**最短路径及最小生成树题目：**

**类型一：**

/\*

\* 1077.cpp

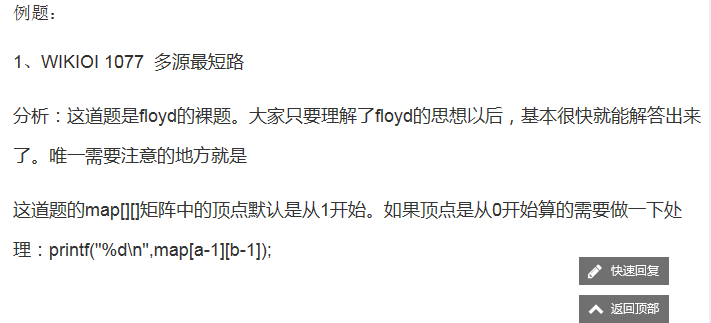
\*

\* Created on: 2014年5月23日

\* Author: pc

\*/





# 题目描述：【codevs1077】多源最短路

**2013年12月30日6170**

## 题目描述 Description

**已知n个点(n<=100)，给你n\*n的方阵，a[i,j]表示从第i个点到第j个点的直接距离。**

**现在有Q个询问，每个询问两个正整数，a和b，让你求a到b之间的最短路程。**

**满足a[i,j]=a[j,i];**

## 输入描述 Input Description

**第一行一个正整数n，接下来n行每行n个正整数，满足a[i,i]=0,再一行一个Q，接下来Q行，每行两个正整数a和b。**

## 输出描述 Output Description

**一共Q行，每行一个整数。**

## 样例输入 Sample Input

**3**

**0 1 1**

**1 0 3**

**1 3 0**

**1**

**2 3**

## 样例输出 Sample Output

**2**

## 数据范围及提示 Data Size & Hint

**n<=100，Q可能非常大。g[i][j]均>=0**

**请使用flyod算法**

**使用C/C++的同学请注意：由于输入数据较大，使用cin和cout会导致程序超时。请使用scanf与printf进行输入和输出。**

## 代码

#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

int e[maxn][maxn];

int n;

const int inf = 99999999;

void initial() {

int i, j;

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

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

if (i == j) {

e[i][j] = 0;

} else {

e[i][j] = inf;

}

}

}

}

/\*\*

\*floyd算法

\*/

void floyd() {

int i, j, k;

for (k = 1; k <= n; ++k) {//遍历所有的中间点

for (i = 1; i <= n; ++i) {//遍历所有的起点

for (j = 1; j <= n; ++j) {//遍历所有的终点

if (e[i][j] > e[i][k] + e[k][j]) {//如果当前i-->j的距离大于i-->k--->j的距离之和

e[i][j] = e[i][k] + e[k][j];//更新从i--->j的最短路径

}

}

}

}

}

int main() {

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

initial();

int i, j;

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

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

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

}

}

floyd();

int q;

scanf("%d", &q);

while (q--) {

int a, b;

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

printf("%d\n", e[a][b]);

}

}

return 0;

}

**类型二**

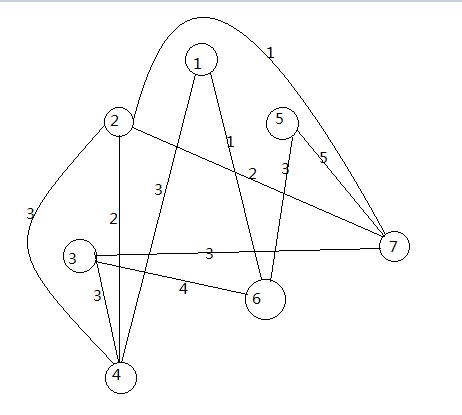
**1：求可变指定起始点到终点的最短路径（起点和终点是变化的）**

**1728: 运送牛奶**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 31  Solved: 9  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1728)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1728)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1728)]

**Description**

德克萨斯纯朴的民眾们这个夏天正在遭受巨大的热浪！！！他们的德克萨斯长角牛吃起来不错，可是他们并不是很擅长生產富含奶油的乳製品。Farmer  John此时以先天下之忧而忧，后天下之乐而乐的精神，身先士卒地承担起向德克萨斯运送大量的营养冰凉的牛奶的重任，以减轻德克萨斯人忍受酷暑的痛苦。 FJ已经研究过可以把牛奶从威斯康星运送到德克萨斯州的路线。这些路线包括起始点和终点先一共经过T  (1  < =  T  < =  2,500)个城镇，方便地标号為1到T。除了起点和终点外地每个城镇由两条双向道路连向至少两个其它地城镇。每条道路有一个通过费用（包括油费，过路费 等等）。考虑这个有7个城镇的地图。城镇5是奶源，城镇4是终点（括号内的数字是道路的通过费用）。



