**最小生成树知识**

**一个连通图的生成树是它的极小连通子图**，在n个顶点的情况下，有n-1条边。也就是说，*生成树是对连通图而言的，是连通图的极小连通子图，包含图中的所有顶点，有且仅有 n-1 条边*。

在图论中，我们常常将树定义为一个**无回路连通图**。对于一个带权的无回路连通图，其每个生成树所有边上的权值之和不能不同，我们把*所有边上权值之和最小的生成树称为图的最小生成树*。常见的求最小生成树的方法有两种：**克鲁斯卡尔(Kruskal)算法和普里姆（Prim）**算法。下面介绍Prim算法。

**那我们如何选择这n个顶点，使得这n个顶点组成的n-1条边的总长度最短呢？**

我们从顶点1开始，把顶点1加入生成树集合(顶点集合)，在一端以顶点1为端点的边中，选择一条最短的，把另一个端点加入生成树；然后选择一条到生成树最短的边，把此时这条边一个端点肯定在生成树中，把另一个端点也加入生成树。依次选择到生成树最短的边……直到选择了n-1条边为止。

**一、prim算法**

**1、理解1**

<http://www.layz.net/LAOJ/suanfa/s9-5.html>

**原理：（简单来说，不断添加最小权值边（与生成树中现有顶点相连的边），知道n-1条边为止）**先从顶点1开始，每次选择到生成树最小的边，然后把这条边另一个端点加入生成树；再选择到生成树最短的边，把另一个端点加入生成树；…… 直到选择了n-1条边为止。

本算法的关键在于**如何求各个顶点（不在生成树内）到生成树的最短边？**

程序中我们定义了一个*数组dis[i]，表示顶点i到生成树的最短距离*，那么就解决了这个问题：

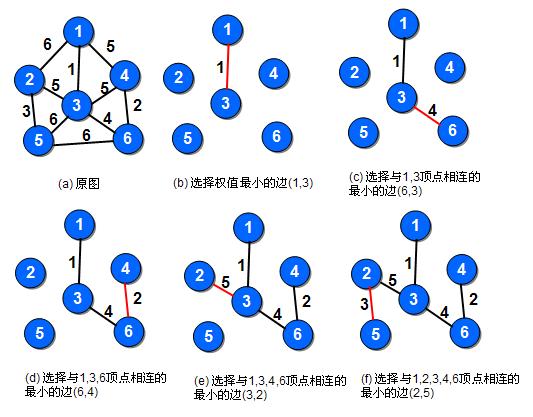


**同时，还有一个问题：如何更新dis[i]呢？**

当我们**将得到k加入到生成树中的同时，我们要判断a[k][i]是不是比dis[i]更小**，如果是的话就更新dis[i]的值。



应用Prim算法构造最小生成树的过程如下：



输出最短权值和的Prim算法程序如下：



如果我们要输出**最小生成树的各条边**，可以定义一个二维数组tree来记录边。

**常见问题：**有n个村庄，从1到n进行编号。你应该修建一些道路，使得任意两个村庄都是相连的。我们说村庄A和村庄B是相连的，要么A和B之间有一条路，要么存在一个存在存在C，A和C之间有一条路，并且C和B是相连的。

一些存在之间已经有一些路存在，现在你的工作就是再修一些路，使得所有的存在都相连起来，且**被修建的所有道路长度最小**。

**2、理解2**

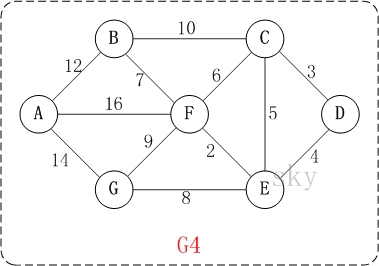
<http://www.cnblogs.com/skywang12345/p/3711510.html>

