NCKU Programming Contest Training Course
Math
2017/03/08

Chien-Wen Chen

ai281918@gmail.com

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

made by ai281918

Prime Number

- Sieve of Eratosthenes (埃拉托斯特尼篩法)
  - 由小到大選擇質數，並刪除其倍數
- 6n±1 Method
  - 拿 2 和 3 這兩個質數先篩過一遍，剩下的數字則用除法驗證是不是質數。

Prime Number

- Sieve of Eratosthenes (埃拉托斯特尼篩法)
  - 由小到大選擇質數，並刪除其倍數

```cpp
#include <cstring>
#include <cmath>
#include <cstdio>
#define MAX 1000000
bool isprime[MAX];
void eratosthenes()
{
    memset(isprime, 1, sizeof(isprime));
    isprime[0] = isprime[1] = false;
    for(int i = 2; i <= sqrt(MAX); ++i)
        if(isprime[i])
            for(int j = i * i; j < MAX; j += i)
                isprime[j] = false;
}
```

Prime Number

- 6n±1 Method
  - 2 和 3 的最小公倍數是 6 ，把所有數字分為 6n 、 6n+1 、 6n+2 、 6n+3 、 6n+4 、 6n+5 六種，可以看出 6n 、 6n+2 、 6n+3 、 6n+4 會是 2 或 3 的倍數，不屬於質數。因此，只要驗證 6n+1 和 6n+5 (= 6n-1)是不是質數就可以了。

Prime Number

- 6n±1 Method

```
#define MAX 1000000
vector<int> prime;
bool isPrime(int n)
{
    for (int i = 0; prime[i] * prime[i] <= n; ++i)
        if (n % prime[i] == 0)
            return false;
    return true;
}
void MakePrime()
{
    prime.push_back(2);
    prime.push_back(3);
    for (int i=5, gap=2; i < MAX; i+=gap, gap=6-gap)
        if (isPrime(i)) prime.push_back(i);
}
```

Prime Number

- 方法二比方法一慢，但較省空間
- 其他方法:
  - 演算法筆記 - Prime

Practice - 1

UVa 10392 - Factoring Large Numbers

Problem Description

One of the central ideas behind much cryptography is that factoring large numbers is computationally intensive. In this context one might use a 100 digit number that was a product of two 50 digit prime numbers. Even with the fastest projected computers this factorization will take hundreds of years.

You don't have those computers available, but if you are clever you can still factor fairly large numbers.

Practice - 1

UVa 10392 - Factoring Large Numbers

Input

The input will be a sequence of integer values, one per line, terminated by a negative number. The numbers will fit in gcc's long long int datatype, however scanf and printf do not appear to handle this datatype correctly, so you must do your own I/O conversion.

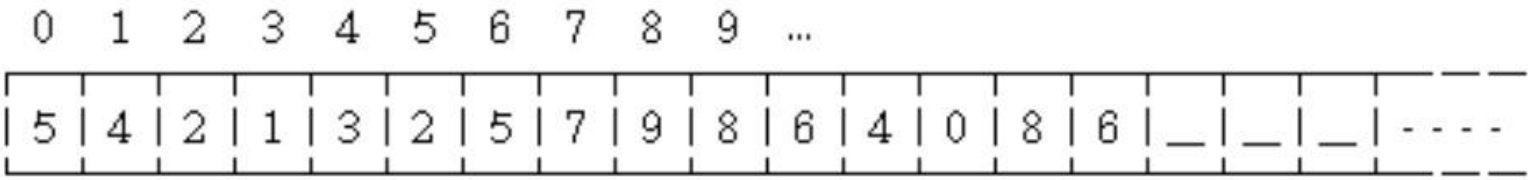
Output

Each positive number from the input must be factored and all factors (other than 1) printed out. The factors must be printed in ascending order with 4 leading spaces preceding a left justified number, and followed by a single blank line.

made by ai281918

Big Number

- Array
- 習慣上將低位數放在index比較小的位置
  - Ex: 680468975231245



- 右方補0

Big Number

- 加法 : 位數各自相加後，由低至高位依序進位
- 減法 : 位數各自相減後，由低至高位依序借位
- 乘法 : 直式乘法
- 除法 : 長除法

Big Number

- 加法：

```
void add(int a[100], int b[100], int c[100])
{
    for (int i = 0; i < 100; ++i)
        c[i] = a[i] + b[i];

    for (int i = 0; i < 100-1; ++i) {
        c[i+1] += c[i] / 10;
        c[i] %= 10;
    }
}
```

Practice - 2

UVa 10106 - Product

Problem Description

The problem is to multiply two integers X,Y . ($0 \le X,Y < 10250$)

Input

The input will consist of a set of pairs of lines. Each line in pair contains one multiplyer.