经过路线5-6-3-4总共需要花费3  (5-> 6)  +  4  (6-> 3)  +  3  (3-> 4)  =  10的费用。 给定一个地图，包含C  (1  < =  C  < =  6,200)条直接连接2个城镇的道路。每条道路由道路的起点Rs，终点Re  (1  < =  Rs  < =  T;  1  < =  Re  < =  T)，和花费(1  < =  Ci  < =  1,000)组成。求从起始的城镇Ts  (1  < =  Ts  < =  T)到终点的城镇Te(1  < =  Te  < =  T)最小的总费用。

**Input**

第一行:  4个由空格隔开的整数:  T,  C,  Ts,  Te

第2到第C+1行:

第i+1行描述第i条道路。有3个由空格隔开的整数:  Rs,  Re和Ci

**Output**

一个单独的整数表示Ts到Te的最短路的长度。（不是费用麼？怎麼突然变直白了 ——译者注）数据保证至少存在一条道路。

**Sample Input**

7 11 5 4

2 4 2

1 4 3

7 2 2

3 4 3

5 7 5

7 3 3

6 1 1

6 3 4

2 4 3

5 6 3

7 2 1

**Sample Output**

7

**HINT**

5-> 6-> 1-> 4  (3  +  1  +  3)

**1．耗时：**

#include <stdio.h>

#include <string.h>

#define MIN(a,b) a<b?a:b

#define M 9999

#define N 2503

int d[N];//起点到该点的权值

int w[N][N];//a到b的权值

int vis[N];

void Dijkstra(int n,int s)

{

    int i,j,k,v,t;

    memset(vis,0,sizeof(vis));

    for(i=1; i<=n; i++)

        d[i]=w[s][i];

    d[s]=0;

    vis[s]=1;

    for(i=1; i<=n; i++)

    {

        t=M;

        for(j=1; j<=n; j++)

        {

            if(t>d[j]&&!vis[j])

            {

                t=d[j];

                v=j;

            }

        }

        vis[v]=1;

        for(k=1; k<=n; k++)

        {

            if(!vis[k])d[k]=MIN(d[k],d[v]+w[v][k]);

        }

    }

}

int main()

{

    int n,m,i,j,a,b,c,T,S,C,E;

    scanf("%d%d%d%d",&T,&C,&S,&E);

        for(i=1;i<=T;i++)

            for(j=1;j<=T;j++)

            w[i][j]=M;

       for(i=0;i<C;i++)

       {

            scanf("%d%d%d",&a,&b,&c);

            w[a][b]=MIN(w[a][b],c);

            w[b][a]=w[a][b];

        }

        Dijkstra(T,S);

        printf("%d\n",d[E]);

    return 0;

}

**2.用队列做的：**

#include<stdio.h>

#include<vector>

#include<queue>

using namespace std;

const int imax = 2505;

#define INF 1e9

vector<pair<int,int> >vec[imax];

int dis[imax];

void Dijkstra(int s)

{

priority\_queue<pair<int,int> >que;

dis[s]=0;

que.push(make\_pair(-dis[s],s));

while(!que.empty())

{

int now = que.top().second;

que.pop();

for(int i = 0; i<vec[now].size(); i++)

{

int v = vec[now][i].first;

if(dis[v]>dis[now]+vec[now][i].second)

{

dis[v]=dis[now]+vec[now][i].second;

que.push(make\_pair(-dis[v],v));

}

}

}

}

void init()

{

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

vec[i].clear(),dis[i]=INF;

}

int main()

{

int n,m,s,t;

int Rs,Re,Ci;

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

{

init();

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

{

scanf("%d%d%d",&Rs,&Re,&Ci);

vec[Rs].push\_back(make\_pair(Re,Ci));

vec[Re].push\_back(make\_pair(Rs,Ci));

}

Dijkstra(s);

printf("%d\n",dis[t]);

}

}

**类型三**

**2.求唯一起点到终点的最短路径（起点和终点固定不变）**

**1950: 中午饿了要定餐（该题测试数据中包含重复路径，是万能模版）**

Time Limit: 1 Sec  Memory Limit: 33 MB  
Submit: 52  Solved: 8  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1950)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1950)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1950)]

**Description**

ACM比赛正在5教进行中，转眼到了中午，这时候出去吃饭怎么可能，只有叫外卖了。外卖要从小门出发，骑自行车来到5教，因为大一学生军训，很多路线有变化，送餐员不知道怎么走了。现在请你来帮帮他，根据你知道的校园各条路的距离，规划出一条从小门到5教的最短路线。

