图论

最小生成树(MST)

最优布线问题

题目描述

学校有n台计算机，为了方便数据传输，现要将它们用数据线连接起来。两台计算机被连接是指它们有数据线连接。由于计算机所处的位置不同，因此不同的两台计算机的连接费用往往是不同的。当然，如果将任意两台计算机都用数据线连接，费用将是相当庞大的。为了节省费用，我们采用数据的间接传输手段，即一台计算机可以间接的通过若干台计算机（作为中转）来实现与另一台计算机的连接。现在由你负责连接这些计算机，任务是使任意两台计算机都连通（不管是直接的或间接的）。

输入

第1行：整数n（2<=n<=100），表示计算机的数目。此后的n行，每行n个整数。第x+1行y列的整数表示直接连接第x台计算机和第y台计算机的费用。

输出

一个整数，表示最小的连接费用。

样例输入	样例输出
3 0 1 2 1 0 1 2 1 0	2

需要求什么?

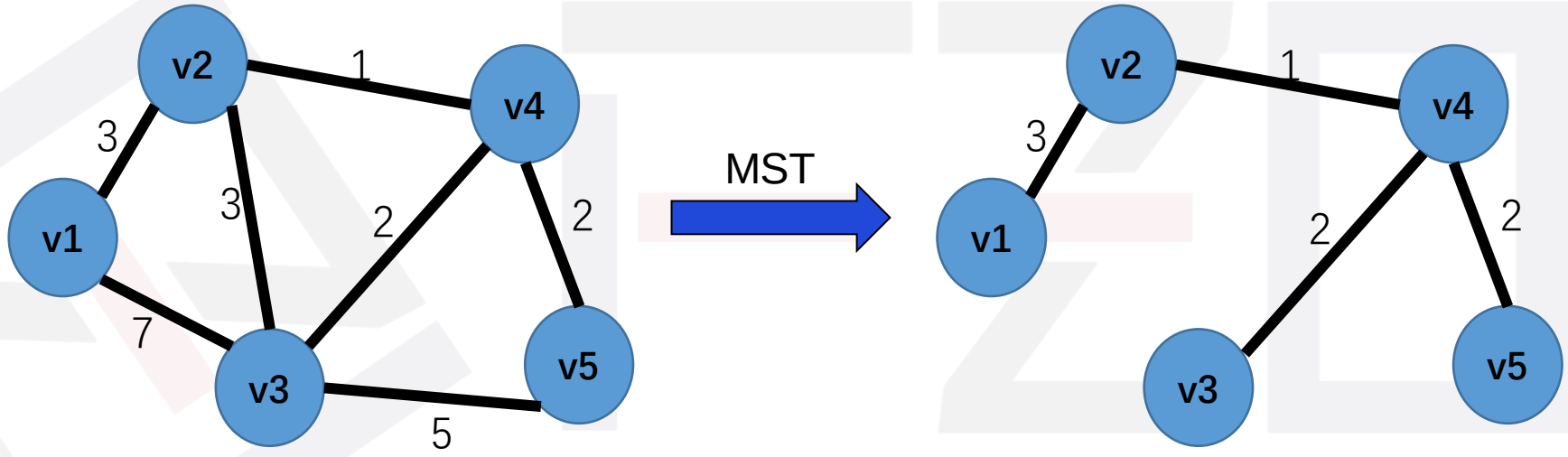
将n个点全部连通的最小权值

需要连接多少条边?

n-1

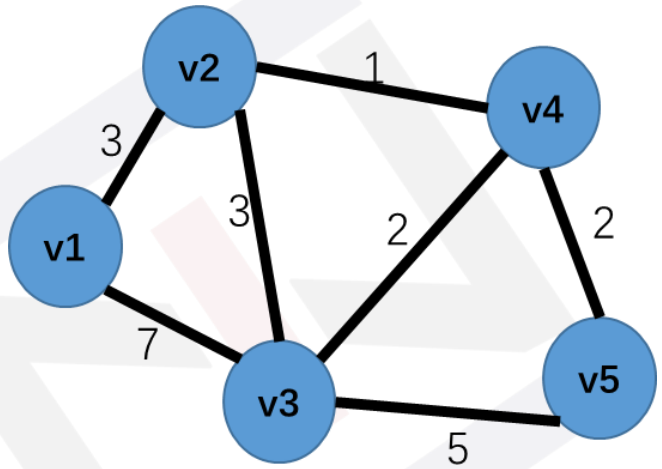
最小生成树(**MST**)：用n-1条边把n个点**连通**的**最小权值**

