CHUYÊN ĐỀ SỐ HỌC

A. MỞ ĐẦU

I. LÝ DO CHỌN ĐỀ TÀI

Số học là một trong những chuyên đề cơ bản khi bắt đầu dạy cho học sinh chuyên Tin ngay từ khi vào lớp 10. Nói như vậy nhưng thực chất các em đã được học từ khi bắt đầu học toán ở bậc tiểu học, THCS. Như vậy để làm tốt những bài tập về số học các em cần phải có một nền tảng về toán số tốt cùng với tư duy của Toán và kiến thức lập trình các em mới có thể giải quyết được thành công những bài toán số trong Tin học. Như vậy mỗi giáo viên khi bắt đầu dạy cho học sinh chuyên Tin đều cho các em làm những bài tập phần số học, thông qua đó trang bị thêm cho các em phần số học và cách giải quyết với từng loại bài toán. Do vậy trong chuyên đề này tôi chọn viết chuyên đề Số học nhằm hệ thống lại những kiến thức cần trang bị cho các em và cũng là tài liệu giảng dạy của mình. Mong muốn được chia sẻ và nhận được sự đóng góp của các quý Thầy cô.

II. MỤC ĐÍCH CỦA ĐỀ TÀI

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. Do đó cái cần để luyện cho các em chính là hệ thống bài tập phong phú và một số lợi ích của các hàm có sẵn trong C++. Trong chuyên đề này tôi giới thiệu một số lớp các bài tập từ đơn giản đến nâng cao để giúp các em giải quyết được bài tập và cũng là củng cố kiến thức cũng như hình thành tư duy Toán trong Tin học cho các em khi mới học lập trình.

B. NỘI DUNG

I. HỆ ĐẾM

Trong Tin học ngoài hệ đếm thập phân còn có hệ đếm nhị phân, hệ đếm Hecxa (cơ số 16)

Tổng quát với một số nguyên dương n bất kì luôn biểu diễn được duy nhất dưới dạng

$N=a_kb^k+ a_{k-1}b^{k-1}+ a_{k-2}b^{k-2}+ a_kb^k+\dots a_1b+ a_ob^o$

Bài 1.Viết chương trình nhập vào số nguyên dương n và cơ số b đưa ra màn hình dạng biểu diễn của nó ở hệ đếm cơ số trong cơ số b. Ví dụ n=446 , b=16 thì dạng biểu diễn là 1BE.

Để giải bài toán trên ta chỉ cần đem n chia liên tiếp cho 16 lấy phần dư, quá trình chia cho đến khi phần nguyên bằng 0 thì dừng lại. Để biết phầ dư từ 10 trở lên là kí tự nào trong hệ b ta sử dụng một xâu mẫu đề biết kí tự cần lấy.

Code chương trình:

```
const mau='0123456789ABCDEFGHI..Z';
Var kq:string;      n:longint;
function doiso(n:longint;b:byte):string;
var kq:string; r:longint;
begin
    kq:='';
```

```
    while n>0 do
     begin
       r:=n mod b;
       kq:=mau[r+1]+kq;     n := n div b;
    end;
exit(kq);
begin
        readln(n,b);
 writeln(doiso(n,b));
end.
```

Tuy nhiên nếu code bằng c++ các bạn sẽ thấy được lợi ích của hàm itoa(n,s,b) để đổi số nguyên n từ hệ thập phân sang hệ cơ số b. strupr(s) để chuyển các kí tự sang in hoa.

Code chương trình

```
#include <bits/stdc++.h>
using namespace std;
char s[256];
int n;
void xuli(){
   itoa(n,s,b);
   int n=strlen(s);
   strupr(s);
   puts(s);
}
int main()
{
   cin>>n>>b;
   xuli();
   return 0;
}
```

Bài 2. Tháng 6 năm 1973 Neil J.A. công bố công trình nghiên cứu về độ lặp bội của các số. Với số nguyên N cho trước, nếu nó có nhiều hơn 1 chữ số, thì người ta thay nó bằng tích các chữ số (trong dạng biểu diễn thập phân). Quá trình thay thế trên được lặp lại cho đến khi nhận được số có một chữ số.Ví dụ, với N = 679 ta có: **679** -> **378** -> **168** -> **48** -> **32** -> **6**.

Số 679 có gốc bội là 5, vì sau 5 lần biến đổi ta được số có 1 chữ số.

Viết chương trình xác định xem với số nguyên N cho trước. Hỏi xem nó có gốc bội là bao nhiêu?

Input: Gồm 1 số nguyên N ($1 \le N \le 10^9$).

Output: Số gốc bội tìm được.

Quá trình tính tích các chữ số của n khi n có lớn hơn 1 chữ số. Như vậy giải bài toán trên ta thấy nếu n>9 thì còn đi tìm tích các chữ số của n. Mỗi lần như vậy ta đếm số lần tính tích của các chữ số của n.

Code chương trình:

```
int n, res, m,u ;
int main() {
  cin>>n;
  res=0;
  while (n>9){
      m=1; u=n;
      do {
        m=m*(u% 10);
        u=u/10;
       }       while (u!=0);
      res++;
      n=m;
    }
  cout<<res;
}
```

Bài 3. Hóa ra ai cũng cần tiền, kể cả phù thủy. Họ sử dụng các đồng vàng, bạc và đồng, gọi tương ứng là Galeon, Sikel và Knat. Một Galeon ăn 17 sikel, một sikel ăn 29 knat. Mọi giá cả nêu sau đều theo các đơn vị kể trên. Trong mỗi giá số sikel không quá 16, số knat – không quá 28.

