CHIA HẾT - SỐ NGUYÊN TỐ – SÀNG SỐ NGUYÊN TỐ

I- CHIA HẾT – SỐ NGUYÊN TỐ

1. Lý thuyết

- Sử dụng các phép toán cơ bản C++: +, -, *, / (chia lấy phần nguyên), % (Chia lấy phần dư)

- Các phép toán so sánh trong C++: >, <, ==, >=, <=, !=

* Các bài toán thường áp dụng: tìm ước, phân tích thành thừa số nguyên tố, lũy thừa, giai thừa, ước chung lớn nhất, bội chung nhỏ nhất,…

*** Thuật toán tìm ước chung lớn nhất**

Thuật toán Euclid:

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

Chú ý: Dựa trên thuật toán tính UCLN ta có thể kiểm tra được 2 số nguyên tố cùng nhau hay không. Ngoài ra cũng có thể dùng để tối giản phân số bằng cách chia cả tử và mẫu cho UCLN.

$$BCNN(a,b)=\frac{a*b}{UCLN(a,b)}$$

*** Thuật toán kiểm tra số nguyên tố**

Thuật toán của ta dựa trên ý tưởng: nếu n >1 không chia hết cho số nguyên nào trong tất cả các số từ 2 đến $\sqrt{n}$ thì n là số nguyên tố.

Hàm kiểm tra nguyên tố nhận vào một số nguyên n và trả lại kết quả là true (đúng) nếu n là nguyên tố và trả lại false nếu n không là số nguyên tố.

```
bool isPrime(long n){
   if (n<2){
         return false;
   }
   for (int i=2;i*i<=n;i++){
         if (n % i == 0){
                  return false;
         }
   }
```

```
    return true;
}
```

Chú ý: Thuật toán này chỉ có lợi khi duyệt số lượng số nguyên tố ít. Vì mỗi lần duyệt 1 số n sẽ phải chạy vòng lặp 2 đến $\sqrt{n}$

2. Một số ví dụ

Ví dụ 1: (UOCSON.CPP) Cho số nguyên dương n ($n<=10^9$). Tìm tất cả các ước số của n

Dữ liệu vào: File UOCSON.INP: Gồm số nguyên dương n

Dữ liệu ra: File UOCSON.OUT: Gồm các ước của n, mỗi ước cách nhau bởi dấu khoảng trắng

Ví dụ

UOCSON.INP	UOCSON.OUT
12	1 2 3 4 6 12

Phân tích:

Ước của n sẽ nằm trong khoảng từ 1 đến n/2 và ước cuối cùng là n.

Tìm ước bằng cách kiểm tra n%i ==0 hay không (1<=i<=n/2)

Chương trình tham khảo:

```
#include <iostream>
#include <fstream>
using namespace std;
#define tfi "UOCSON.INP"
#define tfo "UOCSON.OUT"
long n;
void nhap(){
    freopen(tfi, “r”, stdin);
    cin>>n;
}
void xuly(){
    freopen(tfo, “w”, stdout);
    for(long i=1;i<=n/2;i++){
        if(n%i==0){
            cout<<i<<" ";
```

```
        }
    }
    cout<<n;
}
int main() {
    nhap();
    xuly();
}
```

Ví dụ 2: (TONGCHUSO.CPP) Cho số nguyên dương n ($n<=10^9$). Tính số lượng các chữ số và tổng các chữ số của n

Dữ liệu vào: File TONGCHUSO.INP: Gồm số nguyên dương n

Dữ liệu ra: File TONGCHUSO.OUT:

Dòng 1: gồm 1 giá trị là số lượng chữ số của n

Dòng 2: Giá trị tổng của các chữ số

Ví dụ

TONGCHUSO.INP	TONGCHUSO.OUT
123	3 6

Chương trình tham khảo:

```
#include <stdio.h>
#include <fstream>
using namespace std;
#define tfi "TONGCHUSO.INP"
#define tfo "TONGCHUSO.OUT"
long n;
void nhap(){
    freopen(tfi, “r”, stdin);
    cin>>n;
}
void xuly(){
    long tong=0;
```

```
   int sochuso=0;
   freopen(tfo, "w", stdout);
   long tmp=n;
   while(tmp!=0){
      tong=tong + tmp%10;
      tmp=tmp/10;
      sochuso++;
   }
   cout<<sochuso<<endl;
   cout<<tong;
}
int main() {
   nhap();
   xuly();
}
```

Ví dụ 3: (PRIMEMAX.CPP) Cho số nguyên dương n ($n<=10^9$). Tìm số nguyên tố lớn nhất bé hơn n.