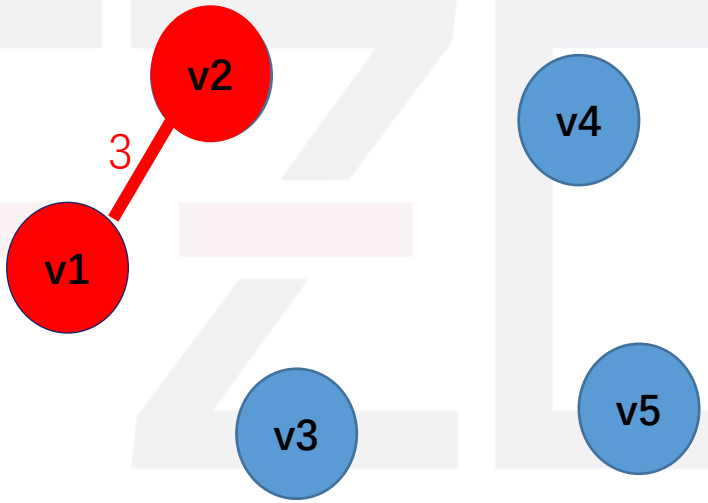
如何求解?



有没有什么思路?

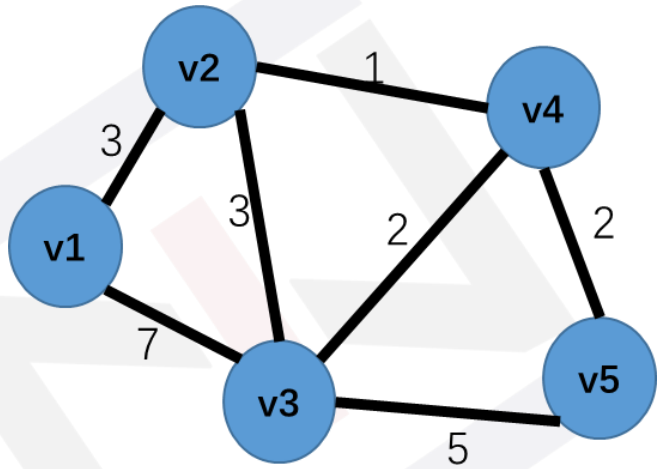
分析



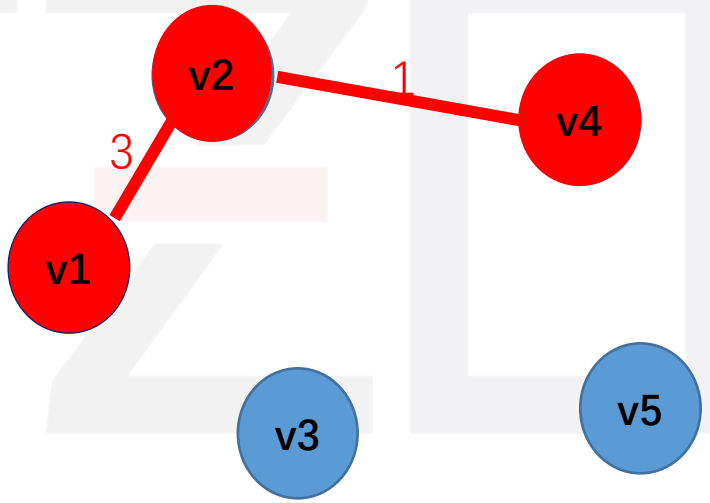


假设v1为树根，
可以确定哪一条边
一定在MST中？

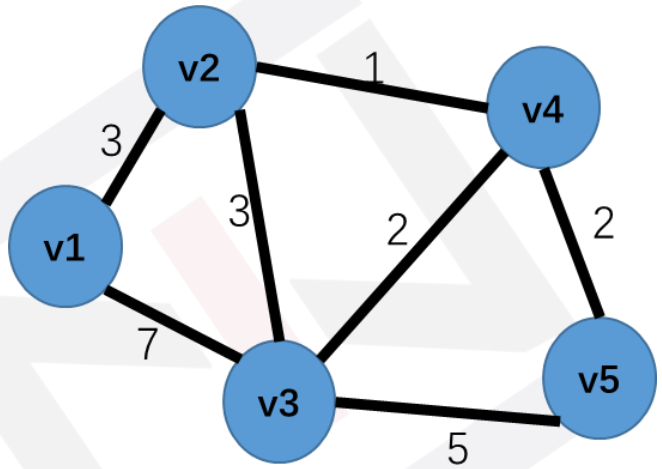
分析



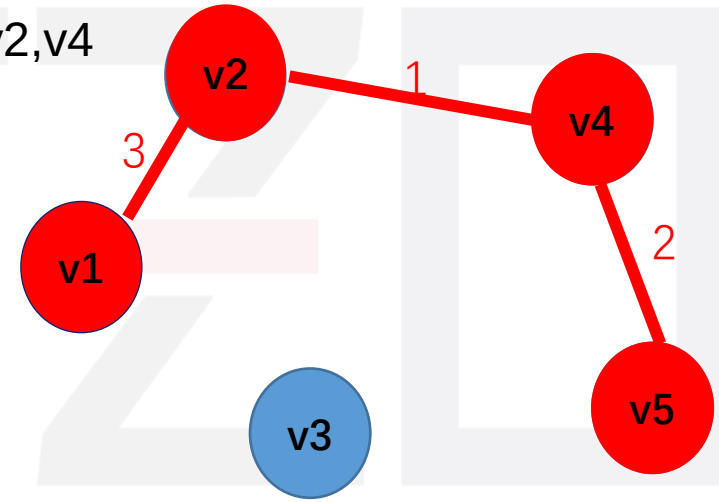
MST v1,v2

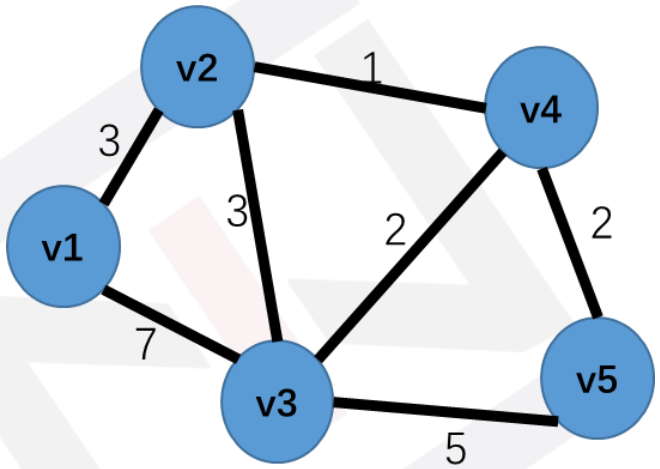


分析

