# 1233还是畅通工程

**Problem Description**

某省调查乡村交通状况，得到的统计表中列出了任意两村庄间的距离。省政府“畅通工程”的目标是使全省任何两个村庄间都可以实现公路交通（但不一定有直接的公路相连，只要能间接通过公路可达即可），并要求铺设的公路总长度为最小。请计算最小的公路总长度。

**Input**

测试输入包含若干测试用例。每个测试用例的第1行给出村庄数目N ( < 100 )；随后的N(N-1)/2行对应村庄间的距离，每行给出一对正整数，分别是两个村庄的编号，以及此两村庄间的距离。为简单起见，村庄从1到N编号。  
当N为0时，输入结束，该用例不被处理。

**Output**

对每个测试用例，在1行里输出最小的公路总长度。

**Sample Input**

3

1 2 1

1 3 2

2 3 4

4

1 2 1

1 3 4

1 4 1

2 3 3

2 4 2

3 4 5

0

**Sample Output**

3

5

代码清单：

//方法一（prim算法）；

//普利姆（Prime）算法（只与顶点相关）

//算法描述：

//普利姆算法求最小生成树时候，和边数无关，只和定点的数量相关，所以适合求稠密网的//最小生成树，时间复杂度为O（n\*n）。

//算法过程：

//1.将一个图的顶点分为两部分，一部分是最小生成树中的结点（A集合），另一部分是未处//理的结点（B集合）。

//2.首先选择一个结点，将这个结点加入A中，然后，对集合A中的顶点遍历，找出A中顶//点关联的边权值最小的那个（设为v），将此顶点从B中删除，加入集合A中。

//3.递归重复步骤2，直到B集合中的结点为空，结束此过程。

//4.A集合中的结点就是由Prime算法得到的最小生成树的结点，依照步骤2的结点连接这//些顶点，得到的就是这个图的最小生成树。

#include <iostream>

using namespace std;

#define MAX 999999

int map[100][100],dis[100];//以节点i为终点的最小边权值；

int mark[100];

int n;

void prim()

{

int i , j , k , min , sum = 0 ;

for( i = 1 ; i <= n ; i++ )

dis[i] = map[i][1] ;//因为节点1已经默认进入最小树，所以初始化应该是节点1到节点i的值；

mark[1] = 1 ;

for( i = 2 ; i <= n ; i++ )

{

min = MAX ;

k = i ;

for( j = 2 ; j <= n ; j++ ) //找出最小边权值；

if(!mark[j] && dis[j] < min )

{

min = dis[j] ;

k = j ;

}

sum += min ;//累加权值；

mark[k] = 1 ;//标记节点k加入生成树；

for( j = 2 ; j <= n ; j++ ) //更新权值；

if( map[j][k] < dis[j] )

dis[j] = map[j][k] ;

}

printf("%d\n",sum);

}

int main()

{

int i , x , y , d ;

while(scanf("%d",&n) != EOF && n)

{

for( i = 1 ; i <= n ; i++ )

{

map[i][i] = 0 ;

mark[i] = 0 ;

}

for( i = 0 ; i < n \* (n - 1) / 2 ; i++ )

{

scanf("%d %d %d",&x,&y,&d);

map[x][y] = map[y][x] = d ;

}

prim();

}

return 0;

}

//方法二（kruskal 算法)

//算法过程：

//1.将图各边按照权值进行排序

//2.将图遍历一次，找出权值最小的边，（条件：此次找出的边不能和已加入最小生成

//

//树集合的边构成环），若符合条件，则加入最小生成树的集合中。不符合条件则继

//

//续遍历图，寻找下一个最小权值的边。

//3.递归重复步骤1，直到找出n-1条边为止（设图有n个结点，则最小生成树的边数应

//

//为n-1条），算法结束。得到的就是此图的最小生成树。

#include<stdio.h>

#include<stdlib.h>

typedef struct path{

int s,e,len;

} path;

int f[101];

int min(int x,int y)

{

if(x<y)

return x;

return y;

}

int find(int x)

{

if(x!=f[x])

f[x]=find(f[x]);

return f[x];

}

void Union(int x,int y)

{

int xx,yy;

xx=find(x);

yy=find(y);

f[xx]=f[yy]=f[x]=f[y]=min(xx,yy);

}

int cmp(const void \*a,const void \*b)

{

path \*c,\*d;

c=(path \*)a;

d=(path \*)b;

return c->len-d->len;

}

int main()

{

path p[5000];

int n,i,j,sum;

while(scanf("%d",&n),n!=0)

{

for(i=0;i<n\*(n-1)/2;i++)

scanf("%d %d %d",&p[i].s,&p[i].e,&p[i].len);

qsort(p,n\*(n-1)/2,sizeof(p[0]),cmp);

j=0;

for(i=1;i<=n;i++)

f[i]=i;

sum=0;

for(i=0;i<n\*(n-1)/2;i++)