普里姆(Prim)算法：是用来求加权连通图的最小生成树的算法。

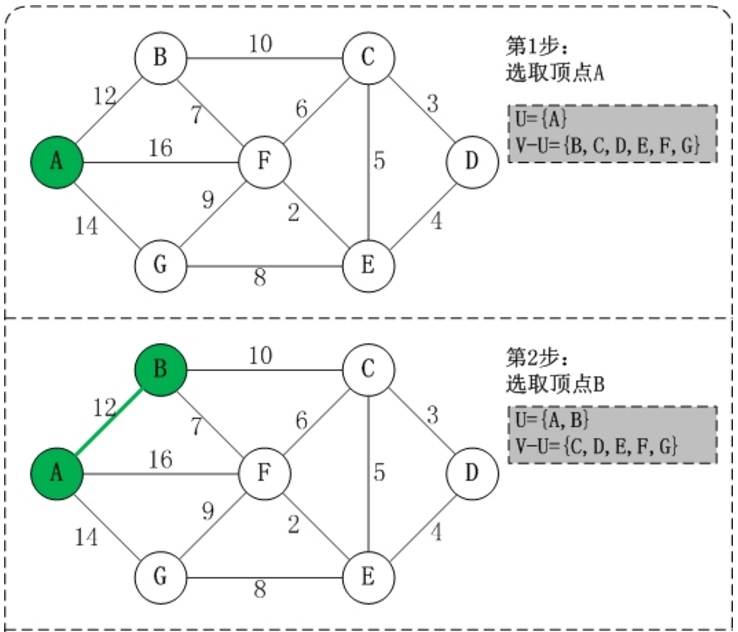
**1）基本思想**

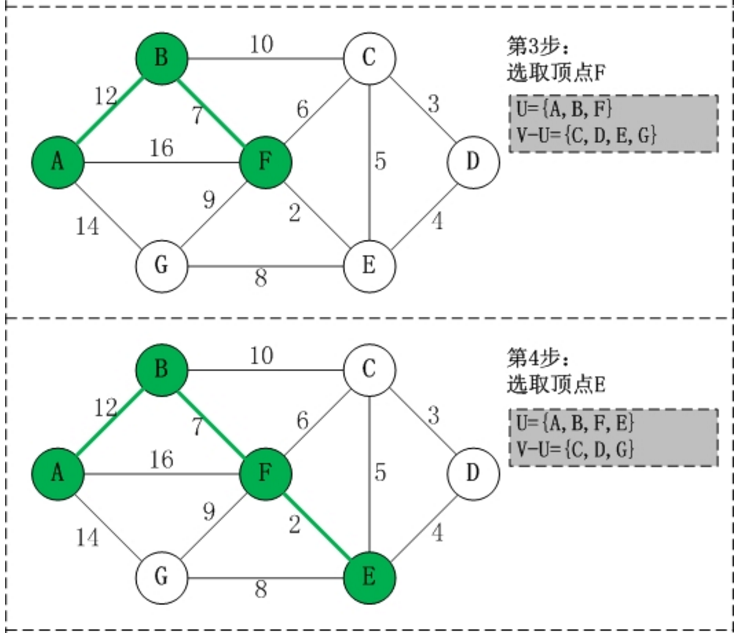
**基本思想：**对于图G而言，V是所有顶点的集合；现在，设置两个新的集合U和T，其中***U用于存放G的最小生成树中的顶点，T存放G的最小生成树中的边***。 从所有uЄU，vЄ(V-U) (V-U表示出去U的所有顶点)的边中**选取权值最小的边(u, v)**，将顶点v加入集合U中，将边(u, v)加入集合T中，如此不断重复，直到U=V为止，**最小生成树**构造完毕，这时集合T中包含了最小生成树中的所有边。