Trước khi vào nhập học ở Hogvard Harry Potter rút ở ngân hàng Gringot một số tiền để mua một số học cụ cần thiết như đậu thần, cú, chậu thiếc, áo choàng, . . . Số tiền Harry rút ra là ***g*** Galeo, ***s*** Sikel và ***k*** Knat. Harry cần mua tất cả là ***n*** thứ. Vật thứ ***i*** có giá là (p_i, q_i, r_i), $i = 1 \div n$, $(0 \le n \le 10^5)$.

Yêu cầu: Hãy xác định số tiền Harry còn lại sau khi sắm mọi thứ. Nếu Harry không đủ tiền thì đưa ra số -1.

Input:

+Dòng đầu tiên chứa 3 số nguyên ***g***, ***s*** và ***k*** $(0 \le g \le 10^5)$,

+Dòng thứ 2 chứa số nguyên ***n***,

+Dòng thứ ***i*** trong ***n*** dòng sau chứa 3 số nguyên p_i, q_i, r_i $(0 \le p_i, q_i, r_i \le 10^5)$.

Output*:*3 số nguyên xác định số tiền còn lại của Harry hoặc số -1.

Để giải bài toán trên chúng ta thấy ngay việc đầu tiên cần tính xem harry có bao nhiêu tiền khi quy về đơn vị knat. Sau đó tính xem số tiền phải trả khi mua n thứ là bao nhiêu knat. Nếu số tiền trả >tiền có in -1. Ngược lại tính số tiền theo các đơn vị galeo, sikel, knat.

Code chương trình:

```
int g,s,k,x,y,p,q,r;
int main() {
```

```
        cin>>g>>s>>k;
        x:=k+s*29+g*17*29;
        cin>>n;
        y=0;
        for (int i=1; i<=n; i++) {
            cin>>p>>q>>r;
            y:=y+r+q*29+p*17*29;
          }
        if (x<y) cout<<-1;  else
          {
            x=x-y;
            k=x %29;
            x=x / 29;
            s=x % 17;
            g=x / 17;
             cout<<g<< " "<<s<< "  "<<k;
       }
}
```

Bài 4.Hãy tìm số nguyên dương nhỏ nhất có chữ số hàng đơn vị là d ($1 \le d \le 9$) sao cho nếu chuyển chữ số hàng đơn vị lên trước chữ số đầu tiên thì ta được một số mới gấp k lần số cũ ($1 \le k \le 9$)

Dữ liệu: Vào từ file văn bản NUMBER.INP

Gồm nhiều dòng, mỗi dòng chứa hai số nguyên dương *d, k* cách nhau ít nhất một dấu cách.

Kết quả: Ghi ra file văn bản NUMBER.OUT

Gồm nhiều dòng, mỗi dòng ghi một số nguyên tìm được ứng với *d,k* ở dòng tương ứng của file dữ liệu. Ghi -1 nếu không có số thỏa mãn

Ví dụ:

NUMBER.INP	NUMBER.OUT
8 4	205128

Ghi chú: Có 50% số test kết quả không vượt quá 10^6.

Hướng dẫn:

sử số cần tìm là $x_n x_{n-1} \ldots x_1 d$. Sau khi đổi d lên trước vị trí đầu tiên thì số cần tìm là $d x_1 x_2 \ldots x_n$. Ta có

$$\begin{array}{r} x_n x_{n-1} \ldots x_1 d \\ \times \qquad\qquad k \\ \hline d x_n x_{n-1} \ldots x_1 \end{array}$$

Thực hiện phép nhân tay cho đến khi nhận được số d và giá trị nhớ bằng 0 ta được số cần tìm.

Ví dụ d=8, k=4 thì

Lấy d.k=8.4=32 do đó x_1=2 nhớ 3

Lấy x_1.k+nhớ=2.4+3 =11 do đó x_2=1 nhớ 1

Lấy x_2.k+nhớ=1.4+1=5 vậy x_3=5 nhớ 0

Lấy x_3.k+nhớ=5.4+0=20 do đó x_4=0 nhớ 2

Lấy x_4.k+nhớ=0.4+2=2 do đó x_5=2 nhớ 0

Lấy x_5.k+nhớ=2.4+0=8 do vậy x_6=8 =d và nhớ 0

Ta có số cần tìm là 205128

Nhận xét rằng các giá trị x và nhớ không vượt quá 10 nếu như số chữ số vượt quá, chẳng hạn 100 mà không quay trở lại d với nhớ=0 thì không có kết quả. Trường hợp này in ra -1

Chú ý khi in sẽ in mảng x ngược lại.

Code chương trình:

```
int d,k,n,x[101];
void tinh()
  {
    x[0]=d; int nho=0;
    for (int i=1; i<=101; i++)
     {
       x[i]=(x[i-1]*k+nho)%10;
       nho=(x[i-1]*k+nho)/10;
       if (nho==0 && x[i]==d && x[i-1]>0)
           {
             n=i-1;
             return;
            }
    }
    n=-1;
}
int main(){
    cin>>d>>k;
    tinh();
    if (n==-1) cout<<n; else
       for (int i=n; i>=0; i--) cout<<x[i];
    return 0;
}
```

Bài 5. Cho số nguyên dương *n*. Người ta phân tích *n* thành tổng các số nguyên dương theo qui tắc như sau: Nếu có thể phân tích *n* thành tổng hai số *x, y* mà hiệu của

chúng đúng bằng *k* cho trước thì phân tích. Nếu không thể phân tích *n* như trên thì để nguyên *n*. Các số *x, y* đến lượt mình lại được phân tích theo qui tắc nói trên.

Hỏi cuối cùng *n* được phân tích thành tổng của bao nhiêu số hạng

Ví dụ, nếu *n*=6; *k*=2 thì đầu tiên 6=4+2. Số 2 không thể phân tích được nữa tuy nhiên số 4 lại có thể phân tích 4=3+1. Số 3 và số 1 không phân tích được nữa. Như vậy cuối cùng 6 được phân tích thành tổng của ba số (6=3+1+2).

