最小生成树（ MST, minimum spannirng tree）

1.概念

生成树（ *spanning tree*）：如果一个无向连通图 G（V，E） 的一个连通子图 G′（V，E′）是一棵包含 *G* 的所有顶点的树，则该子图 G′称为 *G* 的生成树。

生成树 G′是连通图 G 的极小连通子图(若在 G′中再增加任意一条边，则将出现一个回路；若从 G′中去掉任意一条边，G′变成非连通图)。

根据生成树的定义，包含 *n* 个顶点的连通图，其生成树一定包含 *n*-1 条边（∈E）。

一个连通图的最小生成树不是唯一的（起码可以选不同的结点为根）。

最小生成树（ *MST, minimum spannirng tree*）， 也称**最小代价生成树**或**最小权重生成树，**就是指边权和最小的生成树。

实际应用：n 个村庄，要想实现“村村通”，至少要在这 n 个村子之间修 n-1 条道路，选择在哪些村子之间修这 n-1 条路，才能使总的代价最小（总路程最小），这就是最小生成树的典型问题。

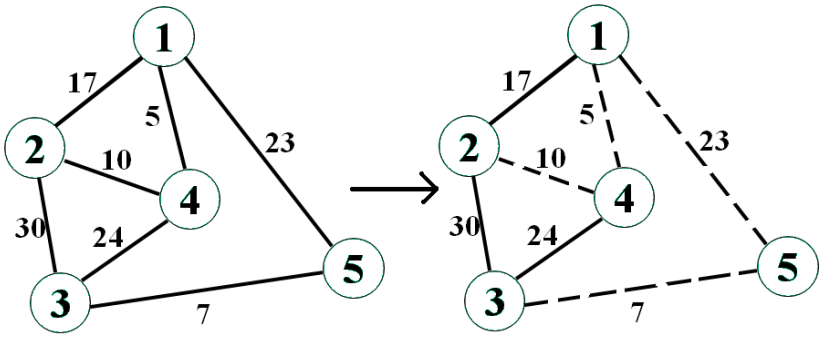

图 1

构造最小生成树的常用算法两种：克鲁斯卡尔（ *Kruskal*）算法和普里姆（ *Prim*）算法，这两种算法都是采用了贪心思想。

2.克鲁斯卡尔（ *Kruskal*）算法：

算法思想：

先构造一个只含 n 个顶点，而边集为空的子图，若将该子图中各个顶点看成是各棵树上的根结点，则它是一个含有 n 棵树的一个森林。从图的边集 E 中选取一条权值最小的边，若该条边的两个顶点分属不同的树，则将其加入子图，也就是说，将这两个顶点分别所在的两棵树合成一棵树；反之，若该条边的两个顶 点已落在同一棵树上，则放弃该边，而应该取下一条权值第二小的边尝试。依次类推，直至森林变为一棵树，也即子图中含有 n-1 条边为止。

实现上述算法需要用到并查集的操作。

算法描述：

step 1,将生成树赋为空。将图中的边按权值从小到大排序。将排在第一位的边标记为当前边。

step 2,若生成树中已有 n-1 条边，则算法停止；否则

step 3,如果将当前边加入生成树中不出现回路，则将当前边加入生成树，将排序序列里当前边的下一条边标记为当前边，跳回 step 2；否则

step 4,将排序序列里当前边的下一条边标记为当前边，跳回 step 3。

现在模拟克鲁斯卡尔算法求图 1 的最小生成树。

首先对边按边权递增的顺序进行排序:

(1,4)-(3,5)-(2,4)-(1,2)-(1,5)-(3,4)-(2,3)

将(1,4)加入生成树，未出现回路;

将(3,5)加入生成树，未出现回路;

将(2,4)加入生成树，未出现回路;

尝试将(1,2)加入生成树，出现回路，跳过(1,2);

将(1,5)加入生成树，未出现回路。

此时生成树中已有 n-1 条边，算法结束。

算法特点：图用边表存储，对边进行操作。

```
// 模板 kruskal
#include<cstdio>
#include<iostream>
#include<cstring>
#include<algorithm>
const int maxe=100100;
const int maxn=10010;
using namespace std;
struct Edge{
    int u,v,w;
};
Edge e[maxe];
int f[maxn];
int n,m;
int cmp(Edge a,Edge b){return a.w<b.w;}
int Find(int x){return f[x]==0?x:f[x]=Find(f[x]);}
int kruskal(){
    int cnt=0;
    long long ans=0;
    for(int i=0;i<m;i++){
        int x=Find(e[i].u);
```

```
            int y=Find(e[i].v);
            if(x!=y){
                f[x]=y;
                ans+=e[i].w;
                if(++cnt>=n-1) return ans;
            }
        }
    }
    int main(){
        memset(e,0,sizeof(0));
        memset(f,0,sizeof(f));
        scanf("%d%d",&n,&m);
        for(int i=0;i<m;i++)
            scanf("%d%d%d",&e[i].u,&e[i].v,&e[i].w);
        sort(e,e+m,cmp);
        cout<<kruskal()<<endl;
        return 0;
    }
```