**Input**

输入包括多组数据。每组数据第一行是两个整数N、M（N<=100，M<=10000），N表示校园内有几个路口，标号为1的路口是小门所在 地，标号为N的路口是5教所在地，M则表示在校园有几条路。N=M=0表示输入结束。接下来M行个整数A，B，C（1<=A,B<=N,1& lt;=C<=1000）,表示在路口A与路口B之间有一条路，C表示A到B的距离。  
输入保证至少存在1条小门到5教的路线。

**Output**

对于每组输入，输出一行，表示送餐员从小门到5教的最短路线

**Sample Input**

2 1

1 2 3

3 3

1 2 5

2 3 5

3 1 2

0 0

**Sample Output**

3

2

**HINT**

#include <stdio.h>

#include <string.h>

#define MIN(a,b) a<b?a:b

#define M 100000

#define N 101

int d[N];//起点到该点的权值

int w[N][N];//a到b的权值

int vis[N];

void Dijkstra(int n)

{

    int i,j,k,v,t;

    memset(vis,0,sizeof(vis));

    for(i=1; i<=n; i++)

        d[i]=w[1][i];

    d[1]=0;

    vis[1]=1;

    for(i=1; i<=n; i++)

    {

        t=M;

        for(j=1; j<=n; j++)

        {

            if(t>d[j]&&!vis[j])

            {

                t=d[j];

                v=j;

            }

        }

        vis[v]=1;

        for(k=1; k<=n; k++)

        {

            if(!vis[k])d[k]=MIN(d[k],d[v]+w[v][k]);

        }

    }

}

int main()

{

    int n,m,i,j,a,b,c;

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

    {

        for(i=1;i<=n;i++)

            for(j=1;j<=n;j++)

            w[i][j]=M;

        while(m--)

        {

            scanf("%d%d%d",&a,&b,&c);

            w[a][b]=MIN(w[a][b],c);

            w[b][a]=w[a][b];

        }

        Dijkstra(n);

        printf("%d\n",d[n]);

    }

    return 0;

}

**2．求已知起点到终点的最短路径：(该题测试数据不包含重复路径，老师的模板是万能的模板)**

#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

const int inf = 9999999;

int s[maxn];//用来记录某一点是否被访问过

int map[maxn][maxn];//地图

int dis[maxn];//从原点到某一个点的最短距离(一开始是估算距离)

int n;

int target;

/\*\*

\* 返回从v---->到target的最短路径

\*/

int dijkstra(int v){

int i;

for(i = 1 ; i <= n ; ++i){//初始化

s[i] = 0;//一开始，所有的点均为被访问过

dis[i] = map[v][i];

}

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

int min = inf;

int pos;

int j;

for(j = 1 ; j <= n ; ++j){//寻找目前的最短路径的最小点

if(!s[j] && dis[j] < min){

min = dis[j];

pos = j;

}

}

s[pos] = 1;

for(j = 1 ; j <= n ; j++){//遍历u的所有的邻接的边

if(!s[j] && dis[j] > dis[pos] + map[pos][j]){

dis[j] = dis[pos] + map[pos][j];//对边进行松弛

}

}

}

return dis[target];

}

int main(){

int m;

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

int i;

int j;

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

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

if(i == j){

map[i][j] = 0;

}else{

map[i][j] = inf;

}

}

}

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

int a,b,c;

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

map[a][b] = map[b][a] = c;//这里默认是无向图。。所以要两个方向都做处理,只做一个方向上的处理会WA

}

target = n;

int result = dijkstra(1);

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

}

return 0;

}

*---------------------------------------------------------------------------------------------------------------------------------*-/\*

\* NEFU\_207.cpp

\*

\* Created on: 2014年5月27日

\* Author: pc

\*/

#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

const int inf = 99999999;

int e[maxn][maxn];

int n,m;

void initial(){

int i;

int j;

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

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

if(i == j){

e[i][j] = 0;

}else{

e[i][j] = inf;

}

}

}

}

void floyd(){

int i;

int j;

int k;

for(k = 1 ; k <= n ; ++k){

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

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

if(e[i][j] > e[i][k] + e[k][j]){

e[i][j] = e[i][k] + e[k][j];

}

}

}

}

}

int main(){

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

initial();

int i;

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

int a,b,c;

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

e[a][b] = e[b][a] = c;

}

floyd();

printf("%d\n",e[1][n]);

