1,

给出n个城市之间的距离 然后在给出q 接下来的q行 每行给出两个城市的序号 表示这两个城市已经建过路了 求还没有建路的城市之间的最小的即可

根据题意是求最小生成树的 直接套用prim算法即可

但是要注意某些城市已经有路了

2,

先用A到K来表示各个田地里面水管的情况 然后给你一个M\*N的矩阵 求所有的田地都能被灌溉所要建造的最少的井

首先我们用1和0来表示一块田的四个方向是否有管道  有的话就是1 没有就是0  比如：A 表示的就是1100 B表示的就是0110 以此这样表示

在同一行我们只需判断当前这块田第三个字符和下一块田的第一个字符是否相等 并且都等于一

在同一列我们只需判断当前这块田第二个字符和下一块田的第四个字符是否相等并且都等于一

3, **题意：找到从起点到终点的一条路，而且这条路经过的边的最大权值与最小权值之差最小，输出这个差**

**并查集+贪心   
枚举最小边，再从小到大一直加大于等于这个最小边的其他边，直到起点跟终点属于同一个集合为止，然后用最大边-最小边的差，看能不能更新mins【结果】**

4,共有n的城市 现在有m个城市之间有路 现在求要让所有的城市都相通 最少还要建多少条路 直接并查集

5,从n个点中选一个根节点，使它能够到达除本身以外的任一点并且花费最少 由于是有向图 所以是最小树形图（有向图的最小生成树） 构建一个虚拟节点 以这个点为根和那n个点建立一个有向图的最小生成树 既是所求

6.现在有n个城市 想让任意两个城市都能相通并且已经有一些城市是相通的 问最少还需要建造多少条路 把没有连通的城市都加到一个环中 判断还需要建造多少路 并查集模板

7.共有n条路，m个村庄，接下来n行，每行给出两个村庄的坐标和修建路的成本，问所有村庄畅通所需要的最少的成本，若不能保存畅通则输出？

Prim算法

1,AC代码：

#include <iostream>

#include <cstring>

#include <cstdio>

#define INF 0x3f3f3f3f

using namespace std;

int map[1001][1001], t, vis[1001],dis[1001];

void prim()

{

memset(dis,1,sizeof(dis));

int sum = 0;

int k;

for(int i = 0; i < t; i++)

{

vis[i] = map[0][i];

}

int min,min1 = INF;

dis[0] = 0;

for(int i = 1; i < t; i++)

{

min = INF;

for(int j = 0; j < t; j++)

{

if(dis[j]!=0&&vis[j]<min)

{

min = vis[j];

k = j;

}

}

dis[k] = 0;

sum+=min;

for(int j = 0; j < t; j++)

if(dis[j]!=0&&map[k][j] < vis[j])

{

vis[j] = map[k][j];

}

}

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

}

int main()

{

while(~scanf("%d",&t))

{

for(int i = 0; i < t; i++)

for(int j = 0; j < t; j++)

scanf("%d",&map[i][j]);

int n, x, y;

scanf("%d",&n);

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

{

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

map[x-1][y-1] = map[y-1][x-1] = 0;

}

prim();

}

return 0;

}

2,AC代码：

#include <iostream>

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

string a[12];

char map[100][100];

int vis[100][100];

int pre[100];

int suan(char x,char y)

{

int num;

int i = x - 'A' + 1;

int j = y - 'A' + 1;

if(a[i][2]=='1'&&a[j][0]=='1')

num = 1;

else

num = 0;

return num;

}

int suan1(char x,char y)

{

int num;

int i = x - 'A' + 1;

int j = y - 'A' + 1;

if(a[i][3]=='1'&&a[j][1]=='1')

num = 1;

else

num = 0;

return num;

}

int find(int x)

{

int r = x;

while(pre[r]!=r)

{

r = pre[r];

}

return r;

}

int main()

{

a[1] = "1100",a[2] = "0110",a[3] = "1001",a[4] = "0011",a[5] = "0101",a[6] = "1010";

a[7] = "1110",a[8] = "1101",a[9] = "1011",a[10] = "0111",a[11] = "1111";

int n,m;