Input*:* Hai số n, k ($n,k \leq 10^9$).

Output*:*Số lượng số thu được khi phân tích n.

Hướng dẫn: Giả sử n phân tích thành tổng của hai số nguyên x và y. Khi đó điều kiện thỏa mãn là x-y=k. Do vậy ta có hệ phương trình

$$\begin{cases} x+y=n \\ x-y=k \end{cases} => \begin{cases} x=\frac{n+k}{2} \\ y=\frac{n-k}{2} \end{cases}$$

Do vậy từ x và y ta dễ dàng tìm được số lượng số thu được khi phân tích n

Code chương trình :

```
int f(int n)
{
  if (n<=k) return 1;
if ((n+k) %2!=0) return 1;
x=(n+k)/2;
y=(n-k)/2;
return (f(x)+f(y));
}
int main()
{
cin>>n>>k;
cout<<f(n) ;
}
```

II. ƯỚC SỐ

Cho hai số nguyên dương a, b luôn tồn tại duy nhất một cách viết a=kb+r với r∈{0, 1, 2, ..,b-1}. Khi đó người ta gọi k là thương của phép chia a cho b, còn r là phần dư của phép chia a cho b.

Để tìm UCLN của hai số nguyên a, b ta có thuật toán Euclid

```
int gcd(int x, int y)
        {
                int  r;
        do
        {
                r=x%y;
```

```
        x=y;
        y=r;
    }
      while (r!=0);
    return x ;
    }
```

Bài 1. Cho các số nguyên dương n, p, q, r ($n,p,q,r \leq 10^9$). Hãy đếm xem có bao nhiêu số nguyên dương trong đoạn [1,n] chia hết cho 2 trong ba số p, q, r nhưng không chia hết cho số còn lại.

Input*:*Gồm nhiều dòng, mỗi dòng ghi 4 số nguyên dương n, p, q, r.

Output:Mỗi dòng ghi kết quả ứng với dòng tương ứng trong input.

Hướng dẫn: Để giải bài toán trên, việc đầu tiên chúng ta phải tìm USCLN sau đó tìm BSCNN của hai số. Từ đó sẽ giải quyết được yêu cầu của bài toán.

Code chương trình :

```
int gcd(int x, int y)
    {
        int  r;
    do
    {
        r=x%y;
        x=y;
        y=r;
    }
      while (r!=0);
    return x ;
    }

int bscnn (int a, int b)
{
            int u=a, v=b;
            int x=gcd(a,b);
            return u*v/x;
}
int  dem(int n; int p; int q; int r)
{
   int a,b;
a=bscnn(p,q);
b=bscnn(a,r);
return n/a-n/b;
}
```

```
int main()
{
while (scanf("%d%d%d%d",&n,&p,&q,&r)>0)
{
        t1=dem(n,p,q,r);
        t2=dem(n,p,r,q);
        t3=dem(n,q,r,p);
       cout<<t1+t2+t3<<endl;
   }
```

Bài 2: Số nguyên ***a*** được coi là tốt hơn số nguyên ***b*** nếu tổng các chữ số của ***a*** lớn hơn tổng các chữ số của ***b***. Với hai số có tổng các chữ số bằng nhau, số bé hơn được coi là tốt hơn. Ví dụ, 124 tốt hơn 123, 3 tốt hơn 111.

Yêu cầu: Cho số nguyên ***n*** ($1 \leq n \leq 10^5$). Hãy tìm ước số tốt nhất của ***n***. Lưu ý là 1 và ***n*** cũng là các ước.

Input*:*Một số nguyên duy nhất n.

Output*:*Kết quả tìm được.

Hướng dẫn: Bài toán trên dễ dàng giải quyết được khi tính được tổng các chữ số của i (i=1..n) sau đó ta sẽ tìm được ước số tốt nhất.

Code chương trình:

```
#include<bits/stdc++.h>
using namespace std;
int n,res;
bool tothon(int a, int b)
{
   int ta=0,x=a;
   do
   {
      ta+=a%10;
      a=a/10;
   } while (a!=0);
   int tb=0,y=b;
   do
   {
      tb+=b%10;
      b=b/10;
   } while (b!=0);
   if ((ta>tb) || (ta==tb && x<y)) return true;
   return false;
}
int main()
```

```
{
   cin>>n;
   res=1;
   for (int i=2; i<=n; i++)
      if (n%i==0)
      if (tothon(i,res)) res=i;
   cout<<res;
   return 0;
}
```

Bài 3. Với số nguyên dương x ($1 \le x \le 10^9$), ký hiệu $s(x)$ là tổng các chữ số các ước của x. Ví dụ $s(6) = 1+2+3+6 = 12$, $s(10) = 1+2+5+1+0 = 9$.

Xét dãy số $a_1 = x$, $a_2 = s(x)$, $a_3 = s(s(x))$, . . ., $a_n = s(a_{n-1})$, . . . Nói dãy số này ổn định , nếu tồn tại một i nào đó sao cho $a_i = a_{i+1}$.

Yêu cầu: Cho số nguyên dương x. Hãy xác định xem dãy số a_n có ổn định hay không, nếu có thì chỉ ra i nhỏ nhất thỏa mãn điều kiện $a_i = a_{i+1}$.

Input: Một dòng duy nhất chứa số nguyên x.

Output*:*Chỉ số n nếu dãy số ổn định. Nếu thử với n>1000 vẫn chưa ổn định thì in -1.

Hướng dẫn: Để giải bài toán trên, việc đầu tiên ta đi tích tổng các chữ số của các ước của ai là sum(ai). sau đó so sánh sum(ai)=ai thì in ra i và dừng lại còn không thì i tăng lên 1, quá trình lại tiếp tục khi i<=1000.

