SỞ GIÁO DỤC – ĐÀO TẠO
THÁI BÌNH

TRÖÔØNG THPT CHUYEÂN

═══***═══

CHUYÊN ĐỀ

" SỐ HỌC"

Giáo viên: ***Nguyễn Tiến Thành***

Đơn vị công tác: Trường THPT Chuyên Thái Bình

Năm học: 2019 - 2020

═══════γη═══════

CHUYÊN ĐỀ CÁC BÀI TOÁN SỐ HỌC

I. MỞ ĐẦU

Trong nhiều bài toán Tin học, việc vận dụng kiến thức số học giúp đưa ra được những thuật toán tối ưu hơn. Mặt khác, những bài toán Tin học có sự vận dụng kiến thức số học cơ bản, đòi hỏi sự cài đặt không quá phức tạp và các em có nền tảng Toán học tốt có thể dễ dàng suy luận được. Từ đó kích thích niềm yêu thích của các em trong việc lập trình. Các thuật toán số học trong tin học là nội dung kiến thức khá quan trọng và được sử dụng nhiều trong thiết kế thuật toán. Tuy nhiên, trong quá trình giảng dạy tôi thấy học sinh vẫn còn khó khăn trong việc phân tích bài toán để có thể áp dụng được thuật toán và cài đặt giải bài toán. Vì vậy tôi chọn chuyên đề này để giúp học sinh có được hệ thống kiến thức cơ bản về toán học để giúp các em dễ dàng hơn trong giải quyết các bài toán cụ thể nhất là đối các bài toán về số học.

Nội dung của chuyên đề được chia thành ba phần: Mở đầu, Nội dung và Kết luận. Trong phần Nội dung (và cũng là phần chính của chuyên đề) chúng tôi đề cập tới các vấn đề chính: Số nguyên tố, Ước số, Bội số, Số Fibonacci, Số Catalan, Xử lí số nguyên lớn, ... Do chuyên đề này là các bài toán số học nên kiến thức cơ sở sẽ được ứng dụng trong từng bài tập cụ thể.

Về mặt lí thuyết số học có rất nhiều tài liệu đã trình bày, ngay cả trong cuốn tài liệu giáo khoa chuyên Tin quyển 1 đã hệ thống rất cụ thể. Chuyên đề là sự sưu tầm, chọn lọc, sắp xếp và hệ thống những vấn đề cơ bản của các bài toán số học theo một mạch kiến thức nhất định dựa trên một số nguồn tài liệu đã có. Cùng với đó, chúng tôi có đưa ra những phân tích, đánh giá để làm sáng tỏ cho mỗi vấn đề được đề cập tới. Với cách tiếp cận mở như vậy, hy vọng chuyên đề này sẽ giúp các em học sinh có được một hệ thống những kiến thức cần thiết, các thầy cô giáo có một chuyên đề chuyên môn bổ ích và thiết thực.

II. NỘI DUNG

1. Số nguyên tố

1.1. Định nghĩa

Một số tự nhiên *p (p>1)* là số nguyên tố nếu *p* có đúng hai ước số là 1 và *p*.

Ví dụ các số nguyên tố: 2, 3, 5, 7, 11, 13, 17, 19, 23, …

1.2. Kiểm tra tính nguyên tố theo định nghĩa.

Ý tưởng chính để kiểm tra số nguyên dương N (N>1) có là số nguyên tố hay không, ta kiểm tra xem có tồn tại số nguyên k $(2 \le k \le \sqrt{N})$ mà k là ước của N (N chia hết cho k) thì N không phải là số nguyên tố, ngược lại N là số nguyên tố.

```
bool IsPrime(int N)
{
   if (N < 2) return flase;
   for(int i = 2; i < sqrt(N); i++)
   if(N%i==0) return flase;
   return true;
}
```

Tuy nhiên ta thấy cách này không hiệu quả khì thời gian kiểm tra lâu. Cải tiến kiểm tra tính nguyên tố của số N bằng cách kiểm tra xem N có chia hết cho số 2, số 3 và các số có dạng $6k \pm 1$ trong đoạn $\left[5, \sqrt{N}\right]$.

```
bool  IsPrime (int N)
{
     if (N==2 || N==3) return true;
     if (N==1 || N%2==0 || N%3==0) return false;
            int k=-1;
     while (k<=int(sqrt(N)))
    {
             k+=6;
       if (N%k==0 || N% (k+2)==0) break;
    }
    return k>int(sqrt(N));
}
```

1.3. Kiểm tra số nguyên tố theo xác suất

Các khái niệm, tính chất của đồng dư thức, và định lý cần nhớ như định lý Ferma. Ở đây tôi đề cập đến định lý Ferma nhỏ và tổng quát hóa của định lý Ferma:

Định lý Ferma nhỏ

Nếu p là một số nguyên tố, thì với số nguyên a bất kỳ, a^p-a sẽ chia hết cho p. Nghĩa là $a^p \equiv a(\mathrm{mod}\, p)$

Một dạng tổng quát của định lý này là: nếu p là số nguyên tố và m và n là các số nguyên dương thỏa mãn $m \equiv n(\mathrm{mod}\, p-1)$, thì $\forall a \in Z : a^m \equiv a^n(\mathrm{mod}\, p)$.

Định lý Fermat còn được tổng quát hóa bởi Định lý Euler: với modulo n bất kỳ và số nguyên a bất kỳ là số nguyên tố cùng nhau với n, ta có: $a^{\varphi(n)} \equiv 1(\mathrm{mod}\, n)$.

Trong đó $\varphi(n)$ là kí hiệu của hàm phi Euler đếm số các số nguyên giữa 1 và n nguyên tố cùng nhau với n. Đây là tổng quát hóa của định lý nhỏ Fermat vì nếu $n=p$ là số nguyên tố thì $\varphi(p) = n-1$.

1.4. Liệt kê các số nguyên tố trong đoạn [*1,N*].

Cách 1. Thử lần lượt các số m trong đoạn [*1,N*], rồi kiểm tra tính nguyên tố của m.

```
void OutPrime(int N)
{
   for(int i = 2; i <= N; i++)
      if(IsPrime(i))
         printf("%d\t", i);
}
```

Ta thấy: Cách này đơn giản nhưng chạy chậm, để cải tiến có thể sử dụng các tính chất của số nguyên tố để loại trước những số không phải là số nguyên tố và không cần phải kiểm tra trước các số này.

Cách 2. Sử dụng sàng số nguyên tố sàng Eratosthene.

Giả sử tất cả đều là số nguyên tố, trước tiên xóa bỏ số 1 ra khỏi tập các số nguyên tố. Số tiếp theo số 1 là số 2, là số nguyên tố, xóa tất cả các bội số của 2 ra khỏi bảng, xét lên 3, loại tất cả các bội số của 3,… thuật toán tiếp tục cho đến khi gặp số nguyên tố lớn hơn $\sqrt{n}$ thì dừng lại. Kết thúc quá trình các số chưa bị loại là số nguyên tố.

```
void Eratosthene(int N)
{
   int a[1000] = {0}; //Tạo mảng và gán tất cả bằng 0
   for(int i = 2; i*i <= N;i++)
      if(!a[i]) //Nếu là số nguyên tố
         //Duyệt các phần tử là bội số của i
         for(int j = i*i; j <= N; j+=i)
            a[j]=1; //Đánh dấu các phần tử là bội số.
   //In các phần tử là số nguyên tố ra màn hình
   for (int i=2;i<=N;i++)
      if(!a[i]) printf("%d ",i);
}
```

2. ƯỚC SỐ, BỘI SỐ

2.1. Số các ước của một số

Giả sử N được phân tích thành thừa số nguyên tố như sau:

$$N = a^i \times b^j \times ... \times c^k$$

Ước số của N có dạng: $a^p \times b^q \times ... \times c^r$ trong đó $0 \le p \le i, 0 \le q \le j, 0 \le r \le k$.

Do đó: Số các ước của N là: $(i+1) \times (j+1) \times ... \times (k+1)$.

Ví dụ: $N = 100 = 2^2 \times 5^2$, số các ước của 100 là: $(2+1)(2+1) = 9$ (các ước của số đó là: 1, 2, 3, 4, 5, 10, 20, 50, 100).

2.2. Tổng các ước của một số

$$N = a^i \times b^j \times ... \times c^k$$

Đặt $N_1 = b^j \times ... \times c^k$

Gọi $F(t)$ là tổng các ước của t, ta có:

$$F(N) = F(N_1 + a \times F(N_1) + ... + a^i \times F(N_1)$$

$$= (1 + a + ... + a^i) \times F(N_1) = \frac{a^{i+1} - 1}{a - 1} \times F(N_1)$$

$$= \frac{(a^{i+1}-1)}{a-1} \times \frac{(b^{j+1}-1)}{b-1} \times \ldots \times \frac{(c^{k+1}-1)}{c-1}$$

Ví dụ: Tổng các ước của 24 là: $\frac{(2^{3+1}-1)}{2-1} \times \frac{(3^{1+1}-1)}{3-1} = 60$

2.3. Ước chung lớn nhất của hai số

Ước số chung lớn nhất (USCLN) của hai số được tính theo thuật toán Euclid $USCLN(a,b) = USCLN(b,(a \bmod b))$.

```
int gcd(int a, int b)
{
    int tmp;
    while(b != 0)
    {
        tmp = a % b;
        a = b;
        b = tmp;
    }
    return a;
}
```

Chú ý: Để có thể sử dụng hàm tìm UCLN trong C++ ta cần thêm thư viện *algorithm.*

Ví dụ:

```
int main(){
    int a = 5, b = 9;
    printf("\ngcd(%d, %d) = %d", a, b, std::__gcd(a,b));
}
```

2.4. Bội chung nhỏ nhất của hai số

Bội số chung nhỏ nhất (BSCNN) của hai số được tính theo công thức:

$$BSCNN(a,b) = \frac{a \times b}{USCLN(a,b)} = \frac{a}{USCLN(a,b)} \times b$$

3. Dãy số Fibonacci

Dãy số Fibonacci được xác định bởi các công thức sau:

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \text{ với } n \geq 2 \end{cases}$$

Một số phần tử đầu tiên của dãy số Fibonacci:

n	0	1	2	3	4	5	6	…
$Fibonacci_n$	0	1	1	2	3	5	8	…

Số Fibonacci là đáp án của các bài toán:

a) Bài toán cổ về sự sinh sản của các cặp thỏ với các giả thiết như sau:

- Các con thỏ không bao giờ chết;
- Hai tháng sau khi ra đời, mỗi cặp thỏ mới sẽ sinh ra một cặp thỏ con (một đực, một cái);
- Khi đã sinh con rồi thì cứ mỗi tháng tiếp theo chúng lại sinh được một cặp con mới.

