Comparative Study of GCD Algorithms in Multiple Languages

(Euclidean and Stein's algorithms)

Kapil P

Kapilpalanisamy73@gmail.com

Introduction:

This is the blog where the multiple languages has been benchmarked on various algorithms that is used to find the GCD of 2 numbers.

Calculating the greatest common divisor (GCD) is a classic problem in computer science and mathematics, fundamental to fields such as cryptography, coding theory, and digital signal processing. This study provides a side-by-side comparison of six primary GCD implementations—Euclidean Iterative, Euclidean Recursive, Stein's Iterative, Stein's Recursive, Subtraction Method, and Language Built-in—benchmarking them in Python, Java, C, and C++ across a range of inputs.This gives the final benchmark of languages performance across various algorithms and the inputs.

Note: The test has been done on the System with Intel i5 11^{th} gen ,8gb Ram , 512gb Rom and 4gb Nvidia GTX 1650 graphics card.

GCD:

The **greatest common divisor** (GCD) of two integers *a* and b is the largest integer that divides both without leaving a remainder.

This can be written as gcd (*a,b*).

Examples:

- Gcd (48,18) = 6
- gcd (20,8) = 4

Key Properties

- gcd (a,0) = |a|
- gcd (a,b) = gcd (b,a)
- If b divides a, then gcd(a,b) = |b|

Applications

- Cryptography .
- Simplifying fractions
- Synchronizing frequencies in signal processing

Languages Used:

Python

Easy to read, modest recursion depth, interpreter overhead.

Java

Uses BigInteger.gcd or user-defined methods,Good recursion, compiled to JVM bytecode.

C

Most direct to hardware; fast, requires manual function implementation.

C++

Modern C++ has std::gcd; also supports manual implementations and templates.

Standard Test Cases has been used to compare the performance

The Test cases are:

Input Size / Pattern	Input A	Input B	GCD Result
1-digit	8	3	1
2-digit	48	18	6
3-digit	210	45	15
4-digit	1234	4321	1
5-digit	12345	54321	3
6-digit	123456	789012	12
7-digit	1000001	7000001	1
8-digit	12345678	87654321	9
9-digit	123456789	987654321	9
10-digit	1234567890	9876543210	90
Both same	55555	55555	55555
One is zero	0	1234567890	1234567890
Large + small	1	999999937	1
Power of 2 values	1048576	32768	32768
Large prime pair	982451653	57885161	1
11-digit	12345678901	10987654321	1
12-digit	1.23457E+11	2.10988E+11	1000000
13-digit	1.23457E+12	3.21099E+12	10000000
14-digit	1.23457E+13	4.3211E+13	100000000
15-digit	1.23457E+14	5.43211E+14	1000000000

Euclidean algorithm:

Euclid's algorithm, is an efficient method for computing the GCD of two integers, the largest number that divides them both without a remainder.

It can be used to reduce fractions to their simplest form, and is a part of many other number-theoretic and cryptographic calculations.

The Euclidean algorithm is based on the principle that the greatest common divisor of two numbers does not change if the larger number is replaced by its difference with the smaller number.

For example,

21 is the GCD of 252 and 105 (as 252 = 21 × 12 and 105 = 21 × 5),

and the same number 21 is also the GCD of 105 and **252 − 105 = 147.**

Since this replacement reduces the larger of the two numbers, repeating this process gives successively smaller pairs of numbers until the two numbers become equal.

And the other way is like the common factors

For example,

a = 12, b = 20
The GCD of 12 and 20 is 4
Explanation: The Common factors of (12, 20) are 1, 2, and 4 and greatest is 4.

There are 2 ways to implement Euclid's algorithm that is:

1. Iterative Method
2. Recursive Method

1. Iterative Method

Iterative method refers to a programming approach that involves repeatedly executing a block of code until a certain condition is met.

Uses a loop to repeatedly replace a with b, and b with a mod b, until b=0.

Highly efficient and commonly used.

1.Python

```python
def gcd_method(a, b):
    while b:
        a, b = b, a % b
    return a
```

def **gcd_method**(a, b):
while b:
a, b = b, a % b
return a

The output of the method is

Methods	Input Size / Pattern	Input A	Input B	GCD Result	Time Taken (s)	Time Taken (ns)	emory Used (I	Timestamp
Euclidean Iterative	1-digit	8	3	1	6.4103E-06	5956	0.96	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	2-digit	48	18	6	3.6531E-06	3444	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	3-digit	210	45	15	2.7907E-06	2676	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	4-digit	1234	4321	1	6.8364E-06	6790	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	5-digit	12345	54321	3	5.8291E-06	5764	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	6-digit	123456	789012	12	1.75061E-05	17519	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	7-digit	1000001	7000001	1	0.000004034	4043	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	8-digit	12345678	87654321	9	4.7027E-06	4701	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	9-digit	123456789	987654321	9	3.1502E-06	3145	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	10-digit	1234567890	9876543210	90	6.0783E-06	6132	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	Both same	55555	55555	55555	2.3162E-06	2354	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	One is zero	0	1234567890	1234567890	0.000002434	2428	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	Large + small	1	999999937	1	2.3539E-06	2381	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	Power of 2 values	1048576	32768	32768	2.2464E-06	2237	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	Large prime pair	982451653	57885161	1	8.7242E-06	8772	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	11-digit	12345678901	10987654321	1	1.27358E-05	12754	0.21	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	12-digit	1.23457E+11	2.10988E+11	1000000	1.72989E-05	17306	0.24	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	13-digit	1.23457E+12	3.21099E+12	10000000	1.85883E-05	18573	0.24	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	14-digit	1.23457E+13	4.3211E+13	100000000	1.27158E-05	12706	0.24	2025-06-29 12:01:22 - 2025-06-29 12:01:22
	15-digit	1.23457E+14	5.43211E+14	1000000000	8.5821E-06	8592	0.24	2025-06-29 12:01:22 - 2025-06-29 12:01:22

2.Java

```java
public static BigInteger gcd(BigInteger a, BigInteger b) {
    while (!b.equals(BigInteger.ZERO)) {
        BigInteger temp = b;
        b = a.mod(b);
        a = temp;
    }
    return a;
}
```

public static BigInteger gcd (BigInteger a, BigInteger b) {

while (!b.equals(BigInteger.ZERO)) {

BigInteger temp = b;

b = a.mod(b);

a = temp;

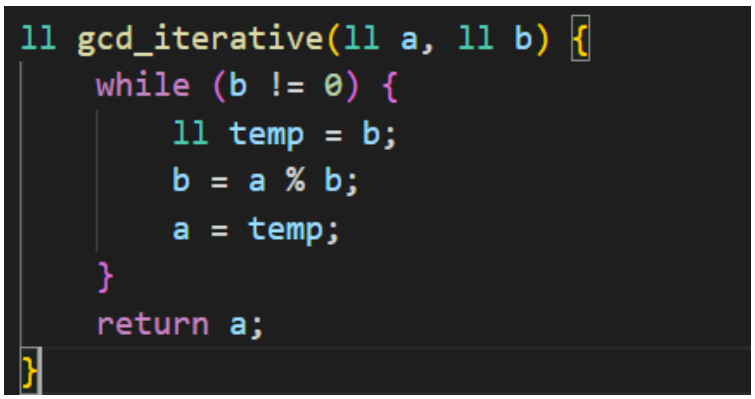
}

return a;

