8.5.4 克鲁斯卡尔算法

克鲁斯卡尔（Kruskal）算法也是一种求带权无向图的最小生成树的构造性算法。

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。

克鲁斯卡尔（Kruskal）算法过程：

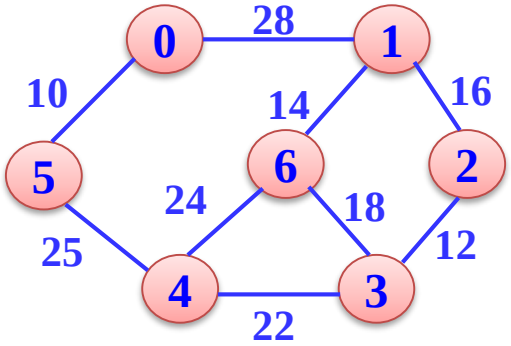
表示最小生成树的边集

（1）置U的初值等于V（即包含有G中的全部顶点），TE的初值为空集（即图T中每一个顶点都构成一个连通分量）。

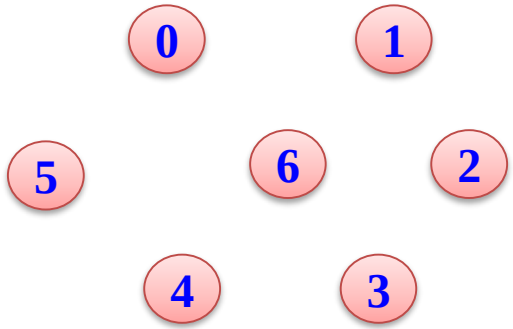
（2）将图G中的边按权值从小到大的顺序依次选取：

❶ 若选取的边未使生成树T形成回路，则加入TE；

❷ 否则舍弃，直到TE中包含$(n-1)$条边为止。



有条件地加入
$(n-1)$条边

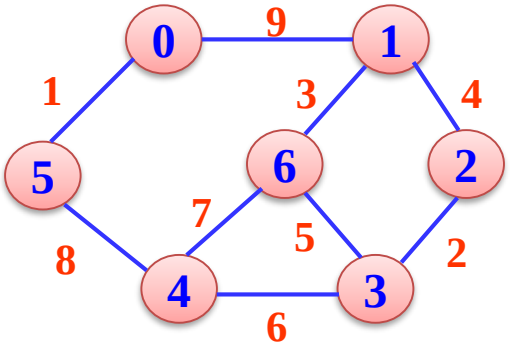


TE={}

Kruskal算法示例的演示

克鲁斯卡尔算法求解最小生成树的过程

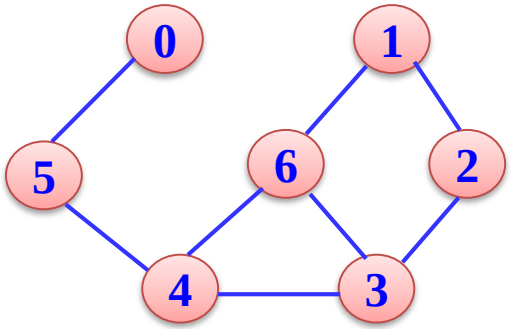
Kruskal算法示例的演示



操作

取8号边

选取了$n-1$条边



最小生成树

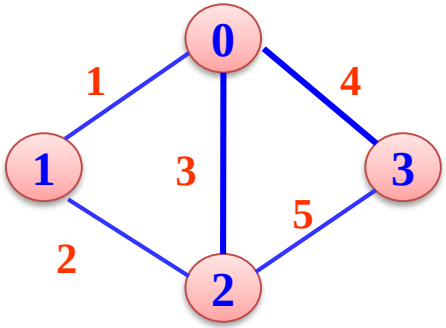
克鲁斯卡尔算法求解最小生成树的过程

算法设计（解决3个问题）：

- 图采用哪种存储结构更合适？　　邻接矩阵
- 边的排序问题？　　这里采用直接插入排序算法
- 如何解决加入一条边后是否出现回路？

采用连通分量编号或顶点集合编号

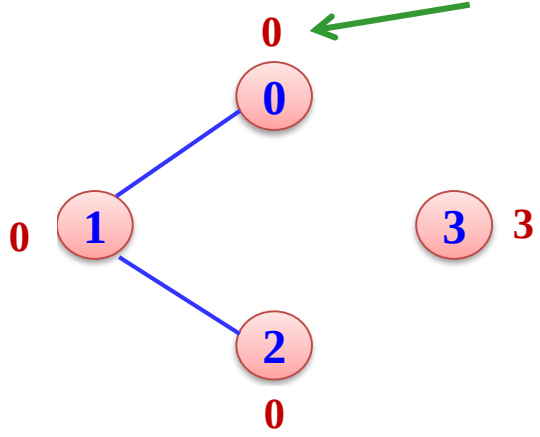
Kruskal算法如何解决出现回路的问题演示



操作

取3号边

vset[0]：连通分量编号



3号边的两个顶点的vset值相同，不能添加！

在实现克鲁斯卡尔算法Kruskal()时，用数组E存放图G中的所有边，其类型如下：

```
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;
```

Edge E[MAXV];

克鲁斯卡尔（Kruskal）算法如下：

```
void Kruskal(MGraph g)
{    int i,j,u1,v1,sn1,sn2,k;
     int vset[MAXV];
     Edge E[MaxSize];          //存放所有边
     k=0;                      //E数组的下标从0开始计
     for (i=0;i<g.n;i++)       //由g产生的边集E
