Bessie is out in the field and wants to get back to the barn to get as much sleep as possible before Farmer John wakes her for the morning milking. Bessie needs her beauty sleep, so she wants to get back as quickly as possible.   
  
Farmer John's field has N (2 <= N <= 1000) landmarks in it, uniquely numbered 1..N. Landmark 1 is the barn; the apple tree grove in which Bessie stands all day is landmark N. Cows travel in the field using T (1 <= T <= 2000) bidirectional cow-trails of various lengths between the landmarks. Bessie is not confident of her navigation ability, so she always stays on a trail from its start to its end once she starts it.   
  
Given the trails between the landmarks, determine the minimum distance Bessie must walk to get back to the barn. It is guaranteed that some such route exists.

**Input**

\* Line 1: Two integers: T and N   
  
\* Lines 2..T+1: Each line describes a trail as three space-separated integers. The first two integers are the landmarks between which the trail travels. The third integer is the length of the trail, range 1..100.

**Output**

\* Line 1: A single integer, the minimum distance that Bessie must travel to get from landmark N to landmark 1.

**Sample Input**

5 5

1 2 20

2 3 30

3 4 20

4 5 20

1 5 100

**Sample Output**

90

**Hint**

INPUT DETAILS:   
  
There are five landmarks.   
  
OUTPUT DETAILS:   
  
Bessie can get home by following trails 4, 3, 2, and 1.

Dijkstra算法

Dijkstra算法是一种按路径长度增序产生各顶点最短路径的算法。每次新扩展一个距离最短的点，更新与其相邻的点的距离。当所有边权都为正时，由于不会存在一个距离更短的没扩展过的点，所以这个点的距离永远不会再被改变，因而保证了算法的正确性。

#include <iostream>

#include<stdio.h>

#include<cstring>

#define maxn 536870912

#define up(x,y) for(int i=x;i<=y;i++)

#define mem(x) memset(x,0,sizeof(x))

using namespace std;

int mapp[2010][2010];

bool b[2010];

int d[2010];

int main()

{

// freopen("input.txt","r",stdin);

int t,n,x,y,e,v,minn;

cin>>t>>n;

up(1,n)

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

if(i==j)mapp[i][i]=0;else mapp[i][j]=maxn;

//注意处理mapp[i][i]的情况

while(t--)

{

cin>>x>>y>>e;

if(mapp[x][y]>e)mapp[x][y]=mapp[y][x]=e;

//注意mapp[x][y]与mapp[y][x]都要改

}

mem(b);

up(1,n)

d[i]=mapp[1][i];//d[i]是i点到1点的距离

up(1,n-1)//迭代除了1点意外的所有点

{

minn=maxn;

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

if(b[j]==0&&d[j]<minn)//如果j没被迭代过

//并且到1点的距离更小

{

minn=d[j];

v=j;

}

b[v]=1;

for(int j=1;j<=n;j++)//迭代所有与v相连的点

if(b[j]==0&&d[j]>mapp[j][v]+d[v])

d[j]=mapp[j][v]+d[v];

}

cout<<d[n]<<"\n";

return 0;

}