Giả sử từ đầu tháng 1 có một cặp mới thì đến giữa tháng thứ n sẽ có bao nhiêu cặp?

b) Đếm số cách xếp *n-1* quân domino kích thước 2×1 phủ kín bảng có kích thước $2 \times (n-1)$.

Hàm tính số Fibonaci thứ *n* bằng phương pháp lặp sử dụng công thức

$$F_n = F_{n-1} + F_{n-2} \text{ với } n \geq 2 \text{ và } F_0 = 0,\ F_1 = 1.$$

```
int fibo (int n)
{
    int f1, f2, fi;
    if (n<=1) return n;
    f2=0; f1=1;
    for (int i=2; i<=n; i++)
    {
        fi=f1+f2;         f2=f1;         f1=fi;
    }
}
```

Sử dụng phương pháp đệ quy:

```
int fibonaci(int n)
{
     if(n==0 || n==1)
          return 1;
     else
          return (fibonaci(n-2)+fibonaci(n-1));
}
```

4. Dãy số CATALAN

Số Catalan được xác định bởi công thức sau:

$$Catalan_n = \frac{1}{n+1} C_{2n}^{n} = \frac{(2n)!}{(n+1)!n!} \text{ với } n \geq 0.$$

Một số phần tử đầu tiên của dãy số Catalan là:

n	0	1	2	3	4	5	6	…
$Catalan_n$	1	1	2	5	14	42	132	…

Số Catalan là đáp án của các bài toán:

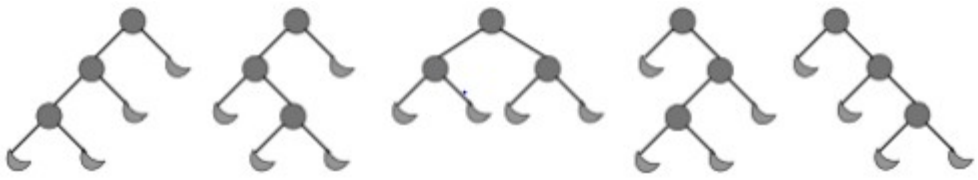
a) Có bao nhiêu cách khác nhau đặt n dấu ngoặc mở và n dấu ngoặc đóng đúng đắn?

Ví dụ: n=3 ta có 5 cách sau*:*

((())), (()()), (())(), ()(()), ()()()

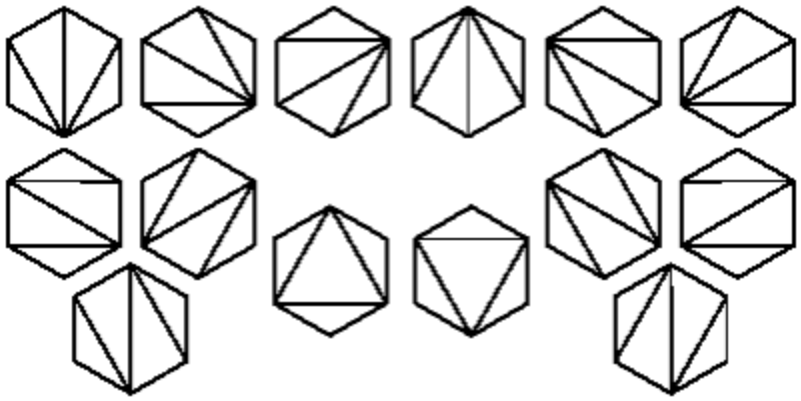
b) Có bao nhiêu cây nhị phân khác nhau có đúng n+1 lá?

Ví dụ: n=3



c) Cho một đa giác lồi (n+2) đỉnh, ta chia thành các tam giác bằng nhau vẽ các đường chéo không cắt nhau trong đa giác. Hỏi có bao nhiêu cách chia như vậy?

Ví dụ: n=4



5. Xử lí số nguyên lớn

5.1. Cộng 2 số nguyên lớn

Phân tích thuật toán

- Bước 1: Chuẩn hóa hai xâu a, b để có độ dài bằng nhau. Nếu xâu nào có độ dài ngắn hơn thì thêm các '0' vào đầu xâu đó.

- Bước 2: Duyệt từ cuối hai xâu về đầu xâu:

+ Tạo xâu kết quả c=a;

+ Tách từng phần tử của hai xâu chuyển sang kiểu số;

+ Tính tổng:

tổng = số 1 + số 2 + nhớ (ban đầu nhớ bằng 0);

nhớ = tổng / 10;

tổng = tổng % 10;

+ Chuyển đổi giá trị ***tổng*** tính được sang ký tự rồi gán vào xâu kết quả.

+ Lưu ý cộng thêm giá trị nhớ lần cuối nếu ***nhớ khác '0'***.

Chương trình tham khảo

```
string Congxau(string a, string b)
{
    string c;
```

```
    long n1=a.length(),n2=b.length(),i,nho=0,Tong;
    if(n1>n2) b.insert(0,n1-n2,'0');
    if(n1<n2) a.insert(0,n2-n1,'0');
    c=a;
    for(i=a.length()-1;i>=0;i--)
    {
        Tong=(a[i]-48)+(b[i]-48)+nho;
        nho=Tong/10;
        Tong=Tong%10;
        c[i]=char(Tong+48);
    }
    if(nho>0)c=char(nho+48)+c;
    return c;
}
```

5.2. Trừ 2 số nguyên lớn (Trừ số lớn cho số bé)

Phân tích thuật toán

- Bước 1: Chuẩn hóa hai xâu a, b để có độ dài bằng nhau. Nếu xâu nào có độ dài ngắn hơn thì thêm các '0' vào đầu xâu đó.

- Bước 2: Duyệt từ cuối hai xâu về đầu xâu:

+ Tạo xâu kết quả c=a;

+ Tách từng phần tử của hai xâu chuyển sang kiểu số;

+ Tính hiệu:

hiệu = số 1 - số 2 - mượn (ban đầu mượn bằng 0);

Nếu hiệu<0 thì {hiệu=hiệu+10; mượn=1;}

Nếu hiệu>0 thì mượn =0;

+ Chuyển đổi giá trị ***hiệu*** tính được sang ký tự rồi gán vào xâu kết quả.

+ Xử lý xâu kết quả nếu xâu có độ dài lớn hơn 1 mà phần tử đầu tiên của mảng xâu là ***'0'***.

Chương trình tham khảo

```
string Truxau(string a, string b)
{
    string c="";
    long n1=a.length(),n2=b.length(),i,Muon=0,Hieu;
    if(n1>n2) b.insert(0,n1-n2,'0');
    for(i=a.length()-1;i>=0;i--)
    {
        Hieu=(a[i]-48)-(b[i]-48)-Muon;
        if(Hieu<0){Hieu+=10;Muon=1;}else Muon=0;
        c=char(Hieu+48)+c;
    }
    while(c.length()>1&&c[0]=='0') c.erase(0,1);
    return c;
}
```

5.3. Nhân một số nguyên lớn với một nguyên số nhỏ

Phân tích thuật toán

- Bước 1: Duyệt từ cuối xâu số lớn về đầu xâu

- Bước 2:

+ Tách từng phần tử của xâu chuyển sang kiểu số và tính tích:

tích = số nhỏ * tg + nhớ (tg là số được tách từ xâu số lớn);

nhớ = tích /10;

Tích = tích % 10;

+ Chuyển đổi giá trị ***tích*** tính được sang ký tự rồi gán vào xâu kết quả.

+ Lưu ý cộng thêm giá trị nhớ lần cuối nếu ***nhớ khác '0'***.

Chương trình tham khảo

```
string Nhan1so(string a, int k)
{
    string b;
    long i,Nho=0,Tich;
    for(i=a.length()-1;i>=0;i--)
    {
        Tich=Nho+(a[i]-48)*k;
        Nho=Tich/10;
        Tich=Tich%10;
        b=b+char(Tich+48);
    }
    if(Nho!=0) b=char(Nho+48)+b;
    while(b.length()>1&&b[0]=='0') b.erase(0,1);
    return b;
}
```

5.4. Nhân 2 số nguyên lớn

Phân tích thuật toán

- Duyệt từ cuối xâu a về đầu xâu.

- Tách từng phần tử của xâu a nhân với xâu b (Thuật toán nhân với số nhỏ).

- Cộng liên tiếp các kết quả thu được (lưu ý trước khi cộng 2 xâu thêm ký tự "0" vào sau xâu thứ 2).

- Xử lý các ký tự "0" trước xâu sau khi cộng.

