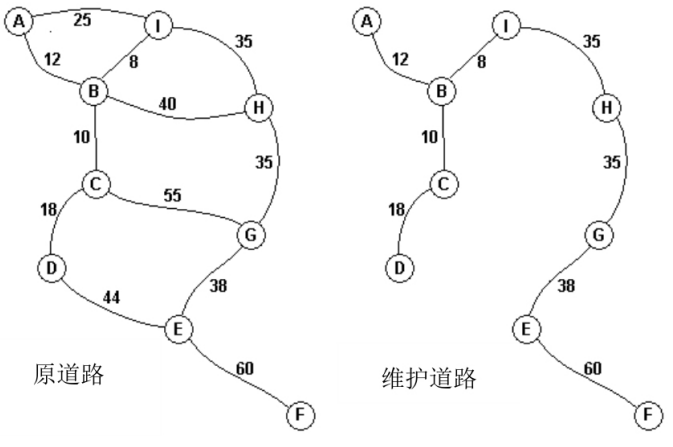
POJ1251

题目描述（POJ1251）：丛林道路网络的维护费用太高，理事会必须选择停止维护一些道路。如下图所示，在下面的地图中，村庄被标记为 A～I。左边的地图显示了现在所有道路及每月的维护费用，每月可以用最少的费用维护一些道路，保证所有村庄都是连通的。右边的地图显示了最便宜的道路维护方案，每月的维护总费用为 216 元。



输入：输入由 1～100 个数据集组成，最后一行只包含 0。每个数据集的第 1 行都为数字 n（1<n<27），表示村庄的数量，对村庄使用字母表的前 n 个大写字母标记。每个数据集都有 n-1 行描述，这些行的村庄标签按字母顺序排序。最后一个村庄没有道路。村庄的每条道路都以村庄标签开头，后面跟着一个从这个村庄到后面村庄的道路数 k。如果 k>0，则该行后面包含 k 条道路的数据。每条道路的数据都是道路另一端的村庄标签，后面是道路的每月维护成本。维护费用是小于 100 的正整数，道路数量不会超过 75 条，每个村庄通往其他村庄的道路都不超过 15 条。

输出：对于每个数据集，都单行输出每月维护连接所有村庄的道路的最低费用。

输入样例	输出样例
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 3 A 2 B 10 C 40 B 1 C 20 0	216 30

题解：这是非常简单的最小生成树问题，只需计算最小生成树的和值即可。使用 Prim 或 Kruskal 算法均可求解。

注意：在数据的输入格式方面，A 2 B 12 I 25 表示 A 关联两条边，包括 A-B 的边（边权为 12）及 A-I 的边（边权为 25）。

算法代码：

```
int prim(int s){
    for(int i=0;i<n;i++)
        dis[i]=m[s][i];
    memset(vis,false,sizeof(vis));
    vis[s]=1;
    int sum=0;
    int t;
    for(int i=1;i<n;i++){
        int min=0x3f3f3f3f;
        for(int j=0;j<n;j++){//找最小
            if(!vis[j]&&dis[j]<min){
                min=dis[j];
                t=j;
            }
        }
        sum+=min;
        vis[t]=1;
        for(int j=0;j<n;j++){//更新
            if(!vis[j]&&dis[j]>m[t][j])
                dis[j]=m[t][j];
        }
    }
    return sum;
}
```

POJ1287

题目描述（POJ1287）：已知该区域中的一组点，以及两点之间每条路线所需的电缆长度。请注意，在两个给定点之间可能存在许多路线。假设给定的可能路线（直接或间接）连接该区域中的每两个点，请设计网络，使每两个点之间都存在连接（直接或间接），并且使用的电缆总长度最小。

输入：输入由多个数据集组成，每个数据集都描述一个网络。数据集的第 1 行包含两个整数：第 1 个整数表示点数 P（P≤50），节点标号为 1～P；第 2 个整数表示点之间的路线数 R。以下 R 行为点之间的路线，每条路线都包括 3 个整数：前两个整数为点标号，第 3 个整数为路线长度 L（L≤100）。数据集之间以空行分隔，输入仅有一个数字 P（P=0）的数据集，表示输入结束。

输出：对于每个数据集，都单行输出所设计网络的电缆的最小总长度。

输入样例

```
1 0

2 3
1 2 37
2 1 17
1 2 68

3 7
1 2 19
2 3 11
3 1 7
1 3 5
2 3 89
3 1 91
1 2 32

5 7
1 2 5
2 3 7
2 4 8
4 5 11
3 5 10
1 5 6
4 2 12

0
```

输出样例

```
0
17
16
26
```

题解：本题是简单的最小生成树问题，可以采用 Prim 或 Kruskal 算法求解。在此使用并查集优化的 Kruskal 算法。

1. 算法设计

（1）初始化。将所有边都按权值从小到大排序，将每个节点的集合号都初始化为自身编号。

（2）按排序后的顺序选择权值最小的边(u,v)。

（3）如果节点 u 和 v 属于两个不同的连通分支，则采用并查集对两个连通分支进行合并，累加边(v,v)的权值。

（4）如果选取的边数小于 n−1，则转向步骤 2；否则算法结束，返回和值。

2. 算法实现

```
int find(int x){//采用并查集找祖宗
    return fa[x]==x?x:fa[x]=find(fa[x]);
}

bool merge(int a,int b){//集合合并
    int x=find(a);
    int y=find(b);
    if(x==y) return 0;
    fa[y]=x;
    return 1;
}
int kruskal(){
    int sum=0;
    sort(edge,edge+m,cmp);
    for(int i=0;i<m;i++){
        if(merge(edge[i].u,edge[i].v)){
            sum+=edge[i].cost;
            if(--n==1)
                return sum;
        }
    }
    return 0;
}
```