          for (j=0;j<g.n;j++)
               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++;
               }
      InsertSort(E,g.e);       //用直接插入排序对E数组按权值递增排序
      for (i=0;i<g.n;i++)      //初始化辅助数组
           vset[i]=i;
```

```
        k=1;                                    //k表示当前构造生成树的第几条边
        j=0;                                    //E中边的下标,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];                       //分别得到两个顶点所属的集合编号
            if (sn1!=sn2)                       //两顶点属于不同的集合
            {       printf("  (%d,%d):%d\n",u1,v1,E[j].w);
                    k++;                        //生成边数增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
        }
}
```

上述算法不是最优的。

改进：堆排序、并查集

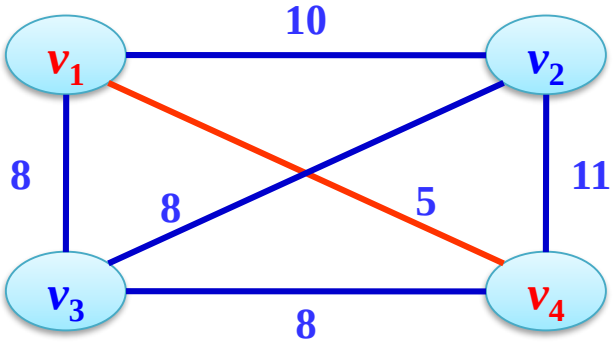
Kruskal算法的时间复杂度为$O(e\log_2 e)$。

思考题

为什么说Kruskal算法更适合稀疏图求最小生成树。

【例8-7】求下面带权图的最小（代价）生成树时，可能是克鲁斯卡（kruskal）算法第二次选中但不是普里姆（Prim）算法（从v_4开始）第2次选中的边是_____。

A.(v_1,v_3)　　B.(v_1,v_4)　　C.(v_2,v_3)　　D.(v_3,v_4)



注：2015年全国考研题

kruskal：

1：(v_1,v_4)，2：(v_1,v_3) 或(v_3,v_4) 或(v_2,v_3)，…

Prim（从v_4开始）：

1：(v_1,v_4)，2：(v_1,v_3) 或(v_3,v_4)，不可能是(v_2,v_3)

答案为C

思考题

有n（n>10）台计算机，已知它们的坐标位置信息，需要连成一个网络。现在求所花最少网线长度，问：

❶ 采用什么算法求解？

❷ 采用哪个算法最好？

——本讲完——