Chương trình tham khảo

```
string Nhanxau(string a, string b)
{
```

```
    string x,Tg1="0",Tg2,c;
    long i,j=0;
    for(i=b.length()-1;i>=0;i--)
    {
        Tg2=Nhan1so(a,(b[i]-48));
        Tg2.insert(Tg2.length(),j,'0');
        j++;
        c=Congxau(Tg1,Tg2);
        Tg1=c;
    }
    return c;
}
```

5.5. Chia số nguyên lớn cho số nguyên nhỏ

Phân tích thuật toán

- Bước 1: Duyệt từ đầu xâu số nguyên lớn
- Bước 2:

+ Tách từng phần tử của xâu đem chia cho số nguyên nhỏ:

chia = số + dư * 10 (dư ban đầu bằng 0);

thương = chia / số nhỏ;

dư = chia % 10;

+ Cộng liên tiếp các ***thương*** được phần nguyên;

+ Lưu lại giá trị ***dư*** cuối cùng được phần dư;

+ Lưu ý: xóa các "0" ở đầu mảng xâu kết quả.

Chương trình tham khảo

```
void chia_so(char a[],long b,char div[],char mod[])
{
    long i,n=strlen(a),du=0,so,chia,thuong;
```

```
    char tg[10],luu[100000]="";
    for(i=0;i<n;i++)
    {
        strncpy(tg,a+i,1);tg[1]='\0';so=atoi(tg);
        chia=du*10+so;du=chia%b;thuong=chia/b;
        itoa(thuong,luu,10);
        strcat(div,luu);
    }
    itoa(du,luu,10);strcpy(mod,luu);
    i=0;
    while(i<strlen(div)-1 && div[i]=='0') i++;
    strcpy(luu,"");
    strncpy(luu,div+i,strlen(div)-i);
    luu[strlen(div)-i]='\0';
    strcpy(div,luu);
}
```

5.6. Chia hai số nguyên lớn

Phân tích thuật toán

- Lấy số ký tự của xâu a bằng số ký tự của xâu b lưu vào xâu chia.

- Chừng nào số xâu chia còn lớn hơn xâu b thì tiến hành trừ liên tiếp xâu chia cho xâu b, ghi lại số lần trừ có thể là thương tìm được.

- Hạ từng ký tự của xâu a xuống xâu chia.

- Lưu ý: xử lý các ký tự "0" vô nghĩa của xâu a và xâu b.

Chương trình tham khảo

```
string Chiaxau(string a,string b,string &mod)
{
    string div="",Tg="",Tg1="";
```

```
    char luu[3];
    long i,n1,n2,dem;
    n1=a.length();n2=b.length();
    if((n1<n2)||(n1==n2)&&(a<b)){mod=a;return "0";}
    else
    {
       Tg=a.substr(0,n2);
       for(i=n2-1;i<a.length();i++)
       {
           dem=0;Tg=Tg+a[i];
           while(Tg.length()>1&&Tg[0]=='0'
                 Tg.erase(0,1);   while((Tg.length()>n2)||
(Tg.length()==n2)&&(Tg>=b))
           {dem++;Tg=Truxau(Tg,b);}
           itoa(dem,luu,10);
           div=div+luu;
       }
            mod=Tg;while((mod.length()>1)&&(mod[0]=='0'))
mod.erase(0,1);
    }
    return div;
}
```

III. MỘT SỐ BÀI TẬP ÁP DỤNG

Bài 1. C11PNUM - Số nguyên tố. *(Nguồn SPOJ)*

https://vn.spoj.com/problems/C11PNUM/

Cho 2 số nguyên N và K ($1 \le N \le 2^{64} - 1$, $3 \le K \le 10$). Tìm số nguyên lớn nhất không vượt quá N và là tích của K số nguyên tố liên tiếp.

Input

- Dòng đầu là số nguyên T tương ứng với số bộ test (1 <= T <= 15)
- T dòng tiếp theo mỗi dòng là 1 cặp số (N, K) cách nhau 1 dấu cách

Output

- Gồm T dòng là kết quả của T bộ test tương ứng, nếu không tìm được số thỏa mãn in ra -1.

Ví dụ

Input	Output
2 100 4 110 3	-1 105

Hướng dẫn

- Sàng nguyên tố.
- Tính các tích k số nguyên tố liên tiếp để cập nhập cho kết quả.

Chương trình tham khảo

```
#include <bits/stdc++.h>
using namespace std;
const int N = 3000000;
typedef unsigned long long ll;
bool isPrime[N];
vector <int> Prime;
int k, test, d;
ll n;
void Eratosthene()
{
      for(int i = 2; i < N; i++) isPrime[i] = 1;
      isPrime[0] = isPrime[1] = 0;
```

```
	for(int i = 2; i*i < N; i ++)
		if(isPrime[i])
			for(int j = i*i; j < N; j += i)
				isPrime[j] = 0;
	for(int i = 0; i < N; i++) if(isPrime[i])
Prime.push_back(i);
	d = Prime.size();
}
void Cal(){
	ll res = 0;
	for(int i = 0; i < d-k+1; i++){
		ll s = 1;
		for(int j = i; j < i+k; j++)
			s *= Prime[j];
		if(s > n) break;
		else res = max(res, s);
	}
	if(res != 0) cout << res;
	else cout << -1;
}
int main(){
	Eratosthene();
	cin >> test;
	while(test --){
		cin >> n >> k;
		Cal();
		cout << '\n';
	}
```

```
}
```

Bộ test tham khảo: *(Có trên SPOJ)*

Bài 2. C11PRIME - Số nguyên tố (Nguồn SPOJ)

https://vn.spoj.com/problems/C11PRIME/

Số nguyên tố là số chỉ chia hết cho 1 và chính nó. Trong một buổi dã ngoại của trường, bất ngờ TMB bị thầy giáo đố một câu như sau: "Một số có dạng p^q là lũy thừa cao của một số nguyên tố khi và chỉ khi p là một số nguyên tố và q > 1. Thầy sẽ cho em một số N bất kỳ và em hãy cho biết đó có phải là lũy thừa cao của một số nguyên tố hay không?". Không phải lúc nào cũng mang theo máy tính bên mình, đây là lúc TMB cần bạn.

Yêu cầu: Cho số N, hãy giúp TMB trả lời câu đố của thầy giáo, nếu N là lũy thừa cao của một số nguyên tố thì in ra 2 số p và q tương ứng, nếu không thì ghi 0.

Giới hạn: $n \leq 10^{18}$

Input: Một dòng duy nhất chứa n.

Output: Một dòng duy nhất là kết quả.

Ví dụ

Input	Output
27	3 3
10	0

Hướng dẫn

- Duyệt q từ 2 đến 63, ta tính căn bậc q của n, sau đó làm tròn và lũy thừa cho q thử xem nó có bằng n không, nếu bằng thì ta kiểm tra xem số đó có phải số nguyên tố không.
- Công thức tính căn bậc q của n: $\sqrt[q]{n} = n^{\frac{1}{q}} = e^{\frac{\ln n}{q}}$
- Ngoài ra, cùng có thể dùng chặt nhị phân để tính căn.

Chương trình tham khảo

```
#include <iostream>
#include <cmath>
using namespace std;
```

```
bool is_prime(long long p) {
    if (p < 2) return false;
    if (p == 2) return true;
    if (p % 2 == 0) return false;
    for (long long d = 3; d * d <= p; d += 2) {
        if (p % d == 0) return false;
    }
    return true;
}

int main() {
    long long n; cin >> n;
    for (int q = 2; q < 64; q++) {
        long long p = round(exp(log(n) / q));
        long long nn = 1;
        for (int i=0; i < q; i++) nn *= p;
        if (nn == n && is_prime(p)) {
            cout << p << ' ' << q;
            return 0;
        }
    }
    cout << 0;
    return 0;
}
```

Bộ test tham khảo: *(Có trên SPOJ)*

Bài 3. Số đặc biệt

An rất yêu thích số nguyên tố, đồng thời cũng rất yêu thích số 5. Do đó, cậu ta luôn coi các *số nguyên tố có tổng các chữ số chia hết cho 5 là số đặc biệt*. Lần này, thầy giáo đưa cho An 2 số nguyên dương $L, R\ (L \le R)$. An rất muốn biết trong đoạn $[L,R]$ có bao nhiêu số đặc biệt nên nhờ các bạn trả lời giúp.

Input: Vào từ file văn bản **SPRIME.INP**

- Dòng đầu tiên chứa số nguyên dương $T \le 100$ là số lượng test trong file.
- T dòng tiếp theo, mỗi dòng chứa hai số nguyên dương $L, R\ (L \le R)$ theo thứ tự, phân tách nhau bởi dấu cách.

Output: Đưa ra file văn bản **SPRIME.OUT** T dòng, mỗi dòng ghi một số là số lượng số đặc biệt trong đoạn $[L,R]$, tương ứng theo thứ tự trong file input. Dòng thứ i trong file output là kết quả của cặp số $[L,R]$ ở dòng $i+1$ trong file input.

Ví dụ

SPRIME.INP	SPRIME.OUT
2 1 10 4 20	1 2

Giải thích:

- Trong đoạn [1, 10] có 1 số đặc biệt là 5.
- Trong đoạn [4, 20] có 2 số đặc biệt là 5 và 19 (1+9 = 10).