典型例题

例 1：局域网(net)（一本通 1391）

【题目描述】

某个局域网内有 n(n≤100)台计算机，由于搭建局域网时工作人员的疏忽，现在局域网内的连接形成了回路，我们知道如果局域网形成回路那么数据将不停的在回路内传输，造成网络卡的现象。因为连接计算机的网线本身不同，所以有一些连线不是很畅通，我们用 f(i,j)表示 i,j 之间连接的畅通程度(f(i,j)≤1000)，f(i,j)值越小表示 i,j 之间连接越通畅，f(i,j)为 0 表示 i,j 之间无网线连接。现在我们需要解决回路问题，我们将除去一些连线，使得网络中没有回路，并且被除去网线的Σf(i,j)最大，请求出这个最大值。

【输入】

第一行两个正整数 n k。

接下来的 k 行每行三个正整数 i j m 表示 i,j 两台计算机之间有网线联通，通畅程度为 m。

【输出】

一个正整数，Σf(i,j)的最大值。

【输入样例】

```
5 5
1 2 8
1 3 1
1 5 3
2 4 5
3 4 2
```

【输出样例】

```
8
```

提示：总的边长和-最小生成树

例 2：城市公交网建设问题（一本通 1348）

【题目描述】

有一张城市地图，图中的顶点为城市，无向边代表两个城市间的连通关系，边上的权为在这两个城市之间修建高速公路的造价，研究后发现，这个地图有一个特点，即任一对城市都是连通的。现在的问题是，要修建若干高速公路把所有城市联系起来，问如何设计可使得工程的总造价最少？

【输入】

n（城市数，1<≤n≤100）

e（边数）

以下 e 行，每行 3 个数 i,j,wij，表示在城市 i,j 之间修建高速公路的造价。

【输出】

n-1 行，每行为两个城市的序号，表明这两个城市间建一条高速公路。

【输入样例】

```
5 8
1 2 2
2 5 9
5 4 7
4 1 10
1 3 12
4 3 6
5 3 3
2 3 8
```

【输出样例】

```
1  2
2  3
3  4
3  5
```

例 3：繁忙的都市(city)（一本通 1392）

【题目描述】

城市 C 是一个非常繁忙的大都市，城市中的道路十分的拥挤，于是市长决定对其中的道路进行改造。城市 C 的道路是这样分布的：城市中有 n 个交叉路口，有些交叉路口之间有道路相连，两个交叉路口之间最多有一条道路相连接。这些道路是双向的，且把所有的交叉路口直接或间接的连接起来了。每条道路都有一个分值，分值越小表示这个道路越繁忙，越需要进行改造。但是市政府的资金有限，市长希望进行改造的道路越少越好，于是他提出下面的要求：

1．改造的那些道路能够把所有的交叉路口直接或间接的连通起来。

2．在满足要求 1 的情况下，改造的道路尽量少。

3．在满足要求 1、2 的情况下，改造的那些道路中分值最大值尽量小。

作为市规划局的你，应当作出最佳的决策，选择那些道路应当被修建。

【输入】

第一行有两个整数 n,m 表示城市有 n 个交叉路口，m 条道路。接下来 m 行是对每条道路的描述，u，v，c 表示交叉路口 u 和 v 之间有道路相连，分值为 c。(1≤n≤300，1≤c≤10000)。

【输出】

两个整数 s，max，表示你选出了几条道路，分值最大的那条道路的分值是多少。

【输入样例】

```
4 5
1 2 3
1 4 5
2 4 7
2 3 6
3 4 8
```

【输出样例】

```
3 6
```

分析：n 个顶点的最小生成树边数为 n-1；最小生成树保证了最后添加的一条边最小。

例 4：联络员(liaison)（1393）

【题目描述】

Tyvj 已经一岁了，网站也由最初的几个用户增加到了上万个用户，随着 Tyvj 网站的逐步壮大，管理员的数目也越来越多，现在你身为 Tyvj 管理层的联络员，希望你找到一些通信渠道，使得管理员两两都可以联络（直接或者是间接都可以）。Tyvj 是一个公益性的网站，没有过多的利润，所以你要尽可能的使费用少才可以。

目前你已经知道，Tyvj 的通信渠道分为两大类，一类是必选通信渠道，无论价格多少，你都需要把所有的都选择上；还有一类是选择性的通信渠道，你可以从中挑选一些作为最终管理员联络的通信渠道。数据保证给出的通行渠道可以让所有的管理员联通。

【输入】

第一行 n，m 表示 Tyvj 一共有 n 个管理员，有 m 个通信渠道；

第二行到 m+1 行，每行四个非负整数，p,u,v,w 当 p=1 时，表示这个通信渠道为必选通信渠道；当 p=2 时，表示这个通信渠道为选择性通信渠道；u,v,w 表示本条信息描述的是 u，v 管理员之间的通信渠道，u 可以收到 v 的信息，v 也可以收到 u 的信息，w 表示费用。

