7.4.3 克鲁斯卡尔算法

- 克鲁斯卡尔（Kruskal）算法是一种按权值的递增次序选择合适的边来构造最小生成树的方法。
- 从连通网G=(V，E)中找最小生成树T=(V，TE)。

克鲁斯卡尔算法的思想：

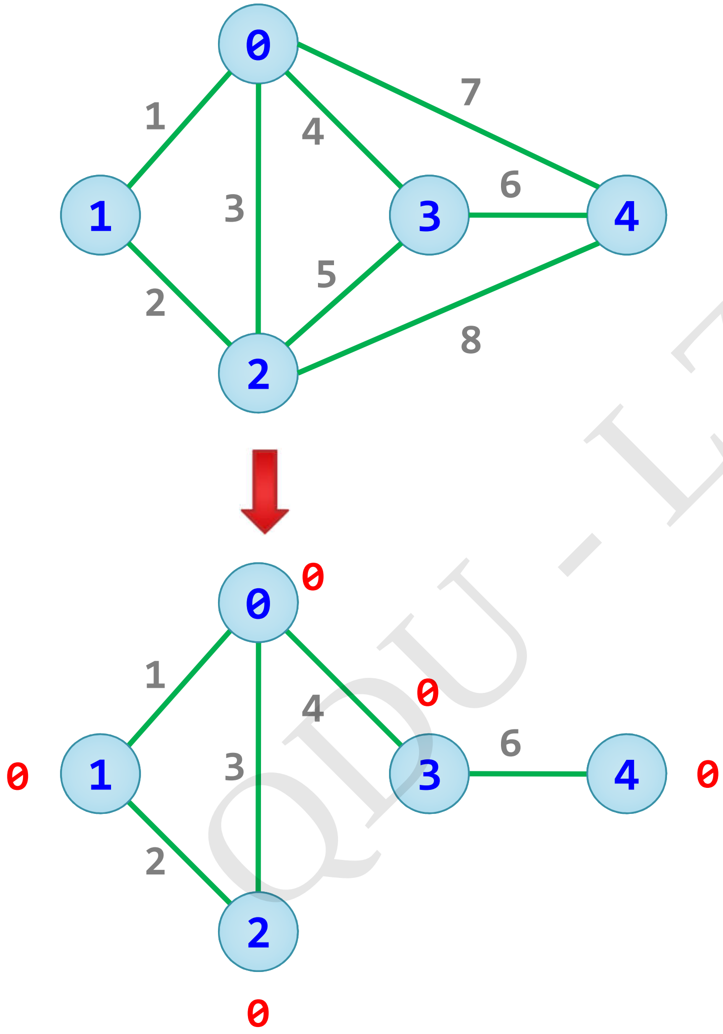
- 初值：U=V，TE={}。
- 对G中的边按权值大小从小到大依次选取。
- (1) 选取权值最小的边(vi，vj)，若边(vi，vj)加入到TE后形成回路，则舍弃该边(边(vi，vj)；否则，将该边并入到TE中，即TE=TE∪{(vi，vj)}。
- (2) 重复(1)，直到TE中包含有n-1条边为止。

- 实现克鲁斯卡尔算法的关键是如何判断选取的边是否与生成树中已保留的边形成回路？
  - 为此设置一个辅助数组vset[0..n-1]，它用于判定两个顶点之间是否连通。
  - 数组元素vset[i]（初值为i）代表编号为i的顶点所属的连通子图的编号。
  - 对于边（i，j），若vset[i]==vset[j]⇨不选；否则选取。
  - 一旦选取边（i，j），将两个连通分量的所有vset值改为vset[i]或者vset[j]。

首先需要对所有边按权值递增排序，为此定义一个具有如下类型的边数组E[]：

```
typedef struct
{   int u;                          //边的起始顶点
    int v;                          //边的终止顶点
    int w;                          //边的权值
} Edge;                             //边数组元素类型
```

✓ 从图的邻接矩阵g中提取出边数组E，然后按边权值递增排序。



构造最小生成树

实现克鲁斯卡尔算法：假设采用直接插入排序法对边集E按权值递增排序。

```
void SortEdge(Edge E[],int e)
//直接插入排序：对E数组按权值递增排序
{   int i,j,k=0;
    Edge temp;
    for (i=1;i<e;i++)
    {    temp=E[i];
         j=i-1;   //从右向左在有序区E[0..i-1]中找E[i]的插入位置
         while (j>=0 && temp.w<E[j].w)
         {    E[j+1]=E[j];      //将权值大于E[i].w的记录后移
              j--;
         }
       E[j+1]=temp;              //在j+1处插入E[i]
    }
}
```

```c
void Kruskal(MatGraph g)      //输出求得的最小生树的所有边
{
   int i,j,u1,v1,sn1,sn2,k;
   int vset[MAXVEX];           //建立数组vset
   Edge E[MAXE];               //建立存放所有边的数组E
   k=0;                        //k统计E数组中边数

   for (i=0;i<g.n;i++)         //由图的邻接矩阵g产生的边集数组E
     for (j=0;j<=i;j++)        //提取邻接矩阵中部分数据
       if (g.edges[i][j]!=0 && g.edges[i][j]!=INF)
           {   E[k].u=i;
               E[k].v=j;
               E[k].w=g.edges[i][j];
               k++;            //累加边数
           }
   SortEdge(E,k);              //对E数组按权值递增排序

   for (i=0;i<g.n;i++)
      vset[i]=i;               //初始化辅助数组
```

```
k=1;            //k表示当前构造生成树的第几条边,初值为1
j=0;            //j为E数组下标,初值为0
while (k<g.n)                  //生成的边数小于n时循环
{  u1=E[j].u;
   v1=E[j].v;                  //取一条边的头尾顶点
   sn1=vset[u1];
   sn2=vset[v1];               //分别得到两顶点所属的集合编号
   if (sn1!=sn2)               //两顶点属不同的集合,取该边
   {  printf("边(%d,%d),权值为%d\n",u1,v1,E[j].w);
      k++;                     //生成的边数增1
      for (i=0;i<g.n;i++)      //两个集合统一编号
        if (vset[i]==sn2)      //集合编号为sn2的改为sn1
          vset[i]=sn1;
   }
   j++;                        //扫描下一条边
 }
}
```

- 上述克鲁斯卡尔算法不是最优算法，在改进后可达到$O(e\log_2 e)$。一般认为克鲁斯卡尔算法的时间复杂度是$O(e\log_2 e)$。
- 由于仅仅与e有关，所以克鲁斯卡尔算法特别适合于稀疏图求最小生成树。

END