最小生成树

Minimum Spanning Tree

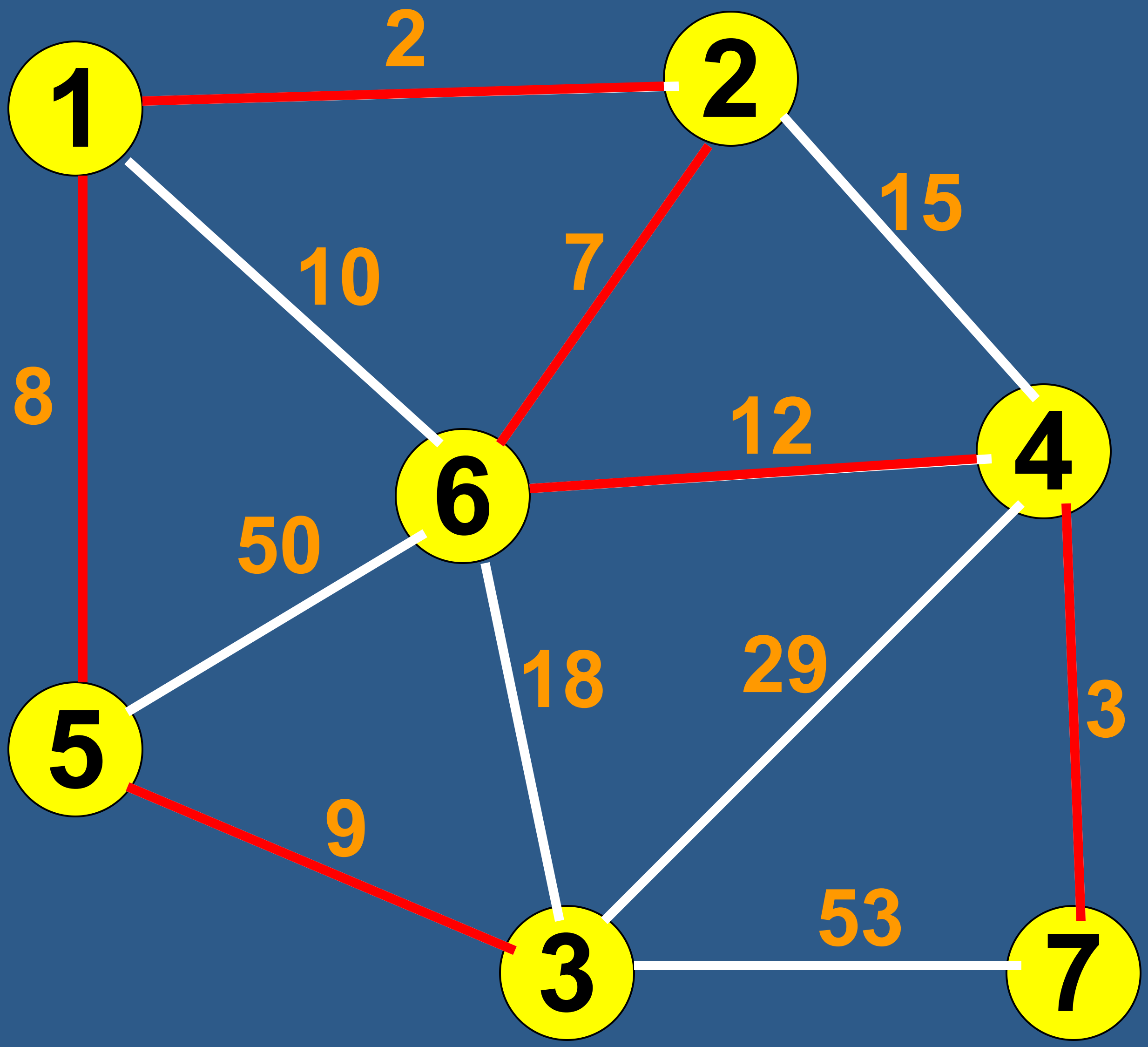
Prim和Kruskal 算法

引例：村长的难题

何老板是某乡村的村长，何老板打算给该村的所有人家都连上网。

该村有n(1<=n<=1000)户人家，编号1到n。由于地形等原因，只有m(1<=m<=50000)对人家之间可以相互牵线。在不同人家间牵线的长度不一定相同。比如在Ai与Bi之间牵线需要Ci米长的网线。