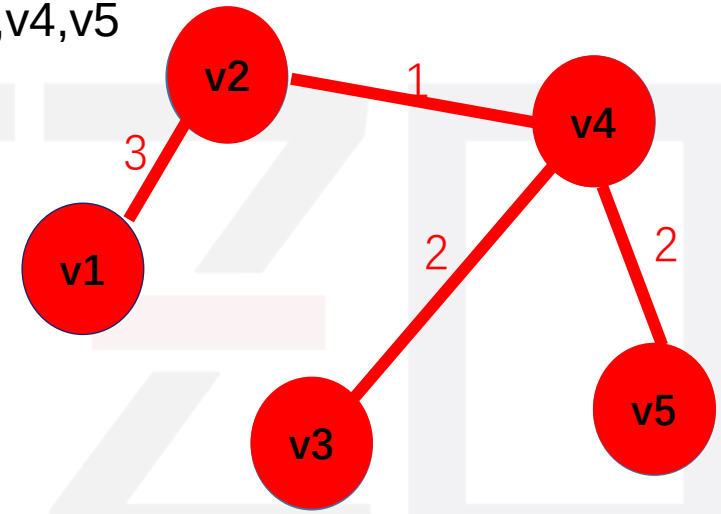


MST v1,v2,v4





MST v1,v2,v4,v5

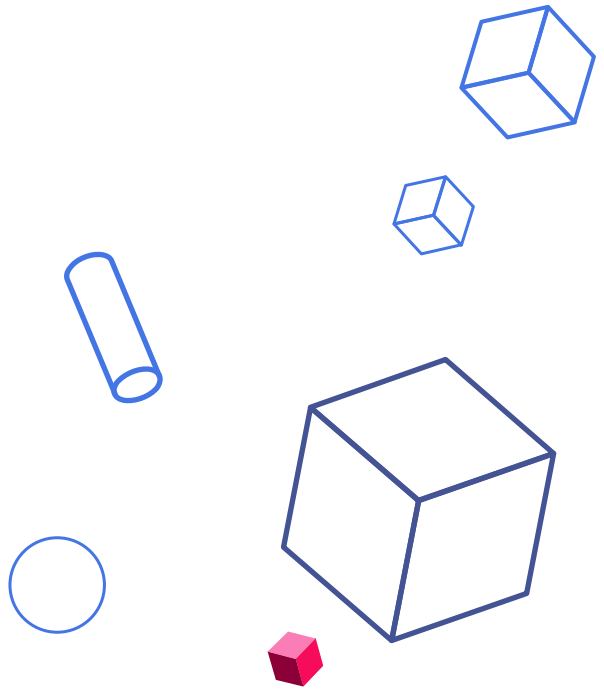


每次找到**未确定MST中**
与MST相连且**权值最小**的点加入 **贪心**

与Dij算法思想类似

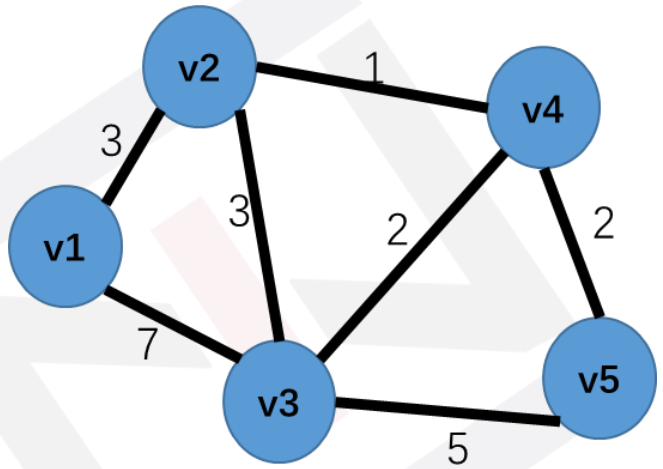
FZOI
西大附中信息学竞赛
High School Affiliated to Southwest University

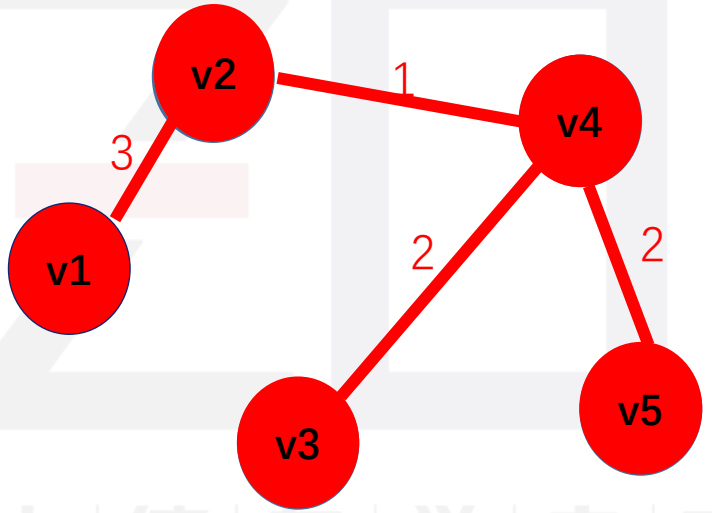
西南大学附属中学
High School Affiliated to Southwest University



