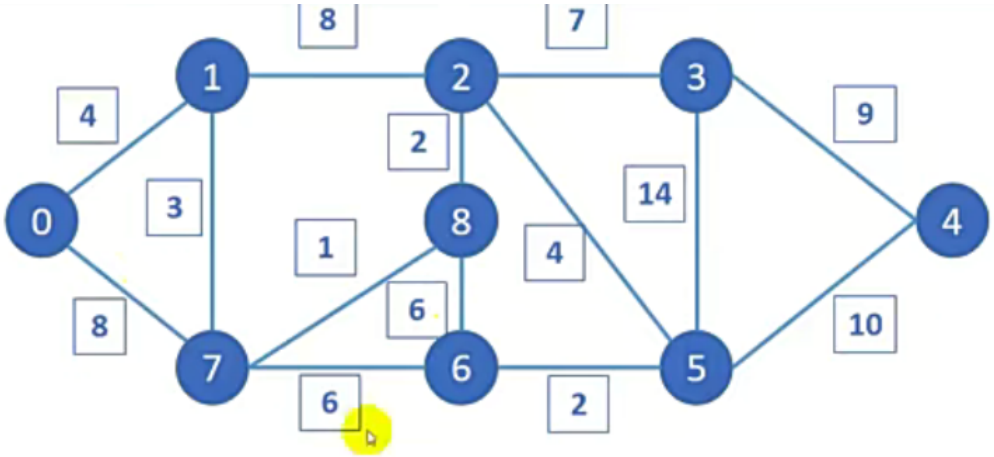
结论：

（1）有向图对应的权重邻接矩阵D是一般不再是对称矩阵；

（2）其主对角线上元素为0.

（3）D_{ij}表示第i个节点到第j个节点的权重。

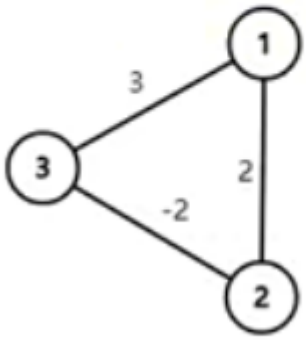
迪杰斯特拉算法



图中有0-8共九个地点，地点之间若用直线连接则表明两地可直接到达，直线旁的数值表示两地的距离。
问题：起点为0，终点为4，怎么走路程最短。
（假设出行方式相同，例如都为步行）

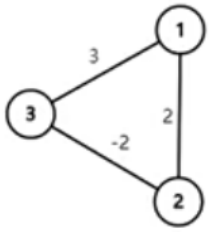
可以用于有向图

但不能处理负权重



1是起点；2是终点

如何修复该缺点?



1是起点；2是终点

Bellman-Ford（贝尔曼-福特）算法

刚刚改变访问状态的节点为0号节点（A）
我们要更新与0号节点相邻的节点信息（B），注意，这里的B节点是未访问的哦
更新的规则如下：
如果（A与B的距离+ A列表中的距离）小于（B列表中的距离），那么我们就将B列表中的距离更新为较小的距离，并将B的父亲节点更新为A

事实上，贝尔曼-福特算法不再将节点区分为是否已访问的状态，因为贝尔曼-福特模型是利用循环来进行更新权重的，且每循环一次，贝尔曼福特算法都会更新所有的节点的信息。

贝尔曼-福特算法不支持含有负权回路的图。
（视频中提到的Floyd(弗洛伊德)算法也不可以）

有兴趣的同学可以参考下面两份资料弄懂其实现原理：
https://blog.csdn.net/a8082649/article/details/81812000
https://www.bilibili.com/video/av43217121

Matlab计算最短路径

```
[P,d] = shortestpath(G,start,end [,'Method',algorithm]
```

功能：返回图G中start节点到end节点的最短路径

输入参数：
（1）G - 输入图（graph 对象 | digraph 对象）
（2）start 起始的节点
（3）end 目标的节点
（4）[,'Method',algorithm]是可选的参数，表示计算最短路径的算法。一般我们不用手动设置，默认使用的是"auto"，具体可设置的参数见下一页PPT。

输出参数：
（1）P – 最短路径经过的节点
（2）d – 最短路径

注意：该函数matlab2015b之后才有哦。

可选的算法

选项	说明
'auto'（默认值）	'auto' 选项会自动选择算法： •'unweighted' 用于没有边权重的 graph 和 digraph 输入。 •'positive' 用于具有边权重的所有 graph 输入，并要求权重为非负数。此选项还用于具有非负边权重的 digraph 输入。 •'mixed' 用于其边权重包含某些负值的 digraph 输入。图不能包含负循环。
'unweighted'	广度优先计算，将所有边权重都视为 1。
'positive'	Dijkstra 算法，要求所有边权重均为非负数。
'mixed'（仅适用于 digraph）	适用于有向图的 Bellman-Ford 算法，要求图没有负循环。尽管对于相同的问题，'mixed' 的速度慢于 'positive'，但 'mixed' 更为通用，因为它允许某些边权重为负数。