Giới hạn:

- 20% số test có $T = 1; L \leq R \leq 20$
- 20% số test tiếp theo có $T = 1; L, R \leq 10^3$
- 30% số test tiếp theo có $2 \leq T \leq 10; L, R \leq 10^5$
- 30% số test cuối cùng có $10 \leq T \leq 100; 0 < L, R \leq 3.10^6$

Hướng dẫn

- Chuẩn bị trước mảng tổng tiền tố S_i là số lượng số thỏa mãn từ l tới r.
- Sử dụng sàng nguyên tố

Chương trình tham khảo

```
#include <bits/stdc++.h>
#define sz(x) int(x.size())
#define MIN(x,y) if (x > y) x = y
#define PB push_back
#define mp make_pair
#define F first
#define S second
#define Task "sprime"
#define maxn 3000001
#define MOD 1000000007
#define remain(x) if (x > MOD) x -= MOD
#define pii pair<int, int>
using namespace std;
bool nt[maxn];
```

```
int s[maxn];
int tongcs( int x)
{
    int sum = 0;
    for ( ; x; x /= 10) sum += x % 10;
    return sum;
}
int main()
{
    ios_base::sync_with_stdio(0);
    freopen(Task".inp", "r", stdin);
    freopen(Task".out", "w", stdout);
    memset(nt, 0, sizeof(nt));
    //for (int i = 1; i < maxn; i++) nt[i] = nto(i);
    s[1] = 0;
    for (int i = 2; i < maxn; i++)
    {
        s[i] = s[i-1];
        if (nt[i] == 0)
        {
         s[i] += (tongcs(i) % 5 == 0);
         for (int j = i+i; j < maxn; j += i) nt[j] = 1;
        }
    }
    int test, L, R;
    cin >> test;
    while (test--)
    {
        cin >> L >> R;
        cout << s[R] - s[L-1] << endl;
    }
    //cout << s[maxn-1];
```

```
    return 0;
}
```

Bộ test tham khảo: *http://www.mediafire.com/file/28041fl4fenhjeb/BAI_3.rar/file*

Bài 4. PTIT017J - ACM PTIT 2017 J - Số các số không chia hết *(Nguồn SPOJ)*

http://www.spoj.com/PTIT/problems/PTIT017J/

Cho 5 số tự nhiên a, b, c, d, e là các số nguyên tố cùng nhau từng đôi một. Hãy cho biết có bao nhiêu số nhỏ hơn hoặc bằng N không chia hết cho bất kỳ số nào trong các số a, b, c, d, e (N<263).

Input

- Dòng đầu tiên là số lượng bộ test T (T ≤ 50).
- Mỗi test gồm hai dòng: dòng đầu tiên ghi lại số N, dòng kế tiếp ghi lại 5 số a, b, c, d, e nguyên tố cùng nhau (cả 5 số đều không quá 1000).

Output

- Ứng với mỗi test đưa ra số lượng các số không chia hết cho bất kể số nào trong các số a, b, c, d, e.

Ví dụ

Input	Output
3 1000 2 3 5 7 11 10000 3 4 5 7 11 1000000000 9 11 13 17 19	207 3116 665093420

Hướng dẫn

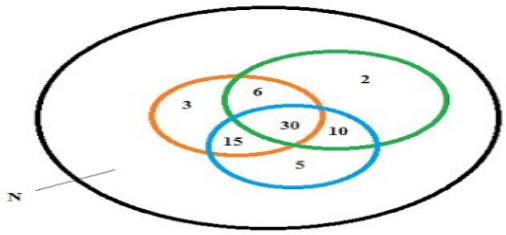
Bài này dựa vào toán rời rạc. Ví dụ:

- Số lượng các số chia hết cho 2 (≤ N) là: [N/2] (trong đó [] là chia lấy nguyên).
- Số lượng các số chia hết cho 3 (≤ N) là: [N/3] (trong đó [] là chia lấy nguyên).
- Số lượng các số chia hết cho cả 2 và 3 (≤ N) là: [N/6] (trong đó [] là chia lấy nguyên).
- Từ đó: số lượng các số không chia hết cho 2 và 3 (≤N) là: N - ([N/2] + [N/3] - [N/6]) (trong đó [] là chia lấy nguyên).

Tổng quát: Với 5 số A, B, C, D, E thì ta có đáp án:

= N - ([N/A] + [N/B] +..+ [N/E] - [N/(A*B)] - [N/(A*C)]-...-N[N/(D*E)] + [N/(A*B*C)] + [N/(A*B*D)] +...+ [N/(C*D*E)]) - [N/(A*B*C*D)] - ... - [N/(B*C*D*E)] + [N/(A*B*C*D*E)])

Ví dụ: Cho 3 số A=2 B=3 C=5, biểu diễn các số chia hết cho A, B, C theo dạng tập hợp;



- Ta thấy S(A+B+C) là tập những số chia hết cho 2 hoặc 3 hoặc 5:

= S(A) + S(B) + S(C) - S(A.B) - S(B.C) - S(C.A) + S(A.B.C) trong đó S (x) là tập các số chia hết cho x.

Việc cần làm giờ là chỉ cần sinh nhị phân chọn các cặp số và tính toán số lượng các số của các cặp tích số.

Chương trình tham khảo

```
#include <bits/stdc++.h>
using namespace std;
int arr[10];
long long N, Value [10], n, k;
void read ()
{
    cin>>N;
    for (int i=1; i<=5; i++)
    {
     cin>>Value[i];
    }
}
vector <long long> v[10];
void tinh ()
{
    long long S = 1;
    int count1=0;
    for (int i=1; i<=n; i++)
```

```
    {
     if (arr[i]==1)
     {
          count1++;
          S*=Value[i];
     }
    }
    if (count1!=0)
    {
     long long tmp = N/S;
     v[count1].push_back(tmp);
    }
}
void sinhNP (int u)
{
    if (u>n)
    {
     tinh ();
    }
    else
    {
     for (int i=0; i<=1; i++)
     {
          arr[u]=i;
          sinhNP (u+1);
     }
    }
}
int main ()
{
    int t;
    cin>>t;
    while (1)
    {
     if (t==0) break;
     t--;
     read ();
     n=5;
```

```
    sinhNP (1);
    long long SUM = 0;
    for (int i=1; i<=5; i++)
    {
        for (int j=0; j<v[i].size (); j++)
        {
            if (i%2!=0) SUM+=v[i][j];
            else SUM-=v[i][j];
        }
    }
    cout<<N-SUM<<endl;
    //Xoa du lieu truoc do.
    for (int i=1; i<=5; i++)
        v[i].clear();
   }
   return 0;
}
```

Bộ test tham khảo: *(Có trên SPOJ)*

Bài 5. Bội chung nhỏ nhất

Cho số nguyên dương n. Xét tất cả các phân tích N thành tổng các số tự nhiên: $N = a_1 + a_2 + \cdots + a_k$.

Yêu cầu: Trong các cách phân tích đó, hãy tìm cách phân tích số n thành tổng các số tự nhiên sao cho bội chung của các số hạng là lớn nhất.

Input: Vào từ file NUMBER.INP gồm một dòng ghi số $n\ (n \leq 100)$

Output: Ghi ra file văn bản NUMBER.OUT

- Dòng đầu ghi bội chung nhỏ nhất của các số hạng trong cách phân tích tìm được.
- Dòng tiếp theo ghi các số hạng đó.

Ví dụ

NUMBER.INP	NUMBER.OUT
3	3 3
51	180180 13 11 9 7 5 4 2

Hướng dẫn

Quy hoạch động:

– Nếu gọi A[i,j] là giá trị lớn nhất của BSCNN khi phân tích i thành tổng của j số tự nhiên.

– B[i,j] là giá trị của số thứ j trong cách phân tích i thành tổng của j và số tự nhiên để A[i, j] đạt Max.

– Khởi tạo:

A[i,1]: =i;

B[i,1]: =i;

– Với $\forall j(2 \le j < N)$ ta xét các khả năng phân tích i $(j \le i < N)$ để BSCNN của các cách phân tích đó đạt Max.

– Ta có: $A[i,j] := Max\{BSCNN(A[i-t,j-1],t)\}$ với $1 \le t \le i-j+1; B[i,j] := t;$

– Giá trị lớn nhất trong các cách phân tích N bằng $Max(A[i,j])$ với $1 \le i \le N$.

Chương trình tham khảo

```
#include <bits/stdc++.h>
using namespace std;
int n,i,j,j1,k,ans,trace;
static int b[101][101],f[101][101];

int LCM(int a,int b)
{
    int m=a, n=b;
    while (n!=0)
    {
        int r=m%n;
        m=n;
        n=r;
    }
    return (long long) a*b/m;
}
void Tracing(int i,int j)
{
    if (i==0||j==0) return;
    Tracing(i-b[i][j],j-1);
    cout<<b[i][j]<<" ";
```

```
}
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    freopen("NUMBER.INP","r",stdin);
    freopen("NUMBER.OUT","w",stdout);
    cin>>n;
    for (i=1;i<=n;++i) f[i][1]=i, b[i][1]=i;
    for (i=1;i<=n;++i)
        for (j=2;j<=n;++j)
            for (k=1,j1=j-1;k<i;++k)
            {
                int &x=f[i][j];
                int t=LCM(f[i-k][j1],k);
                if (x<t)
                {
                    x=t;
                    b[i][j]=k;
                }
            }
    ans=n; trace=1;
    for (j=2;j<=n;++j)
    {
        k=f[n][j];
        if (ans<k) ans=k, trace=j;
    }
    cout<<ans<<"\n";
    Tracing(n,trace);
}
}
```

Bộ test tham khảo: ***http://www.mediafire.com/file/hm71lonsat7x102/BAI_5.rar/file***

Bài 6. PTIT016A - ACM PTIT 2016 A - Bội số chung nhỏ nhất *(Nguồn SPOJ)*

https://www.spoj.com/PTIT/problems/PTIT016A/

Cho hai số tự nhiên A và B. Bài toán đặt ra là xác định số nguyên dương N sao cho bội số chung nhỏ nhất của A+N và B+N đạt giá trị nhỏ nhất.

Input

Chỉ gồm một dòng chứa hai số tự nhiên A, B. Cả hai giá trị không quá 10^9.