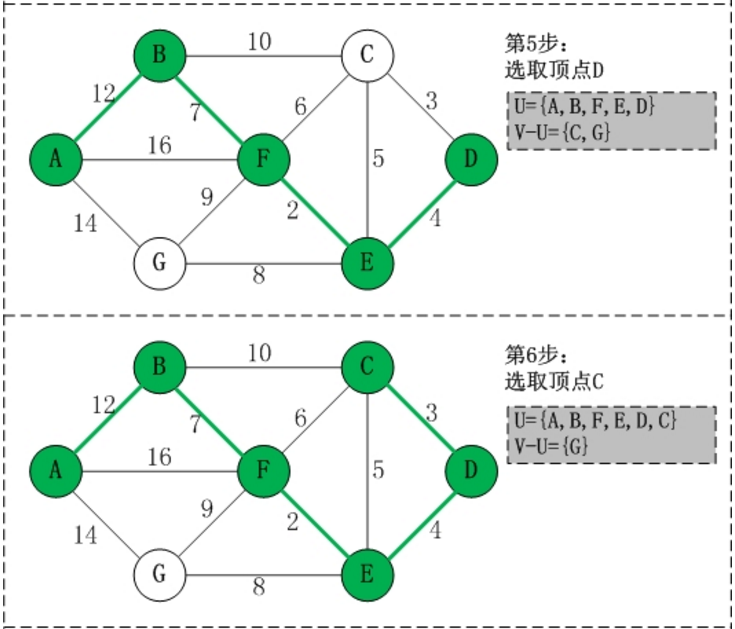
**2）算法图解**

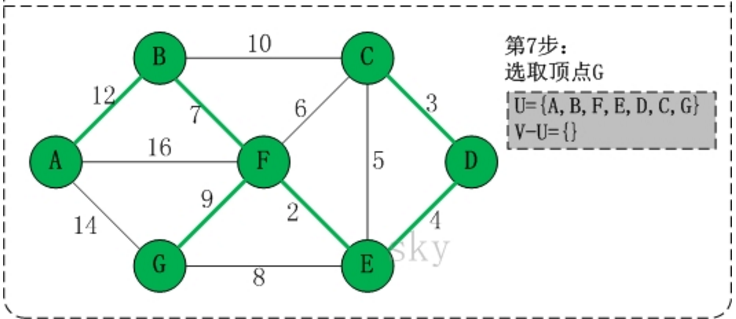


以上图G4为例，来对普里姆进行演示(从**第一个顶点A**开始通过普里姆算法生成最小生成树)。



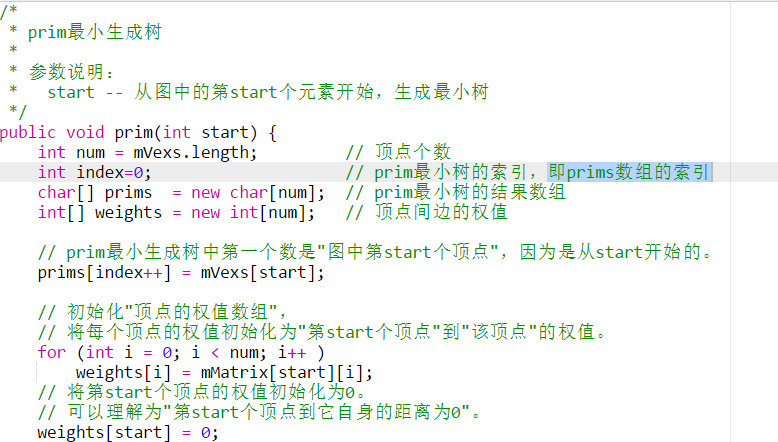


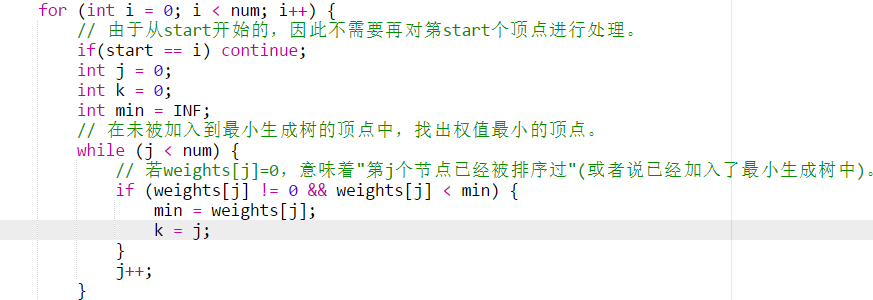


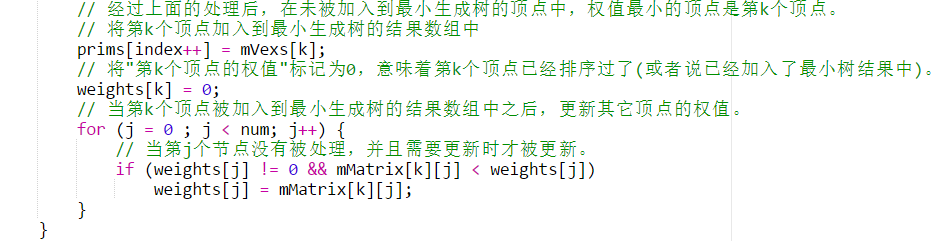


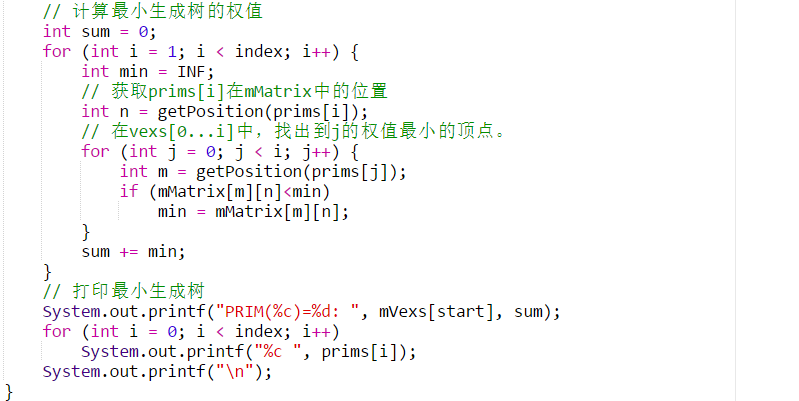
* **初始状态：**V是所有顶点的集合，即V={A,B,C,D,E,F,G}；U和T都是空！
* **第1步：**将顶点A加入到U中。此时，U={A}。
* **第2步：**将顶点B加入到U中。上一步操作之后，U={A}, V-U={B,C,D,E,F,G}；因此，边(A,B)的权值最小。将顶点B添加到U中；此时，U={A,B}。
* **第3步：**将顶点F加入到U中。上一步操作之后，U={A,B}, V-U={C,D,E,F,G}；因此，边(B,F)的权值最小。将顶点F添加到U中；此时，U={A,B,F}。
* **第4步：**将顶点E加入到U中。上一步操作之后，U={A,B,F}, V-U={C,D,E,G}；因此，边(F,E)的权值最小。将顶点E添加到U中；此时，U={A,B,F,E}。
* **第5步：**将顶点D加入到U中。上一步操作之后，U={A,B,F,E}, V-U={C,D,G}；因此，边(E,D)的权值最小。将顶点D添加到U中；此时，U={A,B,F,E,D}。
* **第6步：**将顶点C加入到U中。上一步操作之后，U={A,B,F,E,D}, V-U={C,G}；因此，边(D,C)的权值最小。将顶点C添加到U中；此时，U={A,B,F,E,D,C}。
* **第7步：**将顶点G加入到U中。上一步操作之后，U={A,B,F,E,D,C}, V-U={G}；因此，边(E,G)的权值最小。将顶点G添加到U中；此时，U=V。
* 此时，最小生成树构造完成！它包括的顶点依次是：A B F E D C G。

