It's So Easy to Cut a Rectangle 题解

考虑按题意模拟的过程，假如a是$depth(a)=k$的一个解，那么它会产生哪些$depth(a)=k+1$的解?

设产生的解是b, 则切下1×1之后剩下的$(b-1)\times 1$, 长边和短边之比是$a:1$. 若$(b-1)$是长边，则$\frac{b-1}{1}=\frac{a}{1}$, 解得$b=a+1$. 若1是长边，则$\frac{b-1}{1}=\frac{1}{a}$, 得$b=\frac{1}{a}+1$. 所以实际上这个过程就是$depth(a)=k$的每一个解会产生$depth(a)=k+1$的两个解，$a+1$和$\frac{1}{a}+1$.

例如: $depth$为1的数有2

$depth$为2的数有$\frac{3}{2},3$

$depth$为3的数有$\frac{4}{3},\frac{5}{3},\frac{5}{2},4$

$depth$为4的数有$\frac{5}{4},\frac{7}{5},\frac{8}{5},\frac{7}{4},\frac{7}{3},\frac{8}{3},\frac{7}{2},5$

.....

容易证明，$depth(a)=k$共有2^{k-1}个解。

对于第一问，直接用辗转相除加速按题意模拟: 显然$depth(\frac{a}{b})=depth(\frac{a-b}{b})+1\ (a>b)$, 故$depth(\frac{a}{b})=depth(\frac{b}{a\mod b})+[\frac{a}{b}]$.

对于第二问，我们可以发现一些性质。

首先，假设每层所有解从小到大排序，那么下一层第一个和最后一个解由本层最后一个解产生，下一层第二个和倒数第二个解由本层倒数第二个解产生.....下一层最中间两个解由本层第一个解产生。因此我们可以推出如下性质: **如果把一个数$\frac{a}{b}$变为$\frac{a-b}{b}$, 则在该层中比这个数大的数的个数不变。**

然后，我们可以这样考虑: 把每一层的所有解变为其倒数，然后按照倒序写在前面，这样第k层就有2^k个数。例如第3层共8个数，$\frac{1}{4},\frac{2}{5},\frac{3}{5},\frac{3}{4},\frac{4}{3},\frac{5}{3},\frac{5}{2},5$. 这样我们把$\frac{a}{b}$变成$\frac{a\mod b}{b}$的时候，在该层中比这个数大的数依然不变。而$\frac{a\mod b}{b}$是一个真分数，它和它的倒数在同一层中处于关于中点对称的位置，即比它大的数的个数就是比它的倒数小的数的个数。然后迭代即可。

注：题解只提供一种和标程相吻合且比较好写的做法。有一些做法比较好想但是难写，和本做法大同小异。例如：递归时不递归到$\frac{a\mod b}{b}$而是递归到$\frac{a\mod b+b}{b}$, 然后再将其变为$\frac{a\mod b+b}{a\mod b}$, 相当于跳过了真分数的那步转化，本质是一样的。

代码

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<utility>
#define llong long long
using namespace std;

const int P = 1e9+7;

struct Fraction
{
    llong a,b; // a/b
```

```
    Fraction() {a = b = 1ll;}
    Fraction(llong _a,llong _b) {a = _a,b = _b;}
};

llong quickpow(llong x,llong y)
{
    y %= (P-1);
    llong cur = x,ret = 1ll;
    for(int i=0; y; i++)
    {
        if(y&(1ll<<i)) {y-=(1ll<<i); ret = ret*cur%P;}
        cur = cur*cur%P;
    }
    return ret;
}

pair<llong,llong> f(Fraction x)
{
    pair<llong,llong> ret;
    if(x.b==1) ret = make_pair(x.a-1,quickpow(2ll,x.a-1));
    else
    {
        pair<llong,llong> tmp = f(Fraction(x.b,x.a%x.b));
        ret.first = tmp.first+(x.a/x.b);
        ret.second = (quickpow(2ll,ret.first)-tmp.second+1ll+P)%P;
    }
    return ret;
}

int main()
{
    Fraction x; scanf("%I64d%I64d",&x.a,&x.b);
    pair<llong,llong> ans = f(x);
    ans.second = (ans.second-quickpow(2ll,ans.first-1)+P)%P;
    printf("%I64d %I64d\n",ans.first,ans.second);
    return 0;
}
```