Output

Ghi ra giá trị N tìm được. Nếu có nhiều giá trị N thỏa mãn thì ghi ra giá trị nhỏ nhất.

Ví dụ

Input	Output
4 10	2

Hướng dẫn

– Nhận thấy bội chung nhỏ nhất của A và B bằng tích hai số chia cho ước chung lớn nhất. Do đó để tối thiểu bội chung nhỏ nhất thì ta phải tối đa ước chung lớn nhất

– Ước chung lớn nhất của $A+N$ và $B+N$ theo định lý Euclid thì nó cũng là ước của $(A+N)-(B+N) = A-B.$

Chương trình tham khảo

```
#include<bits/stdc++.h>
using namespace std;
typedef unsigned long long Long;
Long gcd(Long a,Long b)
{
    while(b)
    {
        Long r=a%b;
        a=b;b=r;
    }
    return a;
}
int main()
{
    Long A,B,d,N,i,max,max1,N1;
    cin>>A>>B;
    if(A<B) {d=A;A=B;B=d;}
    if(A==B) {cout<<1;return 0;}
    d=A-B;
    N=(B/d+1)*d-B;
    max=(A+N)/gcd(A+N,B+N)*(B+N);
    for(Long x=2;x*x<=d;x++)
    if(d%x==0)
```

```
    {
              N1=(B/x+1)*x-B;
             if(N1<N)
             {
                max1=(A+N1)/gcd(A+N1,B+N1)*(B+N1);
                if(max>=max1) {max=max1;N=N1;}
             }
             Long y=d/x;
             N1=(B/y+1)*y-B;
             if(N1<N)
             {
             Long max1=(A+N1)/gcd(A+N1,B+N1)*(B+N1);
             if(max>=max1) {max=max1;N=N1;}
             }
    }
    cout<<N;
}
```

Bộ Test tham khảo: *(Có trên SPOJ)*

Bài 7. P144SUMI - ROUND 4I - Chia phần *(Nguồn SPOJ)*

https://www.spoj.com/PTIT/problems/P144SUMI/

Sau giờ học, Tí cùng các bạn rủ nhau đi ăn xúc xích. Mọi người cùng nhau góp tiền. Có tất cả m người nhưng số tiền mà Tí cùng các bạn có chỉ mua được n cái xúc xích.

Để cho công bằng thì Tí sẽ chia đều n chiếc xúc xích cho tất cả mọi người. Với một con dao trong tay, các bạn hãy giúp Tí tính xem Tí cần cắt ít nhất bao nhiêu phát?

Input: Gồm 2 số nguyên n và m $(n,\ m\ \leq 100)$.

Output: In ra một số nguyên duy nhất là đáp án của bài toán.

Ví dụ

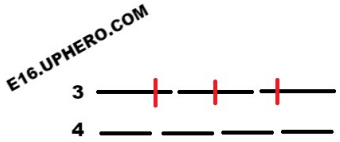
Input	Output
2 6	4
3 4	3
6 2	0

Giải thích Test 1:

Có 2 cái xúc xích và cần chia ra thành 6 phần, Tí sẽ cắt mỗi chiếc xúc xích ra thành 3 phần đều nhau, như vậy cần tổng cộng 4 lát cắt.

Hướng dẫn

– Áp dụng tham lam như hình vẽ:



– Trên hình là cách chia 3 xúc xích cho 4 người (Vạch đỏ là nơi cắt xúc xích)

Như vậy:

– Lấy bội chung của số xúc xích và số người ta sẽ xác định được mỗi người sẽ cần tương ứng với bao nhiêu phần xúc xích. (Lấy bội chung thì lúc chia phần sẽ luôn nguyên).

Ví dụ: BCNN(3 ,4) = 12; ⇒ một người cần 3 phần xúc xích.

– Tham lam: Mỗi người ta sẽ lấy đủ phần xúc xích (thiếu thì lấy thêm không may lấy thừa thì sẽ cắt bớt).

Ví dụ: Người 1 cần 3 phần:

+ Lấy xúc xích 1: được 4 phần; (4 > 3: cắt bớt: cắt++; dư: 1 phần)

⇒ người 1: đủ.

+ Lấy xúc xích 2: được 4 + 1 phần; (5 > 3: cắt bớt; cắt++; dư 2 phần)

⇒ người 2: đủ.

+ Lấy xúc xích 3: được 4 + 2 phần; (6 > 3: cắt bớt; cắt++; dư 3 phần)

⇒ người 3: đủ.

+ Còn lại 3 phần cho người thứ 4;

Chương trình tham khảo

```
#include <bits/stdc++.h>
using namespace std;
int BCNN (int n, int m)
{
     int a=n, b=m;
     int tmp;
     while (m!=0)
```

```
    {
        tmp = n%m;
        n=m;
        m=tmp;
    }
    return (a*b)/n;
}
int main ()
{
    int n, m;
    cin>>n>>m;
    int all_part = BCNN (n, m);
    int part_1 = all_part/n;
    int part_2 = all_part/m;
    int S=0;
    int count_cut=0;
    for (int i=1; i<=m; i++)
    {
        while (S<part_2)
        {
            S+=part_1;
        }
        if (S==part_2) S=0;
        else
        {
            count_cut++;
            S-=part_2;
        }
    }
    cout << count_cut;
    return 0;
}
```

Bộ Test tham khảo: *(Có trên SPOJ)*

Bài 8. P155PROF - ROUND 5F - Dãy số Fibonacci 2 *(Nguồn SPOJ)*

https://www.spoj.com/PTIT/problems/P155PROF/

Bạn được cho 1 dãy số được định nghĩa như sau:

$$F_1 = x;\ F_2 = y;\ F_i = F_{i-1} + F_{i+1}.$$

Cho trước x, y. Hãy tính $F_n \ \%\ 1000000007\ (10^9 + 7)$.

Input

- Dòng đầu tiên gồm 2 số $x, y\ (|x|,\ |y| \le 10^9)$.
- Dòng thứ 2 gồm số nguyên dương $n\ (1 \le n \le 2 \times 10^9)$.

Output

- In ra một số nguyên duy nhất là đáp án của bài toán.

Ví dụ

Input	Output
2 3 3	1
0 -1 2	1000000006

Giải thích test 2: $F_2 = -1,\ -1\ \%\ (1000000007) = 1000000006$.

Hướng dẫn

Quy luật của bài này là cứ 6 số lại lặp lại F_n. Vậy nên chỉ cần biết quy luật thì độ phức tạp chỉ còn là O(n%6); Chỉ cần chú ý thêm về nến nó là số âm thì cộng thêm 1000000007 là được.

Chương trình tham khảo

```
#include <bits/stdc++.h>
using namespace std;
int main ()
{
     long long n;
     long long F1, F2;
     cin>>F1>>F2;
     cin>>n;
     if (n%6==0) n=6;
     else n=n%6;
     long long Fn;
```

```
    for (long long i=3; i<=n; i++)
    {
        Fn=(F2-F1)%1000000007;
        F1=F2;F2=Fn;
    }
    long long KQ;
    if (n==1) KQ=F1%1000000007;
    else if (n==2) KQ=F2%1000000007;
    else KQ=Fn;
    if (KQ<0) cout<<KQ+1000000007;
    else cout<<KQ;
    return 0;
}
```

Bộ Test tham khảo: *(Có trên SPOJ)*

Bài 9. CATALAN - Dãy số Catalan *(Nguồn SPOJ)*

https://www.spoj.com/problems/CATALAN/

Cho số nguyên dương N, dãy Catalan cấp n là dãy $C(1), C(2) \dots C(2n+1)$ gồm các số nguyên không âm thoả mãn : $C(1) = C(2n+1) = 0$ với i bất kì $1 \le i \le 2n$ thì $C(i), C(i+1)$ hơn kém nhau 1 đơn vị.

Với mỗi *n* ta sắp xếp các dãy Catalan theo thứ tự từ điển, đánh số từ 1 trở đi.

Yêu cầu:

- Cho một dãy Catalan, hãy tìm thứ tự của dãy.
- Cho số nguyên dương k hãy tìm dãy có thứ tự k

Input

- Dòng đầu ghi n. (n <= 15)
- Dòng hai ghi một dãy Catalan cấp n
- Dòng 3 ghi một số nguyên dương k (k có thể rất lớn nhưng đảm bảo luôn có nghiệm)

Output

- Dòng 1 ghi số thứ tự dãy ở dòng 2 INPUT.
- Dòng 2 ghi dãy ứng với số thứ tự.

Ví dụ

Input	Output
4 0 1 2 3 2 1 2 1 0 12	12 0 1 2 3 2 1 2 1 0

Hướng dẫn

Cách 1. Sử dụng đệ quy có nhớ.

Cách 2. Quy hoạch động cấu hình.

- Xét lưới vuông (n + 1) x (n + 1). Một quy tắc đi thõa mãn:
  - Xuất phát từ ô (1, 1) đến ô (n + 1, n + 1).
  - Mỗi bước đi chỉ được di chuyển đến ô kề cạnh bên phải hoặc bên dưới.
  - Không được di chuyển qua ranh giới là đường chéo nối đỉnh trái trên và phải dưới của lưới
  - Nếu quy định a[1, 1] = 0 và a[i + 1, j] = a[i, j] + 1, a[i, j + 1] = a[i, j] – 1.
  - Dễ thấy mỗi cách đi từ ô (1, 1) đến ô (n + 1, n + 1) là 1 dãy catalan tương ứng.
- Mảng quy hoạch động:
  - F[i, j]: số cách để đi từ ô (i, j) đến ô (n + 1, n + 1)

    F[i, j] = F[i – 1, j] + F[i, j – 1] với F[n + 1, n + 1] := 1;
  - Giả sử tại ô (u,v) ta di chuyển xuống ô phía dưới thì cách đi này sẽ có số thứ tự lớn hơn cách đi từ ô (u,v) di chuyển sang ô bên phải (với u > v). Do đó ta chỉ quan tâm đến những ô (u, v) mà tại đó thự c hiện di chuyển xuống ô phía dưới, ta cộng vào s thêm F[u, v + 1].
  - Kết quả số thứ tự của dãy catalan tương ứng với cách đi là s + 1.
  - Cho số tìm dãy thì làm ngược lại.

