Dafang_Xu的博客

个人资料

Dafang_Xu

访问： 21355次
积分： 951
等级： BLOG 3
排名： 千里之外

原创： 73篇 转载： 13篇
译文： 1篇 评论： 2条

文章搜索

文章分类

c++基础学习笔记 (5)
POJ 线段树 (8)
POJ-ST算法 (1)
树状数组 (5)
排序算法 (1)
子序列 (3)
数学-傅里叶 (1)
图论-最短路径 (6)
图论-最小生成树 (3)
并查集 (3)
dfs (4)
图论-最大生成树 (2)
图论-差分约束 (3)
哈夫曼 (2)
bfs (2)
贪心 (1)
安卓开发 (3)
数论-排列组合 (2)
数论-约瑟夫环 (2)
Web前端 (3)
数据结构 (1)
数论-素数 (2)
vector (1)
cf水题 (5)
汇编 (0)
POJ水题 (2)
数论-中国剩余定理 (3)

[置顶] 以POJ1061青蛙的约会为例谈扩展欧几里得算法

标签： POJ1086 拓展欧几里得算法 青蛙的约会

2015-12-05 15:54 270人阅读 评论(0) 收藏 举报

分类： 数论-扩展欧几里得

目录(?) [+]

青蛙的约会
Time Limit: 1000MS Memory Limit: 10000K
Total Submissions: 100498 Accepted: 19304
Description
两 只青蛙在网上相识了，它们聊得很开心，于是觉得很有必要见一面。它们很高兴地发现它们住在同一条纬度线上，于是它们约定各自朝西跳，直到碰面为止。可是它 们出发之前忘记了一件很重要的事情，既没有问清楚对方的特征，也没有约定见面的具体位置。不过青蛙们都是很乐观的，它们觉得只要一直朝着某个方向跳下去， 总能碰到对方的。但是除非这两只青蛙在同一时间跳到同一点上，不然是永远都不可能碰面的。为了帮助这两只乐观的青蛙，你被要求写一个程序来判断这两只青蛙 是否能够碰面，会在什么时候碰面。
我们把这两只青蛙分别叫做青蛙A和青蛙B，并且规定纬度线上东经0度处为原点，由东往西为正方向，单位长度1米，这样我们就得到了一条首尾相接的 数轴。设青蛙A的出发点坐标是x，青蛙B的出发点坐标是y。青蛙A一次能跳m米，青蛙B一次能跳n米，两只青蛙跳一次所花费的时间相同。纬度线总长L米。 现在要你求出它们跳了几次以后才会碰面。
Input
输入只包括一行5个整数x，y，m，n，L，其中x≠y < 2000000000，0 < m、n < 2000000000，0 < L < 2100000000。
Output
输出碰面所需要的跳跃次数，如果永远不可能碰面则输出一行”Impossible”
Sample Input
1 2 3 4 5
Sample Output
4
Source
浙江

欧几里得**算法**可以求出a,b的最大公约数。gcd(a,b)=gcd(b,a%b),公式两边b都靠近等号一侧,因此很好记。

```
1 int gcd(int a,int b){return 0==b?a:gcd(b,a%b);}
```

为什么欧几里得算法是对的，记住一个式子：a=bq+r(假设a>b。若a小于b,gcd(a,b)=gcd(b,a)。因此a大于还是小于b无所谓)。因为a%m==0且b%m==0,所以必有r%m==0:a和b的约数必是r(即a%b)的约数。同时若r%n==0,b%n==0,必有a%n==0,因此b和r的约数必是a的约数。因此a、b约数中最大值必是b、r约数中的最大值。

**

扩展欧几里得算法exGcd(a,b,x,y)：求a,b的最大公约数同时可以求得方程组ax+by=gcd的一组特殊解(x,y)。

数论-扩展欧几里得 (1)
dp (2)
KMP (3)
hash (2)
位运算 (1)
php (1)
linux (2)
python编码 (1)

文章存档

2017年05月 (1)
2017年04月 (2)
2017年01月 (1)
2016年11月 (3)
2016年07月 (1)

展开

阅读排行

中国剩余定理算法详解(约 (1972)
Dreamweaver代码自动约 (1653)
哈夫曼树构造算法的正确 (1408)
生成全排列算法详解 (972)
Failed to create directory (907)
codeforces 344 B. Print (420)
Win10下 Gulp,Browser-s (323)
POJ1012Joseph解题报 (318)
Codeforces Round #332 (315)
约瑟夫环O(N)和O(M*N)的 (309)

评论排行

哈夫曼树构造算法的正确 (1)
中国剩余定理算法详解(约 (1)
归并排序&求逆序数 (0)
最好的树状数组学习资料 (0)
线段树（询问、插入）& (0)
线段树 查询删除 后序遍历 (0)
线段树 ST算法 RMQ poj (0)
POJ2528解题报告，区间 (0)
Python write unicode int (0)
线段树 区间更新 访问PO (0)

推荐文章

* 5月书讯：流畅的Python，终于等到你！
* JSON最佳实践
* InfiniBand技术和协议架构分析
* Android 中解决破解签名验证之后导致的登录授权失效问题
* 《Real-Time Rendering 3rd》提炼总结——图形渲染与视觉外观
* CSDN日报20170607 ——《别混淆你想要什么和能否实现》

最新评论

中国剩余定理算法详解(余数互质
Lannister_Stark: 写得真好~!