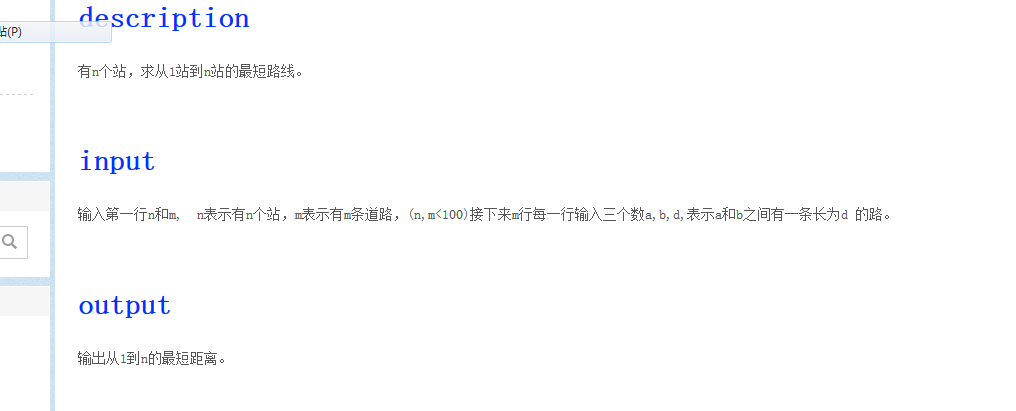
}

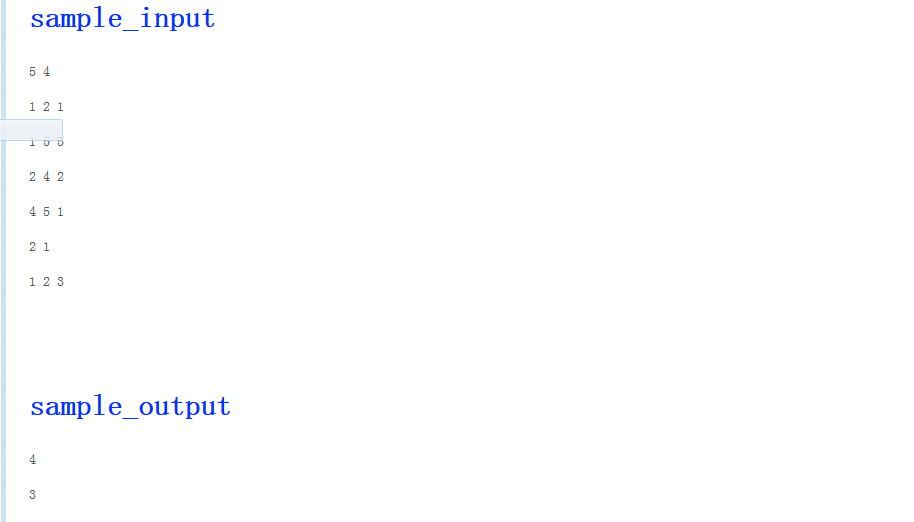
return 0;

}

**floyd：解决多源最短路径问题。求任意两个点之间的最短路径。这当然也就包含了“从a到b的这种情况”。所以这道题也可以使用floyd来解决**

**dijkstra：解决单源最短路径问题 。最典型的就是解决“从a到b的最短路径的距离”的这种问题了。**





#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

const int inf = 9999999;

int s[maxn];//用来记录某一点是否被访问过

int map[maxn][maxn];//地图

int dis[maxn];//从原点到某一个点的最短距离(一开始是估算距离)

int n;

int target;

/\*\*

 \* 返回从v---->到target的最短路径

 \*/

int dijkstra(int v){

    int i;

    for(i = 1 ; i <= n ; ++i){//初始化

        s[i] = 0;//一开始，所有的点均为被访问过

        dis[i] = map[v][i];

    }

    for(i = 1 ; i < n ; ++i){

        int min = inf;

        int pos;

        int j;

        for(j = 1 ; j <= n ; ++j){//寻找目前的最短路径的最小点

            if(!s[j] && dis[j] < min){

                min = dis[j];

                pos = j;

            }

        }

        s[pos] = 1;

        for(j = 1 ; j <= n ; j++){//遍历u的所有的邻接的边

            if(!s[j] && dis[j] > dis[pos] + map[pos][j]){

                dis[j] = dis[pos] + map[pos][j];//对边进行松弛

            }

        }

    }

    return dis[target];

}

int main(){

    int m;

    while(scanf("%d%d",&n,&m)!=EOF){

      if(n==0)

      break;

        int i;

        int j;

        for(i = 1 ; i <= n ; ++i){

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

                if(i == j){

                    map[i][j] = 0;

                }else{

                    map[i][j] = inf;

                }

            }

        }

        for(i = 1 ; i <= m ; ++i){

            int a,b,c;

            scanf("%d%d%d",&a,&b,&c);

            map[a][b] = map[b][a] = c;//这里默认是无向图。。所以要两个方向都做处理,只做一个方向上的处理会WA

        }

        target = n;

        int result = dijkstra(1);

        printf("%d\n",result);

    }

    return 0;

}

### 4.任意两点最短路径问题（给的是坐标距离需要自己求）

#### Problem:313

#### Time Limit:1000ms

#### Memory Limit:65536K

### Description

平面上有n个点(n<=100)，每个点的坐标均在-10000~10000之间。其中的一些点之间有连线，若有连线，则表示可从一个点到达另一个点，即两点间有通路，通路的距离为两点间的直线距离。现在的任务是找出从一点到另一点之间的最短路径。

### Input

多组数据输入.