Dữ liệu vào: File PRIMEMAX.INP: Gồm số nguyên dương n

Dữ liệu ra: File PRIMEMAX.OUT:

Giá trị số nguyên tố lớn nhất bé hơn n. Nếu không có thì ghi -1

Ví dụ

PRIMEMAX.INP	PRIMEMAX.OUT
100	97

Chương trình tham khảo:

```
#include <stdio.h>
#include <fstream>
using namespace std;
#define tfi "PRIMEMAX.INP"
#define tfo "PRIMEMAX.OUT"
long n;
void nhap(){
```

```
    freopen(tfi, “r”, stdin);
    cin>>n;
}
bool isPrime(long n){
    if (n<2){
        return false;
    }
    for (int i=2;i*i<=n;i++){
        if (n % i == 0){
            return false;
        }
    }
    return true;
}
void xuly(){
    freopen(tfo, “w”, stdout);
    long i=n-1;
    while(i>1){
       if(isPrime(i)){
          cout<<i;
          break;
       }
       i--;
    }
    if(i==1) cout<<-1;
}
int main() {
    nhap();
    xuly();
}
```

3. Bài tập vận dụng

Bài 1 (UOCCP.CPP): Cho số nguyên dương n ($n<=10^9$). Tìm các ước số của số n mà ước đó là số chính phương khác 1

Số chính phương là số mà nó là bình phương của 1 số (ví dụ: 4 là số chính phương vì 4=2.2)

Dữ liệu vào: File UOCCP.INP: Gồm số nguyên dương n

Dữ liệu ra: File UOCCP.OUT: Gồm các ước là số chính phương khác 1, mỗi ước cách nhau bởi khoảng trắng. Nếu không có thì ghi -1

UOCCP.INP	UOCCP.OUT
4	4
32	4 16

Bài 2 (SODEP.CPP): Một số nguyên dương được gọi là số đẹp nếu tổng các chữ số của nó chia hết cho số chữ số. Các số được xét không chứa số 0 không có nghĩa. Ví dụ, 31 là một số đẹp vì 3+1 chia hết cho 2. Số 145 không phải là số đẹp vì 1+4 +5 không chia hết cho 3.

Các số đẹp được đánh số từ 1 trở đi theo thứ tự tăng dần của giá trị.

Yêu cầu: Cho số nguyên dương ***n*** ($1 \le n \le 1000000$). Hãy tìm số đẹp thứ ***n***.

Dữ liệu vào: Vào từ file văn bản SODEP.INP, gồm một dòng chứa một số nguyên ***n***.

Dữ liệu ra: Ghi ra file văn bản SODEP.OUT, ghi trên dòng đầu tiên gồm một số đẹp thứ n.

Ví dụ:

SODEP.INP	SODEP.OUT
1	**1**

SODEP.INP	SODEP.OUT
10	**11**

Bài 3: (SOGIAUCO.CPP) - Số giàu có là số mà tổng ước số của nó (không tính nó và số 1) lớn hơn nó. VD: 12<1+2+3+4+6=16 nên 12 là số giàu có

Cho 2 số nguyên dương L,R ($1<=L<R<=10^4$). Đếm số lượng số giàu có trong khoảng [L;R]

Dữ liệu vào: File SOGIAUCO.INP: Gồm 2 số nguyên dương L, R

Dữ liệu ra: File SOGIAUCO.OUT

Dòng 1: giá trị là số lượng các số giàu có trong khoảng [L;R]

Dòng 2: Các số giàu có, mỗi số cách nhau bởi dấu khoảng trắng

Ví dụ

SOGIAUCO.INP	SOGIAUCO.OUT
3 20	3 12 18 20

Bài 4: (PTTHUASO.CPP) Cho số tự nhiên n (n<1000000). Hãy phân tích n thành tích các thừa số nguyên tố.

VD: Nhập vào n = 9 được 9 = 3.3

Dữ liệu vào: Vào từ file văn bản PTTHUASO.INP, gồm một dòng chứa một số nguyên ***n***.

Dữ liệu ra: Ghi ra file văn bản PTTHUASO.OUT, ghi các thừa số nguyên tố, mỗi thừa số cách nhau dấu chấm.

Ví dụ:

PTTHUASO.INP	PTTHUASO.OUT
9	3.3
5	5

Hướng dẫn:

Gán i = 2;

Khi n > 1 thì lặp:

Nếu n chia hết cho i thì in ra i và gán lại n= n / i. Ngược lại tăng i lên 1.

Bài 5: (THUASONN.CPP) Cho số tự nhiên n (n<10000). Tìm các số tự nhiên nhỏ hơn hoặc bằng n mà sau khi làm phép phân tích ra thừa số nguyên tố có nhiều nhân tử nhất (số đó là số nhỏ nhất tìm được).

Ví dụ n=9 . Các số có nhiều nhân tử nhất sau khi làm phép phân tích là: 8 = 2.2.2

Dữ liệu vào: Vào từ file văn bản THUASONN.INP, gồm một dòng chứa một số nguyên ***n***.

