最小生成树

Minimum Spanning Tree

Prim和Kruskal 算法

引例：丛林道路

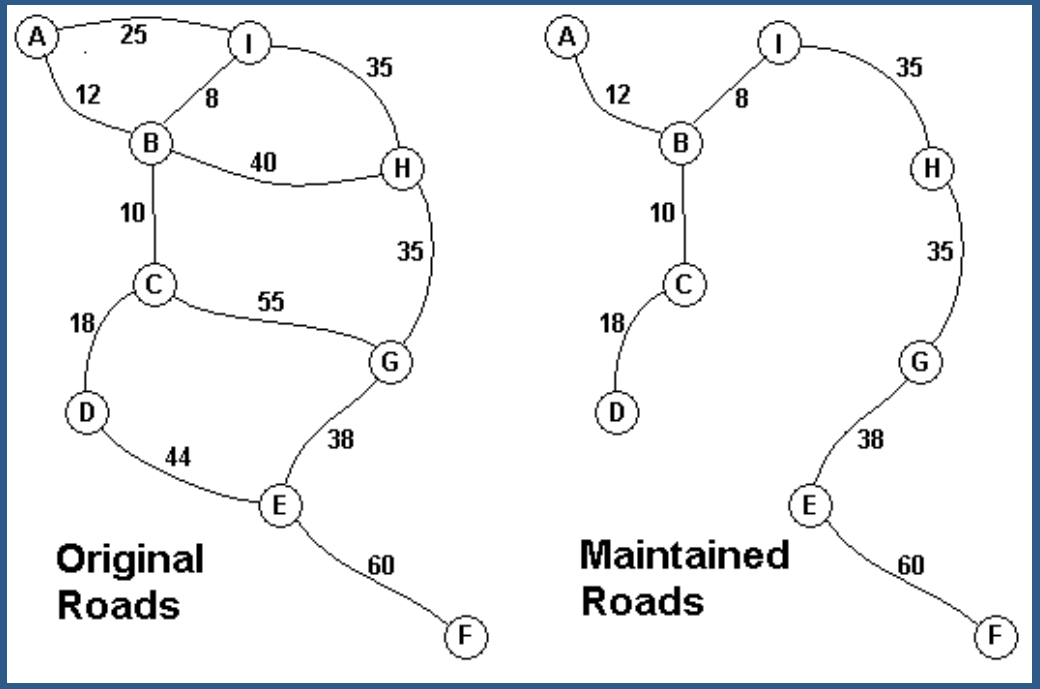
热带小岛“拉格瑞珊”的酋长面临一个问题：通过大量外来资金援助，在岛上的村庄之间修建了很多道路。但是由于岛上的丛林生长得特别快，掩盖了很多公路，而如此庞大的公路网络维护起来费用就相当昂贵。酋长必须选择停止某些公路的维护工作。下面的地图显示了现在正在使用的道路和每条道路每个月需花费的维护费用。酋长要告诉委员会只保留哪些公路便可连接岛上的所有村庄并且使得每个月的维护费用总和最少。下面右边的地图便是最便宜的公路连接方案。

输入格式：

第一行，一个数字n(1<n<27)表示村庄的个数。

接下来n-1按字母表顺序排列。每行开头是一个大写字母，表示村庄的编号。接着是一个数字k，表示连接该村庄的公路数量。如果公路条数k不为0，则后面是和该村庄直接相连的k个村庄的编号（按字母表顺序排列）和维护连接这两个村庄的公路的花费。

注：每个村庄最多有15条公路和其他村庄直接相连，每条公路的维护费用不超过75，公路的总条数不超过100



输出格式：

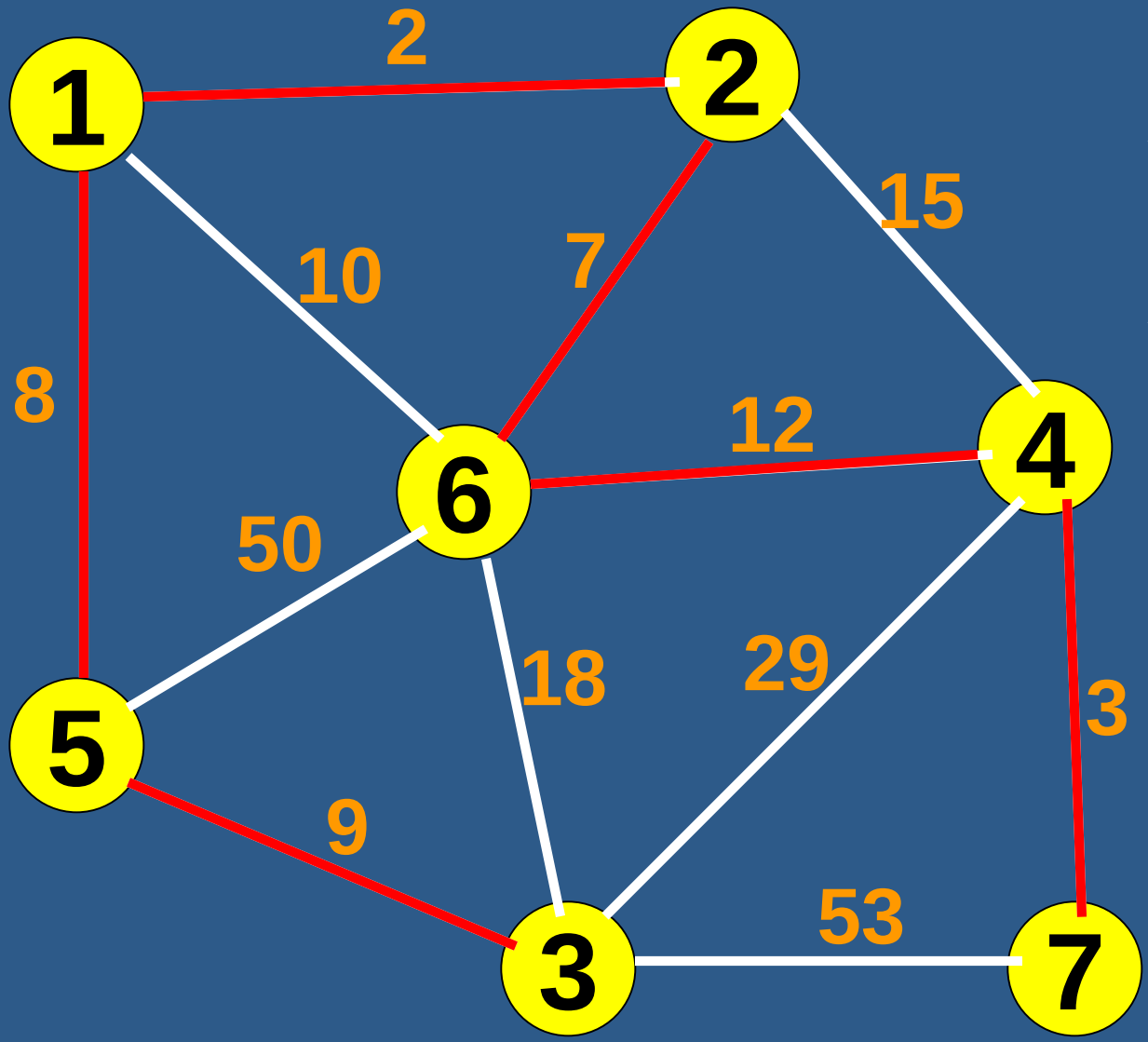
一行：连接所有村庄的公路的最小维护费用。

输入样例：

```
9
A 2 B 12 I 25
B 3 C 10 H 40 I
8
C 2 D 18 G 55
D 1 E 44
E 2 F 60 G 38
F 0
G 1 H 35
H 1 I 35
```

输出样例：

```
216
```



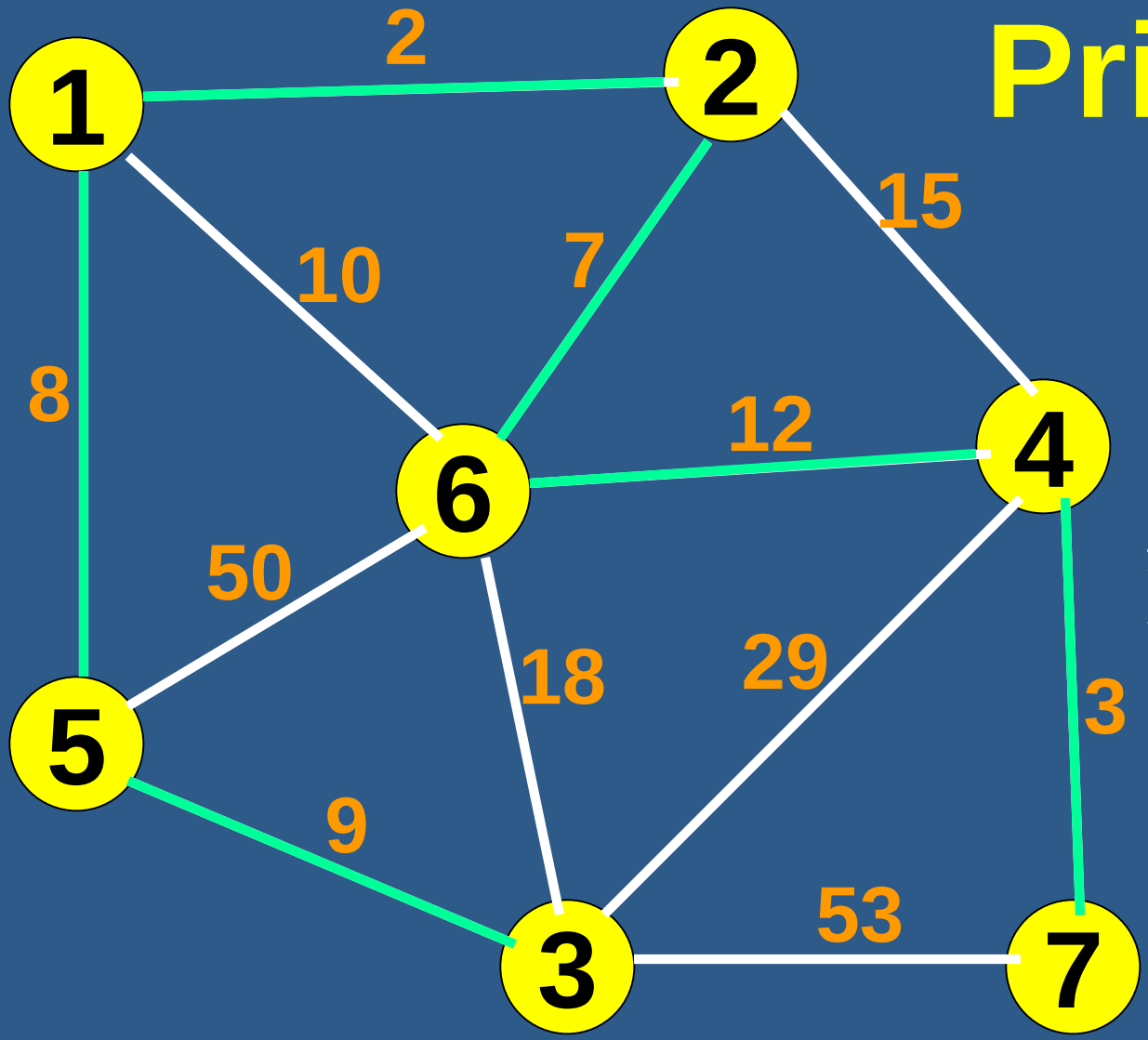
用公路连接n个村庄，找出一种方案，使得总的费用最少。

（无向图）

Prim

Kruskal

Prim 1957



1.任选一个点，加入生成树集合

2.在未加入生成树的点中，找出离生成树距离最近的一个点，将其加入生成树。

3.反复执行2，知道所有点都加入了生成树

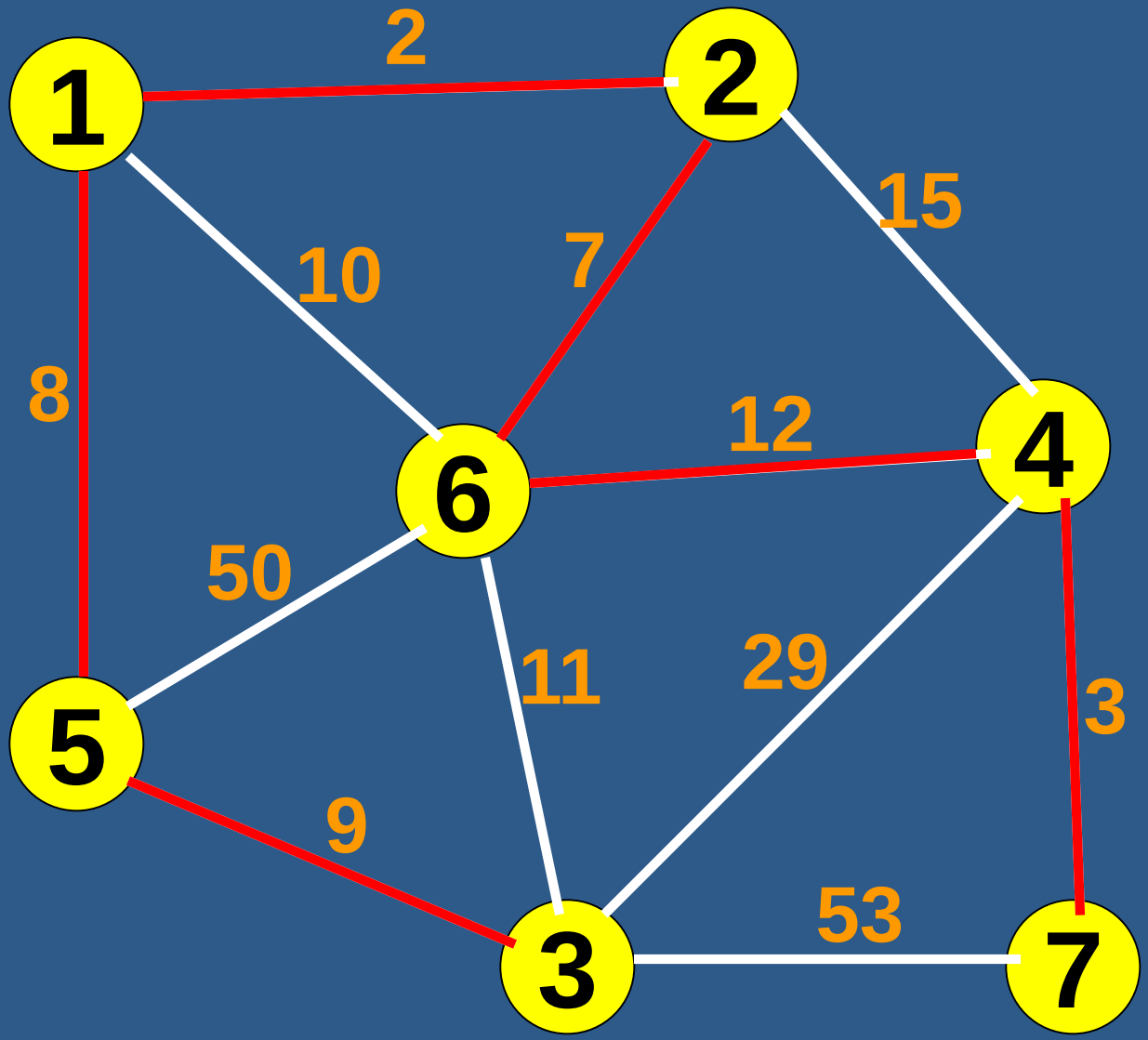
```