每组输入共n+m+3行，其中：

第一行为整数n。

第2行到第n+1行（共n行），每行两个整数x和y，描述了一个点的坐标（以一个空格分隔）。

第n+2行为一个整数m，表示图中连线的个数。

此后的m行，每行描述一条连线，由两个整数I和j组成，表示第I个点和第j个点之间有连线。

最后一行：两个整数s和t，分别表示源点和目标点。

### Output

每组输出一个实数（保留两位小数），表示从s到t 的最短路径长度。

### Sample Input

5

0 0

2 0

2 2

0 2

3 1

5

1 2

1 3

1 4

2 5

3 5

1 5

### Sample Output

3.41

### Hint

### Source

OI

/\*

\* NEFU\_313.cpp

\*

\* Created on: 2014年5月27日

\* Author: pc

\*/

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

const int maxn = 105;

const int inf = INT\_MAX;

int s[maxn];

double dis[maxn];

double map[maxn][maxn];

int n;

int target;

struct Pointt{

double x;

double y;

};

double distance1(Pointt p1, Pointt p2){

return sqrt((p1.x - p2.x)\*(p1.x - p2.x) + (p1.y - p2.y)\*(p1.y - p2.y));

}

double dijkstra(int v){

int i;

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

s[i] = 0;

dis[i] = map[v][i];

}

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

double min = inf;

int pos;

int j;

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

if(!s[j] && dis[j] < min){

min = dis[j];

pos = j;

}

}

s[pos] = 1;

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

if(!s[j] && dis[j] > dis[pos] + map[pos][j]){

dis[j] = dis[pos] + map[pos][j];

}

}

}

return dis[target];

}

void printfMap(){

int i;

int j;

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

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

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

}

printf("\n");

}

}

int main(){

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

Pointt p[n+1];

int i;

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

scanf("%lf%lf",&p[i].x,&p[i].y);

}

int j;

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

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

if(i == j){

map[i][j] = 0;

}else{

map[i][j] = inf;

}

}

}

int m;

scanf("%d",&m);

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

int a,b;

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

map[a][b] = map[b][a] = distance1(p[a],p[b]);

}

int start;

scanf("%d%d",&start,&target);

double result = dijkstra(start);

printf("%.2lf\n",result);

}

return 0;

}

/\*

\* NEFU\_313.cpp

\*

\* Created on: 2014年5月27日

\* Author: pc

\*/

Floyd:

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

const int maxn = 105;

double map[maxn][maxn];

int n;

const int inf = INT\_MAX;

struct Pointt {

double x;

double y;

};

double distance1(Pointt p1, Pointt p2) {

return sqrt((p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y));

}

void initial() {

int i;

int j;

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

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

if (i == j) {

map[i][j] = 0;

} else {

map[i][j] = inf;

}

}

}

}

void floyd() {

int i;

int j;

int k;

for (k = 1; k <= n; ++k) {

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

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

if (map[i][j] > map[i][k] + map[k][j]) {

map[i][j] = map[i][k] + map[k][j];

}

}

}

}

}

int main() {

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

int i;

Pointt p[n + 1];

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

scanf("%lf%lf", &p[i].x, &p[i].y);

}

int m;

scanf("%d", &m);

initial();

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

int a, b;

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

map[a][b] = map[b][a] = distance1(p[a], p[b]);

}

floyd();

int start, end;

scanf("%d%d", &start, &end);

printf("%.2lf\n", map[start][end]);

}

return 0;

}

## 最小生成树问题：

## 5. 1396: Dick's back

时间限制: 1 Sec  内存限制: 128 MB  
提交: 16  解决: 6  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1396)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1396)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1396)]

**题目描述**

In an episode of the Dick Van Dyke show, little Richie connects the freckles on his Dad's back to form a picture of the Liberty Bell. Alas, one of the freckles turns out to be a scar, so his Ripley's engagement falls through.

Consider Dick's back to be a plane with freckles at various (x,y) locations. Your job is to tell Richie how to connect the dots so as to minimize the amount of ink used. Richie connects the dots by drawing straight lines between pairs, possibly lifting the pen between lines. When Richie is done there must be a sequence of connected lines from any freckle to any other freckle.

