**最小生成树**

加权图是一种为每条边关联一个权值或成本的图模型。

最小生成树，给定一幅加权无向图，找到它的一棵最小生成树。

图的生成树是它的一棵含有其所有顶点的无环连通子图。

一幅加权图的最小生成树是它的一棵权值最小的生成树。

在计算最小生成树的过程中，我们约定如下：

1. 只考虑连通图
2. 边的权重不一定表示距离
3. 边的权重可能是0或者负数
4. 所有边的权重都各不相同

如果不同边的权重可以相同，最小生成树就不一定唯一了。

4.3.1 原理

图的一种切分是将图的所有顶点分为两个非空且不重叠的两个集合。横切边是一条连接两个属于不同集合的顶点的边。

切分定理：在一幅加权图中，给定任意的切分，它的横切边中的权重最小者必然属于图的最小生成树。

切分定理是解决最小生成树问题的所有算法的基础。

最小生成树的贪心算法：下面这种方法会将含有V个顶点的任意加权连通图中属于最小生成树的边标记为黑色：初始状态下所有边均为灰色，找到一种切分，它产生的横切边均不为黑色。将它权重最小的横切边标记为黑色。反复，直到标记了V-1条黑色边为止。

4.3.2 加权无向图的数据结构

加权边的API

构造函数：Edge(int v, int w, double weight)

边的权重：double weight()

边两端的顶点之一：int either()

另一个顶点：int other(int v)

将这条边与that比较：int compareTo(Edge that)

字符串表示：toString()

加权无向图的API

构造函数：EdgeWeightedGraph(int V)

顶点数：int V()

边数：int E()

向图中添加一条边e：void addEdge(Edge e)

和v相关联的所有边：Iterable<Edge> adg(int v)

图的所有边：Iterable<Edge> edges()

字符串表示：toString()

编程：带权重的边的数据类型（Edge）

编程：加权无向图的数据类型（EdgeWeightedGraph）

4.3.3 最小生成树的API和测试用例

最小生成树的API

构造函数：MST(EdggeWeightedGraph G)

最小生成树的所有边：Iterable<Edge> edges()

最小生成树的权重：double weight()

4.3.4 Prim算法

Prim算法：每一步会为一棵生长中的树添加一条边。一开始这棵树只有一个顶点，然后会向它添加V-1条边，每次总是将下一条连接树中的顶点与不在树中的顶点且权重最小的边（黑色表示）加入树中（即由树中的顶点所定义的切分中的一条横切边）。

Prime算法的延迟实现计算一幅含有V个顶点和E条边的连通加权无向图的最小生成树所需的空间与E成正比，所需的时间与ElogE成正比（最坏情况下）。

编程：最小生成树的Prim算法的延迟实现。（LazyPrimMST）

4.3.5 Prim算法的即时实现（了解）

4.3.6 Kruskal算法

Kruskal算法的思想是按照边的权重顺序从小到大处理它们，将边加入最小生成树中，加入的边不会与已经加入的边构成环，直到树中含有V-1条边为止。

Kruskal算法的计算一幅含有V个顶点和E条边的连通加权无向图的最小生成树所需的空间和E成正比，所需时间和ElogE成正比。

Kruskal算法一般还是比Prim算法要慢。

编程：最小生成树的Kruskal算法（KruskalMST）

最小生成树Prim算法 LazyPrimMST

最小生成树Kruskal算法KruskalMST