整个村的网络入口在1号人家，何老板的问题是：是否能使得所有人家都连上网？使所有人家都连上网，最少需要多少米网线？



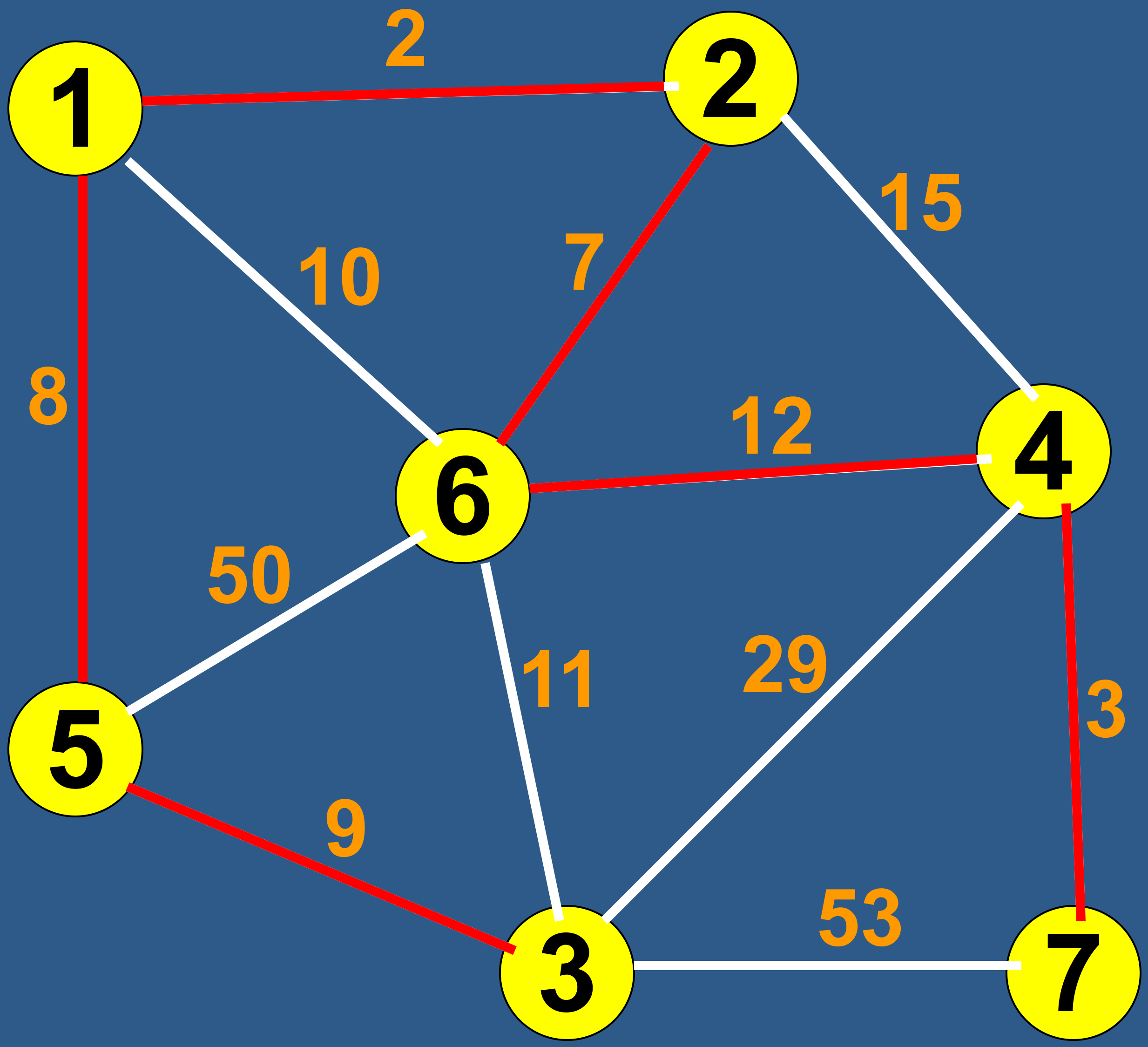
用网线连接n户人家，找出一种方案，使得总的长度最少。

（无向图）

Prim

Kruskal

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)

$O(mlog^m)$ 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,m;             //n为顶点数，m为边数
```