**输入**

The first line contains 0 < *n* <= 100, the number of freckles on Dick's back. For each freckle, a line follows; each following line contains two real numbers indicating the (x,y) coordinates of the freckle.

**输出**

Your program prints a single real number to two decimal places: the minimum total length of ink lines that can connect all the freckles.

**样例输入**

3

1.0 1.0

2.0 2.0

2.0 4.0

**样例输出**

3.41

**提示**

**典型的模板题（prim算法）**

 #include<stdio.h>

    #include<string.h>

    #include<math.h>

    double d[102][2], w[102][102], minCost[102];

    int n, pre[102], hash[102];

     double getDist(double x1,double y1,double x2,double y2){

        return sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2));

    }

    double Prim(){

        memset(hash, 0, sizeof(hash));

        hash[1] = 1;

        for(int i=1; i<=n; ++i){

            minCost[i] = w[1][i];

            pre[i] = 1;

        }

        double sum=0;

        for(int i=1; i<n; ++i){

            int u=-1;

            for(int j=1; j<=n; ++j)if(!hash[j]){

                if(u==-1||minCost[j]<minCost[u])

                    u=j;

            }

            sum += w[pre[u]][u];

            hash[u] = 1;

            for(int j=1; j<=n; ++j)if(!hash[j]){

                if(minCost[j]>w[u][j]){

                    minCost[j] = w[u][j];

                    pre[j] = u;

                }

            }

        }

        return sum;

    }//prim

    int main(){

            scanf("%d",&n);

            for(int i=1; i<=n; ++i)

                scanf("%lf%lf",&d[i][0],&d[i][1]);

            memset(w, 0, sizeof(w));

            for(int i=1; i<=n; ++i)

                for(int j=1; j<=n; ++j)if(i!=j)

                    w[i][j] = getDist(d[i][0],d[i][1],

                            d[j][0],d[j][1]);

            printf("%.2f\n", Prim());

        return 0;

}

**6.求任意点最短路径及最少花费（即路径长度相同时优先选择花费少的）**

**1970: Shortest Path Problem**

时间限制: 1 Sec  内存限制: 128 MB  
提交: 20  解决: 5  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1970)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1970)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1970)]

**题目描述**

 Given N points, M undirected edges, every edge has length D and cost P. Now give you starting point S and ending point T, output the shortest distance from starting point to ending point and its cost. If the shortest path has several routes, print less cost one.

**输入**

     Input N,M, point is marked as 1 to N, and M lines followed, each line has 4 numbers A,B,D,P, indicating there is an edge between A and B, length is D, cost is P. The last line has two numbers S, T. If  the  N,M is 0 0,End the file.

  (1<N<=1000, 0<M<100000, S !=T)

**输出**

Two numbers, shortest distance and cost.

**样例输入**

3 2

1 2 5 6

2 3 4 5

1 3

0 0

**样例输出**

9 11

**提示**

#include <stdio.h>

#include <string.h>

#define MIN(a,b) a<b?a:b

#define M 100000

#define N 101

int d[N];//起点到该点的权值

int cos[N];

int w[N][N];//a到b的权值

int ww[N][N];

int vis[N];

int viss[N];

void Dijkstra(int n,int s)

{

    int i,j,k,v,t;

    memset(vis,0,sizeof(vis));

    for(i=1; i<=n; i++)

      {

        d[i]=w[s][i];

        cos[i]=ww[s][i];

      }

    d[s]=0;

    vis[s]=1;

    cos[s]=0;

   // viss[s]=1;

    for(i=1; i<=n; i++)

    {

        t=M;

        for(j=1; j<=n; j++)

        {

            if(t>d[j]&&!vis[j])

            {

                t=d[j];

                v=j;

            }

        }

        vis[v]=1;

        for(k=1; k<=n; k++)

        {

            if(!vis[k])

            {

                if(d[k]>d[v]+w[v][k])

                {

                    d[k]=d[v]+w[v][k];

                    cos[k]=cos[v]+ww[v][k];

                }

                if(d[k]==d[v]+w[v][k])

                {

                    if(cos[k]>cos[v]+ww[v][k])

                      cos[k]=cos[v]+ww[v][k];

                }

            }

        }

    }

}

int main()

{

    int n,m,i,j,a,b,c,cost,C,E,T,S;

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

    {

        for(i=1;i<=n;i++)

            for(j=1;j<=n;j++)

            {

              w[i][j]=ww[i][j]=M;

            }

        for(i=0;i<m;i++)

        {

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

           // w[a][b]=MIN(w[a][b],c);

          //  ww[a][b]=cost;

            if(w[a][b]>c)

            {

                w[a][b]=w[b][a]=c;

                ww[a][b]=ww[b][a]=cost;

