E2-C题题解

作者：周其顺

题目描述：

C 相对位置

时间限制：1000ms 内存限制：65536kb

通过率：88/125 (70.40%) 正确率：88/824 (10.68%)

题目描述

在二维平面直角坐标系中，共有 n 个点，其中第一个点的位置在**原点**。给出 m 组两个点之间的相对位置。请你求出全部点的具体坐标。

输入

第一行有两个整数 $n(1 \le n \le 10^4)$ 和 $m(1 \le m \le 2 \times 10^5)$，表示点的个数和关系的个数。

接下来有 m 行，每一行有四个整数 $p1, p2, dx, dy$, 代表 $p1$ 和 $p2$ 两个点的位置关系。满足关系式：

$$dx = x_{p2} - x_{p1}, \ dy = y_{p2} - y_{p1}$$

保证 $1 \le p1, p2 \le n, 0 \le |dx|, |dy| \le 10^9$。且每个点坐标如果可以确定，只有唯一解。

解题思路：

题意即给出m组信息，求出尽可能多的点。一个最直接的思路：我们可以定义一个Info结构体，保存Δx、Δy、p1,p2四个信息.

```
typedef struct
{
    int link;
    int p1,p2;
    int dx;     //int 的数据范围是1e9
    int dy;
}Info;
```

在输入的过程中我们将获取的信息保存到一个Info类型的矩阵Map 中，

方便起见，存两份，即在Map[p1][p2]与Map[p2][p1]中均存一份，

但是Δx、Δy，p1,p2进行对应调整，比如我们在Map[1][1]里存一份p1=1,p2=2,dx=1,dy=2;那么就在Map[b][a]中存一份p1=2,p2=1,dx=-1;dy=-2，并把link置1;

等全部的信息输入完毕并且存到了这个Map矩阵里，我们就可以进行遍历计算了，

具体的计算过程如下所述

```
void dfs(int x,int n)
{
    for(int i=1;i<=n;i++)
    {
        if(Map[x][i].link==1&&!isKnown(i))//有连接并且下一个点未知
        {
            List[i].x=List[x].x+Map[x][i].dx;
            List[i].y=List[x].y+Map[x][i].dy;
            dfs(i,n);
        }
    }
}
```

```
bool isKnown(int sequence)//判断点是否已知
{
    if(sequence==1)//第一个点就是已知的
    {
        return true;
    }
    else
    {
      if(List[sequence].x==0&&List[sequence].y==0)//不是第一个点还全是0
      {
          return false;
      }
      else
      {
          return true;
      }
    }
}
```

进行按深度遍历，从第一个点（已知）开始，把与之连接的所有点给一道求出来。

但是这样的做法6个点4个AC, 一个MLE一个TLE, 不难分析出，MLE是显然的，对此的解决方法稍后再说；而TLE的解决方案是容易的，每次进行isKnwon的判断花费了大量时间，我们可以用一个isKnown数组去标记这个点是否已知，这就节省了很多时间。

```
void dfs(int x,int n)
{
    for(int i=0;i<Map[x].size();i++)//都是得和第一个相连才能继续开求的
    {
        if(!KnownList[Map[x][i].p2])//有连接并且下一个点未知
        {
            List[Map[x][i].p2].x=List[Map[x][i].p1].x+Map[x][i].dx;
            List[Map[x][i].p2].y=List[Map[x][i].p1].y+Map[x][i].dy;
            KnownList[Map[x][i].p2]= true;//标记已知
            dfs(Map[x][i].p2,n);
        }
    }
}
```

其实在上述的代码中也对空间的问题进行了解答了，作者对Map矩阵进行了改造成了Vector数组，

于是整个的矩阵变成了变长的结构，就不会再MLE

完整代码：

```
#include <stdio.h>
#include <vector>
using namespace std;
typedef struct
{
long long x;
long long y;
}Point;

typedef struct
{
int link;
int p1,p2;
int dx;    //int 的数据范围是1e9
int dy;
}Info;

void dfs(int x,int n);
vector<Info> Map[12000];
Point List[12000];
bool KnownList[12000];
int main()
{
int n,m; scanf("%d%d",&n,&m);
for(int i=1;i<=n;i++)
```

```
{
KnownList[i]= false;
}
KnownList[1]= true;

for(int i=0;i<m;i++)
{
int p1,p2;int dx,dy;
scanf("%d%d",&p1,&p2);
scanf("%d%d",&dx,&dy);
Info newInfo1,newInfo2;
newInfo1.dx=dx;newInfo1.dy=dy;
newInfo1.p1=p1;newInfo1.p2=p2;
newInfo2.dx=-dx;newInfo2.dy=-dy;
newInfo2.p1=p2; newInfo2.p2=p1;
Map[p1].push_back(newInfo1);
Map[p2].push_back(newInfo2);}

dfs(1,n);

for(int i=1;i<=n;i++)
{
    if(KnownList[i])
    {
        printf("%lld %lld\\n",List[i].x,List[i].y);
    }
    else
    {
        printf("undecidable\\n");
    }
}
return 0;
}

void dfs(int x,int n)
{
for(int i=0;i<Map[x].size();i++)
{
if(!KnownList[Map[x][i].p2])
{
List[Map[x][i].p2].x=List[Map[x][i].p1].x+Map[x][i].dx;
List[Map[x][i].p2].y=List[Map[x][i].p1].y+Map[x][i].dy;
KnownList[Map[x][i].p2]= true;//标记已知
dfs(Map[x][i].p2,n);
}
}
```