}

Euclidean Iterative	1-digit	8	3	1	0.000317174	317174	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	2-digit	48	18	6	0.000012529	12529	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	3-digit	210	45	15	0.000010387	10387	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	4-digit	1234	4321	1	0.000121621	121621	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	5-digit	12345	54321	3	0.000013485	13485	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	6-digit	123456	789012	12	0.000021606	21606	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	7-digit	1000001	7000001	1	0.00001768	17680	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	8-digit	12345678	87654321	9	0.000014896	14896	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	9-digit	123456789	987654321	9	0.000008684	8684	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	10-digit	1234567890	9876543210	90	0.000020638	20638	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	Both same	55555	55555	55555	0.000003397	3397	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	One is zero	0	1234567890	1234567890	0.00000252	2520	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	Large + small	1	999999937	1	0.000011375	11375	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	Power of 2 values	1048576	32768	32768	0.000004385	4385	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	Large prime pair	982451653	57885161	1	0.000039015	39015	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	11-digit	12345678901	10987654321	1	0.000109732	109732	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	12-digit	1.23457E+11	2.10988E+11	1000000	0.000092843	92843	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	13-digit	1.23457E+12	3.21099E+12	10000000	0.000069643	69643	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	14-digit	1.23457E+13	4.3211E+13	100000000	0.00007381	73810	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.000048046	48046	0	2025-06-29 11:51:37 - 2025-06-29 11:51:37

3.C



```
ll gcd_iterative(ll a, ll b) {
    while (b != 0) {
        ll temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

ll gcd_iterative(ll a, ll b) {

while (b != 0) {

ll temp = b;

b = a % b;

a = temp;

}

return a;

}

Euclidean Iterative	1-digit	8	3	1	0.000048	48000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	2-digit	48	18	6	0.000012	12000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	3-digit	210	45	15	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	4-digit	1234	4321	1	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	5-digit	12345	54321	3	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	6-digit	123456	789012	12	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	7-digit	1000001	7000001	1	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	8-digit	12345678	87654321	9	0.000007	7000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	9-digit	123456789	987654321	9	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	10-digit	1234567890	9876543210	90	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	Both same	55555	55555	55555	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	One is zero	0	1234567890	1234567890	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	Large + small	1	999999937	1	0.000007	7000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	Power of 2 values	1048576	32768	32768	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	Large prime pair	982451653	57885161	1	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	11-digit	12345678901	10987654321	1	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	12-digit	1.23457E+11	2.10988E+11	1000000	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	13-digit	1.23457E+12	3.21099E+12	10000000	0.000007	7000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	14-digit	1.23457E+13	4.3211E+13	100000000	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.000006	6000	2025-06-29 11:36:04 - 2025-06-29 11:36:04

4.C++

```cpp
long long gcdIterative(long long a, long long b) {
    while (b != 0) {
        long long temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

long long gcdIterative(long long a, long long b) {
    while (b != 0) {
        long long temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

Euclidean Iterative	1-digit	8	3	1	0.000033	32999	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	2-digit	48	18	6	0.0000342	34165	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	3-digit	210	45	15	0.0000042	4238	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	4-digit	1234	4321	1	0.000004	3980	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	5-digit	12345	54321	3	0.000004	4044	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	6-digit	123456	789012	12	0.0000091	9060	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	7-digit	1000001	7000001	1	0.0000038	3810	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	8-digit	12345678	87654321	9	0.000004	3980	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	9-digit	123456789	987654321	9	0.0000039	3915	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	10-digit	1234567890	9876543210	90	0.0000038	3773	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	Both same	55555	55555	55555	0.0000041	4052	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	One is zero	0	1234567890	1234567890	0.000005	5022	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	Large + small	1	999999937	1	0.0000063	6301	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	Power of 2 values	1048576	32768	32768	0.0000041	4146	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	Large prime pair	982451653	57885161	1	0.0000107	10702	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	11-digit	12345678901	10987654321	1	0.0000043	4305	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	12-digit	1.23457E+11	2.10988E+11	1000000	0.0000041	4118	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	13-digit	1.23457E+12	3.21099E+12	10000000	0.0000043	4306	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	14-digit	1.23457E+13	4.3211E+13	100000000	0.0000039	3949	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.000004	4021	NA	2025-06-29 11:06:54 - 2025-06-29 11:06:54

2. Recursive Method

Solves the same problem recursively: gcd (a, b) =gcd (b, amodb).

This approach is clear and mathematically elegant.

1.Python

```python
def gcd_method(a, b):
    if b == 0:
        return a
    return gcd_method(b, a % b)
```

```
def gcd_method(a, b):
    if b == 0:
        return a
    return gcd_method(b, a % b)
```

Euclidean_recursive	1-digit	8	3	1	0.000007032	6078	0.96	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	2-digit	48	18	6	2.6829E-06	2583	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	3-digit	210	45	15	0.000002017	1924	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	4-digit	1234	4321	1	5.6399E-06	5466	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	5-digit	12345	54321	3	0.000004688	4635	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	6-digit	123456	789012	12	8.9209E-06	8919	0.29	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	7-digit	1000001	7000001	1	3.6058E-06	3568	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	8-digit	12345678	87654321	9	0.000004258	4257	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	9-digit	123456789	987654321	9	2.4019E-06	2423	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	10-digit	1234567890	9876543210	90	4.8629E-06	4832	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	Both same	55555	55555	55555	0.000001793	1838	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	One is zero	0	1234567890	1234567890	0.000001983	1975	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	Large + small	1	999999937	1	2.5139E-06	2441	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	Power of 2 values	1048576	32768	32768	1.8568E-06	1837	0.21	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	Large prime pair	982451653	57885161	1	9.0031E-06	8975	0.32	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	11-digit	12345678901	10987654321	1	1.29521E-05	12967	0.38	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	12-digit	1.23457E+11	2.10988E+11	1000000	1.68001E-05	16744	0.38	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	13-digit	1.23457E+12	3.21099E+12	10000000	0.000018029	18024	0.41	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	14-digit	1.23457E+13	4.3211E+13	100000000	0.000011103	11089	0.29	2025-06-29 12:03:09 - 2025-06-29 12:03:09
	15-digit	1.23457E+14	5.43211E+14	1000000000	7.0201E-06	6974	0.24	2025-06-29 12:03:09 - 2025-06-29 12:03:09

2.Java

```java
    public static BigInteger gcd(BigInteger a, BigInteger b) {
        if (b.equals(BigInteger.ZERO)) return a;
        return gcd(b, a.mod(b));
    }
```

```
public static BigInteger gcd(BigInteger a, BigInteger b) {
    if (b.equals(BigInteger.ZERO)) return a;
    return gcd(b, a.mod(b));
}
```

Euclidean_recursive	1-digit	8	3	1	0.000405214	405214	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	2-digit	48	18	6	0.000012527	12527	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	3-digit	210	45	15	0.000009362	9362	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	4-digit	1234	4321	1	0.000021723	21723	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	5-digit	12345	54321	3	0.000014725	14725	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	6-digit	123456	789012	12	0.000023884	23884	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	7-digit	1000001	7000001	1	0.0000194	19400	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	8-digit	12345678	87654321	9	0.000014572	14572	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	9-digit	123456789	987654321	9	0.000012544	12544	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	10-digit	1234567890	9876543210	90	0.000017884	17884	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	Both same	55555	55555	55555	0.000002547	2547	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	One is zero	0	1234567890	1234567890	0.000002895	2895	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	Large + small	1	999999937	1	0.000007403	7403	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	Power of 2 values	1048576	32768	32768	0.000004503	4503	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	Large prime pair	982451653	57885161	1	0.000036374	36374	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	11-digit	12345678901	10987654321	1	0.000070495	70495	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	12-digit	1.23E+11	2.11E+11	1000000	0.000069806	69806	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	13-digit	1.23E+12	3.21E+12	10000000	0.000082443	82443	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	14-digit	1.23E+13	4.32E+13	100000000	0.000042669	42669	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55
	15-digit	1.23E+14	5.43E+14	1000000000	0.000044106	44106	0	2025-06-29 11:53:55 - 2025-06-29 11:53:55

3.C++

```cpp
long long gcdRecursive(long long a, long long b) {
    return b == 0 ? a : gcdRecursive(b, a % b);
}
```

long long gcdRecursive(long long a, long long b) {
    return b == 0 ? a : gcdRecursive(b, a % b);
}

Euclidean_recursive	1-digit	8	3	1	0.0000339	33893	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	2-digit	48	18	6	0.0000055	5520	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	3-digit	210	45	15	0.0000043	4306	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	4-digit	1234	4321	1	0.0000045	4466	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	5-digit	12345	54321	3	0.0000044	4435	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	6-digit	123456	789012	12	0.0000044	4403	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	7-digit	1000001	7000001	1	0.0000077	7690	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	8-digit	12345678	87654321	9	0.0000043	4347	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	9-digit	123456789	987654321	9	0.0000041	4128	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	10-digit	1234567890	9876543210	90	0.0000044	4414	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	Both same	55555	55555	55555	0.0000044	4429	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	One is zero	0	1234567890	1234567890	0.0000044	4423	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	Large + small	1	999999937	1	0.0000041	4078	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	Power of 2 values	1048576	32768	32768	0.0000042	4152	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	Large prime pair	982451653	57885161	1	0.0000047	4661	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	11-digit	12345678901	10987654321	1	0.0000048	4790	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	12-digit	1.23457E+11	2.10988E+11	1000000	0.0000045	4465	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	13-digit	1.23457E+12	3.21099E+12	10000000	0.0000043	4272	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	14-digit	1.23457E+13	4.3211E+13	100000000	0.0000047	4704	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.0000043	4343	NA	2025-06-29 11:12:50 - 2025-06-29 11:12:50

4.C

```c
// Recursive Euclidean GCD
ll gcd_recursive(ll a, ll b) {
    if (b == 0)
        return a;
    return gcd_recursive(b, a % b);
}
```

ll gcd_recursive(ll a, ll b) {
    if (b == 0)
        return a;
    return gcd_recursive(b, a % b);
}

Euclidean_recursive	1-digit	8	3	1	0.00003	30000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	2-digit	48	18	6	0.000006	6000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	3-digit	210	45	15	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	4-digit	1234	4321	1	0.000005	5000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	5-digit	12345	54321	3	0.000005	5000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	6-digit	123456	789012	12	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	7-digit	1000001	7000001	1	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	8-digit	12345678	87654321	9	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	9-digit	123456789	987654321	9	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	10-digit	1234567890	9876543210	90	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	Both same	55555	55555	55555	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	One is zero	0	1234567890	1234567890	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	Large + small	1	999999937	1	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	Power of 2 values	1048576	32768	32768	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	Large prime pair	982451653	57885161	1	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	11-digit	12345678901	10987654321	1	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	12-digit	1.23457E+11	2.10988E+11	1000000	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	13-digit	1.23457E+12	3.21099E+12	10000000	0.000005	5000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	14-digit	1.23457E+13	4.3211E+13	100000000	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.000004	4000	2025-06-29 11:38:58 - 2025-06-29 11:38:58

Next comes the Steins Algorithm

Steins Algorithm

Stein's algorithm or the binary Euclidean algorithm, is an algorithm that computes the greatest common divisor (GCD) of two nonnegative integers.

Stein's algorithm uses simpler arithmetic operations than the conventional Euclidean algorithm; it replaces division with arithmetic shifts, comparisons, and subtraction.

The algorithm finds the GCD of two nonnegative numbers a and b by repeatedly applying these identities:

1. gcd(a,0) = a => everything divides zero, and a is the largest number that divides a.
2. gcd(2a,2b) = 2 . gcd(a,b) => 2 is a common divisor.
3. gcd(a,2b) = gcd(a,b) => if a is odd: 2 is then not a common divisor.
4. gcd(a,b) = gcs(a,b-a) => if a,b odd and a<=b.

Now this can be implemented in 2 ways:

1. Iterative Method
2. Recursive Method

1.Iterative Method

Relies on bitwise operations instead of division.

Exploits evenness/oddness to halve the numbers via binary shifts.

1.Python

```python
def gcd_method(a, b):
    if a == 0: return b
    if b == 0: return a
    k = 0
    while (a | b) & 1 == 0:
        a >>= 1
        b >>= 1
        k += 1
    while a & 1 == 0:
        a >>= 1
    while b != 0:
        while b & 1 == 0:
            b >>= 1
        if a > b:
            a, b = b, a
        b -= a
    return a << k
```

Stein Iterative	1-digit	8	3	1	9.17E-06	8672	0.96	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	2-digit	48	18	6	0.00000518	4995	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	3-digit	210	45	15	0.000004143	4091	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	4-digit	1234	4321	1	1.57E-05	15692	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	5-digit	12345	54321	3	0.00002466	24594	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	6-digit	123456	789012	12	3.88E-05	38706	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	7-digit	1000001	7000001	1	0.000041805	41716	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	8-digit	12345678	87654321	9	4.55E-05	45575	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	9-digit	123456789	987654321	9	5.51E-05	55179	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	10-digit	1234567890	9876543210	90	0.000070272	70139	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	Both same	55555	55555	55555	0.000005879	5772	0.24	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	One is zero	0	1234567890	1234567890	2.58E-06	2431	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	Large + small	1	999999937	1	4.43E-05	44290	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	Power of 2 values	1048576	32768	32768	3.22E-05	32159	0.24	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	Large prime pair	982451653	57885161	1	6.26E-05	62536	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	11-digit	12345678901	10987654321	1	7.69E-05	76887	0.21	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	12-digit	1.23E+11	2.11E+11	1000000	7.70E-05	76959	0.24	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	13-digit	1.23E+12	3.21E+12	10000000	8.32E-05	83149	0.24	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	14-digit	1.23E+13	4.32E+13	100000000	0.000105558	105558	0.24	2025-06-29 12:04:34 - 2025-06-29 12:04:34
	15-digit	1.23E+14	5.43E+14	1000000000	0.00009164	91583	0.24	2025-06-29 12:04:34 - 2025-06-29 12:04:34

2.Java

```java
public static BigInteger gcd(BigInteger a, BigInteger b) {
    if (a.equals(BigInteger.ZERO)) return b;
    if (b.equals(BigInteger.ZERO)) return a;

    int commonFactorsOf2 = 0;
    while (a.and(BigInteger.ONE).equals(BigInteger.ZERO) && b.and(BigInteger.ONE).equals(BigInteger.ZERO)) {
        a = a.shiftRight(n:1);
        b = b.shiftRight(n:1);
        commonFactorsOf2++;
    }

    while (a.and(BigInteger.ONE).equals(BigInteger.ZERO))
        a = a.shiftRight(n:1);

    while (!b.equals(BigInteger.ZERO)) {
        while (b.and(BigInteger.ONE).equals(BigInteger.ZERO))
            b = b.shiftRight(n:1);

        if (a.compareTo(b) > 0) {
            BigInteger temp = a;
            a = b;
            b = temp;
        }

        b = b.subtract(a);
    }

    return a.shiftLeft(commonFactorsOf2);
}
```

