added: 10 Jun 2008 17:54
Edited: 31 Aug 2011 14:48

Euclid's algorithm find the GCD (greatest common divisor)

Contents [hide]

Given two non-negative integers a and b. Required to find their greatest common divisor, ie the largest number that divides both a $a$and b. English "greatest common divisor" is spelled "greatest common divisor", and its common designation is $\gcd$:

$$\gcd(a,b) = \max_{k=1\ldots\infty\ :\ k|a\ \&\ k|b} k$$

(Here the symbol "$|$"denotes the divisibility, ie," $k|a$"means" $k$divide a")

When it is of the numbers is zero, and the other is non-zero, their greatest common divisor, by definition, it will be the second number. When the two numbers are equal to zero, the result is undefined (any suitable infinite number), we put in this case, the greatest common divisor of zero. Therefore, we can speak of such a rule: if one of the numbers is zero, the greatest common divisor equal to the second number.

Euclid's algorithm , discussed below, solves the problem of finding the greatest common divisor of two numbers $a$and $b$for $O(\log \min(a,b))$.

This algorithm was first described in the book of Euclid's "Elements" (around 300 BC), although it is possible, this algorithm has an earlier origin.

Algorithm

The algorithm itself is very simple and is described by the following formula:

$$\gcd(a,b) = \begin{cases} a, & \text{if b=0} \\ \gcd(b, a \bmod b), & \text{otherwise} \end{cases}$$

Implementation

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

Using the ternary conditional operator C ++, the algorithm can be written even shorter:

```
int gcd (int a, int b) {
        return b ? gcd (b, a % b) : a;
}
```

Finally, we present an algorithm and a non-recursive form:

```
int gcd (int a, int b) {
        while (b) {
                a %= b;
                swap (a, b);
        }
        return a;
}
```

Proof of correctness

First, note that at each iteration of the Euclidean algorithm its second argument is strictly decreasing, therefore, since it is non-negative, then the Euclidean algorithm **always terminates** .

To **prove the correctness** , we need to show that $\gcd(a, b) = \gcd(b, a \bmod b)$for any $a \geq 0, b > 0$.

We show that the quantity on the left-hand side is divided by the present on the right and the right-hand - is divided into standing on the left. Obviously, this would mean that the left and right sides of the same, and that proves the correctness of Euclid's algorithm.

Denote $d = \gcd(a, b)$. Then, by definition, $d|a$and $d|b$.

Next, we expand the remainder of the division $a$on $b$through their private:

$$a \bmod b = a - b \left\lfloor \frac{a}{b} \right\rfloor$$

But then it follows:

$$d \mid (a \bmod b)$$

So, remembering the statement $d|b$, we obtain the system:

$$\begin{cases} d \mid b, \\ d \mid (a \bmod b) \end{cases}$$

We now use the following simple fact: if for any three numbers p, q, rperformed: $p|q$and $p|r$then executed and: $p \mid \gcd(q, r)$. In our situation, we obtain:

$$d \mid \gcd(b, a \bmod b)$$

Or by substituting $d$its definition as we obtain:

$$\gcd(a, b) \mid \gcd(b, a \bmod b)$$

So, we spent half of the proof: show that the left-right divide. The second half of the proof is similar.

Working time

Time of the algorithm is evaluated **Lame theorem** which establishes a surprising connection of the Euclidean algorithm and the Fibonacci sequence:

If $a > b \geq 1$and for some n, the Euclidean algorithm performs no more recursive calls.

Moreover, it can be shown that the upper bound of this theorem - optimal. When it will be done recursive call. In other words, **successive Fibonacci numbers - the worst input** for Euclid's algorithm.

Given that the Fibonacci numbers grow exponentially (as a constant in power n), we find that the Euclidean algorithm runs in multiplication operations.

LCM (least common multiple)

Calculation of the least common multiple (least common multiplier, lcm) reduces to the calculation the following simple statement:

Thus, the calculation of the NOC can also be done using the Euclidean algorithm, with the same asymptotic behavior:

```
int lcm (int a, int b) {
        return a / gcd (a, b) * b;
}
```

(Here is advantageous first divided into , and only then is multiplied by b, as this will help to avoid overflows in some cases)

Literature

- Thomas Cormen, Charles Leiserson, Ronald Rivest, Clifford Stein. **Algorithms: Design and Analysis** [2005]

Coder • месяц назад

А что если нужно найти НОК массива под модулем 10^9+7?

1 • Ответить • Поделиться ›

Medet Kozhabaev • 2 года назад

Как написать на dev-c++

1 • Ответить • Поделиться ›

Гость • 3 месяца назад

разве рекурсия не чрезмерно ресурсоёмкая?

• Ответить • Поделиться ›

Мирас Кенжегалиев • год назад

return b

Если Б = 0 тогда вывод False, в противном случае True я правильно понял?

• Ответить • Поделиться ›

ga ➚ Мирас Кенжегалиев • 11 месяцев назад

DA

• Ответить • Поделиться ›

Андрей • год назад

Здравствуйте! А разве в нерекурсивной форме алгоритма не нужна проверка на a=0? Иначе неверный ответ получается при a=0, b!=0. Спасибо!

• Ответить • Поделиться ›

vadim • 2 года назад

Здравствуйте! Я думаю было бы полезно написать двоичный алгоритм нахождения нод, например когда нужна длинная арифметика написать деление на 2, гораздо проще чем деление по модулю.

• Ответить • Поделиться ›

ramntry • 2 года назад

Если гоняться за краткостью (обычно это до добра не доводит, но gcd слишком "попсовый" алгоритм, чтобы его это касалось) или даже копеечным приростом производительности (избавляясь не только от рекурсии, но и вызова swap), то можно записать так:

```
int gcd(int a, int b) {
        while ((a %= b) && (b %= a));
        return a | b;
}
```

• Ответить • Поделиться ›

Александр Золотарёв ramntry • год назад

если гоняться за краткостью, можно еще короче:

```
int gcd(int a, int b) {
while (b^=a^=b^=a%=b);
return a;
}
```

1 • Ответить • Поделиться ›

Человек Александр Золотарёв • 10 месяцев назад

попробуйте вызвать с b=0

• Ответить • Поделиться ›

Karen Mkrtumyan Человек • 7 месяцев назад

gcd(a,0) ? Это выражение не имеет смысла.

• Ответить • Поделиться ›