ЗМС`09-А3

Или защо грийдито работи... и решение с теория на числата

Спомената задача е интересна с това, че за оптималното ѝ решаване трябва да се използва грийди. След състезанието така и не стана ясно защо грийдито работи. Тук показвам доказателство.

Ето споменатата задача:

Условие:

Задача А3. Генератор на обикновени дроби

Петър е ученик в шести клас. Задачите от групи E и D на всички състезания по информатика вече не са никакъв проблем за него. Ето защо Петър започна сам да си измисля задачи, че даже и да ги решава. Най-новото му творение е програма за генериране на обикновени дроби. Ето как работи тя: Петър въвежда несъкратима обикновена дроб $x = \frac{p}{q}$(p и q са положителни цели числа). Генерирането на нови дроби се извършва чрез командите **L** и **R**. При команда **L** програмата извежда несъкратима дроб, равна на $2x + 1$, а при **R** – несъкратима дроб, равна на $\frac{x}{x+2}$. При всяко следващо въвеждане на команка **L** или **R**, за генерирането на нова дроб програмата използва последната получена. Например, след въвеждане на $\frac{2}{3}$, **R**, **L**, **L**, **R**, тя извежда последователно дробите $\frac{1}{4}, \frac{3}{2}, \frac{4}{1}, \frac{2}{3}$. Сега Петър иска да разбере колко пъти най-малко трябва да въведе **L** или **R**, за да може програмата му да генерира дробта, която е въвел. Помогнете на Петър, като напишете програма **gen**, която решава новата му задача.

Ограничения:

$1 \leq p \leq 10000000, 1 \leq q \leq 10000000.$

Вход

На един ред на стандартния вход са зададени две цели положителни числа – числителят p и знаменателят q на въведената от Петър несъкратима дроб.

Изход

На един ред на стандартния изход програмата трябва да изведе най-малкия брой команди, необходими за да се генерира зададената дроб. Ако въведената от Петър дроб не може да бъде генерирана, програмата трябва да изведе 0.

Пример

Вход

2 3

Изход

4

Анализ:

Да разгледаме двете операции и ефекта които имат върху числителя и знаменателя на дроб $\frac{p}{q}$:

$$L\left(\frac{p}{q}\right) = 2\left(\frac{p}{q}\right) + 1 = \frac{2p+q}{q};$$

$$R\left(\frac{p}{q}\right) = \frac{\left(\frac{p}{q}\right)}{\left(\frac{p}{q}\right)+2} = \frac{p}{2q+p};$$

Вижда се, че двете операции са симетрични относно размяна на числителя и знаменателя. С други думи:

$$L(x) = R\left(\frac{1}{x}\right), R(x) = L\left(\frac{1}{x}\right).$$

Сега трябва да се опитаме да възстановим траекторията. Тук може да идеята на грийдито да се обясни по няколко начина, ето как си го обяснявам аз: Вместо да се опитваме да вървим "напред„ и към началната дроб да прилагаме двете операции по някакъв начин, да се опитаме да тръгнем „назад" и да си поставим следния въпрос: ако имаме фиксирана дроб $\frac{p}{q}$, от кои рационални числа след еднократно прилагане на някоя от двете операции, мжем да получим $\frac{p}{q}$. С други думи, ние търсим решенията на:

$$L(x) = \frac{p}{q}, R(y) = \frac{p}{q}.$$

Тривиално, дробта $\frac{1}{1}$ няма първообраз, т.е. тя не може да бъде получена от никоя дроб след прилагане на една от горепосочените операции.

$$L(x) = \frac{p}{q},$$

$$L\left(\frac{u}{v}\right) = \frac{p}{q},$$

$$\frac{2u+v}{v} = \frac{p}{q},$$

$$v = q, u = \frac{p-q}{2},$$

$$L^{-1}\left(\frac{p}{q}\right) = \frac{p-q}{2q}.$$

Аналогично:

$$R^{-1}\left(\frac{p}{q}\right) = \frac{2p}{q-p}.$$

Важното наблюдение е, че:

$$L(x) > 1, R(x) < 1$$

за всяко $x \in \mathbb{Q}^+$. Това означава, че произволна дроб има най-много 1 прообраз и че ние можем да го намерим, като проверим дали дробта е по-голяма или по-малка от 1. Тривиално, лесно можем да извършим обратните операции и да се върнем „с една стъпка назад“. Можем да продължим това, докато не стигнем началната дроб, или докато не стигнем 1, или докато не зациклим. За реализация на това единия вариант е да търсим цикъл в последователност, за което има хитър оптимален алгоритъм. Грийдито се базира на това наблюдение и на още няколко евристични наблюдения, като това, че ако числителят и знаменателят имат различна четност, то винаги се стига до решение, и наблюдението, че ако числителят и знаменателят са нечетни, трябва да изведем 0 в случаите когато стигнем 1 или някоя дроб от първия вид. Това написах и аз на състезанието, но реших да не го оптимизирам допълнително, което след това ми коства 1(едно) място ☹.