Code chương trình :

```
int sum(int x)
{
   int res=0;
   do
   {
      res+=x %10;
      x=x/10;
   } while (x!=0);
   return res;
}
int sumdigit(int x)
{
   if (x==1) return 1;
   int res=0;
   for (int i=1; i<=int(sqrt(x)); i++)
    if (x%i==0)
   {
      res+=sum(i);
```

```
        if (x/i!=i) res+=sum(x/i);
    }
    return res;
}
int main()
{
    cin>>x;
    int a=x;
    int ds=-1;
    int n=1;
    int an;
    while (n<=1000)
    {
        an=sumdigit(a);
        if (an==a)
        {
            ds=n;
            break;
        }
        a=an;
        n++;
    }
    if (ds!=-1) cout<<ds;
    return 0;
}
```

Bài 4. Ginny bắt đầu học số học và đặc biệt yêu thích phân số. Kiến thức về phân số mà Ginny được làm quen chỉ mới giới hạn trong phạm vi đơn giản: tử số và mẫu số đều là số nguyên, phân số đúng (tử số nhỏ hơn mẫu số và phân số tối giản. Ginny thường nghĩ ra các bài toán để tự giải hoặc trao đổi với các bạn trong lớp. Một trong số các bài toán đó có nội dung như sau.

Yêu cầu: Cho số nguyên ***n*** ($3 \leq \boldsymbol{n} \leq 1\ 000$). Hãy tìm phân số tối giản đúng lớn nhất có tổng tử số và mẫu số bằng n.

Input: Một dòng duy nhất chứa số nguyên n.

Output: Phân số tìm được dưới dạng a/b.

Hướng dẫn: Phân số tối giản là phân số luôn có ước số chung lớn nhất của tử số và mẫu số bằng 1. Do vậy bài toán trở về tìm gcd(a,b) sau đó tìm hai số a, b thỏa mãn có gcd(a,b)=1 và a<b và a+b=n;

Code chương trình :

```
int gcd(int a, int b)
{
```

```
    int r;
    do
    {
       r=a%b;
       a=b;
       b=r;
    } while (r!=0);
    return a;
}
int main()
{
    freopen("inp.txt","r",stdin);
    freopen("out.txt","w",stdout);
    cin>>n;
    int j;
    for (int i=1; i<=n; i++)
    {
       j=n-i;
       if (j<=i) break;
       if (gcd(i,j)==1 && maxkq<double(i)/j)
       {
          maxkq=i/j;
          io=i;
          jo=j;
       }
    }
    cout<<io<<"/"<<jo;
    return 0;
}
```

Bài 5 . Số thân thiện

Số tự nhiên có rất nhiều tính chất thú vị. Ví dụ số 23, số đảo ngược của nó là 32. Hai số này có USCLN là 1. Những số như thế được gọi là số thân thiện, tức là 23 là số thân thiện, số 32 cũng được gọi là số thân thiện.

Hãy nhập vào 2 số nguyên a, b (10<=a, b<=30000). Hãy đếm xem trong khoảng từ a, đến b kể cả a, b có bao nhiêu số thân thiện?

Input: file numfre.inp bao gồm một dòng chứa 2 số a, b cách nhau một khoảng trắng.

Output: file numfre.out bao gồm một dòng là kết quả của bài toán

Ví dụ:

numfre.inp	numfre.out
20 30	3

Hướng dẫn: Ta tìm số đảo ngược của x là y. Sau đó tìm gcd(x,y), nếu gcd(x,y)=1thì kết quả được ghi nhận.

Code chương trình :

```
int gcd(int x, int y)
{
        int  r;
        do
        {
                r=x%y;
                x=y;
                y=r;
        }
           while (r!=0);
        return x ;
}
int invert(int x)
{
   int y=0;
   while (x>0)
    {
      y=y*10+x%10;
      x=x/10;
   }
   return y;
}
int main()
{
      int res=0;
      for (int i=a; i<=b; i++)
      {
          int  j=invert(i);
          if (gcd(i,j)==1) res++;
      }
      printf("%d",res);
}
```

III. SỐ NGUYÊN TỐ

Để kiểm tra một số a có là số nguyên tố hay không các em dễ dàng kiểm tra được n là nguyên tố khi xét với i=2.. $[\sqrt{a}]$ nếu không có số i nào mà n mod i=0. Tuy nhiên ta thấy cách này không hiệu quả khi thời gian kiểm tra lâu. Cải tiến thay vì xét từ 2.. $[\sqrt{a}]$ ta chỉ cần thử đối với các số nguyên tố từ 2.. $[\sqrt{a}]$: 6k, 6k+1, 6k+2, 6k+3, …

```
bool  isprime(int n){
    if (n==2 || n==3 || n==5) return true;
    if (n<6) return false;
   int k=5, d=2;
    while (k<=int(sqrt(n)) {
          if (n%k==0) return false;
          k+=d;
        d=6-d;
    }
    return true;
    }
```

Sàng số nguyên tố

Sàng số nguyên tố là cách nhanh nhất để liệt kê các số nguyên tố trong một phạm vi cho trước. Nguyên lí sàng là: Xóa bỏ số 1 ra khỏi tập số nguyên tố, số tiếp theo số 1 là số 2, là số nguyên tố, xóa tất cả các số là bội của 2 ra khỏi bảng. số 3 là số nguyên tố, xóa tất cả các số là bội của 3, … thuật toán tiếp tục như vậy cho đến khi gặp số nguyên tố lớn hơn $\sqrt{N}$ thì dưng lại.

```
void  sangnt(int n)
{
      for (int i=1; i<=n; i++)  nt[i]=0;
      nt[1]=1;
      for (int i=2; i<=int(sqrt(n)) do
        {
          if (nt[i]==0) {
             int j=2;
             while (i*j<=n)
                  {
                        nt[i*j]=1;
                        j++;
                   }
      }
    }
}
```

Bài 1. Tổng các số nguyên tố

Cho hai số nguyên dương A, B hãy tính tổng các số nguyên tố nằm trong $[A, B]$?

Dữ liệu: Vào từ file văn bản SUMP.INP

- Dòng đầu tiên ghi số nguyên dương $T \leq 10^5$ là số lượng các yêu cầu
- T dòng tiếp theo, mỗi dòng ghi hai số nguyên dương A, B $(1 \leq A, B \leq 10^5)$ là một yêu cầu cần phải thực hiện

Kết quả: Ghi ra file văn bản SUMP.OUT gồm T dòng, mỗi dòng ghi một số nguyên là tổng các số nguyên tố tìm được của câu hỏi tương ứng (theo thứ tự trong file dữ liệu)

Ví dụ

SUMP.INP	SUMP.OUT
2 1 20 10 20	77 60

Ghi chú: Có 75% số test có $T \leq 100$

Hướng dẫn: Ta sử dụng sàng số nguyên tố để có các số nguyên tố trong phạm vi <=50000; sau đó sử dụng mảng s để tính tổng các số nguyên tố. kết quả cần tìm tổng các số nguyên tố trong đoạn [A..B] chính là s[B]-s[A-1];

Code chương trình :

```
#define maxn 100000
void sang(int x)
{
   for (int i=1; i<=x; i++) nt[i]=0;
   nt[1]=1;
   for (int i=2; i<=int(sqrt(x)); i++)
      if (nt[i]==0)
   {
      int j=2;
      while (i*j<=x)
      {
         nt[i*j]=1;
         j++;
      }
   }
}
int main()
{
   freopen("sump.inp","r",stdin);
   freopen("sump.out","w",stdout);
   sang(maxn);s[1]=0;
   for (int i=2; i<=maxn; i++)
```

```
    if (nt[i]==0) s[i]=s[i-1]+i;
   else s[i]=s[i-1];
  cin>>t;
  for (int k=1; k<=t; k++)
  {
    cin>>a>>b;
    cout<<s[b]-s[a-1]<<endl;
  }
  return 0;
}
```

Bài 2: Nói số ***a*** tốt hơn ***b*** nếu tổng bình phương các chữ số của ***a*** (trong hệ cơ số 10) lớn hơn tổng bình phương các chữ số của ***b*** hoặc các tổng này bằng nhau nhưng ***a*** < ***b***.

Yêu cầu: Cho hai số nguyên ***l*** và ***r*** ($2 \le l \le r \le 50\ 000$). Hãy tìm số nguyên tố tốt nhất trong khoảng [***l***, ***r***]. Nếu trong khoảng này không có số nguyên tố nào thì đưa ra số -1.

*Input:*Một dòng chứa hai số l, r.

Output: Kết quả tìm được.

Ví dụ:

Inp.txt	Out.txt
20039 20704	20599

Hướng dẫn: Để tìm số nguyên tố tốt nhất trong khoảng [l..r] việc đầu tiên ta phải làm là sàng nguyên tố trong đoạn <=r. Sau đó duyệt trong đoạn [m..n] số nguyên tố nào thỏa mãn điều kiện là tốt nhất thì lưu lại.

Code chương trình:

```
int m,n,nt[50005];
void sang(int x)
{
  for (int i=1; i<=x; i++) nt[i]=0;
  nt[1]=1;
  for (int i=2; i<=int(sqrt(x)); i++)
    if (nt[i]==0)
  {
    int j=2;
    while (i*j<=n)
    {
      nt[i*j]=1;
      j++;
    }
```

```
    }
}
int sums(int x)
{
    int res=0;
    while (x>0)
    {
        int r=x%10;
        res+=(r*r);
        x=x/10;
    }
    return res;
}
int main()
{
    freopen("inp.txt","r",stdin);
    freopen("out.txt","w",stdout);
    cin>>m>>n;
    sang(n);
    int max=0,kq=-1;
    for (int i=m; i<=n; i++)
        if (nt[i]==0)
    {
        int t=sums(i);
        if (t>max || max==0)
        {
            max=t;
            kq=i;
        }
    }
    cout<<kq;
    return 0;
}
```

Bài 3: Số phong phú

Trong số học, số phong phú là các số mà tổng các ước số của số đó (không kể chính nó) lớn hơn số đó. Ví dụ, số 12 có tổng các ước số (không kể 12) là 1 + 2 + 3 + 4 + 6 = 16 > 12. Do đó 12 là một số phong phú. Bạn hãy lập trình đếm xem có bao nhiêu số phong phú trong đoạn [L,R].

Input: File RICHNUM.INP gồm 2 số L, R ($1 \leq L \leq R <= 10^5$)

Output: File RICHNUM.OUT gồm 1 số nguyên duy nhất là số số phong phú trong đoạn [L, R].

Example:

RICHNUM.INP	RICHNUM.OUT
1 50	9

Giải thích: Từ 1 đến 50 có 9 số phong phú là: 12, 18, 20, 24, 30, 36, 40, 42, 48

Hướng dẫn: Để đếm số phong phú trên đoạn [l..r] ta chỉ cần so sánh tổng các ước của i lớn hơn i thì i là số phong phú và ghi nhận kết quả (res++). Để tính tổng các ước của i ta thấy nếu i là số nguyên tố thì loại i mà không cần tìm tổng các ước. Do đó việc đầu tiên ta sàng nguyên tố. sau đó chỉ đi tính tổng các ước của các số không là số nguyên tố.

Code chương trình :

```
#define maxn 100000
using namespace std;
int l,r,s[maxn+1];
int nt[maxn+1];

void sang(int x)
{
   for (int i=1; i<=x; i++) nt[i]=0;
   nt[1]=1;
   for (int i=2; i<=int(sqrt(x)); i++)
      if (nt[i]==0)
   {
      int j=2;
      while (i*j<=x)
      {
         nt[i*j]=i;
         j++;
      }
   }
}
int main()
{

   freopen("richnum.inp","r",stdin);
   freopen("richnum.out","w",stdout);
   sang(maxn);
   cin>>l>>r;
   s[2]=1; s[2]=1; s[1]=0;
   for (int i=2; i<=r; i++)
      if (nt[i]==0) s[i]=1;
     else{
      s[i]=0;
      int p=nt[i];
      while (i%p==0)
```

```
        {
           s[i]=s[i]+p;
           p=p*nt[i];
        }
        p=p/nt[i];
        int k=i/p;
        s[i]=(s[i]+1)*(s[k]+k)-i;
        }
     int res=0;
     for (int j=l; j<=r; j++)
        if (s[j]>j) res++;
     cout<<res;
     return 0;
  }
```

Bài 4. Giả số nguyên tố

Giả sử *b* là một số nguyên dương. Nếu *p* là hợp số nguyên dương và b^p chia cho *p* được số dư là *b* thì *p* được gọi là giả số nguyên tố cơ sở *b*.

Yêu cầu: Cho *n* là một số nguyên dương hãy liệt các giả số nguyên tố cơ sở 2 trong phạm vi từ 1 đến *n*.

Dữ liệu: Vào từ file văn bản PSEPRIME.INP gồm một dòng chứa số nguyên dương $n \leq 10^6$.

Kết quả: Ghi ra file văn bản PSEPRIME.OUT là các giả số nguyên tố cơ sở 2 trong phạm vi từ 1 đến *n*, mỗi số ghi trên một dòng theo thứ tự tăng dần. Nếu không tìm được số thỏa mãn yêu cầu, ghi ra số 0.

Ví dụ:

PSEPRIME.INP	PSEPRIME.OUT
1000	341 561 645

Hướng dẫn:

+Thuật toán 1:

Bước 1: Sử dụng công thức *(A × B) mod C = ((A mod C) ×(B mode C)) mode C* để tính 2^p mod *p*.

Bước 2: Sử dụng sàng nguyên tố để đánh dấu các số nguyên tố trong phạm vi từ 1 đến *n*.

Bước 3**:** So sánh, kiểm tra lần lượt từng số từ 1 đến *n*.

Với thuật toán này độ phức tạp $O(n^2)$ sẽ có được 60% số test ứng với 60% số điểm của bài có $n \leq 100000$.

+ Thuật toán 2:

Tương tự như thuật toán 1, có sử dụng kỹ thuật chia để trị để tính 2^n mod *n*. Trong bước 3, ta chỉ cần kiểm tra các số lẻ trong phạm vi từ 1 đến *n*.

Với thuật toán này độ phức tạp $O(n\log n)$ có thể được 100% số điểm của bài toán.

Code chương trình :

```
void sang(int n)
{
   for (int i=1; i<=n; i++)  sang[i]=0;
   sang[1]=1;
   for (int i=2; i<=int(sqrt(n)); i++)
      if (sang[i]==0)
   {
      int j=2;
      while (i*j<=n)
      {
         sang[i*j]=1;
         j++;
      }
   }
}
int powemod(int a,int n,int m)
{
   if (n==0) return 1;
   int t=powemod(a,n/2,m);
   if (n%2==0) return (t*t)%m;
   else return (t*t*a)%m;
}
int main()
{
   freopen("pseprime.inp","r",stdin);
   freopen("pseprime.out","w",stdout);
   cin>>n;
   sangnt(n);
   bool ok=false;
   for (int i=3; i<=n; i++)
      if (sang[i]==1)
   {
      if (powemod(2,i,i)==2)
      {
         cout<<i<<"\n";
         ok=true;
      }
   }
```

```
    if (!ok) cout<<0;
    return 0;
}
```

Bài 5. Phân tích số

Cho số nguyên dương M. Hỏi có thể phân tích M thành tổng của 2 hoặc 3 số nguyên tố khác nhau được không?

Dữ liệu: vào từ tệp văn bản NGUYENTO.INP gồm 1 số nguyên dương M ($M \leq 10^4$)

Kết quả: ghi ra tệp văn bản NGUYENTO.OUT gồm 1 số là số cách phân tích N thành tổng của 2 hoặc 3 số nguyên tố khác nhau.

Ví dụ:

NGUYENTO.INP	NGUYENTO.OUT
10	2

Giải thích **:** 1. 10 = 3 + 7 và 2. 10 = 2 + 3 +5

Hướng dẫn:

+ Thuật toán 1:

- Viết 1 hàm kiểm tra tính nguyên tố của một số nguyên
- Dùng 2 vòng for để đếm xem có bao nhiêu cặp hai số có tổng = M

```
For i:=2 to m-1 do
  For j:=i+1 to m do if (NT(i)) and (NT(j)) and (i+j=M) then inc(dem);
```

- Dùng 3 vòng for lồng nhau để đếm xem có bao nhiêu cặp 3 số NT có tổng =M

* Nhận xét làm cách này sẽ chạy mất nhiều thời gian nếu M lớn

+ Thuật toán 2:

- Dùng mảng 1 chiều để lưu các số nguyên tố từ 1 đến M (có K số nguyên tố) (Có thể dùng lệnh đếm từng số hoặc dùng sàng để lấy ra các số nguyên tố)
- Dùng 2 vòng for để đếm ra kết quả:

Code chương trình :

```
void sang(int n)
 {
    for (int i=1; i<=n; i++)  nt[i]=0;
    nt[1]=1;
    for (int i=2; i<=int(sqrt(n)); i++)
       if (nt[i]==0)
    {
       int j=2;
       while (i*j<=n)
       {
          nt[i*j]=1;
          j++;
       }
    }
```

```
}
int main()
{
   cin>>n;
   sang(n);
   int k=0;
   int dem=0;
   for (int i=1; i<=n; i++)
    if (nt[i]==0)    a[++k]=i;
   for (int i=1; i<=k/2; i++)
    {
     if (nt[n-a[i]]==0 && a[i]<n-a[i] ) dem++;
     for(int j=i+1; j<=k-1; j++)
       if (nt[n-a[i]-a[j]]==0&& a[j]<n-a[i]-a[j]) dem++;
   }
   cout<<dem;
   return 0;
}
```

IV. MỘT SỐ BÀI TẬP THAM KHẢO

Bài 1. Táo quân

Có m ông táo và n bà táo được Ngọc Hoàng phân công nhiệm vụ trong năm mới. Đầu tiên Ngọc Hoàng chọn k táo (ông hoặc bà) làm những nhiệm vụ đặc biệt tại các Bộ/Ngành, sau đó Ngọc Hoàng sẽ chọn ra các nhóm, mỗi nhóm gồm đúng 2 ông táo và 1 bà táo để xuống các gia đình dưới hạ giới.

Yêu cầu: Hãy giúp Ngọc Hoàng xác định số nhóm nhiều nhất để phân xuống các gia đình dưới hạ giới.

Ví dụ có $m = 12$ ông táo và $n = 7$ bà táo, có $k = 5$ táo phải làm nhiệm vụ đặc biệt. Ngọc Hoàng có thể chọn tối đa 4 nhóm phân xuống các gia đình (8 ông táo, 4 bà táo). Trong 7 táo còn lại (4 ông và 3 bà) có 5 táo làm nhiệm vụ đặc biệt 2 táo không được phân việc (thất nghiệp!!!)

Dữ liệu: Vào từ file văn bản LARES.INP gồm 1 dòng chứa ba số nguyên dương $m, n, k \leq 10^9$ cách nhau ít nhất một dấu cách.

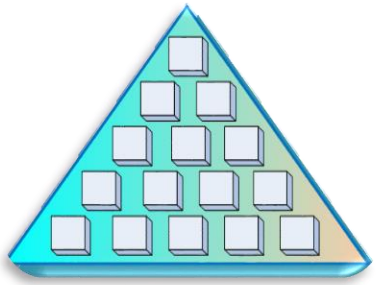
Kết quả: Ghi ra file văn bản LARES.OUT một số nguyên là số nhóm nhiều nhất chọn được để phân xuống các gia đình dưới hạ giới.

Ví dụ

LARES.INP	LARES.OUT
12 7 5	4

Bài 2. Trung Thu

Nhân dịp tết Trung thu thành phố tổ chức một buổi đón trăng chung cho tất cả thiếu nhi của thành phố tại quảng trường chính. Ngoài các tiết mục liên hoan văn nghệ và hoa quả truyền thống thành phố còn đặt hàng làm một hộp kẹo đường phèn lớn. Kẹo đường phèn được đặt trong các hộp hình tam



giác, mỗi viên đường phèn có hình lập phương, sắp thành ***k*** hàng, hàng thứ ***i*** có ***i*** viên. Thống kê cho thấy sẽ có ***m*** em tới dự lễ Trung thu. Ban tổ chức muốn có một hộp kẹo sao cho có thể chia đều cho mỗi em một số lượng viên đường như nhau và không được sót lại viên nào trong hộp.

Nhà máy bánh kẹo có thể sản xuất các loại hộp với số hàng nằm trong phạm vi từ 1 đến ***n***. Như vậy, nếu ***n*** = 20 và số em đến dự là 10 (***m*** = 10) thì có thể dùng các hộp kẹo loại 4 hàng, 15 hàng, 19 hàng hoặc 20 hàng, nghĩa là có 4 cách để Ban tổ chức lựa chọn đặt hàng.

Input: Một dòng duy nhất ghi hai số nguyên n, m (n, m ≤10000)

Output: Số cách lựa chọn.

Bài 3. Lũy thừa

Số biển xe của thầy Bình là 1666 là con số đẹp. Do đó thầy định nghĩa mọi lũy thừa của nó cũng là con số đẹp. Hãy viết chương trình tính tổng các ước của một số đẹp của thầy

Dữ liệu: Vào từ file văn bản BEAUTY.INP một số nguyên không âm $n \leq 10^9$

Kết quả: Ghi ra file văn bản BEAUTY.OUT một số nguyên là tổng các ước không âm của 1666^n. Vì con số này là rất lớn nên thầy chỉ yêu cầu các bạn học sinh in ra phần dư của nó khi chia cho số 47 (là tuổi của thầy, đồng thời cũng là một số nguyên tố!!!)

Ví dụ

BEAUTY.INP	BEAUTY.OUT
0	1

Bài 4. SỐ ĐẶC BIỆT

Một số nguyên dương M được gọi là một số đặc biệt nếu nó thỏa mãn: Tổng các chữ số của M bằng tổng các chữ số của các thừa số nguyên tố là tích của M. Chẳng hạn như số 4937775 là một số đặc biệt vì: 4937775 = 3 . 5 . 5 . 65837

Ta có: 4 + 9 + 3 + 7 + 7 + 7 + 5 = 42

Và: 3 + 5 + 5 + 6 + 5 + 8 + 3 + 7 = 42

Yêu cầu: Cho trước một số nguyên dương N. Tìm số đặc biệt nhỏ nhất lớn hơn N.

Dữ liệu vào: Từ file văn bản SODB.INP

Gồm duy nhất 1 dòng là 1 số nguyên dương N ($N< 10^9$)

Kết quả: Ghi ra file văn bản SODB.OUT

Gồm duy nhất 1 dòng là 1 số đặc biệt nhỏ nhất lớn hơn N

Ví dụ:

SODB.INP	SODB.OUT
4937770	4937775

Bài 5: Xét băng giấy có độ dài 2^K ô và độ rộng một ô. Các ô dược đánh số từ trái sang phải, bắt đầu từ 1. Người ta gập đôi băng giấy sao cho các ô đầu tiên nằm ở

lớp dưới. Như vậy băng giấy trở thành hai lớp và độ dài còn một nửa. Người ta cứ gập đôi như vậy cho đến khi nó có 2^K lớp.

Yêu cầu: Cho K và N ($1 \le K \le 30$, $1 \le N \le 2\ 000\ 000\ 000$), hãy xác định ô thứ N nằm ở lớp thứ mấy từ dưới lên.

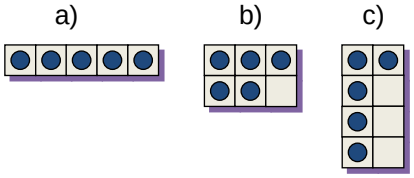
Input: Hai số nguyên K và N.

Output:Kết quả tìm được (-1 nếu băng giấy không có ô N).

Bài 6. XẾP ĐÁ

Cuội rất thích chơi một trò chơi với bộ sưu tập gồm n viên đá của mình: Xếp n viên đá lên một bảng hình chữ nhật chia thành lưới ô vuông đơn vị, sao cho mỗi ô có không quá một viên đá.

Ví dụ với $n = 5$, Cuội có thể xếp chúng vào bảng kích thước 1×5 (Hình a), 2×3 (Hình b) hay 4×2 (Hình c)…



Yêu cầu: Xác định kích thước của bảng có chu vi nhỏ nhất mà Cuội có thể thực hiện được trò chơi.

Dữ liệu: Vào từ file văn bản STONES.INP gồm một dòng chứa số tự nhiên $n < 2^{31}$.

Kết quả: Ghi ra file văn bản STONES.OUT hai số cách nhau một dấu cách là độ dài hai cạnh của bảng tìm được

Ví dụ:

STONES.INP	STONES.OUT
2 **5**	**1 2** **3 2**

Bài 7. Tìm vàng

Tương truyền rằng, ngày xưa có một mưu sĩ thấy dân chúng quá nghèo khổ nên ông ta đã đến thách đố đánh cờ cùng nhà vua nhằm lấy thóc trong kho đem phân phát cho dân nghèo. Nhà vua ra điều kiện nếu đánh thua nhà vua thì mưu sĩ sẽ bị chém đầu, ngược lại mưu sĩ sẽ được trọng thưởng bằng vật chất. Nếu đánh thắng cờ với nhà vua, mưu sĩ chỉ xin một điều đó là trong mỗi ô cờ gồm 8x8 ô thì lần lượt bỏ vào ô thứ 1: 1 hạt thóc, ô thứ 2: 1x2 hạt thóc, ô thứ 3: 1x2x3 hạt thóc,… cho đến ô cuối cùng. Nhà vua nghe qua rất khoái chí và đồng ý ngay. Sau lần đấu cờ đó nhà vua đã mất rất nhiều kho lương thực cho dân nghèo.

Do bản tính hiếu thắng của nhà vua, ông vẫn tiếp tục thách đấu với những tay cao thủ cờ khác trong thiên hạ nhưng bây giờ rút kinh nghiệm ông chỉ xuất trong kho ra bây giờ không phải là thóc nữa mà là vàng. Nguyên tắc để nhận được vàng sau khi đánh thắng nhà vua như sau:

1. Mỗi ô trong bàn cờ có một số. Con số này được gán vào như sau:
   - Ô số 1: 1
   - Ô số 2: 1x2 = 2
   - Ô số 3: 1x2x3 = 6
   - … …

- Ô số 10 1x2x3x….x10 = 3 628 800

… …

- Ô số 21 1x2x3x….x21 = 51 090 942 171 709 440 000

… …

2. Số vàng nhận được chính là con số khác không đầu tiên kể từ hàng đơn vị lên phía trước của ô mà đối thủ sẽ chọn. Ví dụ, chọn ô số 10 thì sẽ được 8 lạng vàng, ô số 21 sẽ được 4 lạng vàng, …

3. Đối thủ chỉ được chọn mỗi lần một ô để nhận vàng. Không được phép chọn các ô bé hơn 5.

4. Bàn cờ dùng thi đấu là bàn cờ 8x8, nhưng bàn cờ để chọn vàng là NxN, các ô được đánh số liên tục từ 1đến N.

Yêu cầu: Tìm số vàng mà đấu thủ nọ nhận được khi chọn một ô.

Dữ liệu: Vào từ file văn bản GOLD.INP gồm 1 số nguyên N là số thứ tự của ô mà đấu thủ chọn ($1 \leq N \leq 10\ 000$)

Kết quả*:* Ghi ra file văn bản GOLD.OUT 1 số duy nhất là số vàng đấu thủ nhận được

Ví dụ:

GOLD.INP	GOLD.OUT
10	8

C. 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. Thời gian viết chuyên đề có hạn, không tránh khỏi những sai sót, rất mong được sự góp ý của các thầy cô và các em học sinh.

Tôi cũng mong muốn khi viết chuyên đề để có được lời giải của bài toán mong các thầy cô cho chi tiết từ hướng dẫn phân tích thuật toán đến cài đặt chương trình để chúng tôi học hỏi được nhiều hơn.

D. TÀI LIỆU THAM KHẢO

1. Tài liệu Chuyên tin quyển 1 – Hồ Sỹ Đàm chủ biên.
2. Một số tài liệu khác của đồng nghiệp.