forthright48

Learning Never Ends

Wednesday, July 1, 2015

Euclidean Algorithm - Greatest Common Divisor

Problem

Given two number A and B, find the greatest number that divides both A and B.

What we are trying to find here is the Greatest Common Divisor(GCD) of A and B. What is GCD? Like the name suggests, it the largest number (let that be G), that can divide both A and B. For example, $GCD(2,8) = 2$, $GCD(3,4) = 1, GCD(12,15) = 3$.

How do we find it?

There are many ways. One way is to list down all the divisors of A and B and then find the largest common divisors from those two lists. This is a naive method and takes too much time.

Another approach is to use Euclidean Algorithm, that works on the principle $GCD(a,b) = GCD(b, a\%b)$. Since $GCD(b, a\%b)$ is a smaller state, it is easier to find than the original. And of course, we can apply the principle on the smaller states repeatedly until the state becomes trivial. The two trivial states for GCD are $GCD(a,a) = a$ and $GCD(a,0) = a$.

Euclidean Algorithm

Recursive Version

The recursive version of GCD is simple and small. Just 4 lines of code are enough to find GCD.

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

Iterative Version

It's possible to find GCD without using recursion. This removes the recursion overhead and makes the code faster.

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

Built-In Version

There is also a builtin function in C++ for finding gcd. You can simply write __gcd(a,b) to find GCD(a,b).

Proof

The Euclidean Algorithm works on the principle $GCD(a,b) = GCD(b, a\%b)$. If we can prove this, then there will be no doubt about the algorithm.

Let $g = GCD(a,b)$ and $a = k \times b + r$, where k is a non-negative integer and r is the remainder. Since g divides a, g also divides $k \times b + r$. Since g divides b, g also divides $k \times b$. Therefore, g must divide r otherwise $k \times b + r$ won't be divisible.

So we proved that g divides b and r.

Now lets say we have $g' = gcd(b,r)$. Since g' divides both b and r, it will divide $k \times b + r$. Therefore, g' will divide a.

Now, can g and g' be two different numbers?

We will prove this using contradiction. Let's say that $g > g'$. We know that g divides both b and r. So how can $gcd(b,r)$ be g' when we have a number greater than g' that divides both b and r? So g cannot be greater than g'.

Using the same logic, we find there is a contradiction when $g < g'$. Therefore, the only possibility left is $g = g'$.

$\therefore GCD(a,b) = GCD(b,r) = GCD(b, a\%b)$.

Complexity

It's kind of tricky. For now, just look at this answer from StackOverflow. The complexity of Eculidean Algorithm according to the answer is $O(log_{10}A + log_{10}B)$.

Apparently, there is a whole chapter in Knuth's book TAOCP. I will write a separate post on the complexity of this algorithm when I understand it. For now, just know that it works fast.

Properties

1. Every common divisor of a and b is a divisor of $gcd(a,b)$.
2. The gcd is a commutative function: $gcd(a,b) = gcd(b,a)$.
3. The gcd is an associative function: $gcd(a, gcd(b,c)) = gcd(gcd(a,b), c)$.
4. The gcd of three numbers can be computed as $gcd(a,b,c) = gcd(gcd(a,b), c)$, or in some different way by applying commutativity and associativity. This can be extended to any number of numbers.

More properties can be found on Wiki page

Coding Pitfalls

Notice that the algorithm work correctly for non-negative inputs only. Try to find $GCD(4,-2)$ with the above algorithm. The correct answer should be 2. GCD will always be positive. But it returns -2. Even though the algorithm works correctly only for a non-negative number, it can easily be extended to work with negative numbers. You need to either send the absolute value of the inputs to the algorithm or use the absolute value of the return value.

Next, notice that $GCD(0,0) = 0$. It should be infinity. Also, if you try to work with the return value of $GCD(0,0)$, then you might get RTE due to division by zero.

Try to be careful in the two scenarios above.

Resources

1. Wikipedia - Greatest Common Divisor

Related Problems

1. UVa 11417 - GCD
2. UVa 11827 - Maximum GCD

Posted by Mohammad Samiul Islam

G+1 Recommend this on Google

Labels: GCD, Number Theory

No comments:

Post a Comment

Leave comments for Queries, Bugs and Hugs.

Enter your comment...

Comment as: Select profile... ▾

Publish Preview

Newer Post Home

Subscribe to: Post Comments (Atom)

Blog Archive

- ▼ 2015 (35)
  - ► Sep (7)
  - ► Aug (13)
  - ▼ Jul (15)
    - Simple Hyperbolic Diophantine Equation
    - Linear Diophantine Equation
    - Extended Euclidean Algorithm
    - Introduction to Modular Arithmetic
    - Divisor Summatory Function
    - Sum of Divisors of an Integer
    - Prime Number Theorem
    - Upper Bound for Number Of Divisors

Labels

Analysis Arithmetic Function Backtrack Big Int Binary Bitwise Complexity Contest D&C Divisors Factorial Factorization GCD Language LCM Logarithm Math Modular Arithmetic Number Theory Optimization Primality Test Prime Proof Repeated Squaring Sieve SPOJ Theorem UVa

Highly Composite Numbers

Number of Divisors of an Integer

Prime Factorization of an Integer

Sieve of Eratosthenes - Generating Primes

Primality Test - Naive Methods

Lowest Common Multiple of Two Number

Euclidean Algorithm - Greatest Common Divisor