```
bool cmp(Line a,Line b) //按边长由小到大排序
{
       return a.len<b.len;
}
```

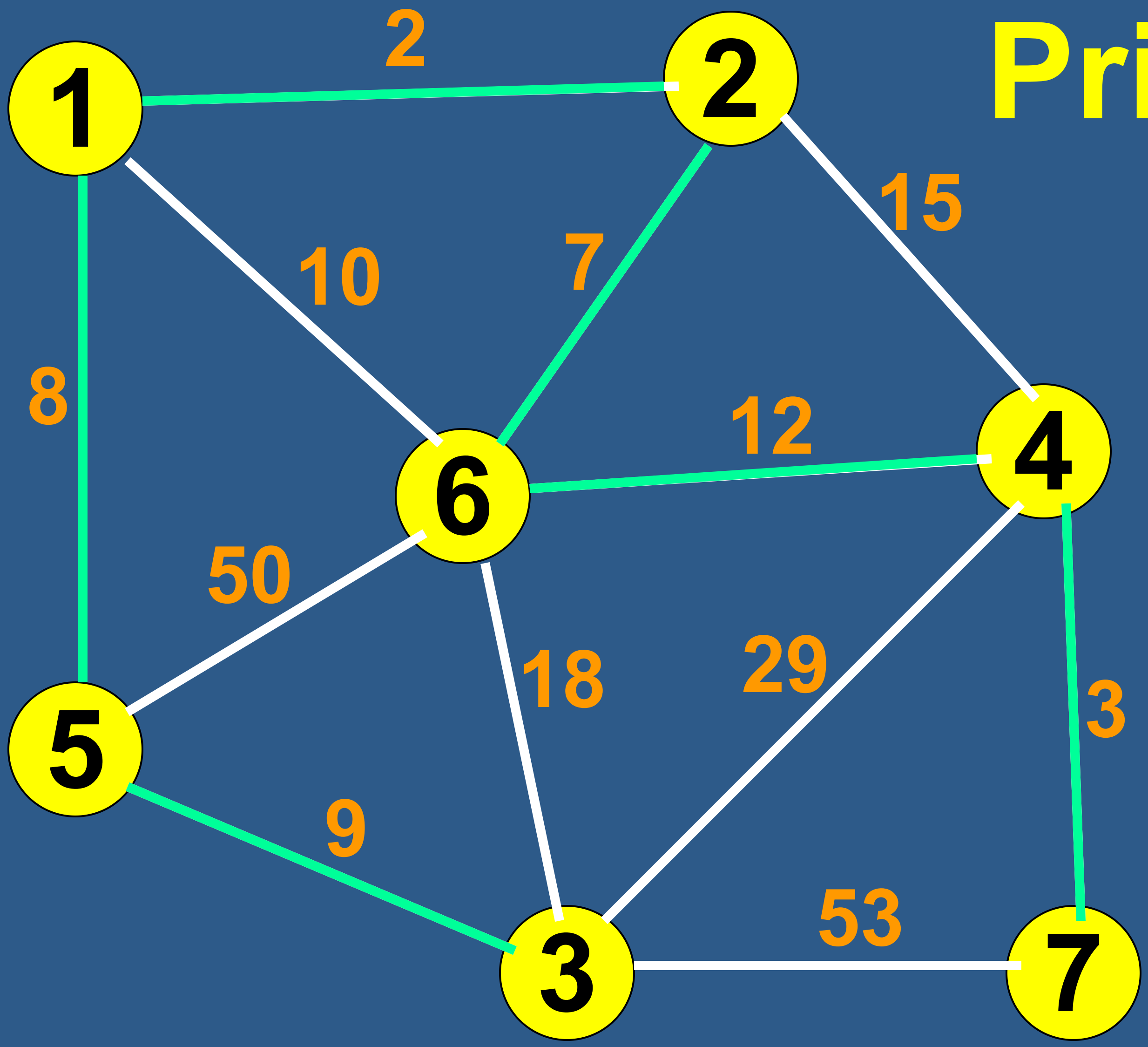
```
void init()//初始化
{
   scanf("%d%d",&n,&m);
   for(int i=1;i<=m;i++)
         scanf("%d%d%d",&Edge[i].a,Edge[i].b,Edge[i].len); //读入图的信息
   for(int i=1;i<=n;i++)Father[i]=i; //初始化并查集
   sort(Edge+1,Edge+1+m,cmp);        //使用快速排序将边按权值从小到大排列
}
```