关闭

**从式子ax+by=gcd可以看出必须先求出gcd,才能求出解。而递归中的却如此。

1、假设已经找到gcd,那么y=0,x=gcd，显然是该方程的解。而且在每一层递归函数exGcd(a,b,x,y)里都有一个解(x,y)。

2、在递归回溯过程中，假设在第n+1层exGcd(a(n+1),b(n+1),x,y)找到了(x0,y0)是方程的解。那么有a(n+1)*x0+b(n+1)*y0=c(1式)。

3、根据递归过程，在第n层调用了exGcd(bn,an%bn,x,y)进入第n+1层。因此在第n+1层的解满足a(n+1)=bn,b(n+1)=an%bn，即(1)式等价于：bn*x0+(an%bn)%y0=c(2式)。又因为an%bn=an-bn*(an/bn)。所以(2式子)等价于bn*x0+(an-bn*(an/bn))*y0=bn(x0-(an/bn)*y0)+an*y0=c(3式)。

3、假设回溯到第n层时，解为(x1,y1)，an*x1+bn*y1=c(4式)。对比(3式)和(4式)发现x1=y0,y1=x0-y0*(an/bn)。所以第n层的解可以通过第n+1层的解递推出来。在最深层有解x=gcd,y=0,因此层层回溯，每层都会有一个关于当前层exGcd(an,bn,x,y)的解(xn,yn)使得an*xn+bn*yn=gcd(an,bn)。因此回溯到第一层时就得到了关于第一层的解。

代码如下:

```
long long exGcd(long long a, long long b, long long &x, long long &y){
    if(b==0){
        x=1;
        y=0;
        return a;
    }
    long long g=exGcd(b,a%b,x,y);
    long long temp=x;
    x=y;
    y=temp-(a/b)*y;//注意此处不能写成y*a/b!
    return g;
}
```

扩展欧几里得算法应用之一就是：求二元一次方程组的解(x,y), a*x+b*y=c。

注意，这里右边是c，不是gcd(a,b)。假设其中x是我们关心的，所以研究x的解的情况。计算步骤：

1、用扩展欧几里得算法求解方程：a*x+b*y=gcd(a,b). 可以同时得到gcd以及 一组特解(x0,y0)

2、如果c%gcd!=0,那么根据式子a*x+b*y=c可知其无整数解（因为a,b都可以提一个公因数gcd,但是c不能提因子gcd，等号左右两边不等价）。

3、那么a*x+b*y=c的x的特殊解x1等于a*x+b*y=gcd(a,b)的特殊解x0乘以c/gcd: x1=x0*c/gcd。4、这一步的构造很关键，因为引入了整数t，t不同解不同，这就是为什么会有无数解。因为x1*a+y1*b=x1*a+y1*b+(t*a*b/gcd-t*a*b/gcd)=a(x1+t*b/gcd)+b(y1-t*a/gcd)=c，所以方程a*x+b*y=c的x的通解就是x=x1+t*b/gcd=x0*c/gcd+t*b/gcd。

5、x有无数种情况，我们关心特殊情况，比如在实际问题中经常需要找大于0且最小的x的值。如何寻找？找x即找t，下面对t进行分析。 x=x0*c/gcd+t*b/gcd。这里的x肯定大于0，减去t*b/gcd剩下的x0*c/gcd也大于0，所以直接对x=x0*c/gcd进行分析。不妨让t=x/(b/gcd)得到x中有t个b/gcd,然后在x中减去这t个b/gcd：t=x/(b/gcd),x=x-t*(b/gcd)。此时得到的x可能小于等于0，因此要做判断，x小于等于0时加上b/gcd即可。

首先，此题有两个变量，跳的次数P和跳的圈数Q。需要求次数的最小值。当然联想到拓展欧几里得算法的应用：求解二元一次线性方程组。两只青蛙相遇的条件是：（x+mP） mod L =(y+nP) mod L=0,但是这样写不是方程式的形式，所以改成等价的方程的形式：（x+mP)-(y+nP) =Q L,等价于x-y+P(m-n)=QL,x-y=(n-m)P+QL。令c=x-y,a=n-m,b=L,则aP+bQ=c,这就是标准的二元一次方程组，两个未知量求其中一个未知量（跳到次数P）的最小值。

```
#include <iostream>
#include <cstring>
#include<cstdio>
#include <algorithm>
using namespace std;
long long exGcd(long long  a,long long b,long long &x,long long &y){
    if(b==0){
        x=1;
        y=0;
        return a;
    }
    long long d=exGcd(b,a%b,x,y);
```

哈夫曼树构造算法的正确性证明
十五_cray: 因此每次生成的节点取最小值，即可保证，总的和值是最小的。-----这一点是没有依据的。因为当前合并过...

```
    long long tmpX=x;
    x=y;
    y=tmpX-y*(a/b);//不能写成y*a/b
    return d;
}
int main()
{
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);
    long long ansX,x,y,n,m,L,x0,y0,a,b,c,gcd;
    while(scanf("%lld%lld%lld%lld%lld",&x,&y,&m,&n,&L)==5){
        c=x-y;
        a=n-m;
        b=L;
        gcd=exGcd(a,b,x0,y0);
        if(c%gcd!=0) cout<<"Impossible"<<endl;
        else{
        ansX=x0*(c/gcd);
        long long t=ansX/(b/gcd);
        ansX=ansX-t*(b/gcd);
        if(ansX<0) ansX+=b/gcd;
        cout<<ansX<<endl;
        }

    }
    return 0;
}
```

欢迎留言，积极讨论，一起进步！

顶 0　　踩 0

上一篇 全排列POJ1256Anagram

下一篇 error: ISO C++ forbids declaration of `FILE' with no type

相关文章推荐

参考知识库

算法与数据结构知识库

16778 关注 | 2320 收录

猜你在找

查看评论

暂无评论

发表评论

用户名： nobleman__

评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap