

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▾[Practice](#) ▾[Resources](#) ▾[Login](#)

Have you registered for Codestudio Beginner Contest 24 yet!

[Register Now](#)[Home](#) [Codestudio](#) > [Library](#) > [Competitive Programm...](#) > [Advanced Level](#) >

GCD Euclidean Algorit...

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

Problem of the day

Consistent and structured practice daily can land you in

**Explore****Related Articles**

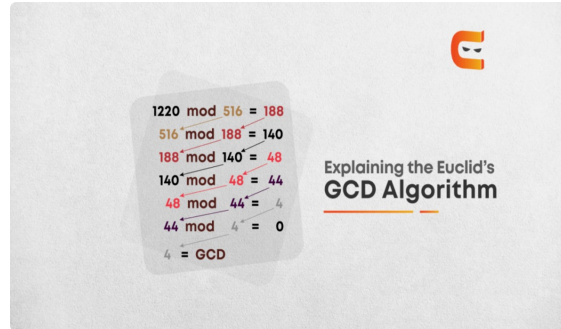
- Catalan Numbers
- Count Number of Valid Parenthesis
- Minimum Number of Arrows to Burst Balloons
- Chinese remainder theorem

Algorithm



Kushleen Waraich
Last Updated: Oct 31, 2022

Share :



One of the earliest known numerical algorithms is that developed by Euclid (the father of geometry) in about 300 B.C. for computing the Greatest Common Divisor (GCD) of two positive integers. Euclid's algorithm is an efficient method for calculating the GCD of two numbers, the largest number that divides both of them without any remainder.

Algorithm:

- The first way of doing it if we try to subtract the smallest number from the greatest, GCD remain the same and in this way, if we keep repeating this step we'll finally get [GCD](#).
- But as above process subtraction could be time-consuming then instead of subtracting what we do, we divide the smaller number, the algorithm stops when we find remainder 0.

Let $GCD(x,y)$ be the GCD of positive integers x and y . If $x = y$, then obviously $GCD(x,y) = GCD(x,x) = x$. Euclid's insight was to observe that, if $x > y$, then $GCD(x,y) = GCD(x-y,y)$.

Actually, this is easy to prove. Suppose that d is a divisor of both x and y . Then there exist integers q_1 and q_2 such that $x = q_1 d$ and $y = q_2 d$. But then $x - y = q_1 d - q_2 d = (q_1 - q_2)d$. Therefore d is also a divisor of $x-y$.

Using similar reasoning, one can show the converse, i.e., that any divisor of $x-y$ and y is also a divisor of x . Hence, the set of common divisors of x and y is the same as the set of common divisors of $x-y$ and y . In particular, the largest values in these two sets are the same, which is to say that $GCD(x,y) = GCD(x-y,y)$.

For improving the other we can also use, (For less no. of iterations) $GCD(x,y) = GCD(x \% y,y)$.

For illustration, the Euclidean algorithm can be used to find the greatest common divisor of $a = 1071$ and $b = 462$.

$$1071 \bmod 462 = 147$$

$$462 \bmod 147 = 21$$

$$147 \bmod 21 = 0$$

Since the last remainder is zero, the algorithm ends

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

Implementation: If we compare this [algorithm](#) with a naïve approach (brute force), what we generally try to do. Let's see How?

Naïve Approach:

```
//We'll start from 1 to smallest of the two number until we find a number
that divides into both of them.

int gcd (int a, int b) {
    int ans=1;
    for(int i=1; i<=min(a,b); i++) {
        if (a%i==0 && b%i==0){
            ans=i;
        }
    }
}
```

But Euclid's method is faster than this naïve method, So let's we follow the Euclidean method to find out the GCD of 4598 and 3211. We represent the two number to the following way, Dividend should be the large number and Divisor is another number. we will repeat the process until the remainder is equal to zero.

Dividend = Divisor * Quotient + Remainder

4598 = 3211 * 1 + 1387

3211 = 1387 * 2 + 437

1387 = 437 * 3 + 76

437 = 76 * 5 + 57

76 = 57 * 1 + 19

57 = 19 * 3 + 0

At the point, the remainder is 0 and we get the GCD 19. Notice that every step dividend and divisor are exchanged and remainder and divisor exchanged. When we remainder is 0 that means the remaining divisor our GCD.

Euclid's GCD Algorithm:

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

Time Complexity: $O(\log \min(a, b))$

Binary Euclidean Algorithm: This algorithm finds the gcd using only subtraction, binary representation, shifting and parity testing. We will use a divide and conquer technique. The following function calculate $\text{gcd}(a, b, \text{res}) = \text{gcd}(a, b, 1) \cdot \text{res}$. So to calculate $\text{gcd}(a, b)$ it suffices to call $\text{gcd}(a, b, 1) = \text{gcd}(a, b)$.

12.3: Greatest common divisor using binary Euclidean algorithm.

```
1 def gcd(a, b, res):
2     if a == b:
3         return res * a
4     elif (a % 2 == 0) and (b % 2 == 0):
5         return gcd(a // 2, b // 2, 2 * res)
6     elif (a % 2 == 0):
7         return gcd(a // 2, b, res)
8     elif (b % 2 == 0):
9         return gcd(a, b // 2, res)
10    elif a > b:
11        return gcd(a - b, b, res)
12    else:
13        return gcd(a, b - a, res)
```

This algorithm is superior to the previous one for very large integers when it cannot be assumed that all the arithmetic operations used here can be done in constant time. Due to the binary representation, operations are performed in linear time based on the length of the binary representation, even for very big integers. On the other hand, modulo applied in algorithm 10.2 has worse time complexity. It exceeds $O(\log n \cdot \log \log n)$, where $n = a + b$. Thus the time complexity is $O(\log(a \cdot b)) = O(\log a + b) = O(\log n)$.

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

Least Common Multiple:

The least common multiple (lcm) of two integers a and b is the smallest positive integer that is divisible by both a and b. There is the following relation:

$$\text{lcm}(a, b) = a \cdot b / \text{gcd}(a, b)$$

Knowing how to compute the $\text{gcd}(a, b)$ in $O(\log(a+b))$ time, we can also compute the $\text{lcm}(a, b)$ in the same time complexity.

Extended Euclidean Algorithm:

Although Euclid GCD algorithm works for almost all cases we can further improve it and this algorithm is known as the Extended Euclidean Algorithm. This algorithm not only finds GCD of two numbers but also integer coefficients x and y such that:

$$ax + by = \text{gcd}(a, b)$$

Input: $a = 35, b = 15$

Output: $\text{gcd} = 5$

$x = 1, y = -2$

(Note that $35 \cdot 1 + 15 \cdot (-2) = 5$)

Basically, what this algorithm does, it updates the results of $\text{gcd}(a, b)$ using the results calculated by recursive call $\text{gcd}(b \% a, a)$. Let values of x and y obtained by the recursive calls are x_1 and y_1 . Then x and y are as follows:

$$x = y_1 - (b/a) \cdot x_1$$

$$y = x_1$$

```
int gcd(int a, int b, int& x, int& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    int x1, y1;
    int d = gcd(b, a % b, x1, y1);
    x = y1;
    y = x1 - y1 * (a / b);
    return d;
}
```

This implementation of the extended Euclidean algorithm produces correct results for negative integers as well.

Approach	Time Complexity	Space Complexity
Brute Force Solution	$O(\min(\text{number1}, \text{number2}))$	$O(1)$
Euclidean Algorithm	$O(\max(\text{number1}, \text{number2}))$	$O(1)$
Optimized Euclidean Algorithm	$O(\log(\text{number1} + \text{number2}))$	$O(\log(\text{number1} + \text{number2}))$

Some Problems Based on Euclid's Algorithm:

- Program to find the LCM of two numbers using GCD.
- Program to find GCD of floating-point numbers.
- Program to find the common ratio of three numbers.
- Program to find GCD of an [array](#) of integers.
- Program to find the sum of squares of N natural numbers.
- For more such problems or enhance your skills, you can also practice on our coding platform [Codezen](#).

To read more, click [here](#).

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Previous Article](#)

[Next Article](#)

Was this article helpful ?








0 upvotes

Share this article with friends and colleague :



Comments

Write your thoughts...

B *I*     ▾    ▾

Post

No comments yet

Be the first to share what you think

Categories: Coding courses for beginners | Web Development Courses | Data Science & Machine Learning Courses | Competitive Programming Course | Android App Development Courses | Courses for interview preparation

Popular Courses: C++ Foundation with Data Structures | Java Foundation with Data Structures | Python Foundation with Data Structures | Competitive Programming | Full Stack Web Development

Career Tracks: Ninja Competitive Programmer Track | Ninja Android Developer Career Track | Ninja Web Developer Career Track - NodeJS & ReactJS | Ninja Web Developer Career Track - NodeJS | Ninja Data Scientist Career Track | Ninja Machine Learning Engineer Career Track

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▾[Practice](#) ▾[Resources](#) ▾[Login](#)

CODING NINJAS

[About Us](#)[Press](#)[Privacy](#)[Policy](#)[Terms &](#)[Conditions](#)[Bug](#)[Bounty](#)[Hire from](#)[CodeStudio](#)

PRODUCTS

[Problem of the day](#)[Interview Problems](#)[Interview Experiences](#)[Interview](#)[Conditions](#)[Bundle](#)[Guided](#)[Paths](#)[Library](#)[Test Series](#)[Contest](#)

COMMUNITY FOLLOW US ON

[CodeStudio](#)[Blog](#)[Events](#)[Campus](#)[Ninjas](#)

We accept payments using:



No Cost EMI

100% safe & secure payment



Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▼[Practice](#) ▼[Resources](#) ▼[Login](#)

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▼[Practice](#) ▼[Resources](#) ▼[Login](#)

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▼[Practice](#) ▼[Resources](#) ▼[Login](#)

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▼[Practice](#) ▼[Resources](#) ▼[Login](#)

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▼[Practice](#) ▼[Resources](#) ▼[Login](#)

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▼[Practice](#) ▼[Resources](#) ▼[Login](#)

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▼[Practice](#) ▼[Resources](#) ▼[Login](#)

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →

[Guided Paths](#)[Contests](#)[Interview Prep](#) ▼[Practice](#) ▼[Resources](#) ▼[Login](#)

Set your goal

Important for focused learning



Prepare for tech interviews



Learn and practise coding



Become an expert competitive coder



Already setup?

Next →