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

{

if(n==-1&&m==-1)

break;

memset(vis,0,sizeof(vis));

int sum = n\*m;

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

{

for(int j = 1; j <= m; j++)

{

scanf(" %c",&map[i][j]);

}

}

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

pre[i] = i;

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

{

for(int j = 1; j <= m; j++)

{

//printf(" %c",map[i][j]);

if(j+1<=m)

{

int ans = suan(map[i][j],map[i][j+1]);

int fx = find((i-1)\*m+j); ///把二维数组用一位数组存 求当前点在一位数组里面是第几个

int fy = find((i-1)\*m+j+1);

if(fx!=fy&&ans == 1) ///注意： 如果下一块田已经访问过 不减一

{

pre[fx] = fy;

sum--;

}

}

if(i+1<=n)

{

int ans1 = suan1(map[i][j],map[i+1][j]);

int fx = find((i-1)\*m+j); ///把二维数组用一位数组存 求当前点在一位数组里面是第几个

int fy = find((i+1-1)\*m+j);

if(fx!=fy&&ans1==1)

{

pre[fx] = fy;

sum--;

}

}

}

}

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

}

return 0;

}

3,AC代码：

#include <iostream>

#include <cstdio>

#include <algorithm>

using namespace std;

int pre[1001];

struct Node

{

int start,end,speed;

}node[1001];

int cmp(Node a,Node b)

{

return a.speed > b.speed;

}

int find(int x)

{

int r = x;

while(pre[r]!=r)

{

r = pre[r];

}

int i = x,j;

while(i!=j)

{

j = pre[i];

pre[i] = r;

i = j;

}

return r;

}

int main()

{

int n,m;

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

{

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

scanf("%d%d%d",&node[i].start,&node[i].end,&node[i].speed);

int k;

sort(node,node + m, cmp);

scanf("%d",&k);

int x,y;

for(int t = 1; t <= k; t++)

{

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

int ans = 999999999;

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

{

for(int j = 1; j <= n; j++)

pre[j] = j;

for(int q = i; q < m; q++)

{

int fx = find(node[q].start);

int fy = find(node[q].end);

if(fx!=fy)

{

pre[fy] = fx;

}

if(find(x)==find(y))

{

if(ans > node[i].speed - node[q].speed)

ans = node[i].speed - node[q].speed;

break;

}

}

if(k==m)

break;

}

if(ans == 999999999)

printf("-1\n");

else

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

}

}

return 0;

}

4,AC代码：

#include <cstdio>

#include <iostream>

using namespace std;

int pre[1001];

int find(int x)

{

int i = x;

while(pre[i]!=i)

{

i = pre[i];

}

int a = x,b;

while(a!=i)

{

b = pre[a];

pre[a] = i;

a = b;

}

return i;

}

int main()

{

int n,m,x,y;

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

{

if(n==0)

break;

int sum = n - 1;

scanf("%d",&m);

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

pre[i] = i;

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

{

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

int f1 = find(x);

int f2 = find(y);

if(f1!=f2)

{

pre[f1] = f2;

sum--;

}

}

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

}

return 0;

}

5，AC代码：

#include <stdio.h>

#include <string.h>

#include <algorithm>

using namespace std;

#define MAX 1005

#define INF 0x3f3f3f3f

int pre[MAX],id[MAX],vis[MAX],n,m,pos;

double in[MAX]; ///in存的是最小的入边 pre存的是一个边的起点

struct node

{

int u,v;

double w;

}edge[MAX \* MAX];

int solve(int root, int V, int E)

{

double ret = 0;

while(true)

{

///找每个节点的最小入边

for(int i = 0; i < V; i++)

{

in[i] = INF;

}

for(int i = 0; i < E; i++)

{

int u = edge[i].u;

int v = edge[i].v;

if(edge[i].w < in[v]&&u!=v)