```
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;
}
```

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
{                                                //dis记录各节点到生成树的最小距离
      int   dis[101],path[101],i,j,k,Min;        //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
      for(i=1;i<=n;i++)
      {   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])
                { 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
                  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)

例1：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**，得到权值总和为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**

次小生成树

次小生成树 方法一：

1.先用prim或kruskal求出最小生成树;

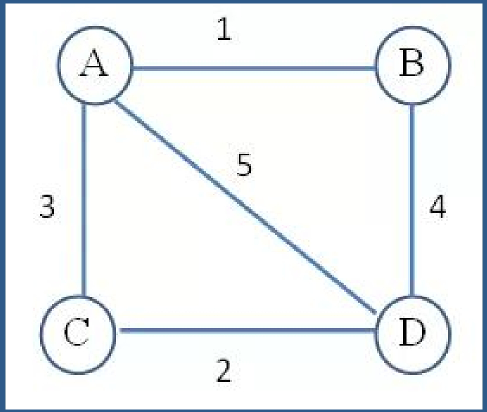
2.依次删除最小生成树上的每一条边，每删掉一条边，重新求一次最小生成树，记录下新生成树的边权之和，然后还原该边，继续讨论删除下一条边。

3.删边讨论过程中，新生成的树中，边权和最大的，就是所求的次小生成树

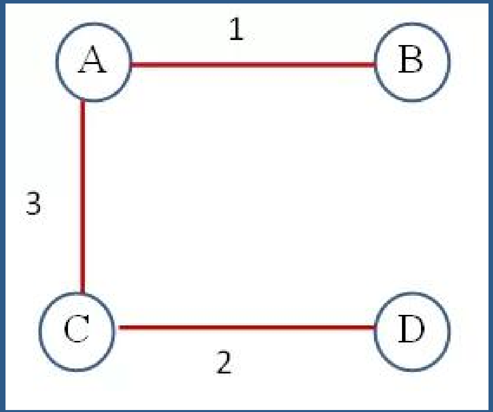
4.这种方法，prim的时间复杂度为$O(e*n^2)$，
kruskal时间复杂度为$O(n*eloge)$

次小生成树 方法二：

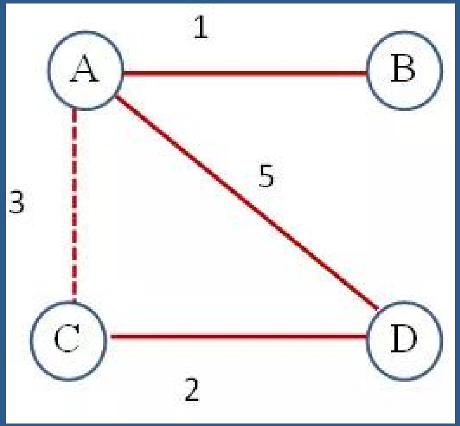
1.首先求最小生成树T；
2.枚举添加不在T中的边，则添加后一定会形成环；
3.找到环上边值第二大的边(**即环中属于T中的最大边**)，把它删掉，计算当前生成树的权值，取所有枚举修改的生成树的最小值，即为次小生成树；