Dữ liệu ra: Ghi ra file văn bản THUASONN.OUT, ghi số nhỏ nhất bé hơn n mà phân tích được nhiều nhân tử nhất.

Ví dụ:

THUASONN.INP	THUASONN.OUT
9	8
24	16

Bài 6: (THUASOLT.CPP) Cho số tự nhiên n (n<10000). Viết chương trình cho phép phân tích một số ra thừa số nguyên tố và ghi kết quả dưới dạng tích các lũy thừa. Ví dụ: 300 = 2^2.3.5^2

Dữ liệu vào: Vào từ file văn bản THUASOLT.INP, gồm một dòng chứa một số nguyên ***n***.

Dữ liệu ra: Ghi ra file văn bản THUASOLT.OUT, ghi dạng thừa số lũy thừa phân tích được.

Ví dụ:

THUASOLT.INP	THUASOLT.OUT
16	2^4
24	2^3.3

Hướng dẫn:

Dùng một mảng để lưu lũy thừa. Mảng này có giá trị các phần tử ban đầu đều bằng 0. Nếu n chia hết cho i thì tăng M[i] lên 1.

Khi in kiểm tra: Nếu M[i] >0 thì in i^M[i].

Bài 7: (TONGNGTO.CPP) Mọi số tự nhiên đều có thể viết được dưới dạng tổng của hai số nguyên tố. Viết chương trình xét xem trong đoạn [n1...n2] số nào cho phép tách thành tổng hai số nguyên tố nhiều trường hợp nhất.

Dữ liệu vào: Vào từ file văn bản TONGNGTO.INP, gồm một dòng chứa 2 số nguyên ***n1, n2*** cách nhau bởi khoảng trắng (1<n1<n2<1000000).

Dữ liệu ra: Ghi ra file văn bản TONGNGTO.OUT, ghi ra số cho phép tách thành hai số nguyên tố nhiều trường hợp nhất. (trường hợp có nhiều đáp án thì ghi ra số nhỏ nhất)

Ví dụ:

TONGNGTO.INP	TONGNGTO.OUT
1 100	90

Bài 8: Số đặc biệt (SODB.CPP)

Tèo có một dãy số gồm N số nguyên dương. Tèo gọi 1 số M lớn hơn 1 là số đặc biệt nếu như nó thỏa mãn tất cả số dư của các phần tử trong dãy số trên khi chia cho M đều bằng nhau.

Nhiệm vụ của bạn là xác định tất cả các số đặc biệt.

Dữ liệu vào: SODB.INP

Dòng đầu tiên là N (2<= N <= 100).

Dòng thứ hai gồm N số nhỏ hơn 10^9, mỗi số cách nhau bởi khoảng trắng. Không tồn tại 2 số nào giống nhau.

Dữ liệu ra: SODB.OUT

In ra tất cả các số đặc biệt mà Tèo quan tâm theo thứ tự tăng dần.

Ví dụ:

SODB.INP	SODB.OUT
3 6 34 38	2 4
5 5 17 23 14 83	3

Bài 9: Số hữu nghị (SOHN.CPP)

Hai số tự nhiên A, B được coi là hữu nghị nếu như số này bằng tổng các ước số của số kia và ngược lại. Lập trình tìm và chiếu lên màn hình các cặp số hữu nghị trong phạm vi từ 1 đến N (1<N<=1000000). (Lưu ý: số 1 được coi là ước số của mọi số còn mỗi số không được coi là ước số của chính nó).

Dữ liệu vào: SOHN.INP

Gồm số nguyên N

Dữ liệu ra: SOHN.OUT

Mỗi dòng là cặp số hữu nghị, mỗi số cách nhau bởi khoảng trắng.

Ví dụ:

SOHN.INP	SOHN.OUT
500	220 284

Bài 10: Số nguyên tố cùng độ cao (hprime.cpp)

Độ cao của số tự nhiên là tổng các chữ số của số đó. Với mỗi cặp số tự nhiên n và h cho trước hãy liệt kê các số nguyên tố không vượt quá n và có độ cao h, $10 \le n \le 1000000$, $1 \le h \le 54$

Dữ liệu vào: File **hprime.inp** gồm 2 giá trị n và h cách nhau khoảng trống.

Dữ liệu ra: File **hprime.out** gồm các số nguyên tố cùng độ cao, mỗi số cách nhau một khoảng trống. Nếu không có số nào thỏa thì ghi - 1

Ví dụ:

hprime.inp	hprime.out
30 5	5 23

hprime.inp	hprime.out
40 9	-1

Bài 11: Độ cao nguyên tố (DOCAONT.cpp)

Độ cao của số tự nhiên là tổng các chữ số của số đó. Tìm tất cả các số có độ cao là số nguyên tố trong khoảng từ 1 đến n

Dữ liệu vào: File **DOCAONT.inp** gồm giá trị n (1<n<=1000000).