Stein Iterative	1-digit	8	3	1	0.000064643	64643	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	2-digit	48	18	6	0.00006533	65330	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	3-digit	210	45	15	0.000028923	28923	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	4-digit	1234	4321	1	0.00007442	74420	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	5-digit	12345	54321	3	0.000075837	75837	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	6-digit	123456	789012	12	0.000145275	145275	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	7-digit	1000001	7000001	1	0.000169915	169915	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	8-digit	12345678	87654321	9	0.00012414	124140	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	9-digit	123456789	987654321	9	0.000139067	139067	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	10-digit	1234567890	9876543210	90	0.000187919	187919	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	Both same	55555	55555	55555	0.00000622	6220	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	One is zero	0	1234567890	1234567890	0.000000879	879	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	Large + small	1	999999937	1	0.000115433	115433	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	Power of 2 values	1048576	32768	32768	0.00008343	83430	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	Large prime pair	982451653	57885161	1	0.000141852	141852	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	11-digit	12345678901	10987654321	1	0.000163811	163811	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	12-digit	1.23E+11	2.11E+11	1000000	0.000094052	94052	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	13-digit	1.23E+12	3.21E+12	10000000	0.000134705	134705	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	14-digit	1.23E+13	4.32E+13	100000000	0.000077543	77543	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42
	15-digit	1.23E+14	5.43E+14	1000000000	0.000163781	163781	0	2025-06-29 11:55:42 - 2025-06-29 11:55:42

3.C++

```cpp
long long steinGCD(long long a, long long b) {
    if (a == 0) return b;
    if (b == 0) return a;

    int shift;
    for (shift = 0; ((a | b) & 1) == 0; ++shift) {
        a >>= 1;
        b >>= 1;
    }
    while ((a & 1) == 0)
        a >>= 1;

    do {
        while ((b & 1) == 0)
            b >>= 1;
        if (a > b)
            swap(a, b);
        b -= a;
    } while (b != 0);

    return a << shift;
}
```

Stein Iterative	1-digit	8	3	1	0.000034	34019	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	2-digit	48	18	6	0.0000062	6178	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	3-digit	210	45	15	0.0000046	4624	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	4-digit	1234	4321	1	0.0000045	4477	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	5-digit	12345	54321	3	0.0000045	4472	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	6-digit	123456	789012	12	0.0000048	4791	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	7-digit	1000001	7000001	1	0.0000047	4726	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	8-digit	12345678	87654321	9	0.0000044	4381	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	9-digit	123456789	987654321	9	0.000006	6016	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	10-digit	1234567890	9876543210	90	0.0000044	4449	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	Both same	55555	55555	55555	0.0000044	4400	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	One is zero	0	1234567890	1234567890	0.0000046	4639	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	Large + small	1	999999937	1	0.0000046	4627	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	Power of 2 values	1048576	32768	32768	0.0000041	4079	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	Large prime pair	982451653	57885161	1	0.0000048	4753	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	11-digit	12345678901	10987654321	1	0.0000051	5141	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	12-digit	1.23457E+11	2.10988E+11	1000000	0.0000043	4327	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	13-digit	1.23457E+12	3.21099E+12	10000000	0.0000065	6525	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	14-digit	1.23457E+13	4.3211E+13	100000000	0.0000346	34636	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.0000057	5689	NA	2025-06-29 11:15:27 - 2025-06-29 11:15:27

4.C

```cpp
// Stein's Binary GCD - Iterative
ll gcd_stein_iterative(ll u, ll v) {
    if (u == 0) return v;
    if (v == 0) return u;

    int shift = count_trailing_zeros(u < v ? u : v);
    u >>= count_trailing_zeros(u);

    while (v != 0) {
        v >>= count_trailing_zeros(v);
        if (u > v) {
            ll temp = v;
            v = u;
            u = temp;
        }
        v = v - u;
    }

    return u << shift;
}
```

Stein Iterative	1-digit	8	3	1	0.000034	34000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	2-digit	48	18	6	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	3-digit	210	45	15	0.000004	4000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	4-digit	1234	4321	2	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	5-digit	12345	54321	3	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	6-digit	123456	789012	192	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	7-digit	1000001	7000001	1	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	8-digit	12345678	87654321	18	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	9-digit	123456789	987654321	9	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	10-digit	1234567890	9876543210	90	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	Both same	55555	55555	55555	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	One is zero	0	1234567890	1234567890	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	Large + small	1	999999937	1	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	Power of 2 values	1048576	32768	32768	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	Large prime pair	982451653	57885161	1	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	11-digit	12345678901	10987654321	1	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	12-digit	1.23457E+11	2.10988E+11	1000000	0.000004	4000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	13-digit	1.23457E+12	3.21099E+12	10000000	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	14-digit	1.23457E+13	4.3211E+13	100000000	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.000005	5000	2025-06-29 11:40:57 - 2025-06-29 11:40:57

2.Recursive Method

Recursive take on Stein's method; uses the same bitwise decomposition as the iterative form.

1.Python

```python
def gcd_method(a, b):
    if a == b or b == 0:
        return a
    if a == 0:
        return b
    if (~a & 1):
        if b & 1:
            return gcd_method(a >> 1, b)
        else:
            return gcd_method(a >> 1, b >> 1) << 1
    if (~b & 1):
        return gcd_method(a, b >> 1)
    if a > b:
        return gcd_method((a - b) >> 1, b)
    return gcd_method((b - a) >> 1, a)
```

Stein Recursive	1-digit	8	3	1	0.000012523	11630	0.96	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	2-digit	48	18	6	0.000012225	12020	0.21	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	3-digit	210	45	15	9.55E-06	9421	0.21	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	4-digit	1234	4321	1	3.69E-05	36908	0.23	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	5-digit	12345	54321	3	5.33E-05	53320	0.35	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	6-digit	123456	789012	12	8.89E-05	88854	0.7	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	7-digit	1000001	7000001	1	0.00010112	101128	0.73	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	8-digit	12345678	87654321	9	0.000107888	107848	0.73	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	9-digit	123456789	987654321	9	0.000152707	152710	0.82	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	10-digit	1234567890	9876543210	90	0.000173908	173943	1.04	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	Both same	55555	55555	55555	2.22E-06	2254	0.21	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	One is zero	0	1234567890	1234567890	2.53E-06	2463	0.21	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	Large + small	1	999999937	1	7.74E-05	77430	0.76	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	Power of 2 values	1048576	32768	32768	5.01E-05	50176	0.63	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	Large prime pair	982451653	57885161	1	0.000144382	144445	1.07	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	11-digit	12345678901	10987654321	1	0.000177887	177880	1.16	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	12-digit	1.23E+11	2.11E+11	1000000	0.000163467	163447	1.32	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	13-digit	1.23E+12	3.21E+12	10000000	0.000171781	171749	1.23	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	14-digit	1.23E+13	4.32E+13	100000000	0.000212532	212574	1.51	2025-06-29 12:05:56 - 2025-06-29 12:05:56
	15-digit	1.23E+14	5.43E+14	1000000000	0.000212337	212253	1.29	2025-06-29 12:05:56 - 2025-06-29 12:05:56

2.Java

```java
public static BigInteger gcd(BigInteger a, BigInteger b) {
    if (a.equals(b)) return a;
    if (a.equals(BigInteger.ZERO)) return b;
    if (b.equals(BigInteger.ZERO)) return a;

    if (a.and(BigInteger.ONE).equals(BigInteger.ZERO)) {
        if (b.and(BigInteger.ONE).equals(BigInteger.ONE))
            return gcd(a.shiftRight(n:1), b);
        else
            return gcd(a.shiftRight(n:1), b.shiftRight(n:1)).shiftLeft(n:1);
    }

    if (b.and(BigInteger.ONE).equals(BigInteger.ZERO))
        return gcd(a, b.shiftRight(n:1));

    if (a.compareTo(b) > 0)
        return gcd(a.subtract(b).shiftRight(n:1), b);
    else
        return gcd(b.subtract(a).shiftRight(n:1), a);
}
```

Stein Recursive	1-digit	8	3	1	0.00006747	67470	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	2-digit	48	18	6	0.000046254	46254	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	3-digit	210	45	15	0.000027551	27551	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	4-digit	1234	4321	1	0.000086751	86751	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	5-digit	12345	54321	3	0.000090877	90877	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	6-digit	123456	789012	12	0.000145104	145104	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	7-digit	1000001	7000001	1	0.000177011	177011	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	8-digit	12345678	87654321	9	0.000169222	169222	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	9-digit	123456789	987654321	9	0.000159044	159044	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	10-digit	1234567890	9876543210	90	0.00017711	177110	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	Both same	55555	55555	55555	0.000000718	718	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	One is zero	0	1234567890	1234567890	0.000002134	2134	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	Large + small	1	999999937	1	0.000128247	128247	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	Power of 2 values	1048576	32768	32768	0.000098271	98271	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	Large prime pair	982451653	57885161	1	0.000183594	183594	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	11-digit	12345678901	10987654321	1	0.000153484	153484	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	12-digit	1.23E+11	2.11E+11	1000000	0.000131835	131835	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	13-digit	1.23E+12	3.21E+12	10000000	0.000096625	96625	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	14-digit	1.23E+13	4.32E+13	100000000	0.000162081	162081	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29
	15-digit	1.23E+14	5.43E+14	1000000000	0.000146398	146398	0	2025-06-29 11:56:29 - 2025-06-29 11:56:29

3.C++

```cpp
long long steinRecursive(long long a, long long b) {
    if (a == b || b == 0) return a;
    if (a == 0) return b;

    if ((a & 1) == 0 && (b & 1) == 0)
        return steinRecursive(a >> 1, b >> 1) << 1;
    else if ((a & 1) == 0)
        return steinRecursive(a >> 1, b);
    else if ((b & 1) == 0)
        return steinRecursive(a, b >> 1);
    else if (a > b)
        return steinRecursive((a - b) >> 1, b);
    else
        return steinRecursive((b - a) >> 1, a);
}
```

Stein Recursive	1-digit	8	3	1	0.0000402	40241	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	2-digit	48	18	6	0.0002502	250227	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	3-digit	210	45	15	0.0000043	4266	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	4-digit	1234	4321	1	0.0000113	11269	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	5-digit	12345	54321	3	0.0000044	4425	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	6-digit	123456	789012	12	0.0000042	4238	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	7-digit	1000001	7000001	1	0.0000046	4601	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	8-digit	12345678	87654321	9	0.0000048	4822	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	9-digit	123456789	987654321	9	0.000004	3989	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	10-digit	1234567890	9876543210	90	0.000004	4047	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	Both same	55555	55555	55555	0.0000036	3635	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	One is zero	0	1234567890	1234567890	0.0000054	5360	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	Large + small	1	999999937	1	0.0000048	4789	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	Power of 2 values	1048576	32768	32768	0.0000049	4903	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	Large prime pair	982451653	57885161	1	0.0000106	10625	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	11-digit	12345678901	10987654321	1	0.0000054	5427	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	12-digit	1.23457E+11	2.10988E+11	1000000	0.0000052	5226	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	13-digit	1.23457E+12	3.21099E+12	10000000	0.0000057	5680	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	14-digit	1.23457E+13	4.3211E+13	100000000	0.0000049	4901	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.0000049	4872	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38

4.C

```cpp
// Recursive Binary GCD (Stein's Algorithm)
ll gcd_stein_recursive(ll a, ll b) {
    if (a == b || b == 0) return a;
    if (a == 0) return b;

    // both even
    if ((a & 1) == 0 && (b & 1) == 0)
        return gcd_stein_recursive(a >> 1, b >> 1) << 1;
    // a even
    else if ((a & 1) == 0)
        return gcd_stein_recursive(a >> 1, b);
    // b even
    else if ((b & 1) == 0)
        return gcd_stein_recursive(a, b >> 1);
    // both odd
    else if (a > b)
        return gcd_stein_recursive((a - b) >> 1, b);
    else
        return gcd_stein_recursive((b - a) >> 1, a);
}
```

Stein Recursive	1-digit	8	3	1	0.000045	45000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	2-digit	48	18	6	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	3-digit	210	45	15	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	4-digit	1234	4321	1	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	5-digit	12345	54321	3	0.000004	4000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	6-digit	123456	789012	12	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	7-digit	1000001	7000001	1	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	8-digit	12345678	87654321	9	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	9-digit	123456789	987654321	9	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	10-digit	1234567890	9876543210	90	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	Both same	55555	55555	55555	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	One is zero	0	1234567890	1234567890	0.000004	4000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	Large + small	1	999999937	1	0.000004	4000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	Power of 2 values	1048576	32768	32768	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	Large prime pair	982451653	57885161	1	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	11-digit	12345678901	10987654321	1	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	12-digit	1.23457E+11	2.10988E+11	1000000	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	13-digit	1.23457E+12	3.21099E+12	10000000	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	14-digit	1.23457E+13	4.3211E+13	100000000	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.000005	5000	2025-06-29 11:42:26 - 2025-06-29 11:42:26

Now we move to the Next method that is Subtratcion method

5.Subtraction Method

Repeatedly subtracts the smaller number from the larger one until both are equal.

Simple but inefficient for large or distant pairs.

1.Python

```python
def gcd_method(a, b):
    if a == 0: return b
    if b == 0: return a
    while a != b:
        if a > b:
            a -= b
        else:
            b -= a
    return a
```

Gcd Subtraction	1-digit	8	3	1	8.8196E-06	8239	0.96	2025-06-29 12:11:26 - 2025-06-29 12:11:26
	2-digit	48	18	6	4.4983E-06	4492	0.21	2025-06-29 12:11:26 - 2025-06-29 12:11:26
	3-digit	210	45	15	3.2252E-06	3003	0.21	2025-06-29 12:11:26 - 2025-06-29 12:11:26
	4-digit	1234	4321	1	7.77808E-05	77610	0.21	2025-06-29 12:11:26 - 2025-06-29 12:11:26
	5-digit	12345	54321	3	0.000087068	86880	0.21	2025-06-29 12:11:26 - 2025-06-29 12:11:26
	6-digit	123456	789012	12	1.50939E-05	15051	0.21	2025-06-29 12:11:26 - 2025-06-29 12:11:26
	7-digit	1000001	7000001	1	0.093506869	93509895	0.21	2025-06-29 12:11:26 - 2025-06-29 12:11:26
	8-digit	12345678	87654321	9	0.010187278	10189185	0.21	2025-06-29 12:11:26 - 2025-06-29 12:11:26
	9-digit	123456789	987654321	9	8.661496483	8661498480	0.21	2025-06-29 12:11:26 - 2025-06-29 12:11:35
	10-digit	1234567890	9876543210	90	9.224442932	9224444860	0.21	2025-06-29 12:11:35 - 2025-06-29 12:11:44
	Both same	55555	55555	55555	0.00000322	3036	0.21	2025-06-29 12:11:44 - 2025-06-29 12:11:44
	One is zero	0	1234567890	1234567890	3.3034E-06	3192	0.21	2025-06-29 12:11:44 - 2025-06-29 12:11:44
	Large + small	Nil	Nil	Nil	Nil	Nil	Nil	Nil
	Power of 2 values	1048576	32768	32768	2.20286E-05	21413	0.24	2025-06-29 12:11:44 - 2025-06-29 12:11:44
	Large prime pair	982451653	57885161	1	0.000064855	64720	0.21	2025-06-29 12:11:44 - 2025-06-29 12:11:44
	11-digit	12345678901	10987654321	1	0.006979079	6980518	0.21	2025-06-29 12:11:44 - 2025-06-29 12:11:44
	12-digit	1.23457E+11	2.10988E+11	1000000	2.92989E-05	29073	0.24	2025-06-29 12:11:44 - 2025-06-29 12:11:44
	13-digit	1.23457E+12	3.21099E+12	10000000	6.38557E-05	63654	0.24	2025-06-29 12:11:44 - 2025-06-29 12:11:44
	14-digit	1.23457E+13	4.3211E+13	100000000	0.001787606	1788724	0.24	2025-06-29 12:11:44 - 2025-06-29 12:11:44
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.01440374	14405249	0.24	2025-06-29 12:11:44 - 2025-06-29 12:11:44