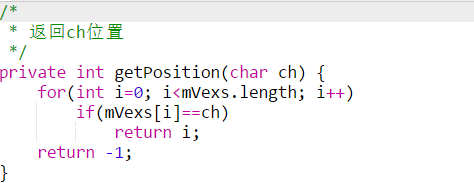
**3）代码实现**





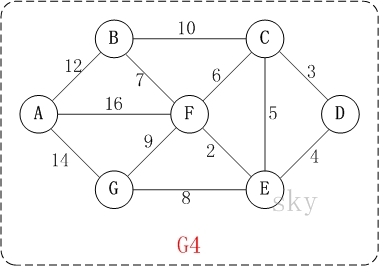




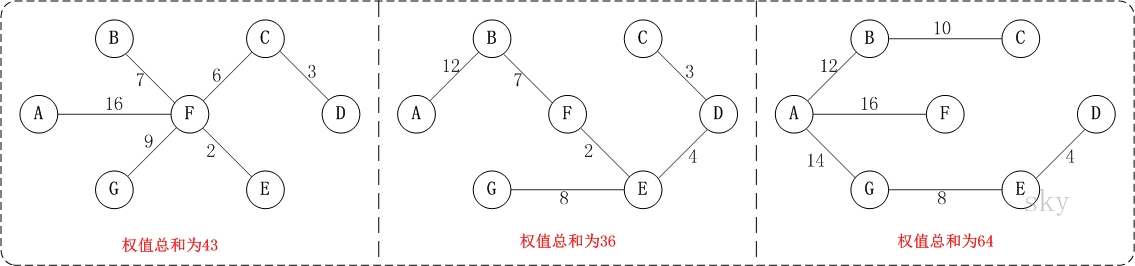


**二、kruskal算法（克鲁斯卡尔）**

在含有n个顶点的连通图中选择n-1条边，构成一棵**极小连通子图**，并使该连通子图中n-1条边上权值之和达到最小，则称其为连通网的最小生成树。



例如，对于如上图G4所示的连通网可以有多棵权值总和不相同的生成树。



**1、理解1**

<http://www.cnblogs.com/skywang12345/p/3711504.html>

克鲁斯卡尔(Kruskal)算法，是用来求加权连通图的最小生成树的算法。

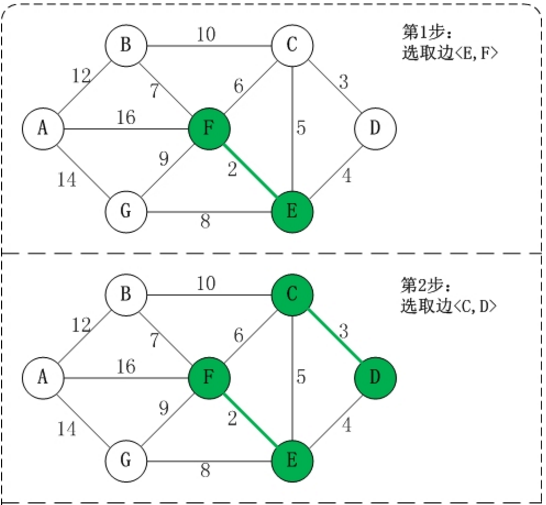
**1）基本思想及做法**

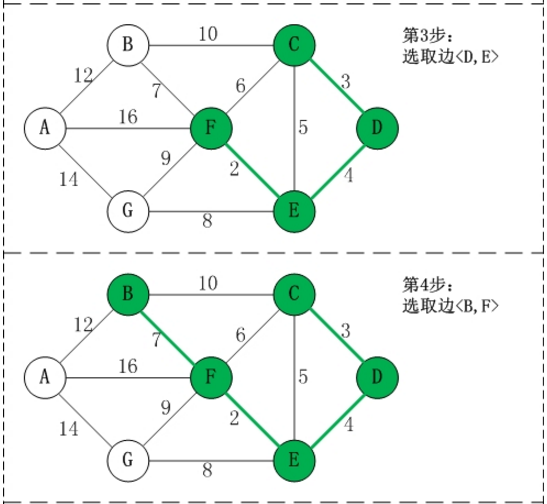
**基本思想：**按照权值*从小到大的顺序选择n-1条边，并保证这n-1条边不构成回路*。（即先对边从小到大排序，而后依次选择）

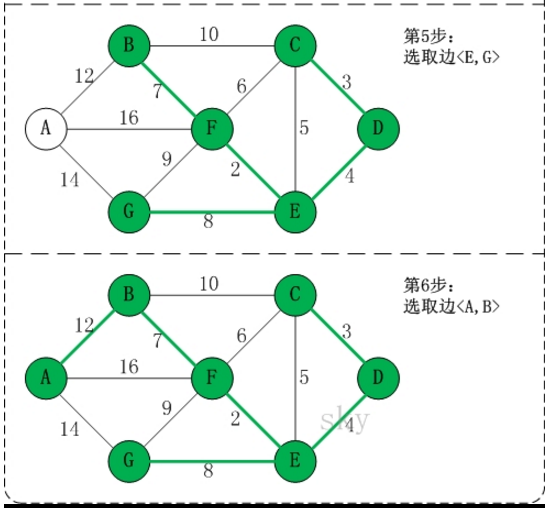
**具体做法：**首先**构造一个只含n个顶点的森林**，然后依权值从小到大从连通网中**选择边加入到森林**中，并使森林中不产生回路，直至森林变成一棵树为止。

