Section 3.1 Minimal Spanning Trees 最小生成树(MST)

译 by Lucky Crazy & Felicia Crazy

例题：最短网络 [Russ Cox, Winter 1999 USACO Open] 农民约翰被选为他们镇的镇长！他其中一个竞选承诺就是在镇上建立起互联网，并连接到所有的农场。当然，他需要你的帮助。约翰已经给他的农场安排了一条高速的网络线路，他想把这条线路共享给其他农场。为了用最小的消费，他想铺设最短的光纤去连接所有的农场。你将得到一份各农场之间连接费用的列表，你必须找出能连接所有农场并所用光纤最短的方案。每两个农场间的距离不会超过 100000 数学模型：

给出: 一幅连通的有权无向图。

一棵生成树的图是一张无向的连通图(也就是图中任意一对节点都是连通的).

而一棵最小生成树是一棵权和最小的生成树(即树所有的权值和最小). 利用 Prim 算法建造最小生成树：

给出: 所有节点和权值的列表。

利用贪心算法建造最小生成树，即在每次都将已有的生成树相连的一个权值最小的节点连入树中。

从某一个节点开始生成。

确定所有不在树中结点连接到树中所需的最小权值（如下例中的 distance），并记录下与之相连的结点（如下例中的 source)。 如果有结点无法连入树中，那么假设它的权值为无穷大 （比图中所有的权值都大）.

在每一步中,找出一个和生成树相连的权值最小的节点（即在 distance 栏中）并且用一条边将它连入树中。（由于最小生成树的性质，它必定是不含圈的。否则断开这个圈仍然可以保证每个节点都在树中——译者）

如果你想知道详细的分析或该算法的正确性证明，请参考《Cormen, Leiserson, Rivest》的第 24 章。

伪代码:

```
 # distance(j) is distance from tree to node j
 # source(j) is which node of so-far connected MST
 #                   is closest to node j
1   For all nodes i
2     distance(i) = infinity       # no connections
3     intree(i) = False           # no nodes in tree
4     source(i) = nil
5   treesize = 1                 # add node 1 to tree
6   treecost = 0
7   intree(1) = True
8   For all neighbors j of node 1   # update distances
9      distance(j) = weight(1,j)
10 source(j) = 1
11 while (treesize < graphsize)
12 find node with minimum distance to tree; call it node i
13 assert (distance(i) != infinity, "Graph Is Not Connected")
   # add edge source(i),i to MST
14 treesize = treesize + 1
15 treecost = treecost + distance(i)
16 intree(i) = True # mark node i as in tree
   # update distance after node i added
17 for all neighbors j of node i
18 if (distance(j) > weight(i,j))
19 distance(j) = weight(i,j)
20 source(j) = i
```

该算法的时间复杂度为 O(N2)。使用堆可以得到 O(N log N)的复杂度。图例：

考虑下图的权，边情况:

目标：生成最小生成树。 该算法将在 1）节点开始，它与节点 2），6），3）相连，权值情况如下：

Node distance intree source 1 infinity True nil 2 30 False 1 3 20 False 1 6 25 False 1

已知不存在权为无穷大的情况。（intree=False & source=nil）

可连接的最小权为 20, 所以节点 3）被连入树中：

Node distance intree source 1 infinity True nil 2 9 False 3 3 20 True 1 6 25 False 1 7 7 False 3

注意：节点 3）现在已经“在树中”了，节点 2）的权已经从 20 变成的 9，且 source 也变成的 3。

可连接的最小权为 7, 所以节点 3）和 7）被连接:

Node distance intree source 1 infinity True nil 2 9 False 3 3 20 True 1 6 10 False 7 7 7 True 3

可连接的最小权为节点 2）的 9。 连接节点 3)和 2)：

Node distance intree source 1 infinity True nil 2 9 True 3 3 20 True 1 4 21 False 2 5 9 False 2 6 10 False 7 7 7 True 3

连接节点 2）和 5)：

Node distance intree source 1 infinity True nil 2 9 True 3 3 20 True 1 4 8 False 5 5 9 True 2 6 10 False 7 7 7 True 3 8 12 False 5

下一步连接节点 5) 和 4）:

Node distance intree source 1 infinity True nil 2 9 True 3 3 20 True 1 4 8 True 5 5 9 True 2 6 10 False 7 7 7 True 3 8 12 False 5 连接节点 6）和 7）:

Node distance intree source 1 infinity True nil 2 9 True 3 3 20 True 1 4 8 True 5 5 9 True 2 6 10 True 7 7 7 True 3 8 11 False 6

最后，连接节点 6）和 8)：

Node distance intree source 1 infinity True nil 2 9 True 3 3 20 True 1 4 8 True 5 5 9 True 2 6 10 True 7 7 7 True 3 8 11 True 6

最小生成树完成。注意：

必须知道某些情况下某些结点无法连入树中，应避免重复计算这些结点。（即 intree=False & source=nil，在例中没有该情况）。题型提示：

如果某些问题需要一张最优的连通图，并且需要用以一个最小的花费来连接该系统或该系统的任意两个部分，这样的问题就极类似最小生成树问题。

拓展：

如果你的生成树有任何约束条件的话（任意两个结点可能离得非常远，或者平均距离必须最小），这个算法就玩完了，而且让程序适应这样的约束条件非常困难。

显而易见，任意两个结点之间不能有多边（你就留下权值最小的边，忽略其余的边）。

Prim 算法无法扩展到有向图（如果你想要的是强连通图的话）。例题：包裹寄送

给出：一些城市的位置，和轮船公司连接每对城市的航线的花费。找出使得一个包裹能够从任意一座城市送到任意的另外一座城市的花费最小。高速公路建设

当然，为了经济效益，他们想要花最少的钱来做这件事。高速公路的花费正比于它的长度。给出 L.S. 州的所有城市的 x,y 坐标，设计使得所有城市互相连通的最便宜的建造方案。 Bovile 电话(已删节) [USACO Training Camp 1998, Contest 2]

给出：一些奶牛和田野中的干草堆（奶牛和干草堆在一起），连接任意的位置需要一定的花费。只用干草堆和奶牛，计算哪些干草堆应该包含在干草堆网络中，并且使总花费最小。

分析：对于每一组可能的干草堆（也就是，共有 2 n 组），计算这组干草堆和奶牛的最小生成树。计算最小生成树的组合，使得花费最小。