2.Java

```java
public static BigInteger gcd(BigInteger a, BigInteger b) {
    if (a.equals(BigInteger.ZERO)) return b;
    if (b.equals(BigInteger.ZERO)) return a;

    while (!a.equals(b)) {
        if (a.compareTo(b) > 0)
            a = a.subtract(b);
        else
            b = b.subtract(a);
    }
    return a;
}
```

Gcd Subtraction	1-digit	8	3	1	0.000026701	26701	0	2025-06-29 11:57:20 - 2025-06-29 11:57:20
	2-digit	48	18	6	0.000009813	9813	0	2025-06-29 11:57:20 - 2025-06-29 11:57:20
	3-digit	210	45	15	0.000009427	9427	0	2025-06-29 11:57:20 - 2025-06-29 11:57:20
	4-digit	1234	4321	1	0.00024597	245970	0	2025-06-29 11:57:20 - 2025-06-29 11:57:20
	5-digit	12345	54321	3	0.000293523	293523	0	2025-06-29 11:57:20 - 2025-06-29 11:57:20
	6-digit	123456	789012	12	0.000047907	47907	0	2025-06-29 11:57:20 - 2025-06-29 11:57:20
	7-digit	1000001	7000001	1	0.015167131	15167131	2048.55	2025-06-29 11:57:20 - 2025-06-29 11:57:20
	8-digit	12345678	87654321	9	0.000997856	997856	1116.99	2025-06-29 11:57:20 - 2025-06-29 11:57:20
	9-digit	123456789	987654321	9	0.27699197	276991970	-2463.77	2025-06-29 11:57:20 - 2025-06-29 11:57:21
	10-digit	1234567890	9876543210	90	0.206081043	206081043	1931.13	2025-06-29 11:57:21 - 2025-06-29 11:57:21
	Both same	55555	55555	55555	0.000034493	34493	0	2025-06-29 11:57:21 - 2025-06-29 11:57:21
	One is zero	0	1234567890	1234567890	0.000000422	422	0	2025-06-29 11:57:21 - 2025-06-29 11:57:21
	Large + small	1	999999937	1	20.13450663	20134506630	-2017.8	2025-06-29 11:57:21 - 2025-06-29 11:57:41
	Power of 2 values	1048576	32768	32768	0.000025789	25789	0	2025-06-29 11:57:41 - 2025-06-29 11:57:41
	Large prime pair	982451653	57885161	1	0.000009971	9971	0	2025-06-29 11:57:41 - 2025-06-29 11:57:41
	11-digit	12345678901	10987654321	1	0.002240024	2240024	789.96	2025-06-29 11:57:41 - 2025-06-29 11:57:41
	12-digit	1.23E+11	2.11E+11	1000000	0.000022927	22927	0	2025-06-29 11:57:41 - 2025-06-29 11:57:41
	13-digit	1.23E+12	3.21E+12	10000000	0.000017301	17301	87.77	2025-06-29 11:57:41 - 2025-06-29 11:57:41
	14-digit	1.23E+13	4.32E+13	100000000	0.000321428	321428	175.55	2025-06-29 11:57:41 - 2025-06-29 11:57:41
	15-digit	1.23E+14	5.43E+14	1000000000	0.002541354	2541354	1492.15	2025-06-29 11:57:41 - 2025-06-29 11:57:41

3.C++

```cpp
long long gcdSubtraction(long long a, long long b) {
    if (a == 0) return b;
    if (b == 0) return a;

    while (a != b) {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }
    return a;
}
```

Stein Recursive	1-digit	8	3	1	0.0000402	40241	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	2-digit	48	18	6	0.0002502	250227	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	3-digit	210	45	15	0.0000043	4266	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	4-digit	1234	4321	1	0.0000113	11269	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	5-digit	12345	54321	3	0.0000044	4425	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	6-digit	123456	789012	12	0.0000042	4238	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	7-digit	1000001	7000001	1	0.0000046	4601	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	8-digit	12345678	87654321	9	0.0000048	4822	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	9-digit	123456789	987654321	9	0.000004	3989	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	10-digit	1234567890	9876543210	90	0.000004	4047	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	Both same	55555	55555	55555	0.0000036	3635	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	One is zero	0	1234567890	1234567890	0.0000054	5360	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	Large + small	1	999999937	1	0.0000048	4789	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	Power of 2 values	1048576	32768	32768	0.0000049	4903	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	Large prime pair	982451653	57885161	1	0.0000106	10625	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	11-digit	12345678901	10987654321	1	0.0000054	5427	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	12-digit	1.23457E+11	2.10988E+11	1000000	0.0000052	5226	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	13-digit	1.23457E+12	3.21099E+12	10000000	0.0000057	5680	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	14-digit	1.23457E+13	4.3211E+13	100000000	0.0000049	4901	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.0000049	4872	NA	2025-06-29 11:17:38 - 2025-06-29 11:17:38

4.C

```c
// gcd using repeated subtraction
ll gcd_subtraction(ll a, ll b) {
    if (a == 0) return b;
    if (b == 0) return a;
    while (a != b) {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }
    return a;
}
```

Gcd Subtraction	1-digit	8	3	1	0.000049	49000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	2-digit	48	18	6	0.000012	12000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	3-digit	210	45	15	0.000005	5000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	4-digit	1234	4321	1	0.000006	6000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	5-digit	12345	54321	3	0.000006	6000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	6-digit	123456	789012	12	0.000006	6000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	7-digit	1000001	7000001	1	0.000417	417000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	8-digit	12345678	87654321	9	0.000048	48000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	9-digit	123456789	987654321	9	0.022871	22871000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	10-digit	1234567890	9876543210	90	0.02058	20580000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	Both same	55555	55555	55555	0.000004	4000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	One is zero	0	1234567890	1234567890	0.000004	4000	2025-06-29 11:44:05 - 2025-06-29 11:44:05
	Large + small	1	999999937	1	2.211147	2211147000	2025-06-29 11:44:08 - 2025-06-29 11:44:08
	Power of 2 values	1048576	32768	32768	0.00001	10000	2025-06-29 11:44:08 - 2025-06-29 11:44:08
	Large prime pair	982451653	57885161	1	0.000007	7000	2025-06-29 11:44:08 - 2025-06-29 11:44:08
	11-digit	12345678901	10987654321	1	0.000028	28000	2025-06-29 11:44:08 - 2025-06-29 11:44:08
	12-digit	1.23457E+11	2.10988E+11	1000000	0.000005	5000	2025-06-29 11:44:08 - 2025-06-29 11:44:08
	13-digit	1.23457E+12	3.21099E+12	10000000	0.000006	6000	2025-06-29 11:44:08 - 2025-06-29 11:44:08
	14-digit	1.23457E+13	4.3211E+13	100000000	0.000021	21000	2025-06-29 11:44:08 - 2025-06-29 11:44:08
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.000049	49000	2025-06-29 11:44:08 - 2025-06-29 11:44:08

Next method is using inbuilt methods.