            }

            if(w[a][b]==c)

            {

                if(ww[a][b]>cost)

                ww[a][b]=ww[b][a]=cost;

            }

        }

        scanf("%d%d",&S,&E);

        Dijkstra(n,S);

        printf("%d %d\n",d[E],cos[E]);

    }

    return 0;

}

8

### 宫锁珠帘

#### Problem:208

#### Time Limit:1000ms

#### Memory Limit:65536K

### Description

新年期间湖南卫视又上映了不少宫闱大戏，晴川穿走了，又来了个宫锁珠帘的袁珊珊，袁珊珊不够美，没锁住观众的心，特别是咱们整ACM的，对这些宫闱大戏当然是不屑一顾了，可这新亮剑的李云龙咱们还是有得一看的，最近听说A地又驻扎了一个骑兵营，李云龙心里就乐了，心想老子吃了骑兵营这么多亏，这下终于有机会自个儿也能整个啥骑兵营威风威风了，于是拿出地图决定抄最短的路立马拿下这个骑兵营。

### Input

每组数据第一行包含两个正整数N和M(0 < N < 100,0 < M < 100)，分别代表李云龙现有地图上显示的村子或者驻扎地的总数目，分别以0～N-1编号。

接下来是M行道路信息。每一行有三个整数A,B,X(0 <= A,B < N,A!=B,0< X < 10000),表示A和B之间有一条长度为X的双向道路。

再接下一行有两个整数S,T(0 <= S,T < N)，分别代表起点和终点。

### Output

输出从S到T的最短路的长度。若不存在这样的一条路，则输出-1.

### Sample Input

5 4

0 1 5

0 3 4

3 2 7

3 2 6

0 2

4 1

2 3 1

1 2

### Sample Output

10

-1

### Hint

题目与分析：

这道题抽象一下，还是“求从a到b的最短距离”。。同样可以使用floyd和dijkstra来做。。

这道题与前面的不同的地方在于：两个点之间可能有多条路(我们保存那条最短的即可)。

另外，还要理解dijkstra和floyd算法中使用到的map[][]矩阵的含义。

map[i][i] = 0.自己到自己的距离为0

map[i][j] = inf .表示两点之间无法连通

以下是分别用dijkstra、floyd、spfa这三种算法来做的代码，需要注意的是这道题顶点序号的范围是0~n-1，而之前做的题目的定点序号范围都是1~n。

/\*

\* NEFU\_208.cpp

\*

\* Created on: 2014年5月27日

\* Author: pc

\*/

#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

const int inf = 10005;

int n;

int s[maxn];

int dis[maxn];

int map[maxn][maxn];

int target;

int dijkstra(int v){

int i;

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

s[i] = 0;

dis[i] = map[v][i];

}

for(i = 0 ; i < n-1 ; ++i){//这里的意思实际上是将剩下的n-1个点全部放到S集合中

int min = inf;

int pos;

int j;

for(j = 0 ; j < n ; ++j){//寻找最短路径点

if(!s[j] && dis[j] < min){

min = dis[j];

pos = j;

}

}

s[pos] = 1;

for(j = 0 ; j < n ; ++j){

if(!s[j] && dis[j] > dis[pos] + map[pos][j]){

dis[j] = dis[pos] + map[pos][j];

}

}

}

return dis[target];

}

int main(){

int m;

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

int i;

int j;

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

for(j = 0 ; j < n ; ++j){

if(i == j){

map[i][j] = 0;

}else{

map[i][j] = inf;

}

}

}

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

int a,b,c;

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

if(map[a][b] > c){

map[a][b] = map[b][a] = c;

}

}

int start,end;

scanf("%d%d",&start,&end);

target = end;

int result = dijkstra(start);

if(result == inf){

printf("-1\n");

}else{

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

}

}

return 0;

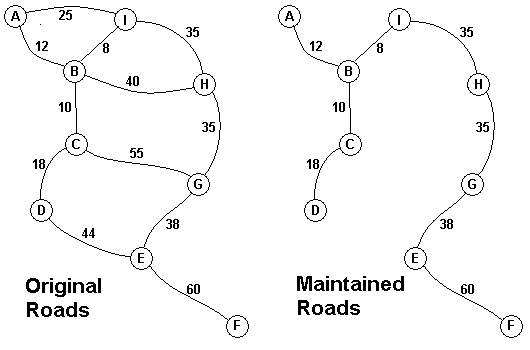
}

**并查集求最小生成树**

## **1824: J. Jungle**

Time Limit: 1 Sec  Memory Limit: 33 MB  
Submit: 8  Solved: 4  
[[Submit](http://218.27.6.151/oj/submitpage.php?id=1824)][[Status](http://218.27.6.151/oj/problemstatus.php?id=1824)][[Web Board](http://218.27.6.151/oj/bbs.php?pid=1824)]