void prim(int x)                                  //开始时任选一点x加入生成树，故一开始树中只有一个点x
{
    int   dis[101],path[101],i,j,k,Min;           //dis记录各节点到生成树的最小距离
    for(i=1;i<=n;i++)                             //path[i]记录生成树中与节点i最近的一个节点的编号，用于记录路径
    {   dis[i]=map[i][x];     path[i]=x;  }       //初始化，将每个节点到生成树的最小距离赋值为它到点x的距离,将生成树中与i最近的节点赋值为x(因为此时生成树中只有一个节点x)
    for(i=1;i<=n-1;i++)                           //除x外，还有n-1个节点要讨论
    {
        Min=inf;                                  //inf为自定义的一个表示无穷大的数。
        for(j=1;j<=n;j+=)                         //找出在未加入生成树的节点中，离当前生成树距离最近的一个节点
            if ((dis[j]!=0)&&(dis[j]<Min))
            {   Min=dis[j];    k=j;    };
        dis[k]=0;                                 //将找出的离生成树距离最近的节点t到生成树的距离赋值为0，表示它已经加入到树中了
        for(j=1;j<=n;j++)                         //k加入生成树后，可能有其他节点到生成树的最短距离发生变化，调整他们的path值
            if(dis[j]>map[j][k])                  //讨论未加入到生成树的节点j与k的距离是否比j原来到生成树的最短距离dis[j]要短，如果是，则用新的更短的距离取代原来的距离，并将生成树离j最近的节点改成t
            { dis[j]=map[j][k];  path[j]=k; }
    }
}
```