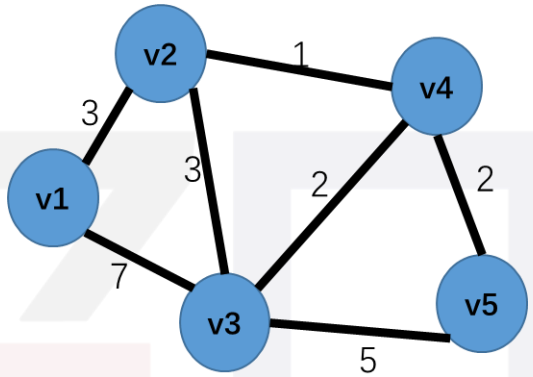
Prim算法

val[v]表示顶点v与MST相连的最小边权
f[v]表示是否在MST中





	1	2	3	4	5
val[v]	0	3	2	1	2
f[v]	1	0	0	0	0



val[v]表示顶点v与MST相连的最小边权
f[v]表示是否在MST中

① 初始化val[i]=∞,val[1]=0,mst=0;
② 选择一个不在MST中点k(f[k]==0)，并且val[k]的值最小;
③ 标记点k(f[k]=1),mst+=val[k];
④ 以k为中间点，更新修改val[i]的值;
⑤ 重复步骤2到步骤4，**n次**过后求得mst

小结：Prim是从点出发，生成MST

算法实现

1. **初始化**：f[i]=0,val[i]=∞; val[1]=0;
2. for (i = 1; i <= n ; i++)
   a)找到不在MST中的点k(f[k]=0)，并且val[k]的值最小；
   b)k确定为MST,mst+=val[k];f[k]=1;
   for:判断是否能**更新与k相连的顶点**j的val

```
if(val[j]>w[k][j]) {
  val[j]=w[k][j];
}
```

时间复杂度： o(n^2)

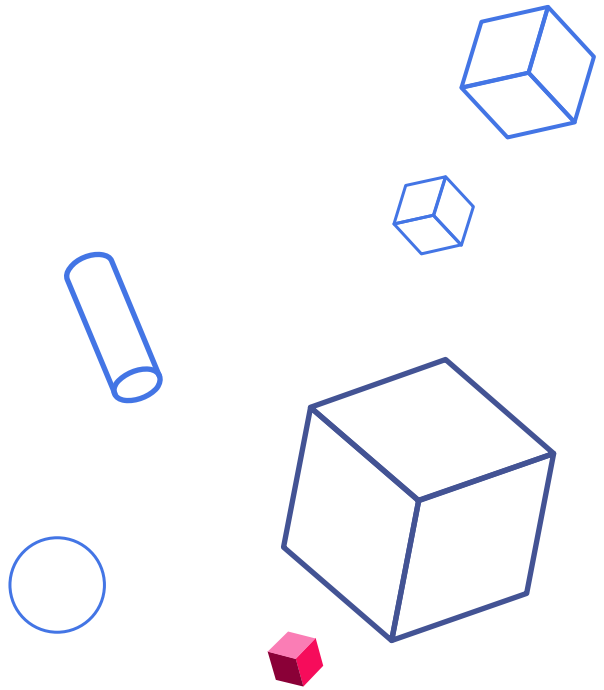
代码


```
struct edge
{
		int last,to,len;
}a[100001];
int first[10001],len=0;
//邻接表
bool f[10001];//记录是否在树上
int dis[10001];//记录结点到树的距离
void add......//存边
void prim()
{
		int i;
   for(i=1;i<=n;i++)dis[i]=999999;//初始化
		int cnt=0;//树内点的数量
		int sum=0;//树内边权总和
		dis[1]=0;
		f[1]=1;
		cnt=1;
		//先确定根结点，一般以1作为根结点
		while(cnt<n)//直到n个结点均在树上
		{
			int id,minn=1000001;
			//id记录找到的结点的编号，minn是它到树的距离
			for(i=1;i<=n;i++)
				if(f[i]==0&&dis[i]<minn)
				{
					id=i;
					minn=dis[i];
				}
			f[id]=1;
			cnt++;
			//将这个点加入树
			sum+=dis[id];
			//刷新边权总和
			for(i=first[id];i;i=a[i].last)
			//刷新结点到树的距离
				if(f[a[i].to]==0&&a[i].len<dis[a[i].to])
					dis[a[i].to]=a[i].len;

		}
}
```

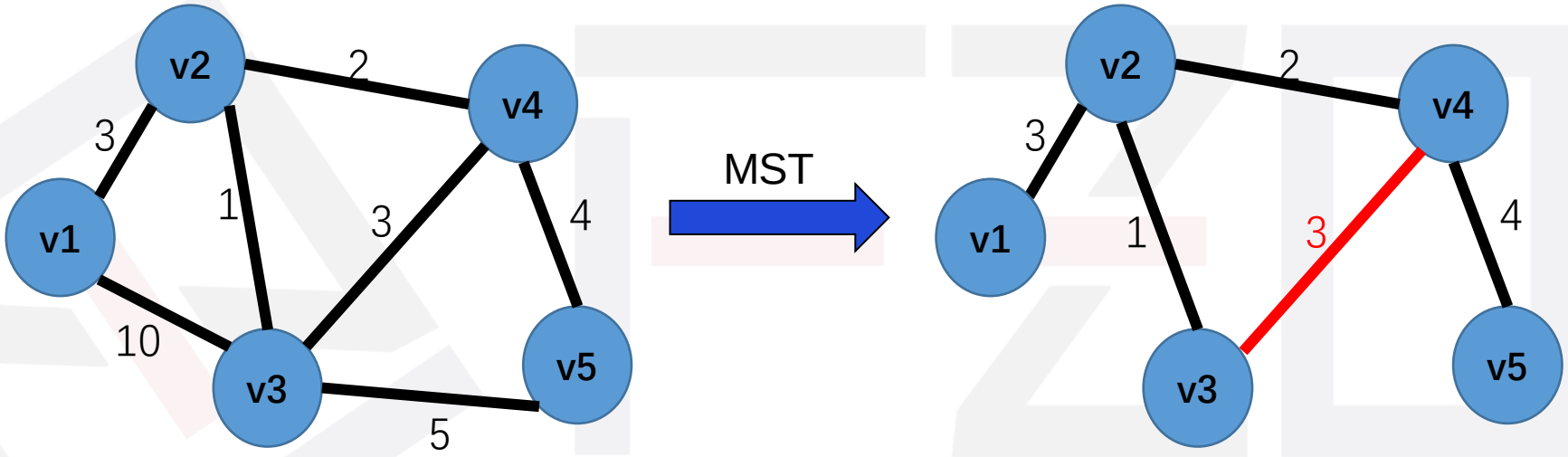
FZOI
西大附中信息学竞赛
High School Affiliated to Southwest University

西南大学附属中学
High School Affiliated to Southwest University



Kruskal算法

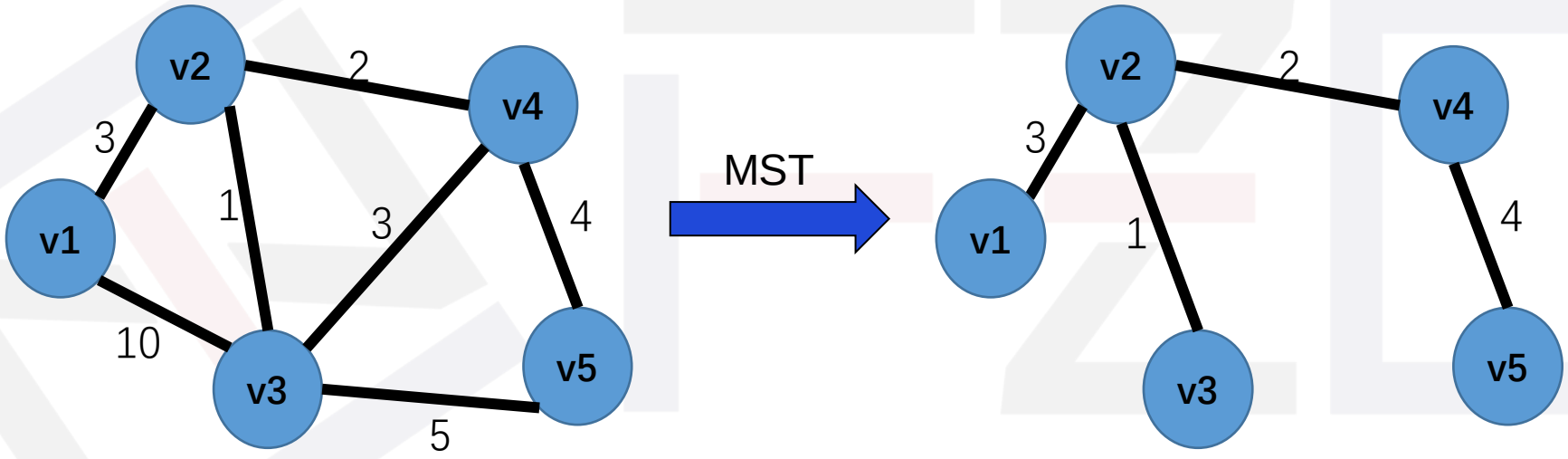
Kruskal算法



发现：MST的树边都是权值较小的n-1条

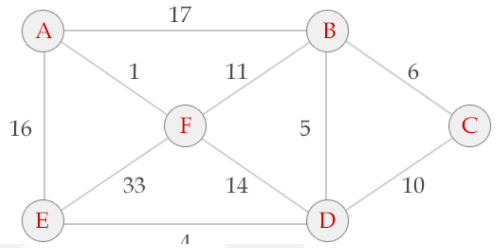
是最小的n-1条吗?