{

pre[v] = u;

in[v] = edge[i].w;

if(u==root)

pos = i;

}

}

///判断是否存在最小树形图

for(int i = 0; i < V; i++)

{

if(i==root)

continue;

if(in[i]==INF)

return -1; ///除了根以外有点没有入边,则根无法到达它 说明它是独立的点 一定不能构成树形图

}

///找环

int cnt = 0;

memset(vis,-1,sizeof(vis));

memset(id,-1,sizeof(id));

in[root] = 0;

for(int i = 0; i < V; i++)

{

ret+=in[i];

int v = i;

while(vis[v]!=i&&id[i]==-1&&v!=root) ///用来建环的

{

vis[v] = i;

v = pre[v];

}

if(v!=root&&id[v]==-1)

{

for(int u = pre[v]; u!=v; u = pre[u])

id[u] = cnt; ///标记节点u为第几个环

id[v] = cnt++;

}

}

if(cnt==0)

break;///无环

for(int i = 0; i < V; i++)

{

if(id[i]==-1)

id[i] = cnt++;

}

///建立新图，缩点 重新标记

for(int i = 0; i < E; i++)

{

int u = edge[i].u;

int v = edge[i].v;

edge[i].u = id[u];

edge[i].v = id[v];

if(id[u]!=id[v])

edge[i].w-=in[v];

}

V = cnt;

root = id[root];

}

return ret;

}

int main()

{

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

{

double sum = 0;

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

{

scanf("%d%d%lf",&edge[i].u,&edge[i].v,&edge[i].w);

edge[i].u++;edge[i].v++;

sum+=edge[i].w;

}

sum++;

for(int i = m; i < m + n; i++)

{

edge[i].u = 0;

edge[i].v = i - m + 1;

edge[i].w = sum;

}

double ans = solve(0,n + 1, n + m);

///n+1为总结点数,m+n为总边数

///ans代表以超级节点0为根的最小树形图的总权值,

///将ans减去sum,如果差值小于sum,说明节点0的出度只有1,说明原图是连通图

///如果差值>=sum,那么说明节点0的出度不止为1,说明原图不是连通图

if(ans==-1||ans - sum > sum)

printf("impossible\n");

else

printf("%.0lf %d\n",ans - sum, pos - m);

printf("\n");

}

return 0;

}

6.AC代码：

#include <cstdio>

#include <iostream>

using namespace std;

int pre[1001];

int find(int x)

{

int i = x;

while(pre[i]!=i)

{

i = pre[i];

}

int a = x,b;

while(a!=i)

{

b = pre[a];

pre[a] = i;

a = b;

}

return i;

}

int main()

{

int n,m,x,y;

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

{

if(n==0)

break;

int sum = n - 1;

scanf("%d",&m);

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

pre[i] = i;

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

{

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

int f1 = find(x);

int f2 = find(y);

if(f1!=f2)

{

pre[f1] = f2;

sum--;

}

}

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

}

return 0;

}

7.AC代码：

#include <iostream>

#include <cstring>

#include <cstdio>

#include <math.h>

#define INF 0x3f3f3f3f

using namespace std;

int map[1001][1001], n, m, vis[1001],dis[1001];

void prim()

{

memset(dis,1,sizeof(dis));

int sum = 0;

int k;

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

{

vis[i] = map[0][i];

}

int min;

dis[0] = 0;

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

{

min = INF;

for(int j = 0; j < m; j++)

{

if(dis[j]!=0&&vis[j]<min)

{

min = vis[j];

k = j;

}

}

if(min==INF)

{

printf("?\n");

return ;

}

dis[k] = 0;

sum+=min;

for(int j = 0; j < m; j++)

if(dis[j]!=0&&map[k][j] < vis[j])

{

vis[j] = map[k][j];

}

}

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

}

int main()

{

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

{

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

for(int j = 0; j < m; j++)

map[i][j] = INF;

int x, y, z;

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

{

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

map[x - 1][y - 1] = map[y-1][x-1] = z;

}

prim();

}

return 0;

}