## **Description**



The Head Elder of the tropical island of Lagrishan has a problem. A burst of foreign aid money was spent on extra roads between villages some years ago. But the jungle overtakes roads relentlessly, so the large road network is too expensive to maintain. The Council of Elders must choose to stop maintaining some roads. The map above on the left shows all the roads in use now and the cost in aacms per month to maintain them. Of course there needs to be some way to get between all the villages on maintained roads, even if the route is not as short as before. The Chief Elder would like to tell the Council of Elders what would be the smallest amount they could spend in aacms per month to maintain roads that would connect all the villages. The villages are labeled A through I in the maps above. The map on the right shows the roads that could be maintained most cheaply, for 216 aacms per month. Your task is to write a program that will solve such problems.   
  
The input consists of one to 100 data sets, followed by a final line containing only 0. Each data set starts with a line containing only a number n, which is the number of villages, 1 < n < 27, and the villages are labeled with the first n letters of the alphabet, capitalized. Each data set is completed with n-1 lines that start with village labels in alphabetical order. There is no line for the last village. Each line for a village starts with the village label followed by a number, k, of roads from this village to villages with labels later in the alphabet. If k is greater than 0, the line continues with data for each of the k roads. The data for each road is the village label for the other end of the road followed by the monthly maintenance cost in aacms for the road. Maintenance costs will be positive integers less than 100. All data fields in the row are separated by single blanks. The road network will always allow travel between all the villages. The network will never have more than 75 roads. No village will have more than 15 roads going to other villages (before or after in the alphabet). In the sample input below, the first data set goes with the map above.   
  
The output is one integer per line for each data set: the minimum cost in aacms per month to maintain a road system that connect all the villages. Caution: A brute force solution that examines every possible set of roads will not finish within the one minute time limit.

## **Input**

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

## **Output**

216

30

## **Sample Input**

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

## **Sample Output**

216

30

## **HINT**

## **Source**

#include<stdio.h>

int set[27];

void merge(int x ,int y)

{

set[x]=y;

}

int find(int x)

{

int tt;

tt=x;

while(set[tt]!=tt)

tt=set[tt];

return tt;

}

int main()

{

int t,i,j,k,sj,sk,p,q,n,r,a[27][27];

char c,d;

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

{

for(i=0;i<=26;i++)

for(j=0;j<=26;j++)

{

a[i][j]=101;

set[j]=j;

}

q=t;

//printf("---%d\n",q);

while(--q)

{

//printf("---%d\n",q);

scanf("%\*c%c %d",&c,&n);

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

{

scanf("%\*c%c %d",&d,&k);

a[c-'A'][d-'A']=k;

}

}

r=sj=sk=0;

q=t;

while(--q){

p=101;

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

{

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

{

if(p>a[j][k])

{

if(find(j)!=find(k))

{

p=a[j][k];

sj=j;

sk=k;

}

}

}

}

r+=p;

merge(find(sj),find(sk));

// printf("-%d\n",q);

}

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

}

return 0;

}

# 畅通工程

**Time Limit: 1000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 26026    Accepted Submission(s): 11350**

**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

?

**Source**

[浙大计算机研究生复试上机考试-2007年](http://acm.hdu.edu.cn/search.php?field=problem&key=%D5%E3%B4%F3%BC%C6%CB%E3%BB%FA%D1%D0%BE%BF%C9%FA%B8%B4%CA%D4%C9%CF%BB%FA%BF%BC%CA%D4-2007%C4%EA&source=1&searchmode=source)

**Recommend**

lcy

#include <stdio.h>

#include <stdlib.h>

#include <algorithm>

using namespace std;

const int N = 150;

int m,n,u[N],v[N],w[N],p[N],r[N];

int cmp(const int i,const int j)

{

return w[i] > w[j];

}

int find(int x)

{

return p[x]==x?x:p[x]=find(p[x]);

}

int kruskal()

{

int cnt=0,x,y,i,ans=0;

//n是点数，m是边数，汝佳那本书上是如此

//并查集初始化

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

p[i]=i;

//边编号

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

r[i]=i;

sort(r,r+m,cmp);

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

{

//取出未加入的边权最小的边的编号

int e=r[i];

x=find(u[e]);

y=find(v[e]);

if(x!=y)

{

ans += w[e];

p[x]=y;

cnt++;

}

}

//找不到最小生成树

if(cnt<n-1)

ans=0;

return ans;

}

int main()

{

int i,ans;

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

{

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

{

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

}

ans=kruskal();

if(ans)

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

else //说明不存在最小生成树

puts("存在最小生成树!");

}

return 0;

}