【输出】

最小的通信费用。

【输入样例】

```
5 6
1 1 2 1
1 2 3 1
1 3 4 1
1 4 1 1
2 2 5 10
2 2 5 5
```

【输出样例】

9

【样例解释】

1-2-3-4-1 存在四个必选渠道，形成一个环，互相可以到达。需要让所有管理员联通，需要联通 2 号和 5 号管理员，选择费用为 5 的渠道，所以总的费用为 9。

【注意】

U,v 之间可能存在多条通信渠道，你的程序应该累加所有 u,v 之间的必选通行渠道

【数据范围】

对于 30%的数据，n≤10,m≤100;

对于 50%的数据， n≤200,m≤1000

对于 100%的数据，n≤2000,m≤10000

分析：先把必选的放在生成树中，然后再在可选的中选择。

3.普里姆（ *Prim*）算法

算法思想：

首先把这个结点包括进生成树里，然后在那些其一个端点已在生成树里、另一端点还未在生成树里的所有边中找出权最小的一条边，并把这条边、包括不在生成树的另一端点包括进生成树，…。依次类推，直至将所有结点都包括进生成树为止。

算法描述：

step 1,将生成树赋为空。任选一点放进生成树里。

step 2,在那些其一个端点已在生成树里、另一端点还未在生成树里的所有边中找出权最小的一条边，并把这条边以及不在生成树的另一端点包括进生成树。

step 3,重复 step 2，直至将所有结点都包括进生成树为止。

下面模拟普里姆算法求图 1 的最小生成树。

假设一开始将结点 1 放入生成树中;

将边（1,4）及结点 4 放入生成树里;

将边（4,2）及结点 2 放入生成树里;

将边（1,5）及结点 5 放入生成树里;

将边（5,3）及结点 3 放入生成树里。

所有结点均在生成树中，构造完毕。

算法特点：图用链接矩阵存储，对顶点进行操作。

```
// prim 算法模板
#include<cstdio>
#include<iostream>
#include<cstring>
using namespace std;
#define INF 0x7f7f7f7f
const int maxn=1010;
int a[maxn][maxn];
int vis[maxn];
int d[maxn];
```

```
int n;
int prim(){
    memset(d,0x7f,sizeof(d));
    memset(vis,0,sizeof(vis));
    for(int i=1;i<=n;i++)d[i]=a[1][i];
    d[1]=0;
    vis[1]=1;
    int ans=0;
    for(int i=1;i<n;i++){
        int k=0;
        for(int j=1;j<=n;j++)
            if(!vis[j]&&d[j]<d[k]) k=j;
        vis[k]=1;//if k=0
        ans+=d[k];
        for(int j=1;j<=n;j++)
            if(!vis[j]) d[j]=min(d[j],a[k][j]);
    }
    return ans;
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)cin>>a[i][j];
    cout<<prim()<<endl;
    return 0;
}
```

典型例题

例 1：最短网络 Agri-Net（一本通 1350；luogu 1546）

题目背景

农民约翰被选为他们镇的镇长!他其中一个竞选承诺就是在镇上建立起互联网，并连接到所有的农场。当然，他需要你的帮助。

题目描述

约翰已经给他的农场安排了一条高速的网络线路，他想把这条线路共享给其他农场。为了用最小的消费，他想铺设最短的光纤去连接所有的农场。

你将得到一份各农场之间连接费用的列表，你必须找出能连接所有农场并所用光纤最短的方案。每两个农场间的距离不会超过 100000

输入输出格式

输入格式：

第一行： 农场的个数，N（3<=N<=100）。

第二行..结尾：后来的行包含了一个 N*N 的矩阵,表示每个农场之间的距离。理论上，他们是 N 行，每行由 N 个用空格分隔的数组成，实际上，他们限制在 80 个字符，因此，某些行会紧接着另一些行。当

然，对角线将会是 0，因为不会有线路从第 i 个农场到它本身。

输出格式：

只有一个输出，其中包含连接到每个农场的光纤的最小长度。

输入输出样例

输入样例#1：

```
4
0 4 9 21
4 0 8 17
9 8 0 16
21 17 16 0
```

输出样例#1：

```
28
```

说明

4.最小生成树的性质：

(1)切割性质：（各边边权均不相同）一条边是连接图中某非全集非空集的点集合 S 和其补集中所有的边的最小边，那么这条边就在最小生成树中。

(2)回路性质：（各边边权均不相同）图若有回路，那么回路中的最长边一定不在最小生成树中。

(3)最小瓶颈生成树：使最大边权值尽量小的生成树

最小生成树就是这么一棵树，因为 kruscal 算法的过程

(4)最小瓶颈路：找 u 到 v 的一条路径满足最大边权值尽量小

先求最小生成树，然后 u 到 v 的路径在树上是唯一的，答案就是这条路径

思考：

非连通图的最小生成森林

最大生成树（先加最大边。作用：u 到 v 的路中，最小边最大）