Output

For each input pair of lines the output line should consist one integer the product.

made by ai281918

Greatest Common Divisor

- 輾轉相除法 (Euclidean Algorithm)

```
int gcd(int a, int b)
{
    if (a == 0) return b;
    return gcd(b % a, a);
}
```

Practice - 3

acm International Collegiate Programming Contest

IBM event sponsor

UVa 408 – Uniform Generator

made by ai281918

Extended Euclidean Algorithm

- 找到aX + bY = gcd(a, b)的整數解 X,Y
- Ex (from wiki)
  - 47x + 30y = 1

Extended Euclidean Algorithm

- 47 = 30 * 1 + 17
- 30 = 17 * 1 + 13
- 17 = 13 * 1 + 4
- 13 = 4 * 3 + 1

Extended Euclidean Algorithm

- 17 = 47 * 1 + 30 * (-1)
- 13 = 30 * 1 + 17 * (-1)
- 4 = 17 * 1 + 13 * (-1)
- 1 = 13 * 1 + 4 * (-3)

Extended Euclidean Algorithm

- 1 = 13 * 1 + 4 * (-3)
- 1 = 13 * 1 + [17 * 1 + 13 * (-1)] * (-3)
- 1 = 17 * (-3) + 13 * 4
- 1 = 17 * (-3) + [30 * 1 + 17 * (-1)] * 4
- 1 = 30 * 4 + 17 * (-7)
- 1 = 30 * 4 + [47 * 1 + 30 * (-1)] * (-7)
- 1 = 47 * (-7) + 30 * 11

Extended Euclidean Algorithm

- gcd(a, b) = gcd(b, a%b)
- aX + bY = gcd(a, b) = gcd(b, a%b) = bX' + (a%b)Y'
- aX + bY = bX' + [a – (a/b)b]Y' = aY' + b(X'-(a/b)Y')
  - X = Y'
  - Y = X' – (a/b)Y'

Extended Euclidean Algorithm

```
int exGCD(int a, int b, int &X, int &Y)
{
    if (b == 0) {
        X = 1;
        Y = 0;
        return a;
    } else {
        int gcd = exGCD(b, a % b, X, Y);
        int tmp = X;
        X = Y;
        Y = tmp - (a/b)*Y;
        return gcd;
    }
}
```

Practice - 4

UVa 10104 - Euclid Problem

Problem Description

From Euclid it is known that for any positive integers A and B there exist such integers X and Y that AX + BY = D, where D is the greatest common divisor of A and B. The problem is to find for given A and B corresponding X, Y and D.

Input

The input will consist of a set of lines with the integer numbers A and B, separated with space (A, B < 1000000001).

Output

For each input line the output line should consist of three integers X, Y and D, separated with space. If there are several such X and Y , you should output that pair for which |X| + |Y | is the minimal. If there are several X and Y satisfying the minimal criteria, output the pair for which X ≤ Y

Extended Euclidean Algorithm

- $\frac{m!}{n!}$ % p (p是一個很大的質數)

Extended Euclidean Algorithm

- $aX + bY = \gcd(a, b)$
  - $a = n!$
  - $b = p$
  - $\gcd(a, b) = 1$

Extended Euclidean Algorithm

- n!X + pY = 1 (use Extended Euclidean Algorithm get (X,Y))
- n!X + pY = 1 → mod p
- (n!X) %p = 1 --- (1)
- $\frac{m!}{n!}$ % p = ans --- (2)
- (1) * (2) → ($\frac{m!}{n!}$ * n!X) % p = ans → (m! * X) % p = ans

Practice - 5

acm International Collegiate Programming Contest

IBM event sponsor

Facebook Hacker Cup 2017 Round1

Beach Umbrellas

made by ai281918

Epsilon

- Float :
  - 數值範圍 : -3.4e-38 ~ 3.4e38
  - 十位數精確度位數 : 6~7

- Double :
  - 數值範圍 : -1.7e308~1.7e308
  - 十位數精確度位數 : 14~15

Epsilon

- Example

```cpp
#include <cstdio>
#include <cmath>
int main()
{
    double a = asin(sqrt(2.0) / 2) * 4.0;
    double b = acos(-1.0);

    printf("a = %.20lf\n", a);
    printf("b = %.20lf\n", b);
    printf("a - b = %.20lf\n", a - b);
    printf("a == b? %s\n", a == b ? "True" : "False");
}
```

made by ai281918

Epsilon

- Result

```
a = 3.1415926535897936000 0
b = 3.1415926535897931000 0
a - b = 0.0000000000000044409
a == b? False
```

Epsilon

- 引入eps判斷浮點數是否相等
  - eps = 1e-8

整數	浮點數
a == b	|a – b| < eps
a != b	|a – b| > eps
a < b	a – b < -eps
a > b	a – b > eps

Practice - 6

acm International Collegiate Programming Contest

IBM event sponsor

UVa 906 – Rational Neighbor

made by ai281918

Problem List

- UVa
  - 338, 375, 408, 906, 10035, 10090, 10104, 10106, 10311, 10392, 10407, 10494, 10606, 11408, 11538

- 門檻：5題
- 第二次修課同學，請從紅字中挑選5題來完成門檻