Chương trình tham khảo

```
#include <bits/stdc++.h>
long n,k,i,j,min,kq,tong,C[41],r[41],F[41][41];
long mini(long x,long y)
```

```
  {
  if(x>y) return y;
  else return x;
  }
main()
{
scanf("%ld",&n);
n=2*n+1;
for(i=1;i<=n;i++)
  scanf("%ld",&C[i]);
scanf("%ld",&k);
F[n][0]=1;
F[n-1][1]=1;
for(i=n-2;i>=1;i--)
  {
  if(i%2==0) j=1;
  else j=0;
  while(j<=min)
    {
    min=mini(n-i,i-1);
    if(j>0)
      F[i][j]=F[i+1][j-1];
    if(j+1<=n-(i+1))
      F[i][j]=F[i][j]+F[i+1][j+1];
    j=j+1;
    }
  }
for(i=3;i<=n-2;i++)
  {
  if(C[i]>C[i-1]&&C[i-1]>0)
    kq=kq+F[i][C[i-1]-1];
  }
printf("%ld\n",kq+1);
r[1]=0;
r[2]=1;
for(i=3;i<=n;i++)
  {
```

```
  min=mini(n-i,i-1);
  if(r[i-1]+1>min)
    r[i]=r[i-1]-1;
  else if(r[i-1]==0)
    r[i]=1;
  else
    {
    if(k>F[i][r[i-1]-1])
      {
      r[i]=r[i-1]+1;
      k=k-F[i][r[i-1]-1];
      }
    else r[i]=r[i-1]-1;
    }
  }
for(i=1;i<=n;i++)
  printf("%ld ",r[i]);
}
```

Bộ test tham khảo: *(Có trên SPOJ)*

Bài 10. BIGNUM - Xử lý số nguyên lớn *(Nguồn SPOJ)*

Cho hai số nguyên dương A và B (A, B có không quá 1000 chữ số)

Yêu cầu

Tính A + B, A - B, A * B

Input

- Dòng 1: số nguyên A
- Dòng 2: số nguyên B

Output

- Dòng 1: Kết quả A + B
- Dòng 2: Kết quả A - B
- Dòng 3: Kết quả A * B

Ví dụ

Input	Output
10 11	21 -1 110

Hướng dẫn: Xử lí số nguyên lớn.

Chương trình tham khảo *(Một cách khác cài đặt xử lí số nguyên lớn)*

```
#include <bits/stdc++.h>
using namespace std;
const int BASE = 10000;
typedef vector <int> BigInt;
BigInt Init(string s) {
    BigInt a; if (s.size() == 0) { a.push_back(0); return a;
}
    while (s.size() % 4 != 0) s = '0' + s;
    for (int i = 0; i < (int)s.size(); i += 4) {
        int value = 0;
        for (int j = 0; j < 4; j++)
            value = value * 10 + (s[i + j] - '0');
        a.insert(a.begin(), value);
    } return a;
}
void Print(const BigInt a) {
    int L = a.size(); printf("%d", a[L - 1]);
    for (int i = L - 2; i >= 0; i--) printf("%04d", a[i]);
    printf("\n");
}
BigInt operator + (const BigInt a, const BigInt b) {
    BigInt c; int carry = 0;
    for (int i = 0; i < (int)a.size() || i < (int)b.size();
i++) {
        if (i < (int)a.size()) carry += a[i];
        if (i < (int)b.size()) carry += b[i];
        c.push_back(carry % BASE); carry /= BASE;
    }
    if (carry) c.push_back(carry); return c;
}
BigInt operator - (const BigInt a, const BigInt b) {
    BigInt c; int carry = 0;
    for (int i = 0; i < (int)a.size(); i++) {
```

```
        int s = a[i] - carry;
        if (i < (int)b.size()) s -= b[i];
        if (s < 0) s += BASE, carry = 1; else carry = 0;
        c.push_back(s);
    }
    while (*c.rbegin() == 0 && c.size() > 1) c.pop_back();
return c;
}
BigInt operator * (const BigInt a, const int mul) {
    BigInt c; int carry = 0;
    for (int i = 0; i < (int)a.size(); i++) {
        carry += a[i] * mul; c.push_back(carry % BASE);
carry /= BASE;
    }
    if (carry) c.push_back(carry); return c;
}
BigInt operator * (const BigInt a, const BigInt b) {
    BigInt c(a.size() + b.size() + 5, 0);
    for (int i = 0; i < (int)a.size(); i++) {
        int carry = 0;
        for (int j = 0; j < (int)b.size(); j++) {
            int k = i + j; c[k] += a[i] * b[j] + carry;
            carry = c[k] / BASE; c[k] %= BASE;
        }
        if (carry) c[i + b.size()] += carry;
    }
    while (*c.rbegin() == 0 && c.size() > 1) c.pop_back();
return c;
}
bool operator < (BigInt a, BigInt b) {
    while (a.size() && *a.rbegin() == 0) a.pop_back();
    while (b.size() && *b.rbegin() == 0) b.pop_back();
    if (a.size() != b.size()) return a.size() < b.size();
    for (int i = a.size() - 1; i >= 0; i--)
        if (a[i] != b[i]) return a[i] < b[i];
    return false;
}
int main() {
    string sa, sb;
    cin >> sa >> sb;
    BigInt a = Init(sa);
    BigInt b = Init(sb);
    Print(a + b);
    if (a < b) {
        printf("-");
```

```
        Print(b - a);
    } else {
        Print(a - b);
    }
    Print(a * b);
    return 0;
}
```

Bộ test tham khảo: *(Có trên SPOJ)*

Bài 11. FUNCTION

Cho hàm số tuyến tính $f(x) = Ax + B$. Định nghĩa $g_0(x) = x$ và $g_n(x) = f(g_{n-1}(x))$ với $n \geq 1$. Cho trước các số *A*, *B*, *n*, *x*. Tính $g_n(x) \ mod \ cho \ 10^9 + 7$.

Input: Một dòng duy nhất gồm các số $A, B, n, x. \ (1 \leq A, B, x \leq 10^9, 1 \leq n \leq 10^{18})$

Output: Một dòng duy nhất là đáp án.

Ví dụ:

Input	Output
3 4 1 1	7

Hướng dẫn

+ Theo đề bài: $g_0(x) = x$ và $g_n(x) = f(g_{n-1}(x))$ với n ≥ 1.

$$\rightarrow g_1(x) = f(g_0(x)) = f(x) = Ax + B$$

$$\rightarrow g_2(x) = f(g_1(x)) = A(Ax + B) + B = A^2x + AB + B$$

…

$$\rightarrow g_n(x) = A(A(A(\ldots\ldots(Ax + B) + B) + B) + \ldots.) + B$$

$$= A^n x + A^{n-1}B + A^{n-2}B + .. + B$$

$$= A^n x + B(A^{n-1} + A^{n-2} + \ldots + A + 1)$$

$$= A^n x + B(A^n - 1)/(A - 1) \ \ (do \ (A^n - 1) = (A - 1) * (A^{n-1} + A^{n-2} + \ldots + A + 1))$$

+ Tính:

$$g_n(x) \ \% \ (1e9 + 7) = (A^n \% \ mod) * (x \ \% \ mod) + (B \ \% \ mod) * ((A^n - 1)/(A - 1)) \ \% \ mod$$

+ Tính $A^n \% \ mod$ sử dụng kĩ thuật đệ quy.

+ Tính $((A^n - 1)/(A - 1)) \% mod$: Ta có định lý Fermat nhỏ: nếu p là số nguyên tố thì với số tự nhiên b ta có $b^{p-1} \% \ p \ = \ 0$

+ Áp dụng: với p là số nguyên tố ta có:

$$(a/b)\%p = \left(\left(a*b^{p-2}\right)/b^{p-1}\right)\%p = \left(\left(a*b^{p-2}\right)\right)\%p \text{ (đồng dư)}$$

$$\rightarrow g_n(x)\%(1e9+7) = \left(A^n\%mod\right)*\left(x\%mod\right)+\left(B\%mod\right)*\left(\left(A^n-1\right)/(A-1)\right)\%mod$$

$$= \left(A^n\%mod\right)*\left(x\%mod\right)+\left(B\%mod\right)*\left(\left(A^n-1\right)*(A-1)^{mod-2}\right)\%mod$$

Chương trình tham khảo

```
#include <bits/stdc++.h>
#define Task "function"
using namespace std;
const int m = 1e9 + 7;
long long a,b,x;
unsigned long long n;
long long power(long long k, long long h)
{

    if(h == 0) {
        return 1;
    } else {
        long long temp = power(k, h/2)%m;
        if(h % 2 == 0)
            return temp*temp%m %m;
        else
            return (temp*temp%m) * k%m;
    }

}
int main()
{
    ios_base::sync_with_stdio(0); cin.tie(NULL);
cout.tie(NULL);
    freopen(Task".inp", "r", stdin);
    freopen(Task".out", "w", stdout);
    cin >> a >> b >> n >> x;
    if(a==1) cout << (x + b*n)%m;
    else
    {
```

```
            long long s = (power(a,n) * (x%m))%m;
            s=s + (b%m*(power(a,n)-1)%m*power(a-1,m-2)%m)%m;
            s=s%m;
            cout << s;
        }
    }
```

Bộ test tham khảo: http://www.mediafire.com/file/8cmo4sun5qftn1z/BAi_11.rar/file

Bài 12. GCDMAX

Cho dãy $a_1, a_2, ..., a_n$. Một dãy $b_1, b_2, ..., b_n$ được gọi là đẹp nếu $|a_i - b_i| \le 1$.