**2）算法图解**

以上图G4为例，来对克鲁斯卡尔进行演示(假设，用**数组R保存最小生成树结果**)。







* **第1步：**将边<E,F>加入R中。边<E,F>的权值最小，因此将它加入到最小生成树结果R中。
* **第2步：**将边<C,D>加入R中。上一步操作之后，边<C,D>的权值最小，因此将它加入到最小生成树结果R中。
* **第3步：**将边<D,E>加入R中。上一步操作之后，边<D,E>的权值最小，因此将它加入到最小生成树结果R中。
* **第4步：**将边<B,F>加入R中。上一步操作之后，边<C,E>的权值最小，但<C,E>会和已有的边**构成回路**；因此，跳过边<C,E>。同理，跳过边<C,F>。将边<B,F>加入到最小生成树结果R中。
* **第5步：**将边<E,G>加入R中。上一步操作之后，边<E,G>的权值最小，因此将它加入到最小生成树结果R中。
* **第6步：**将边<A,B>加入R中。上一步操作之后，边<F,G>的权值最小，但<F,G>会和已有的边构成回路；因此，跳过边<F,G>。同理，跳过边<B,C>。将边<A,B>加入到最小生成树结果R中。
* 此时，最小生成树构造完成！它包括的边依次是：<E,F> <C,D> <D,E> <B,F> <E,G> <A,B>。

**3）算法分析**

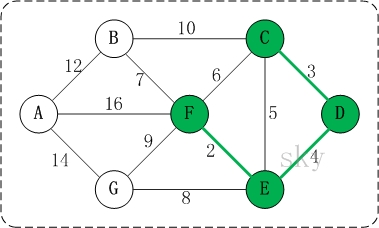
根据前面介绍的克鲁斯卡尔算法的基本思想和做法，我们能够了解到，克鲁斯卡尔算法重点需要解决的以下两个问题：

**问题一** 对图的*所有边按照权值大小进行排序*。

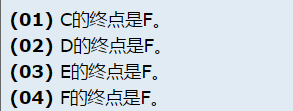
**问题二** 将边*添加到最小生成树中时，怎么样判断是否形成了回路*。

**问题一**很好解决，采用*排序算法进行排序*即可。

**问题二**，处理方式是：*记录顶点在"最小生成树"中的终点，顶点的终点是"在最小生成树中与它连通的最大顶点"*(关于这一点，后面会通过图片给出说明)。然后每次需要将一条边添加到最小生存树时，判断**该边的两个顶点的终点是否重合，重合的话则会构成回路（回路的终点判断）**。以下图来进行说明：



在将<E,F> <C,D> <D,E>加入到最小生成树R中之后，这几条边的顶点就都有了终点：



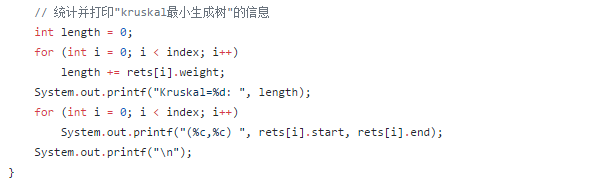
关于**终点**，就是*将所有顶点按照从小到大的顺序排列好之后*；**某个顶点的终点就是"与它连通的最大顶点"**。 因此，接下来，虽然<C,E>是权值最小的边。但是C和E的重点都是F，即它们的终点相同，因此，将<C,E>加入最小生成树的话，会形成回路。这就是判断回路的方式。

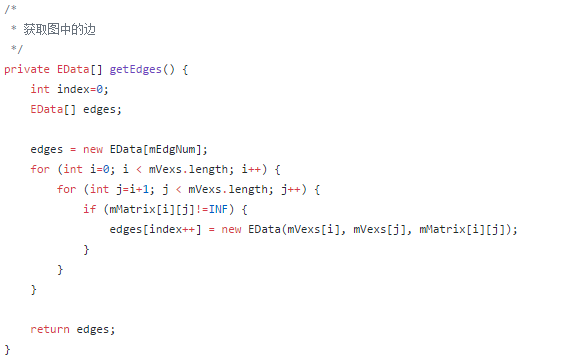
**4）代码实现**

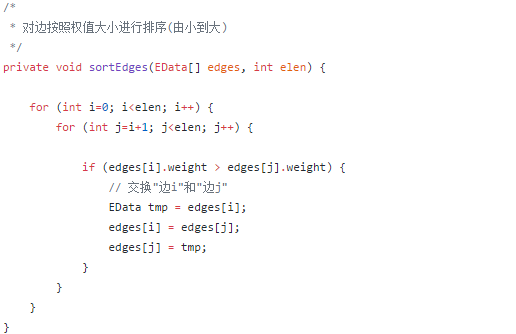
<https://github.com/wangkuiwu/datastructs_and_algorithm/blob/master/source/graph/kruskal/udg/java/MatrixUDG.java>