POJ2031

题目描述（POJ2031）：空间站由许多单元组成，所有单元都是球形的。在该站成功进入其轨道后不久，每个单元都固定在其预定的位置。两个单元可能彼此接触，甚至重叠。在极端情况下，一个单元可能完全包围另一个单元。所有单元都必须连接，因为机组成员应该能够从任何单元走到任何其他单元。如果存在下面三种情况，则可以从单元 A 走到另一个单元 B：

（1）A 和 B 相互接触或重叠；

（2）A 和 B 通过“走廊”连接；

（3）有一个单元 C，从 A 到 C，且从 B 到 C 是可能的（传递）。

需要设计一种配置，看看用走廊连接哪些单元可以使整个空间站连通。建造走廊的成本与其长度成正比。因此，应该选择走廊总长度最短的计划。

输入：输入由多个数据集组成。每个数据集的第 1 行都包含一个整数 n（0<n≤100），表示单元的数量。以下 n 行是对单元的描述，其中每一行都包含 4 个值，表示球体的中心坐标 x、y 和 z，以及球体的半径 r，每个值都为小数（小数点后 3 位）。x、y、z 和 r 均为正数且小于 100.0。输入的结尾由包含 0 的行表示。

输出：对于每个数据集，都单行输出建造走廊的最短总长度（小数点后 3 位）。

注意：如果不需要建造走廊，则走廊的最短总长度为 0.000。

输入样例	输出样例
3 10.000 10.000 50.000 10.000 40.000 10.000 50.000 10.000 40.000 40.000 50.000 10.000 2 30.000 30.000 30.000 20.000 40.000 40.000 40.000 20.000 5 5.729 15.143 3.996 25.837 6.013 14.372 4.818 10.671 80.115 63.292 84.477 15.120 64.095 80.924 70.029 14.881 39.472 85.116 71.369 5.553 0	20.000 0.000 73.834

题解：本题属于最小生成树问题，可以采用 Prim 或 Kruskal 算法求解。

1. 算法设计

（1）计算任意两个单元之间的距离，如果两个单元有接触或重叠，则距离为 0.000。

（2）采用 Prim 算法求解最小生成树。

（3）输出最小生成树的权值之和。

2. 算法实现

```
struct cell{
    double x,y,z,r;//球形单元的圆心、半径
}c[maxn];

double clu(cell c1,cell c2){//计算两个球形单元的距离
    double x=(c1.x-c2.x)*(c1.x-c2.x);
    double y=(c1.y-c2.y)*(c1.y-c2.y);
    double z=(c1.z-c2.z)*(c1.z-c2.z);
    double d=sqrt(x+y+z);
    if(d-c1.r-c2.r<=0)
        return 0.000;
    else
        return d-c1.r-c2.r;
}
```

```
double prim(int s){//返回值为double类型
    for(int i=0;i<n;i++)
        low[i]=m[s][i];
    memset(vis,false,sizeof(vis));
    vis[s]=1;
    double sum=0.000;
    int t;
    for(int i=1;i<n;i++){//执行n-1次
        double min=inf;
        for(int j=0;j<n;j++){//找最小
            if(!vis[j]&&low[j]<min){
                min=low[j];
                t=j;
            }
        }
        sum+=min;
        vis[t]=1;
        for(int j=0;j<n;j++){//更新
            if(!vis[j]&&low[j]>m[t][j])
                low[j]=m[t][j];
        }
    }
    return sum;
}
```

POJ2421

题目描述（POJ2421）：有 N 个村庄，编号为 1～N，需要建造一些道路，使每两个村庄之间都可以相互连接。两个村庄 A 和 B 是相连的，当且仅当 A 和 B 之间有一条道路，或者存在一个村庄 C，A 和 C 相连且 C 和 B 相连。已知一些村庄之间已经有一些道路，你的工作是修建一些道路，使所有村庄都连通起来，所有道路的长度之和最小。

输入：第 1 行是整数 N（3≤N≤100），表示村庄的数量；然后是 N 行，其中第 i 行包含 N 个整数，第 j 个整数表示村庄 i 和村庄 j 之间的距离（距离为[1,1000]内的整数）；接着是整数 Q（0≤Q≤N×(N+1)/ 2），表示已建成道路的数量；最后是 Q 行，每行都包含两个整数 a 和 b（1≤a<b≤N），表示村庄 a 和村庄 b 之间的道路已经建成。

输出：单行输出需要构建的所有道路的最小长度。

输入样例	输出样例
3 0 990 692 990 0 179 692 179 0 1 1 2	179

题解：本题属于最小生成树问题，不同的是本题有一些道路已经建成，将这些道路的边权设置为 0，然后采用 Prim 或 Kruskal 算法求解最小生成树即可。

算法代码：

```
int prim(int s){
    memset(vis,false,sizeof(vis));
    memset(low,0,sizeof(low));
    for(int i=1;i<=n;i++)
        low[i]=m[s][i];
    vis[s]=1;
    int sum=0;
    int t;
    for(int i=1;i<n;i++){//执行 n-1 次
        int min=inf;
        for(int j=1;j<=n;j++){//找最小
            if(!vis[j]&&low[j]<min){
                min=low[j];
                t=j;
            }
        }
        sum+=min;
        vis[t]=1;
        for(int j=1;j<=n;j++){//更新
            if(!vis[j]&&low[j]>m[t][j])
                low[j]=m[t][j];
        }
    }
    return sum;
}

int main(){
    int q,a,b;
    while(cin>>n){
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)
                cin>>m[i][j];
        cin>>q;
        while(q--){
            cin>>a>>b;
            m[a][b]=m[b][a]=0;
        }
        cout<<prim(1)<<endl;
    }
    return 0;
}
```