Một dãy *b* đẹp sẽ càng đẹp nếu ước chung lớn nhất của các phần tử của nó càng lớn càng tốt. Hãy tìm ước chung lớn nhất có thể của một dãy *b* đẹp.

Input

- Dòng đầu là số nguyên dương n ($1 \le n \le 50000$).
- Dòng thứ hai là số nguyên dương mô tả dãy a. $(1 \le a_i \le 10^9)$

Output

- Một dòng duy nhất là đáp án cần tìm.

Ví dụ

Input	Output
5 4 2 3 1 2	2

Hướng dẫn

- Nếu số tự nhiên n có số lượng ước $\le \sqrt[3]{n}$
- Đề cho $1 \le a_i \le 10^9$ nên ta có thể:

  $For\ (j = 1; j <= 1000; j++)$

  { Thử xem $a[i], a[i]-1, a[i]+1$ có phải ước của j không;
  Nếu có lưu lại mảng b ;
  }
- In ra mảng b;

Chương trình tham khảo

```
#include<bits/stdc++.h>
using namespace std;
```

```
int n,a[50005],f[50005];
int main()
{
    freopen("GCDMAX.inp","r",stdin);
    freopen("GCDMAX.out","w",stdout);
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    int ds=-1;
    f[1]=a[1]-1;
    for(int i=2;i<=n;i++)
    {
        f[i]=max(f[i],__gcd(a[i]-1,f[i-1]));
        f[i]=max(f[i],__gcd(a[i],f[i-1]));
        f[i]=max(f[i],__gcd(a[i]+1,f[i-1]));
    }
    ds=max(ds,f[n]);
    for(int i=1;i<=n;i++) f[i]=0;
    f[1]=a[1];
    for(int i=2;i<=n;i++)
    {
        f[i]=max(f[i],__gcd(a[i]-1,f[i-1]));
        f[i]=max(f[i],__gcd(a[i],f[i-1]));
        f[i]=max(f[i],__gcd(a[i]+1,f[i-1]));
    }
    ds=max(ds,f[n]);
    for(int i=1;i<=n;i++) f[i]=0;
    f[1]=a[1]+1;
    for(int i=2;i<=n;i++)
    {
        f[i]=max(f[i],__gcd(a[i]-1,f[i-1]));
        f[i]=max(f[i],__gcd(a[i],f[i-1]));
        f[i]=max(f[i],__gcd(a[i]+1,f[i-1]));
    }
    ds=max(ds,f[n]);
    printf("%d",ds);
}
```

Bộ test tham khảo: http://www.mediafire.com/file/asu2zq3qxt84kpy/BAI_12.rar/file

III. MỘT SỐ BÀI TẬP THAM KHẢO

Bài 13. P161PROA - Số gần nguyên tố *(Nguồn SPOJ)*

Chúng ta đều biết số nguyên tố là số nguyên dương mà chỉ có duy nhất 2 ước phân biệt. Iron man luôn thích những cái đặc biệt và mới mẻ, và anh ra đưa ra 1 định nghĩa mới "Số gần nguyên tố" – là các số nguyên dương mà có đúng 3 ước phân biệt.

Bạn được cho 1 mảng có n phần tử, hãy kiểm tra xem từng phần tử trong mảng có phải là số gần nguyên tố hay không.

Input

- Dòng đầu tiên nhập vào số tự nhiên n ($1 \le n \le 10^5$) là số phần tử của mảng.
- Dòng tiếp theo nhập n số nguyên dương x[i] ($1 \le x[i] \le 10^{12}$)

Output

- In ra trên n dòng: dòng thứ i in "YES" nếu x[i] là số gần nguyên tố, ngược lại thì in "NO".

Ví dụ

Input	Output
3 4 5 6	YES NO NO

Bài 14. BLGEN – Chuỗi gen đặc trưng (Nguồn SPOJ đề thi olympic 30/4/2014 Môn tin học khối 10)

Tế bào của một cá thể sinh vật ngoài hành tinh mới được phát hiện gồm rất nhiều gen, mỗi gen trong chuỗi gen của tế bào đều có số lượng nào đó các nucleotide (ký hiệu là nu). Các chuyên gia thường quan tâm chuỗi gen của mỗi cá thể dưới góc độ một chuỗi số lượng tương ứng các nu (gọi tắt là chuỗi nu), do đó chuỗi sẽ như là một dãy số nguyên dương đồng thời số số hạng của dãy này sẽ được gọi là độ dài của chuỗi. Mỗi gen được xem là đặc biệt nếu số nu của nó hoặc là bình phương của một số nguyên hoặc là lập phương của một số nguyên tố.

Để nghiên cứu khả năng biến đổi gen của loài sinh vật nói trên, các nhà khoa học xem xét hai mẫu chuỗi nu của hai cá thể và quan tâm đến mức độ "giống nhau" giữa chúng theo cách tìm ra chuỗi con chỉ gồm các gen đặc biệt mà cùng xuất hiện ở cả hai chuỗi nu (mỗi chuỗi con như vậy đều được gọi là chuỗi đặc trưng chung của hai chuỗi nu). Lưu ý rằng, chuỗi con của một chuỗi nu X, là chuỗi thu được từ

X bằng cách giữ nguyên tất cả hoặc loại bỏ đi một số nào đó các gen mà vẫn giữ thứ tự xuất hiện trong chuỗi X.

Yêu cầu : Xác định độ dài lớn nhất L của chuỗi đặc trưng chung của hai chuỗi nu cho trước.

Input

- Dòng đầu ghi lần lượt các số hạng của chuỗi nu thứ nhất.
- Dòng tiếp theo ghi lần lượt các số hạng của chuỗi nu thứ hai.
- Tất cả các số hạng của hai chuỗi đều nguyên dương và không vượt quá 10^{19}
- Độ dài của mỗi chuỗi nu đều không vượt quá 1000.

Output

- Ghi ra duy nhất một số nguyên L tìm được.

Ví dụ

Input	Output
2 9 8 4 1 27 4 6 5 6 9 1 8 2 6 27 1 4	4

(Giải thích: L = 4, một trong các chuỗi đăc trưng chung là: 9, 1, 27, 4)

Ràng buộc : 60% số test ứng với 60% số điểm của bài ứng với tình huống độ dài của hai chuỗi nu không vượt quá 255 và giá trị của mỗi số hạng đều không vượt quá 10^6.

Bài 15. Bội số chung nhỏ nhất

Cho số nguyên n, tìm bội số chung nhỏ nhất của các số 1, 2, …, n.

Input

Tệp văn bản BSCNN.INP gồm một dòng chứa số n ($n \leq 1000$).

Output

Tệp văn bản BSCNN.OUT ghi bội số chung nhỏ nhất của 1, 2, … n.

Ví dụ

Input	Output
3	6

Bài 16. FIBVAL SPOJ – VOI 2012 Bản Vanxơ Fibonacci *(Nguồn SPOJ)*

Bản vanxơ Fibonacci là một bản nhạc mà giai điệu của nó bắt nguồn từ một trong những dãy số nổi tiếng nhất trong Lý thuyết số – dãy số Fibonacci. Hai số đầu tiên của dãy là số 1 và số 2, các số tiếp theo được xác định bằng tổng của 2 số liên tiếp ngay trước nó trong dãy.

Bản vanxơ Fibonacci thu được bằng việc chuyển dãy số Fibonacci thành dãy các nốt nhạc theo qui tắc chuyển một số nguyên dương thành nốt nhạc sau đây:

- Số 1 tương ứng với nốt Đô (C).
- Số 2 tương ứng với nốt Rê (D).
- Số 3 tương ứng với nốt Mi (E).
- Số 4 tương ứng với nốt Fa (F).
- Số 5 tương ứng với nốt Sol (G).
- Số 6 tương ứng với nốt La (A).
- Số 7 tương ứng với nốt Si (B).
- Số 8 tương ứng với nốt Đô (C).
- Số 9 tương ứng với nốt Rê (D).

và cứ tiếp tục như vậy. Ví dụ, dãy gồm 6 số Fibonacci đầu tiên 1, 2, 3, 5, 8 và 13 tương ứng với dãy các nốt nhạc C, D, E, G, C và A.

Để xây dựng nhịp điệu vanxơ người ta đi tìm các đoạn nhạc có tính chu kỳ trong bản vanxơ Fibonacci. Đoạn nhạc được gọi là có tính chu kỳ nếu như có thể chia nó ra thành $k \geq 2$ đoạn giống hệt nhau.

Ví dụ, đoạn nhạc GCAGCA là đoạn có tính chu kỳ, vì nó gồm 2 đoạn giống nhau GCA.

Yêu cầu: Cho trước hai số nguyên dương u, v (u < v), hãy xác định độ dài đoạn nhạc dài nhất có tính chu kỳ của bản nhạc gồm dãy các nốt nhạc của bản vanxơ Fibonacci bắt đầu từ vị trí u kết thúc ở vị trí v.

Ràng buộc: 50% số tests ứng với 50% số điểm có $u_i < v_i \leq 100$.

Input

- Dòng thứ nhất chứa số nguyên dương k ($k \leq 100$) là số lượng test.
- Dòng thứ i trong số k dòng tiếp theo chứa 2 số nguyên dương u_i, v_i được ghi cách nhau bởi dấu cách ($u_i < v_i \leq 10^9$) là vị trí bắt đầu và kết thúc của 1 bản nhạc.