输出最短路径总长度

```
for(i=1;i<=n;i++)
   if(path[i]!=i)
        total=total+map[i][path[i]]; cout<<total;
```

prim时间复杂度$O(n^2)$

堆优化O(nlogn)



Kruskal 1956

注：在选择新的边加入时，该边的两个端点不能在同一棵已生成的树上！

每次选最短的边加入生成树

O(mlogm)

Kruskal

Kruskal算法基本思想：

每次选不属于同一生成树的且权值最小的边的顶点，将边加入生成树，并将所在的2个生成树合并，直到只剩一个生成树

排序使用Quicksort

检查是否在同一生成树用并查集

总复杂度O(eloge)

$O(m\log^m)$ m表示边的数量

```
#define    maxn 101
#define   maxe 10001
struct line{
          int a,b; //边的2个顶点
          int len; //边的长度
        };
line edge[maxe];   //保存所有边的信息
int father[maxn]    //father存i的父亲节点
int n,e;                 //n为顶点数，e为边数
```

```
void quicksort(int l,int r) //按边长进行快速排序
{
    int mid,i,j;   line temp;
    mid=edge[(l+r)/2].len;
    i=l;j=r;
    while(i<=j)
   {
        while(edge[i].len<mid)i++;
        while(edge[j].len>mid)j--;
        if(i<=j)
        {
          temp=edge[i];
          edge[i]=edge[j];
          edge[j]=temp;
          i++;j--;
        }
    }
    if(l<j)quicksort(l,j);
    if(i<r)quicksort(i,r);
}
```

```
void init()//初始化
{
    int i;
    scanf("%d%d",&n,&e);
    for(i=1;i<=e;i++)
        scanf("%d%d%d",&edge[i].a,edge[i].b,edge[i].len); //读入图的信息
    for(i=1;i<=n;i++)father[i]=i;                  //初始化并查集
    quicksort(1,e);                                //使用快速排序将边按权值从小到大排列
}
```

```
int getfather(int x)//并查集，用来判断2个顶点是否属于同一个生成树
{
    if(x!=father[x])father[x]=getfather(father[x]);
    return father[x];
}

 void kruskal()
 {
    int x,y,k,cnt,tot;    //k为当前边的编号,tot统计最小生成树的边权总和
                          //cnt统计进行了几次合并。n-1次合并后就得到最小生成树
    cnt=0;k=0; tot=0;
    while(cnt<n-1)     //n个点构成的生成树总共只有n-1条边
    {
        k++;
        x=getfather(edge[k].a);
        y=getfather(edge[k].b);
        if(x!=y)
        {
          father[x]=y; //合并到一个生成树
          tot=tot+edge[k].len;
          cnt++;
        }
    }
    printf("%d\n",tot);
 }
```

```
int main()
{
    init();
    kruskal();
    return 0;
}
```

例：征兵 poj3723

Windy要组建一支军队，有N个女孩和M个男孩来应征入伍，每招募一个士兵，需要向其支付10000块的入伍费，这样Windy需要花费(N+M)*10000块钱，这是一笔不小的费用。

Windy发现这些人中有的是亲戚关系，即如果第i号女生和第j号男生是亲戚，那么招募其中一个后，再利用他们的亲戚关系去招募另外一个，就可以少花费D_{ij}块钱。现在已知这些人中有R对存在亲戚关系，问Windy至少要用多少钱才能组建成这支军队。

算法框架：

1.将每一对人看作一条边，边的权值为D_{ij}

2.求**最大生成树**，得到权值综合为cost;

3.最后支付的费用为(N+M)*10000-cost;

问题1：这些亲戚关系能否一定让这N+M个人得到一棵生成树？

问题2：prim或kruskal能否适用此题?

此题我们构成的图不一定连通，可用kruskal：

1.将边按权值由大到小排序；

2.按kruskal规则，每次选权值最大的边出来；

3.最后形成的不一定是一棵最大生成树，可能是一个**最大生成森林**

课后练习：1228,1819,1261，1457