Примерен код – C++

```
/*
TASK: gen
LANG: C++
*/
//by krassi_holmz, 2009
#include <iostream>
using namespace std;

typedef long long unsigned num;

int main(){
    num p, q, p0, q0, i;
    bool odod;
    cin >> p0 >> q0;
    odod = (p0&1) && (q0&1);
    if(p0 == 1 && q0 == 1){
        cout << 0 << endl;
        goto LONDON;
    }
    p = p0;
    q = q0;
    i = 1;

    //iter
    if((p&1) && (q&1)){
```

```
        //if both odd, then
        if(p > q){
            p-=q;
            p>>=1;
        }else{
            q-=p;
            q>>=1;
        }
    }else{
        if(p > q){
            p-=q;
            q<<=1;
        }else{
            q-=p;
            p<<=1;
        }
    }

    for(;;i++){
        //cout << p << ' ' << q << endl;
        //system("pause");
        if(odod && (((p&1)==0) || ((q&1)==0))){
            cout << 0 << endl;
            goto LONDON;
        }
        if(p == q){
            cout << 0 << endl;
            break;
        }else if(p == p0 && q == q0){
            cout << i << endl;
            break;
        }else{
                //iter
        if((p&1) && (q&1)){
            //if both odd, then
            if(p > q){
                p-=q;
                p>>=1;
            }else{
                q-=p;
                q>>=1;
            }
        }else{
            if(p > q){
                p-=q;
                q<<=1;
            }else{
                q-=p;
                p<<=1;
            }
        }
        }
    }
    LONDON:
    //system("pause");

    return 0;
```

```
}
```

Примерен код – Mathematica

```
In[52]:= f[Rational[p_, q_]] := If[p > q, (p - q)/(2 q), (2 p)/(q - p)];
         f[n_] := (n - 1)/2;
         f[0] := 0;
         FixedPointList[f, 1/4, 10]
Out[55]= {1/4, 2/3, 4, 3/2, 1/4, 2/3, 4, 3/2, 1/4, 2/3, 4}
```

Нека оттам откъдето сме стигнали, да погледнем по-математически на въпроса – за да разбулим тайната на работещото грийди ще използваме средства от теорията на числата. След разглеждане на няколко редици, генерирани от грийдито се вижда, че генерираните дроби притежават инвариант – сумата на числителя и знаменателя не се променя(тук е важно да оставим дробта в несъкратима форма). Това означава, че ако фиксираме сумата на числителя и знаменателя в началото, то цялата редица се определя от числителите на редицата от генерираните дроби! Но това е много добра новина, защото така можем да определим какво става, като променяме само по едно число – числителят! Оказва се, че промяната на числителя на всяка стъпка е изключително проста:

$$p \to (2p)\%n$$

Това строго се доказва, като се разгледат два случая за числителя и се разпишат съответните прообразувания.

Нека имаме дробта $h = \frac{p}{n-p}, p < \frac{n}{2}, h < 1$. Трябва да приложим обратната на $L(x)$ карта:

$$L^{-1}\left(\frac{p}{n-p}\right) = \frac{p-(n-p)}{2(n-p)} = \frac{2p-n}{2n-2p}.$$

Имаме:

$$(2p-n)+(2n-2p) = n.$$

$$2p-n \equiv 2p\ (mod\ n).$$

Нека сега $p > \frac{n}{2}, h > 1$. Сега трябва да приложим обратната на $R(x)$ карта:

$$R^{-1}\left(\frac{p}{n-p}\right) = \frac{2p}{(n-p)-p},$$

$$2p + \big((n-p)-p\big) = n.$$

$$2p \equiv 2p(mod\ n).$$

Така стигнахме до следното:

Следствие:

Числителят на дробта, получена след k стъпки на грийдито с начална дроб $\frac{p}{q}$, е точно $\big(2^k p\big)\%(p+q)$.

Така доказахме, че всъщност зад двете операции се крие фундаментална теория на числата. Сега въпроса от задачата, преведен на езика на теорията на числата, се свежда до следното:

Намерете показателя на 2 по модул $p+q$.

Примерен код – Mathematica

```
In[69]:= Solution[p_, q_] := MultiplicativeOrder[2, p + q] /.
           MultiplicativeOrder[a_, b_] → 0;

In[70]:= Solution[2, 3]

Out[70]= 4

In[74]:= Solution[1, 4]

Out[74]= 4

In[75]:= Solution[1, 5]

Out[75]= 0
```

Примерен код – C++

```
//gen.cpp
//Solution by number theory
//by krassi_holmz, 2009
#include <iostream>
using namespace std;

typedef long long unsigned num;
```

```
int main(){
	num i, p, n;
	cin >> p >> n;
	n += p;
	if(n%2 == 0){
		cout << 0 << endl;
	}else{
		p = 2;
		i = 1;
		while(p != 1){
			p *= 2;
			p %= n;
			i++;
		}
		cout << i << endl;
	}
	return 0;
}
```