{

if(j==n-1)

break;

if(find(p[i].s)==find(p[i].e))

continue;

Union(p[i].s,p[i].e);

sum=sum+p[i].len;

j++;

}

printf("%d\n",sum);

}

return 0;

}

# 1863 畅通工程

**Problem Description**

省政府“畅通工程”的目标是使全省任何两个村庄间都可以实现公路交通（但不一定有直接的公路相连，只要能间接通过公路可达即可）。经过调查评估，得到的统计表中列出了有可能建设公路的若干条道路的成本。现请你编写程序，计算出全省畅通需要的最低成本。

**Input**

测试输入包含若干测试用例。每个测试用例的第1行给出评估的道路条数 N、村庄数目M ( < 100 )；随后的 N   
行对应村庄间道路的成本，每行给出一对正整数，分别是两个村庄的编号，以及此两村庄间道路的成本（也是正整数）。为简单起见，村庄从1到M编号。当N为0时，全部输入结束，相应的结果不要输出。

**Output**

对每个测试用例，在1行里输出全省畅通需要的最低成本。若统计数据不足以保证畅通，则输出“?”。

**Sample Input**

3 3

1 2 1

1 3 2

2 3 4

1 3

2 3 2

0 100

**Sample Output**

3

?

代码清单：

//Kruskal

#include <stdio.h>

#include <algorithm>

using namespace std;

struct EDGE

{

int u,v,cost;

}eg[100001];

int n,m,father[100001];

bool cmp(EDGE e1,EDGE e2)

{

return e1.cost<e2.cost;

}

// 并查集 初始化函数

void Init( int m )

{

int i;

for(i=1;i<=m;i++)

father[i]=i;

}

// 并查集 查找函数

int Find(int x)

{

while(father[x]!=x)

x=father[x];

return x;

}

// 并查集 合并函数

void Combine(int a,int b)

{

int temp\_a,temp\_b;

temp\_a=Find(a);

temp\_b=Find(b);

if(temp\_a!=temp\_b)

father[temp\_a]=temp\_b;

}

// 最小生成树 Kruskal 算法

int Kruskal( void )

{

EDGE e;

int i,res;

sort(eg,eg+n,cmp);

// 并查集 初始化

Init(m);

// 构建最小生成树

res=0;

for( i=0;i<n;++i )

{

e=eg[i];

if( Find(e.u)!=Find(e.v) )

{

Combine(e.u,e.v);

res+=e.cost;

}

}

return res;

}

int main()

{

int i,ans;

bool bl;

while( scanf("%d%d",&n,&m) && n )

{

for( i=0;i<n;++i )

scanf("%d%d%d",&eg[i].u,&eg[i].v,&eg[i].cost);

ans=Kruskal();

// 是否所有的点都在同一个集合

bl=true;

for(i=2;i<=m;++i)

if( Find(1)!=Find(i) )

{

bl=false;

break;

}

if( bl )

printf("%d\n",ans);

else

printf("?\n");

}

return 0;

}

//Prim

#include <stdio.h>

#include <string.h>

#define RANGE 101

#define MAX 0x7fffffff

int cost[RANGE][RANGE];

int mincost[RANGE];

bool used[RANGE];

// n个点，m条边

int n,m;

int Min(int a,int b)

{

return a<b?a:b;

}

void prim( void )

{

// sum 记录最小生成树权值

int i,v,u,sum;

// 从1到各个点距离，初始化used数组

for( i=1;i<=n;++i )

{

used[i]=false;

mincost[i]=cost[1][i];

}

sum=0;

while( true )

{

v=-1;

// 从没有连接到的点中，找最近的点

for( u=1;u<=n;++u )

if( !used[u] && (v==-1 || mincost[u]<mincost[v]) )

v=u;

if( v==-1 ) break;

if( mincost[v]==MAX ) break;

used[v]=true;

sum+=mincost[v];

// 更新到各个点的距离

for( u=1;u<=n;++u )

mincost[u]=Min( mincost[u],cost[v][u] );

}

// 判断能否构成最小生成树

for( i=1;i<=n;++i )

{

if( used[i]==false )

{

printf("?\n");

return;

}

}

printf("%d\n",sum);

}

int main()

{

int i,j;

int u,v,c;

while( scanf("%d%d",&m,&n) && m )

{

// init cost by MAX

for( i=1;i<=n;++i )

for( j=1;j<=i;++j )

{

if( i==j )

cost[i][j]=0;

else

cost[i][j]=cost[j][i]=MAX;

}

for( i=0;i<m;++i )

{

scanf("%d%d%d",&u,&v,&c);

cost[u][v]=cost[v][u]=c;

}

prim();

}

return 0;