Output

- Ghi ra k dòng, dòng thứ i chứa 1 số nguyên là độ dài đoạn nhạc tìm được tương ứng với test thứ i. Nếu không tìm được đoạn nào có tính chu kỳ thì ghi ra số -1.

Ví dụ

Input	Output
2 1 3 4 10	-1 2

Bài 17. P156SUME spoj PTIT – ROUND 6E – Ước chung của chuỗi

Một chuỗi a được gọi là ước của chuỗi b nếu tồn tại một số nguyên dương x sao cho khi ta viết x lần chuỗi a thì sẽ thu được chuỗi b. Ví dụ chuỗi "abab" có 2 ước là "ab" và "abab".

Bạn được cho 2 cho 2 chuỗi s1 và s2, hãy đếm xem chúng có tất cả bao nhiêu ước chung?

Input

- Dòng đầu tiên là 1 chuỗi s1, dòng thứ 2 là chuỗi s2.
- Cả 2 chuỗi đều gồm các chữ cái thường, độ dài 2 chuỗi không quá 10^5.

Output

- In ra một số nguyên là kết quả của bài toán.

Ví dụ

Input	Output
xyztxyzt xyzt	1
aaa aa	1

Bài 18. LQDFIBO - Xâu Fibonacci *(Nguồn SPOJ)*

Cho 2 xâu khác rỗng S_1và S_2 có độ dài không lớn hơn 100. Xét các dãy $F_1, F_2, F_3, ..., F_n$ trong đó:

$$F_1 = S_1$$
$$F_2 = S_2$$

....

$F_k = F_{k-1} + F_{k-2}$ với k>2

Cho xâu S không quá 100 ký tự và số nguyên $N\ (3 \le N \le 1000)$ .Hãy xác định S xuất hiện bao nhiêu lần trong F_n.

Input

- Dòng 1 chứa số nguyên N
- Dòng 2 chứa xâu S_1
- Dòng 3 chứa xâu S_2
- Dòng 4 chứa xâu S

Output

- Đưa ra một dòng chứa kết quả

Ví dụ:

Input	Output
8 A B AB	8

Bài 19. NEWJ - Số học 2 *(Nguồn SPOJ)*

Tìm tất cả số các nguyên x thoả mãn (x*x) mod n = a mod n. Trong đó n là số nguyên tố và ước chung lớn nhất của a và n = 1, $0 \le x \le n - 1$.

Input

- Dòng 1: số nguyên K là số bộ test ($1 \le K \le 120000$). K dòng tiếp theo mỗi dòng gồm 2 số nguyên a , n ($1 \le a, n \le 1000000$).

Output

- Với mỗi test ghi ra tất cả các số nguyên x thoả mãn theo thứ tự tăng dần trên 1 dòng . Nếu không có số nguyên x nào thoả mãn thì ghi ra “Khong co”.

Ví dụ

Input	Output

5 4 17 3 7 2 7 14 31 10007 20011	2 15 Khong co 3 4 13 18 5382 14629

Bài 20. GCDOPER

Cho dãy $a_1, a_2, ..., a_n$.

Có thể thực hiện một số thao tác sau: Lấy 2 số cạnh nhau từ dãy a, giả sử là x,y, thay một trong hai vị trí đó bằng $gcd(x, y)$.

Hỏi cần bao nhiêu bước để đưa dãy về toàn số 1.

Input

- Dòng đầu là số nguyên dương $n\ (1 \le n \le 1000)$.
- Dòng thứ hai là n số nguyên dương mô tả dãy $a.\ (1 \le a_i \le 10^9)$.

Output

- Một dòng duy nhất là số bước ít nhất. Nếu không tồn tại cách in ra -1.

Ví dụ

Input	Output
2 2 2 3 4 6	5

Tham khảo: http://www.mediafire.com/file/a46mtnaruh6e9ec/Bai_20.rar/file

Bài 21. P153PROB - ROUND 3B - Dãy số Catalan *(Nguồn SPOJ)*

Bạn được cho một dãy số được định nghĩa truy hồi như sau:

$$C_n = C_{n-1}C_0 + C_{n-2}C_1 + ... + C_0C_{n-1}$$

Với $C_0 = C_1 = 1$ thì dãy trên chính là dãy Catalan, có thể được tính theo cách khác:

$$C_n = \frac{(2n)!}{n!*(n+1)!}$$

Nhiệm vụ của bạn tính số S_n biết:

$$S_n = \sum_{k=0}^{n} C_k C_{n-k}$$

Input: Dòng duy nhất chứa số n $(0 \leq n \leq 5000)$.

Output: Dòng duy nhất chứa kết quả tìm được.

Ví dụ

Input	Output
3	14

Bài 22. Phân tích số

Cho một số nguyên dương S, S có thể phân tích thành dãy các số nguyên dương x_1, x_2, …, x_n sao cho $x_1 + x_2 + \ldots + x_n = S$. Bài toán đặt ra là phân tích S thành tổng các số nguyên dương sao cho ước chung lớn nhất của các hạng tử này là 1. Cụ thể, cần tính số lượng dãy x_1, x_2, …, x_n sao cho:

$$x_1 + x_2 + \ldots + x_n = S \text{ và } GCD(x_1,x_2,\ldots, x_n) = 1$$

Ví dụ: S=4 ta có các dãy sau {1, 1, 1, 1}, {2, 1, 1}, {1, 1, 2}, {1, 2, 1}, {1, 3}, {3, 1}.

Yêu cầu: Cho số nguyên dương S tính số cách phân tích S thành tổng các số nguyên dương sao cho ước chung lớn nhất giữa các số là 1.

Input: file Analyse.inp có cấu trúc như sau:

- Dòng đầu tiên chứa số nguyên dương t ($t \leq 10$) là số bộ test.
- Tiếp theo là t dòng, mỗi dòng chứa một số nguyên dương s ($s \leq 10^9$).

Output: ghi kết quả vào **file Analyse.out**

- Gồm t dòng, mỗi dòng là kết quả tương ứng mod module $10^9 + 7$.

Ví dụ

Analyse.inp	Analyse.out
2 1 4	1 6

Giải thích

- Trường hợp thứ nhất, chỉ có dãy {1} thoả mãn.
- Trường hợp thứ hai, có 6 dãy thoả mãn: {1, 1, 1, 1}, {1, 1, 2}, {1, 2, 1}, {1, 3}, {2, 1, 1}, {3, 1}.

Subtask:

- 30% test tương ứng với 30% số điểm có $S \leq 10$.
- 70% test tương ứng với 30% số điểm có $S \leq 10^9$.

Bài 23. Dãy số Fibonacci

Nhật Khôi rất thích nghiên cứu về toán. Bài toán hiện tại mà cậu ấy đang nghiên cứu là dãy Fibonacci với quy luật như sau:

$$f_0 = 0,\ f_1 = x;$$
$$f_n = f_{n-1} + f_{n-2}\ (\forall\ n > 1).$$

Nhật Khôi rất thích thú khi đã tính được tới số Fibonacci thứ n. Sau đó cậu quyết định đi ngủ. Trong lúc ngủ, không biết rằng Nga Hằng Nguyễn đã chui từ đâu ra và phá nát mất 2 số f_0 và f_1của Nhật Khôi. Nhật Khôi ngồi khóc một mình trong 4 bức tường vì cậu ấy không thể tìm ra được số x của mình. Điều mà Nhật Khôi vẫn còn nhớ trong đầu đó là số f_0 đầu tiên chắc chắn là số 0 và số n và giá trị f_n. Nhưng Nhật Khôi đã quên số x rồi.

Yêu cầu: Hãy giúp Nhật Khôi tìm lại số x của mình nhé!

Input: Vào từ file văn bản FIBONACCI.INP gồm hai số là lượt là n và f_n ($2 \le n \le 1000$).

Output: Ghi ra file văn bản FIBONACCI.OUT gồm một số nguyên duy nhất là số x.

Ràng buộc:

- Có 40% số lượng tests thỏa mãn điều kiện: $0 \le f_n \le 1000000$;
- 60% số lượng tests còn lại thỏa mãn điều kiện: $0 \le f_n \le 10^{18}$.

Ví dụ

FIBONACCI.INP	FIBONACCI.OUT
6 8	1

IV. KẾT LUẬN

Trong chuyên đề này tôi mới trình bày những bài tập cơ bản nhất về số học để học sinh tiếp cận và vận dụng được kiến thức số học kết hợp với kiến thức lập trình để giải quyết các bài toán có hiệu quả. Còn rất nhiều bài tập hay và khó mà trong chuyên đề này tôi chưa đề cập đến.

Chuyên đề có sử dụng một số bài tập trên trang lập trình trực tuyến SPOJ (có trích dẫn nguồn đầy đủ).

Tôi xin chân thành cảm ơn những tác giả có bài toán được lựa chọn trong chuyên đề. Thời gian viết chuyên đề có hạn, không tránh khỏi những sai sót rất mong sự đóng góp ý kiến của quý Thầy Cô để chuyên đề được tốt hơn.

Tôi xin chân thành cảm ơn!

V. TÀI LIỆU THAM KHẢO

[1] Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, Nguyên Thanh Tùng, tài liệu giáo khoa chuyên tin quyển 1, NXB Giáo dục, 2009;
[2] Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, Nguyên Thanh Tùng, tài liệu chuyên tin học bài tập quyển 1, NXB Giáo dục, 2013;
[3] Một số tài liệu khác của đồng nghiệp;
[4] Trang web https://vn.spoj.com/

Ý KIẾN CỦA TỔ NHÓM CHUYÊN MÔN

Ý KIẾN CỦA HỘI ĐỒNG KHOA HỌC CẤP TRÊN