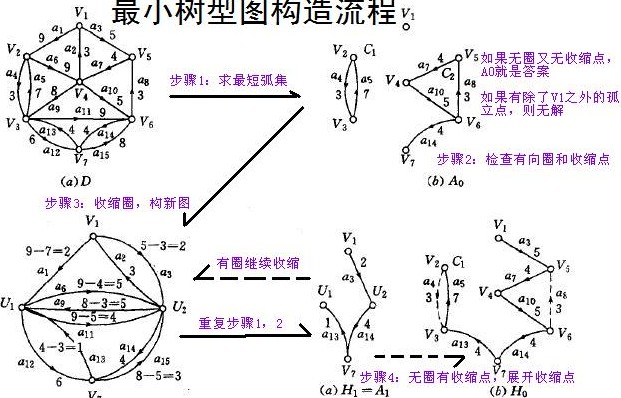
最小树形图，就是给有向带权图中指定一个特殊的点root，求一棵以root为根的有向生成树T，并且T中所有边的总权值最小。最小树形图的第一个算法是1965年朱永津和刘振宏提出的复杂度为O(VE)的算法。  
判断是否存在树形图的方法很简单，只需要以v为根作一次图的遍历就可以了，所以下面的算法中不再考虑树形图不存在的情况。  
在所有操作开始之前，我们需要把图中所有的自环全都清除。很明显，自环是不可能在任何一个树形图上的。只有进行了这步操作，总算法复杂度才真正能保证是O(VE)。  
首先为除根之外的每个点选定一条入边，这条入边一定要是所有入边中最小的。现在所有的最小入边都选择出来了，如果这个入边集不存在有向环的话，我们可以证明这个集合就是该图的最小树形图。这个证明并不是很难。如果存在有向环的话，我们就要将这个有向环所称一个人工顶点，同时改变图中边的权。假设某点u在该环上，并设这个环中指向u的边权是in[u]，那么对于每条从u出发的边(u, i, w)，在新图中连接(new, i, w)的边，其中new为新加的人工顶点; 对于每条进入u的边(i, u, w)，在新图中建立边(i, new, w-in[u])的边。为什么入边的权要减去in[u]，这个后面会解释，在这里先给出算法的步骤。然后可以证明，新图中最小树形图的权加上旧图中被收缩的那个环的权和，就是原图中最小树形图的权。  
上面结论也不做证明了。现在依据上面的结论，说明一下为什么出边的权不变，入边的权要减去in [u]。对于新图中的最小树形图T，设指向人工节点的边为e。将人工节点展开以后，e指向了一个环。假设原先e是指向u的，这个时候我们将环上指向u的边 in[u]删除，这样就得到了原图中的一个树形图。我们会发现，如果新图中e的权w'(e)是原图中e的权w(e)减去in[u]权的话，那么在我们删除掉in[u]，并且将e恢复为原图状态的时候，这个树形图的权仍然是新图树形图的权加环的权，而这个权值正是最小树形图的权值。所以在展开节点之后，我们得到的仍然是最小树形图。逐步展开所有的人工节点，就会得到初始图的最小树形图了。  
如果实现得很聪明的话，可以达到找最小入边O(E)，找环 O(V)，收缩O(E)，其中在找环O(V)这里需要一点技巧。这样每次收缩的复杂度是O(E)，然后最多会收缩几次呢？由于我们一开始已经拿掉了所有的自环，我门可以知道每个环至少包含2个点，收缩成1个点之后，总点数减少了至少1。当整个图收缩到只有1个点的时候，最小树形图就不不用求了。所以我们最多只会进行V-1次的收缩，所以总得复杂度自然是O(VE)了。由此可见，如果一开始不除去自环的话，理论复杂度会和自环的数目有关。

==================================================================================================  
下面是朱刘算法的构造图：

[](http://photo.blog.sina.com.cn/showpic.html#blogid=6af663940100ls4h&url=http://s8.sinaimg.cn/orignal/6af6639448ebac93ec5a7)  
  
有固定根的最小树形图求法O(VE)：

【转】最小树形图 - 闲着无聊 - 闲着OI的博客首先消除自环，显然自环不在最小树形图中。然后判定是否存在最小树形图，以根为起点DFS一遍即可。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客之后进行以下步骤。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客设cost为最小树形图总权值。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客0.置cost=0。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客1.求最短弧集合Ao （一条弧就是一条有向边）

【转】最小树形图 - 闲着无聊 - 闲着OI的博客除源点外，为所有其他节点Vi，找到一条以Vi为终点的边，把它加入到集合Ao中。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客（加边的方法：所有点到Vi的边中权值最小的边即为该加入的边，记prev[vi]为该边的起点，mincost[vi]为该边的权值）

【转】最小树形图 - 闲着无聊 - 闲着OI的博客2.检查Ao中的边是否会形成有向圈，有则到步骤3，无则到步骤4。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客（判断方法：利用prev数组，枚举为检查过的点作为搜索的起点，做类似DFS的操作）

【转】最小树形图 - 闲着无聊 - 闲着OI的博客3.将有向环缩成一个点。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客假设环中的点有（Vk1，Vk2，… ，Vki）总共i个，用缩成的点叫Vk替代，则在压缩后的图中，其他所有不在环中点v到Vk的距离定义如下：

【转】最小树形图 - 闲着无聊 - 闲着OI的博客【转】最小树形图 - 闲着无聊 - 闲着OI的博客gh[v][Vk]=min 【转】最小树形图 - 闲着无聊 - 闲着OI的博客{ gh[v][Vkj]-mincost[Vkj] } (1<=j<=i)而Vk到v的距离为

【转】最小树形图 - 闲着无聊 - 闲着OI的博客【转】最小树形图 - 闲着无聊 - 闲着OI的博客gh[Vk][v]=min 【转】最小树形图 - 闲着无聊 - 闲着OI的博客{ gh[Vkj][v] }              (1<=j<=i)

【转】最小树形图 - 闲着无聊 - 闲着OI的博客同时注意更新prev[v]的值，即if(prev[v]==Vkj) prev[v]=Vk

【转】最小树形图 - 闲着无聊 - 闲着OI的博客另外cost=cost+mincost[Vkj] (1<=j<=i)

【转】最小树形图 - 闲着无聊 - 闲着OI的博客到步骤1.

【转】最小树形图 - 闲着无聊 - 闲着OI的博客4.cost加上Ao的权值和即为最小树形图总权值。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客如要输出最小树形图较烦，没实现过。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客找环O(V)，收缩O(E)，总复杂度O(VE)。

【转】最小树形图 - 闲着无聊 - 闲着OI的博客那幅对我有莫大帮助的流程图如下，

【转】最小树形图 - 闲着无聊 - 闲着OI的博客对于不固定根的最小树形图，wy教主有一巧妙方法。摘录如下：

【转】最小树形图 - 闲着无聊 - 闲着OI的博客新加一个点，和每个点连权相同的边，这个权大于原图所有边权的和，这样这个图固定跟的最小树形图和原图不固定跟的最小树形图就是对应的了。