Kruskal算法

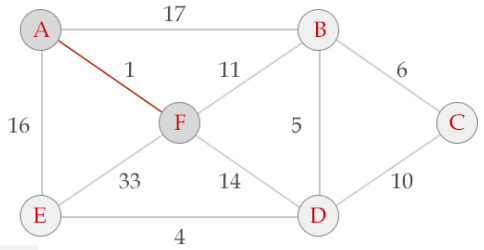


将边权从小到大枚举，如果边的两个顶点属于不同的集合，就把这条边加入MST

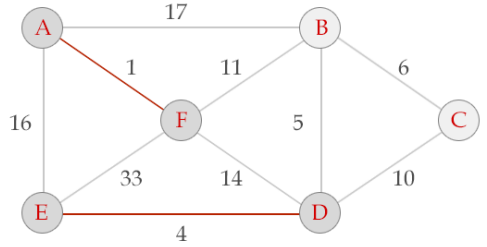
Kruskal算法



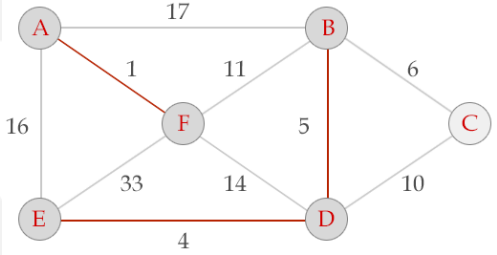
{A}{B}{C}{D}{E}{F}



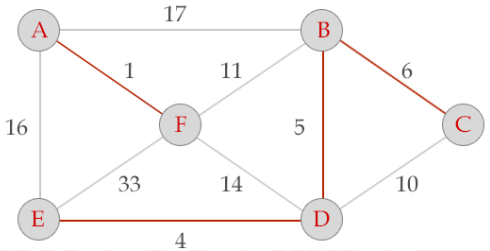
{A,F}{B}{C}{D}{E}



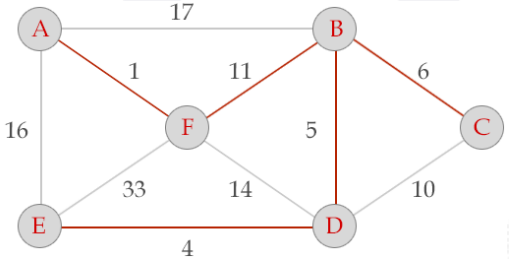
{A,F}{B}{C}{D,E}



{A,F}{C}{B,D,E}



{A,F}{B,C,D,E}



{A,F,B,C,D,E}

如何判断属于不同集合？ **并查集**

Kruskal算法

1. 对边权**排序**
2. for （i = 1; i <= M(边集）; i++)
   a) if(第i条边两个顶点u, v不属于同一集合)
      - **合并**u, v所在集合
      - mst+=val[i];
      - k++;
      - if(k==n-1) break;

时间复杂度：**Mlog(M)**

策略：**贪心**

小结：Kruskal是从边出发，生成MST

代码


```
struct edge
{
		int u,v,w;
}a[100001];
//边集数组
int boss[10001];//并查集，boss[i]表示i的祖先
int find(int x)
{
		if(boss[x]==x)return x;//找到祖先
		else
		{
				boss[x]=find(boss[x]);//路径压缩
				return boss[x];
		}
}
void Kurscal()
{
		int i;
		for(i=1;i<=n;i++)boss[i]=i;//初始化
//n个结点，每个结点的祖先默认为它自己，也就是每个结点自己一个集合
		stable_sort(a+1,a+1+m,cmp);//m条边，将边按照边权从小到大排序
		int cnt=0;//当前最小生成树里边的数量
		int len=0;//当前最小生成树边权总和
		for(i=1;i<=m;i++)
		{
				int x=find(a[i].u),y=find(a[i].v);
				//x表示a[i].u的祖先，y表示a[i].v的祖先
				if(x!=y)
				//说明两点不在同一集合内，即这两点不连通
				{
						boss[x]=y;//标记祖先
						cnt++;//边数增加
						len+=a[i].w;//边权和增加
				}
				if(cnt==n-1)break;
				//如果已经选了n-1条边，那最小生成树就建好了
		}
}
```

Thanks

For Your Watching