6.Inbuilt Methods

Uses the standard gcd function provided by each language's library.

Often optimized beyond user-space implementations.

1.Python

```python
def gcd_method(a, b):
    return math.gcd(a, b)
```

Gcd Builtin	1-digit	8	3	1	8.18E-06	6155	0.96	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	2-digit	48	18	6	7.07E-06	6673	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	3-digit	210	45	15	3.01E-06	2928	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	4-digit	1234	4321	1	3.02E-06	2884	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	5-digit	12345	54321	3	4.81E-06	4488	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	6-digit	123456	789012	12	0.000002983	2931	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	7-digit	1000001	7000001	1	2.88E-06	2803	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	8-digit	12345678	87654321	9	1.76E-05	17499	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	9-digit	123456789	987654321	9	2.31E-06	2265	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	10-digit	1234567890	9876543210	90	2.62E-06	2596	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	Both same	55555	55555	55555	1.64E-05	16398	0.24	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	One is zero	0	1234567890	1234567890	0.000003309	3282	0.24	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	Large + small	1	999999937	1	0.000002346	2323	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	Power of 2 values	1048576	32768	32768	3.03E-06	2992	0.24	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	Large prime pair	982451653	57885161	1	2.44E-06	2411	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	11-digit	12345678901	10987654321	1	2.39E-06	2372	0.21	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	12-digit	1.23E+11	2.11E+11	1000000	0.000002916	2883	0.24	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	13-digit	1.23E+12	3.21E+12	10000000	4.54E-06	4354	0.24	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	14-digit	1.23E+13	4.32E+13	100000000	0.000004393	4269	0.24	2025-06-29 12:08:53 - 2025-06-29 12:08:53
	15-digit	1.23E+14	5.43E+14	1000000000	4.03E-06	3850	0.24	2025-06-29 12:08:53 - 2025-06-29 12:08:53

2.Java

```
BigInteger result = input.a.gcd(input.b);
```

Gcd Builtin	1-digit	8	3	1	0.000399245	399245	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	2-digit	48	18	6	0.000032312	32312	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	3-digit	210	45	15	0.000007521	7521	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	4-digit	1234	4321	1	0.00000795	7950	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	5-digit	12345	54321	3	0.000006599	6599	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	6-digit	123456	789012	12	0.000010669	10669	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	7-digit	1000001	7000001	1	0.000007941	7941	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	8-digit	12345678	87654321	9	0.000008087	8087	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	9-digit	123456789	987654321	9	0.000009501	9501	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	10-digit	1234567890	9876543210	90	0.000020626	20626	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	Both same	55555	55555	55555	0.000004859	4859	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	One is zero	0	1234567890	1234567890	0.000004135	4135	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	Large + small	1	999999937	1	0.00000727	7270	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	Power of 2 values	1048576	32768	32768	0.000008841	8841	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	Large prime pair	982451653	57885161	1	0.00001103	11030	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	11-digit	12345678901	10987654321	1	0.000015248	15248	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	12-digit	1.23E+11	2.11E+11	1000000	0.000013572	13572	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	13-digit	1.23E+12	3.21E+12	10000000	0.000016395	16395	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	14-digit	1.23E+13	4.32E+13	100000000	0.00001801	18010	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48
	15-digit	1.23E+14	5.43E+14	1000000000	0.000018328	18328	0	2025-06-29 11:58:48 - 2025-06-29 11:58:48

3.C++

```cpp
long long gcdModulusOnly(long long a, long long b) {
    while (a != 0 && b != 0) {
        if (a > b) a %= b;
        else b %= a;
    }
    return a | b; // one of them will be zero
}
```

Gcd Builtin	1-digit	8	3	1	0.0000517	51724	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	2-digit	48	18	6	0.0000089	8913	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	3-digit	210	45	15	0.0000051	5125	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	4-digit	1234	4321	1	0.0000049	4910	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	5-digit	12345	54321	3	0.0000053	5310	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	6-digit	123456	789012	12	0.0000048	4784	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	7-digit	1000001	7000001	1	0.0000077	7732	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	8-digit	12345678	87654321	9	0.0000056	5624	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	9-digit	123456789	987654321	9	0.0000048	4800	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	10-digit	1234567890	9876543210	90	0.0000048	4817	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	Both same	55555	55555	55555	0.0000052	5203	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	One is zero	0	1234567890	1234567890	0.0000047	4726	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	Large + small	1	999999937	1	0.0000053	5279	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	Power of 2 values	1048576	32768	32768	0.0000055	5549	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	Large prime pair	982451653	57885161	1	0.0000048	4802	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	11-digit	12345678901	10987654321	1	0.0000053	5283	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	12-digit	1.23457E+11	2.10988E+11	1000000	0.0000058	5769	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	13-digit	1.23457E+12	3.21099E+12	10000000	0.0000047	4668	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	14-digit	1.23457E+13	4.3211E+13	100000000	0.0000053	5280	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.0000047	4655	NA	2025-06-29 11:22:08 - 2025-06-29 11:22:08

4.C

```cpp
// GCD using only modulus (Euclidean without conditionals)
ll gcd_modulus_only(ll a, ll b) {
    ll temp;
    while (a && b) {
        // Swap a = a % b and vice versa until one becomes 0
        temp = a % b;
        a = b;
        b = temp;
    }
    return a + b; // one of them is 0, the other is GCD
}
```

Gcd Builtin	1-digit	8	3	1	0.000037	37000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	2-digit	48	18	6	0.000006	6000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	3-digit	210	45	15	0.000005	5000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	4-digit	1234	4321	1	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	5-digit	12345	54321	3	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	6-digit	123456	789012	12	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	7-digit	1000001	7000001	1	0.000006	6000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	8-digit	12345678	87654321	9	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	9-digit	123456789	987654321	9	0.000005	5000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	10-digit	1234567890	9876543210	90	0.000005	5000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	Both same	55555	55555	55555	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	One is zero	0	1234567890	1234567890	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	Large + small	1	999999937	1	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	Power of 2 values	1048576	32768	32768	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	Large prime pair	982451653	57885161	1	0.000005	5000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	11-digit	12345678901	10987654321	1	0.000005	5000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	12-digit	1.23457E+11	2.10988E+11	1000000	0.000005	5000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	13-digit	1.23457E+12	3.21099E+12	10000000	0.000006	6000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	14-digit	1.23457E+13	4.3211E+13	100000000	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36
	15-digit	1.23457E+14	5.43211E+14	1000000000	0.000004	4000	2025-06-29 11:45:36 - 2025-06-29 11:45:36

Algorithm Analysis

Method	Step-by-Step	Typical Complexity
Euclidean Iter.	Loop: $a, b = b, a \bmod b$	$O(\log n)$
Euclidean Rec.	If $b = 0$ return a; else recurse	$O(\log n)$
Stein Iter.	Uses bitwise shifts and subtraction in loop	$O(\log n)$
Stein Rec.	Recursive bitwise splitting	$O(\log n)$
Subtraction	Continue $a = a - b$ (or vice versa)	$O(\max(a, b))$
Built-in	Library optimized	$O(\log n)$