}

# 1879继续畅通工程

**Problem Description**

省政府“畅通工程”的目标是使全省任何两个村庄间都可以实现公路交通（但不一定有直接的公路相连，只要能间接通过公路可达即可）。现得到城镇道路统计表，表中列出了任意两城镇间修建道路的费用，以及该道路是否已经修通的状态。现请你编写程序，计算出全省畅通需要的最低成本。

**Input**

测试输入包含若干测试用例。每个测试用例的第1行给出村庄数目N ( 1< N < 100 )；随后的 N(N-1)/2 行对应村庄间道路的成本及修建状态，每行给4个正整数，分别是两个村庄的编号（从1编号到N），此两村庄间道路的成本，以及修建状态：1表示已建，0表示未建。  
  
当N为0时输入结束。

**Output**

每个测试用例的输出占一行，输出全省畅通需要的最低成本。

**Sample Input**

3

1 2 1 0

1 3 2 0

2 3 4 0

3

1 2 1 0

1 3 2 0

2 3 4 1

3

1 2 1 0

1 3 2 1

2 3 4 1

0

**Sample Output**

3

1

0

**代码清单：**

**/\***

**hdu1879**

**典型的最小生成树，在做并查集时做的这道题，**

**所以使用并查集实现的克鲁斯卡尔算法**

**\*/**

**#include <iostream>**

**#include <stdio.h>**

**#include <algorithm>**

**using namespace std;**

**struct node {**

**int start ,end,expense,flag;**

**}data[5005];**

**int father[105];**

**void make\_set(int n)**

**{**

**for(int i=1;i<=n;i++)**

**father[i]=i;**

**}**

**int find\_set(int x)**

**{**

**if(x^father[x])**

**father[x]=find\_set(father[x]);**

**return father[x];**

**}**

**int union\_set(int x,int y)**

**{**

**x=find\_set(x);**

**y=find\_set(y);**

**if(x==y)**

**return 0;**

**father[x]=y;**

**return 1;**

**}**

**bool cmp(node a,node b)**

**{**

**return a.expense<b.expense;**

**}**

**int main()**

**{**

**int n;**

**while(scanf("%d",&n)!=EOF)**

**{**

**if(!n)**

**break;**

**make\_set(n);**

**int ans=0;**

**int m=(n-1)\*n/2;**

**for(int i=0;i<m;i++)**

**{**

**scanf("%d%d%d%d",&data[i].start,&data[i].end,&data[i].expense,&data[i].flag);**

**if(data[i].flag)//当道路修通时，规定一节点为另一节点的父亲**

**father[data[i].start]=data[i].end;**

**}**

**sort(data,data+m,cmp);//按道路的花费升序排列**

**//在不构成环的前提下，选择最短的边，有贪心的思想**

**for(int i=0;i<m;i++)**

**{**

**if(union\_set(data[i].start,data[i].end))**

**ans+=data[i].expense;**

**}**

**printf("%d\n",ans);**

**}**

**return 0;**

**}**

**//最小生成树的prim算法**

**include <iostream>**

**#include <stdio.h>**

**#include <string.h>**

**using namespace std;**

**int n;**

**int low[110];//存储将要联通的候选边**

**int visited[110];//是否已经并入联通集合**

**int map[110][110];//二维矩阵存储图**

**const int maxn=0x3fffffff;//无穷大**

**int prim()**

**{**

**int min;**

**int pos=0,ans=0;**

**memset(visited,0,sizeof(visited));**

**visited[pos]=1;//从第一个点开始联通**

**for(int i=0;i<n;i++)//循环初始化**

**{**

**if(!visited[i])**

**{**

**low[i]=map[pos][i];**

**}**

**}**

**for(int i=1;i<n;i++)**

**{**

**min=maxn;**

**for(int j=0;j<n;j++)**

**{**

**if(min>low[j]&&!visited[j])**

**{**

**min=low[j];//找到所有可连通边中的最短边**

**pos=j;//找到将要并入的下一个点**

**}**

**}**

**ans+=min;**

**visited[pos]=1;**

**for(int j=0;j<n;j++)//新点并入后更新最短边候选集合**

**{**

**if(!visited[j]&&map[pos][j]<low[j])**

**low[j]=map[pos][j];**

**}**

**}**

**return ans;**

**}**

**int main()**

**{**

**while(scanf("%d",&n)!=EOF)**

**{**

**if(!n)break;**

**int a,b,c,d;**

**int m=(n-1)\*n/2;**

**for(int i=0;i<m;i++)**

**{**

**scanf("%d%d%d%d",&a,&b,&c,&d);**

**if(d)//若两点已经连通，不需新建道路，花费为零**

**{**

**map[a-1][b-1]=map[b-1][a-1]=0;**

**}**

**else**

**map[a-1][b-1]=map[b-1][a-1]=c;**

**}**

**printf("%d\n",prim());**

**}**

**return 0;**

**}**