最初的带权五向图G

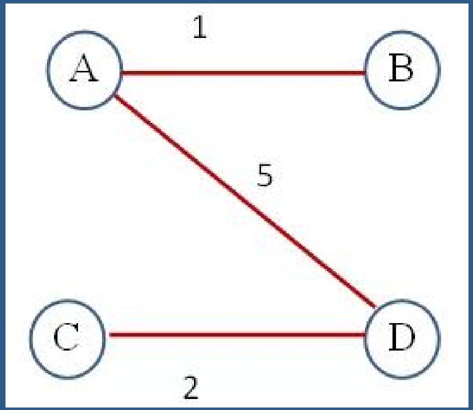


先求出图G的最小生成树T



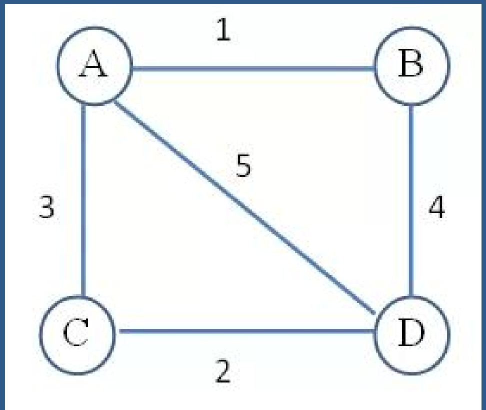
添加一条不再T中的边AD

将环ACD中第二大边AC删掉

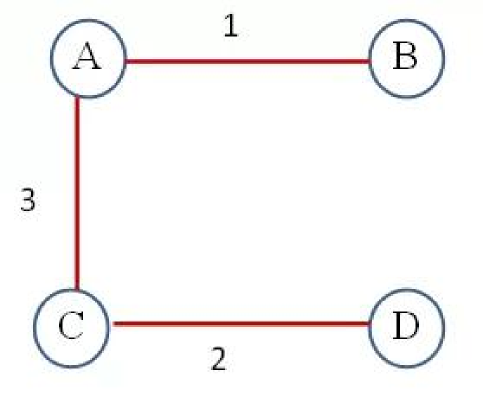


当前生成树的权值和为8

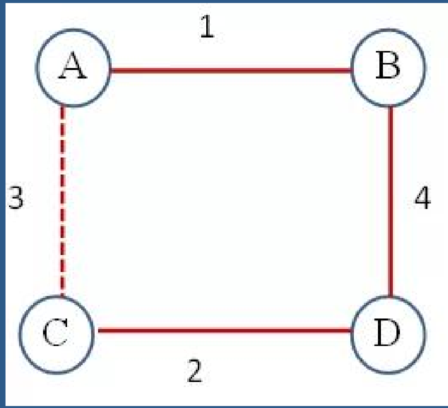
次小生成树 方法二：



最初的带权五向图G

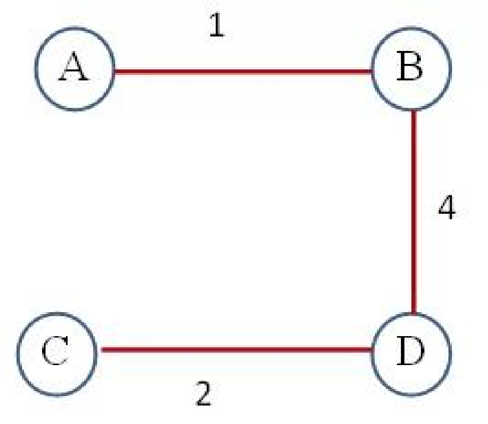


还原成最初图G的最小生成树T



添加一条不再T中的边边BD

将环ABCD中第二大边AC删掉



当前生成树的权值和为7

此时，我们求出了图G的次小生成树，权值为7

次小生成树 方法二：

具体实现步骤：

1.先用prim或kruskal求出最小生成树T；

2.从每个结点出发遍历(BFS或DFS)一次最小生成树T，用二维数组MaxLen[u][v]记录结点u出发到结点v的路径上，经过的边中，权值最大的一条边的权值；

3.然后枚举不在T中的边(x,y)，将其加入生成树后，将所在环的第二大边删除，删除后新的生成树的权值 =T- MaxLen[x][y] + Edge[x,y]的值；

4.min{ T- MaxLen[x][y] + Edge[x,y] }即是次小生成树的权值总和；

5.时间复杂度

求最小生成树：O(n^2)或O(eloge)

从每个点出发遍历最小生成树求MaxLen[x][y]: O(n^2)

依次添加每条不在T中的边：O(e)

总时间复杂度O(MST)+O(n^2)+O(e)

练习题目：POJ1679

思维训练：tree 出题人"陈立杰" bzoj 2654

给你一个无向带权连通图，每条边是黑色或白色。让你求一棵最小权的恰好有need条白色边的生成树。题目保证有解。

输入格式：

第一行V,E,need分别表示点数，边数和需要的白色边数。
接下来E行每行s,t,c,col表示这边的端点(点从0开始标号)，边权，颜色(0白色1黑色)。

输出格式：

一行表示所求生成树的边权和。

样例输入：

2 2 1
0 1 1 1
0 1 2 0

样例输出：

2

数据规模：

V<=50000,E<=100000
所有数据边权为[1,100]中的正整数

课后练习：1228,1819,1261,1457