Dữ liệu ra: File **DOCAONT.out** gồm các số có độ cao là số nguyên tố trong khoảng từ 1 đến n, mỗi số cách nhau một khoảng trống.

Ví dụ:

DOCAONT.INP	DOCAONT.OUT
20	2 3 5 7 11 12 14 16 20

Bài 12: Số nguyên tố cùng nhau (NTCN.CPP)

Hai số nguyên dương được gọi là nguyên tố cùng nhau nếu ước số chung lớn nhất của chúng bằng 1.

Cho N số nguyên dương $A_1,A_2,...,A_N$. Gọi M là giá trị lớn nhất trong các số $A_1,A_2,...,A_N$.

Viết chương trình tìm số nguyên dương X lớn nhất không vượt quá M mà X nguyên tố cùng nhau với tất cả các số $A_1,A_2,...,A_N$.

Dữ liệu vào: Cho trong file văn bản **NTCN.INP** gồm:

- Dòng đầu là số nguyên dương N ($N \le 100$).
- N dòng tiếp theo, mỗi dòng chứa một giá trị tương ứng $A_1,A_2,...,A_N$ ($A_i \le 1000; i=1,2,...,N$).

Kết quả: Đưa ra file văn bản **NTCN.OUT** chứa số nguyên X tìm được thỏa mãn điều kiện của bài toán.

Ví dụ:

NTCN.INP	NTCN.OUT
3 4 12	13

15	

II- SÀNG SỐ NGUYÊN TỐ

1. Ý tưởng

Sàng Eratosthenes dùng để tìm các số nguyên tố nhỏ hơn hoặc bằng số nguyên N nào đó. Nó còn có thể được sử dụng để kiểm tra một số nguyên nhỏ hơn hoặc bằng N hay không.

Nguyên lí hoạt động của sàng là vào mỗi lần duyệt, ta chọn một số nguyên tố và loại ra khỏi sàng tất cả các bội của số nguyên tố đó mà lớn hơn số đó. Sau khi duyệt xong, các số còn lại trong sàng đều là số nguyên tố.

Thuật toán:

Bước 1: Tạo mảng đánh dấu tất cả các số từ 1 đến n đều là số nguyên tố

Bước 2: Với mỗi số nguyên tố bé hơn hoặc bằng $\sqrt{n}$

+ Đánh dấu các bội của số nguyên tố đó.

```
void sangnt (int n){
   vector<bool> prime(n+1, true);
   prime[0] = prime[1] = false;
   for(int i = 2; i <= n; i++) {
      if (prime[i]) {
         for(int j = i*i; j <= n; j+=i) {
            mark[j] = false;
         }
      }
   }
}
```

2. Một số ví dụ:

Ví dụ 1: **(SANGNT.CPP)**

Nhập vào 1 số nguyên N (N<=1000000000). Liệt kê các số nguyên tố từ 1 đến N

Dữ liệu vào: file SANGNT.INP chứa số nguyên N.

Dữ liệu ra: file SANGNT.OUT các số nguyên tố trong khoảng 1 đến N, mỗi số cách nhau bằng 1 khoảng trắng

SANGNT.INP	**SANGNT.INP**
10	2 3 5 7

Chương trình tham khảo:

```
#include <stdio.h>
#include <vector>
#include <fstream>
using namespace std;
#define tfi "SANGSNT.INP"
```

```
#define tfo "SANGSNT.OUT"
ifstream fi;
ofstream fo;
int n;
vector <bool> isprime;
void nhap(){
   fi.open(tfi);
   fi>>n;
   fi.close();
}
void sangnt(int n){
   for(int i=0;i<=n;i++){
      isprime.push_back(true);
   }
   isprime[0]=isprime[1]=false;
   for(int i=2;i*i<=n;i++){
      if(isprime[i]){
         for(int j=i*2;j<=n;j+=i){
            isprime[j]=false;
         }
      }
   }
}
void xuat(){
   fo.open(tfo);
   for(int i=2;i<=n;i++){
      if(isprime[i]) fo<<i<<" ";
   }
   fo.close();
}
int main() {
   nhap();
```

```
    sangnt(n);
    xuat();
}
```

3. Bài tập vận dụng

Bài 1: (SNTLR.CPP)

Cho 2 số nguyên L, R (1<=L<=R<=1000000000). Liệt kê các số nguyên tố trong khoảng từ L đến R

Dữ liệu vào: file SNTLR.INP chứa số nguyên L, R cách nhau bằng 1 khoảng trắng.

Dữ liệu ra: file SNTLR.OUT các số nguyên tố trong khoảng L đến R, mỗi số cách nhau bằng 1 khoảng trắng

SNTLR.INP	**SNTLR.INP**
10 20	11 13 17 19