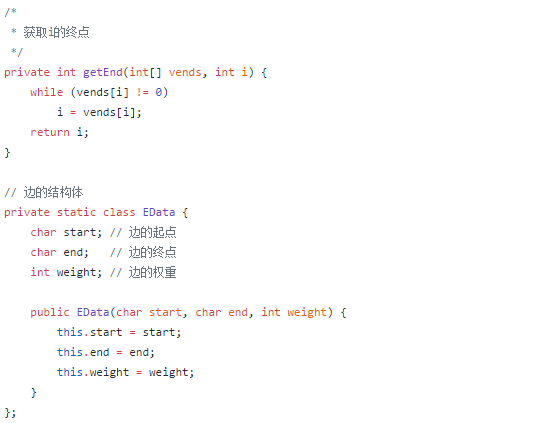
构建新的数据结构，保存每条边的相关信息

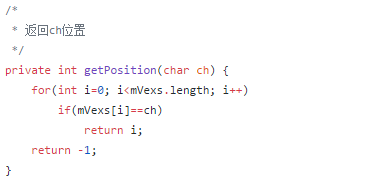












**2、理解2**

<http://www.layz.net/LAOJ/suanfa/s9-6.html>

**1）基本概念及原理**

**Kruskal算法以边为主导地位，始终选择当前可用的最短边**。那我们如何依次选择这n-1条边，使得这n-1条边的总长度最短呢？

要选择最短的边，我们首先想到给这些边**从小到大排序**，然后从小到大依次选择可用的边……直到选择了n-1条可用边为止。

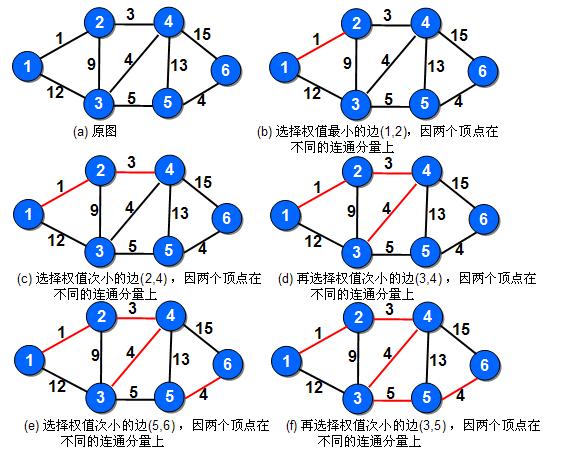
**Kruskal算法的原理：**先对所有的边按权值从小到大进行排序，然后从最小的可用边开始选，依次选择每一条边，直到选择了n-1条可用边为止。

**2）算法伪代码**

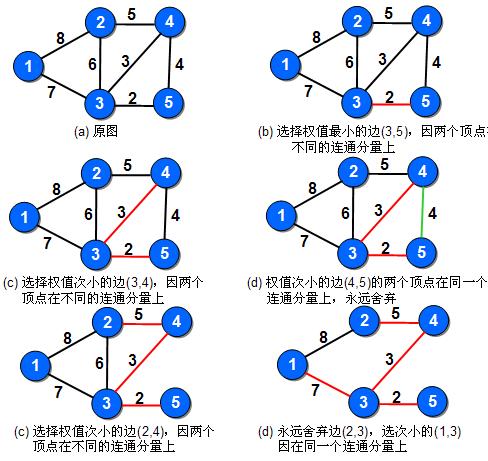


**3）实例**

应用Kruskal算法构造最小生成树的过程如下：



从上面可以看出，*Kruskal算法的关键在于如何判断一条边的两个顶点是否属于同一个连通分量*。如果是，则舍弃这条边，因为这样会产生回路。如果不是，则选用。例如下图(d)中，边(4,5)的权值为4，是剩下的边中最小的，选择该边，但是因为该边的两个顶点在同一个连通分量中，如果选用这条边，这个连通分量中就出现了回路（环）。而我们知道，最小生成树中是一棵树，是不能有环的。



**如何判断一条边的两个顶点是否属于同一个连通分量？**我们可以使用前面学习的**并查集**。如果一条边的两个顶点有共同的祖先，那么它们就属于同一个连通分量。

Kruskal算法注意的几点：

**1、边的存储**

struct edge{

int a;

int b;

int w;

} e[m];

pascal为：e:array[0..100]of record a,b,w:integer;end;

**2、Kruskal算法子程序**

