

NCKU Programming Contest Training Course Math 2018/03/14

Lin Yun Wen

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









Big Number

GCD, Extended Euclid's Algorithm





- 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。





- We use sieve to create a prime array
 - Chose the smallest number at each iteration and delete the multiple of this 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

```
#include <cmath>
    #include <cstring>
    #define MAX 10000000
    bool is_prime[MAX];
    void eratosthenes()
       memset(is_prime, 1, sizeof(is_prime));
       is_prime[0] = false;
8
        is_prime[1] = false;
10
    for (int i = 2; i <= sqrt(MAX); ++i)</pre>
11
    if (is_prime[i])
12
13
    for(int j = i+i; j < MAX; j += i)</pre>
    is_prime[j] = false;
14
```







- 6n±l Method
 - - 2和3的最小公倍數是6,把所有數字分為6n、6n+l、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。





6n±1 Method

```
#include <vector>
    #define MAX 10000000
    vector<int> prime;
    bool is_prime(int n) {
    for (int i = 0; prime[i]*prime[i] <= n; ++i)</pre>
    (n % prime[i] == 0)
    return false;
    return true;
    void make_prime() {
10
    prime.push_back(2);
11
    prime.push_back(3);
12
13
    \cdots for (int i = 5, gap = 2; i < MAX; i+=gap, gap = 6 - gap)
14
    if (is_prime(i))
15
    prime.push_back(i);
16
```





- 方法二比方法一慢,但較省空間
- But just remember that the code in previous page is fast enough to solve almost every prime problems

- 其他方法:
 - <u>演算法 筆記 Prime</u>



Practice - 1



UVa 10392 - Factoring Large Numbers







Big Number

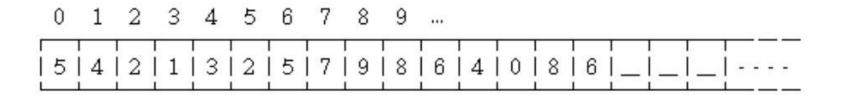
GCD, Extended Euclid's Algorithm





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







- 加法:位數各自相加後,由低至高位依序進位
- 减法: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]) {

void add(int a[100], int b[100], int c[100], int c[100]) {

void add(int a[100], int b[100], int c[100], int
```



Practice - 2



UVa 10106 - Product

Problem Description

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

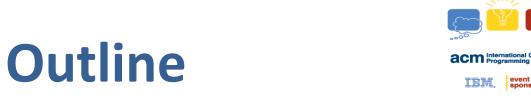
Input

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.





Big Number

GCD, Extended Euclid's Algorithm





Greatest Common Divisor

輾轉相除法 (Euclidean Algorithm)





- gcd(462, 1071)
 - $-\gcd(147,462)$
 - gcd(21, 147)
 - $-\gcd(0,7)$

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

- 21071 中不斷減去462 直到小於462 (可以減2 次,即商40=2),餘數是 147:
 - $-1071 = 2 \times 462 + 147.$
- 然後從462中不斷減去147直到小於147(可以減3次,即q1 = 3),餘數是21:
 - $-462 = 3 \times 147 + 21.$
- 再從147中不斷減去21直到小於21(可以減7次,即q2 = 7),沒有餘數: $-147 = 7 \times 21 + 0$.
- 此時,餘數是0,所以1071和462的最大公因數是21,



Practice - 3



UVa 408 – Uniform Generator





Extended Euclidean Algorithm event sponsor

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

$$-47x + 30y = 1$$



Extended Euclidean Algorithm em Programming Contest Sponsor

•
$$47 = 30 * 1 + 17$$

•
$$30 = 17 * 1 + 13$$

•
$$17 = 13 * 1 + 4$$

•
$$13 = 4 * 3 + 1$$

•
$$4 = 1 * 4 + 0$$

```
gcd(30, 47)
gcd(17, 30)
gcd(13, 17)
gcd(4, 13)
gcd(1, 4)
gcd(0, 1)
```



Extended Euclidean Algorithm event sponsor

•
$$13 = 4 * 3 + 1$$

•
$$4 = 1 * 4 + 0$$

•
$$17 = 47 * 1 + 30 * (-1)$$

•
$$13 = 30 * 1 + 17 * (-1)$$

•
$$4 = 17 * 1 + 13 * (-1)$$

•
$$| = |3 * | + 4 * (-3)$$

$$47x + 30y = 1$$





Extended Euclidean Algorithm em Programming Contest Sponsor

- | = |3 * | + 4 * (-3)
- I = 13 * I + [17 * I + 13 * (-1)] * (-3)
- I = I7 * (-3) + I3 * 4
- I = 17 * (-3) + [30 * I + 17 * (-1)] * 4
- I = 30 * 4 + 17 * (-7)
- I = 30 * 4 + [47 * I + 30 * (-I)] * (-7)
- I = 47 * (-7) + 30 * II



Extended Euclidean Algorithm event sponsor

- 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 event sponsor

```
int exGCD(int a, int b, int &X, int &Y) {
17
    \cdot \cdot \cdot if (b == 0) {
18
    X = 1;
19
20
    Y = 0;
21
    return a;
22
    } else {
23
    ···· int gcd = exGCD(b, a % b, X, Y);
    24
    X = Y:
25
26
    \cdots Y = tmp - (a/b)*Y;
27
    return gcd;
28
29
```



Practice - 4



UVa 10104 - Euclid Problem





Extended Euclidean Algorithm em leterational Collegia Programming Conference Programming Co

•
$$\frac{m!}{n!}$$
 % P (P是一個很大的質數)



Extended Euclidean Algorithm event sponsor

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

方程式 ax+by=1 有整數解 iff 整數 a 和 b 互質



Extended Euclidean Algorithm enternational College Programming Control Programming Con

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

$$\rightarrow \left(\frac{m!}{n!} \times n! X\right) \% p = ans \rightarrow (m! \times X) \% p = ans$$







Facebook Hacker Cup 2017 Round I Beach Umbrellas





• 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:-I.7e308~I.7e308

- 十位數精確度位數:I4~I5





Example





Result

```
linyunwen@Lin-Yun-Wens-MacBook-Air ~/D/L/c/ACM> ./sample_epsilon

a = 3.14159265358979356009

b = 3.14159265358979311600

a-b = 0.000000000000000044409

a == b? False
```





• 引入 eps 判斷浮點數是否相等

$$- eps = Ie-8$$

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



UVa 906 - Rational Neighbor