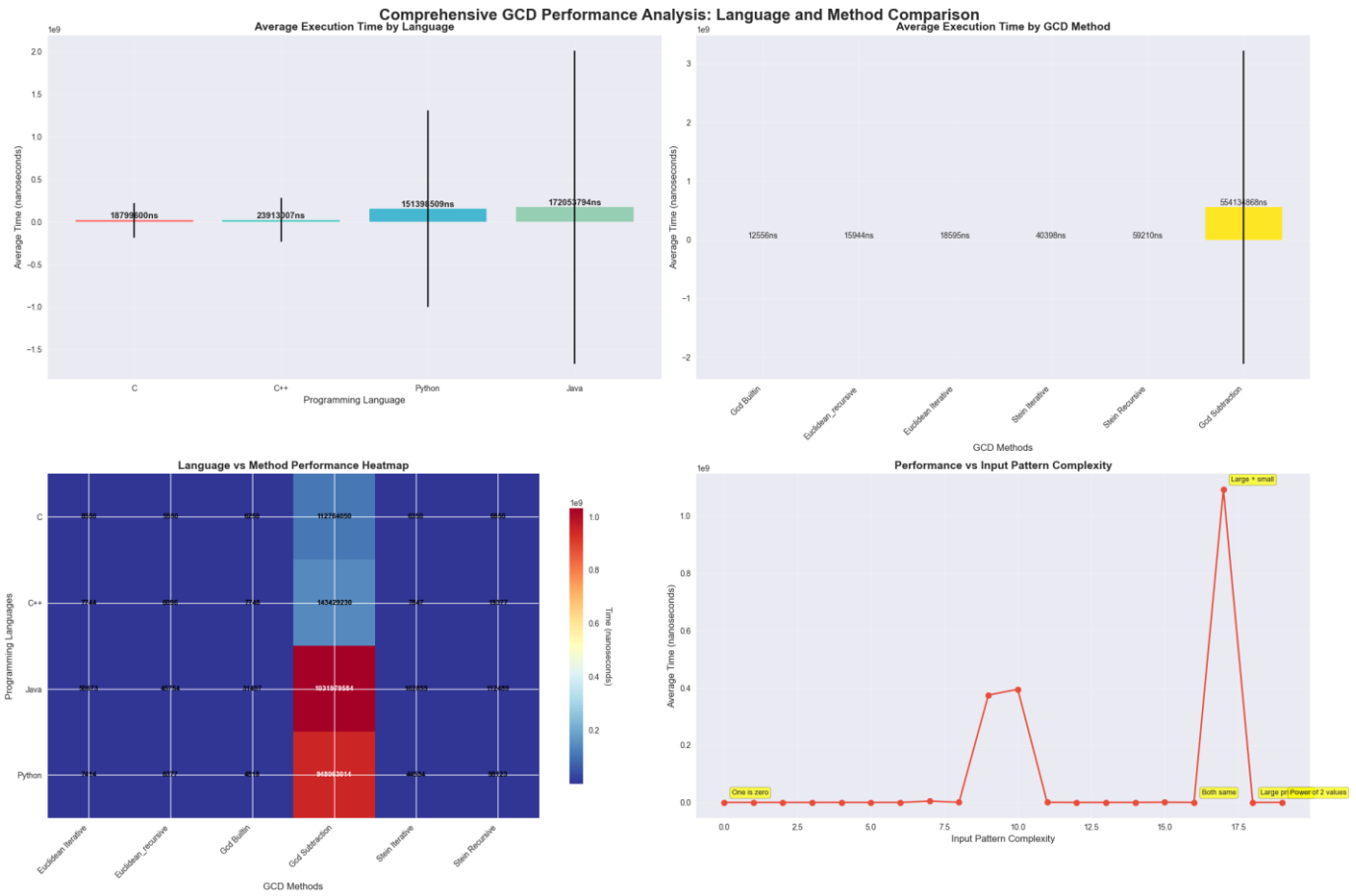
Comprehensive Results

We are here to the end part of the blog where we see the solid benchmarks of each language's performance according to the algorithms.

Benchmark Summary (For 10-Digit Inputs)

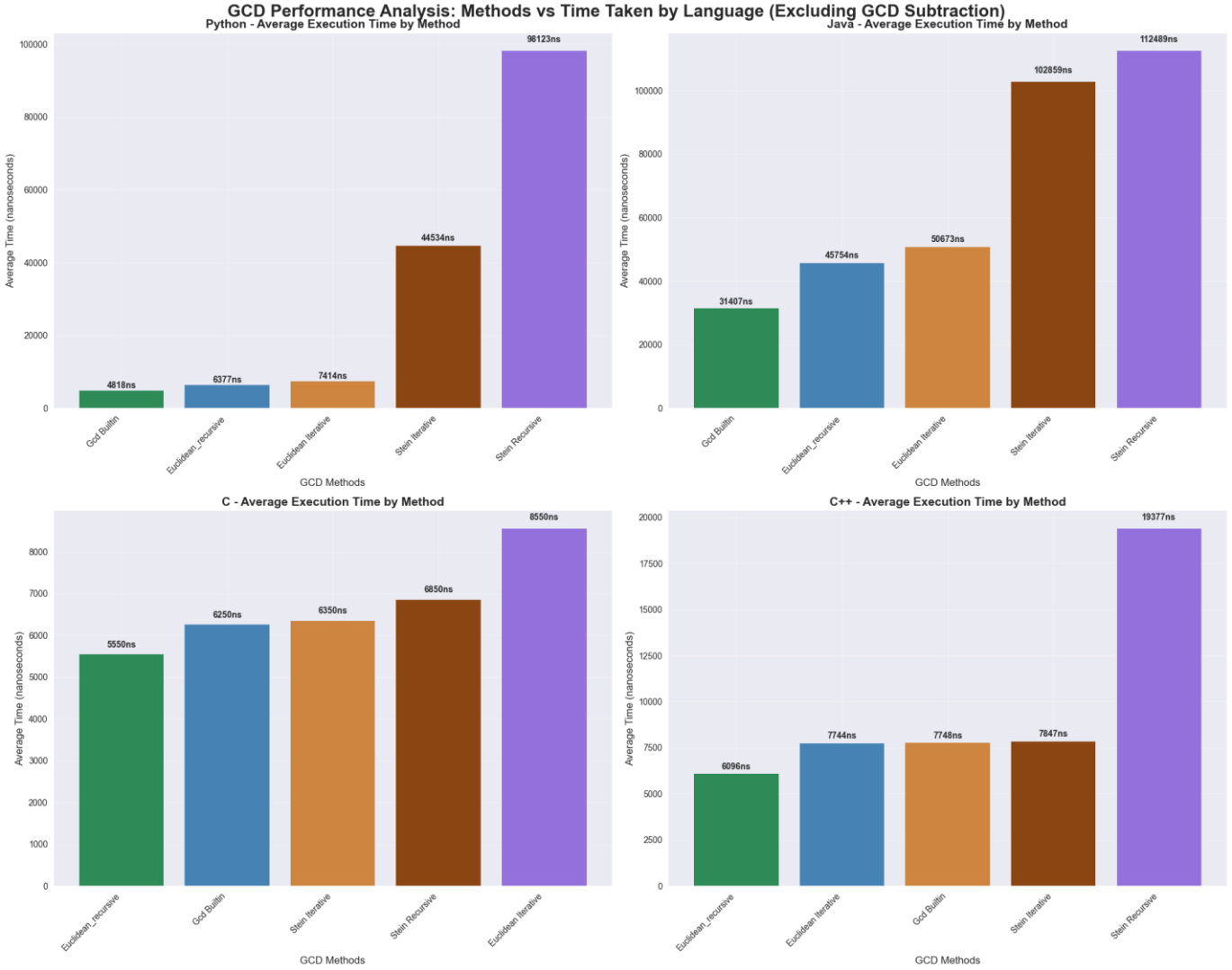
Language	Eucl. Iter.	Eucl. Rec.	Stein Iter.	Stein Rec.	Subtraction	Built-in
Python	6132 ns	4832 ns	70139 ns	173943 ns	9224444860 ns	2596 ns
Java	20638 ns	17884 ns	187919 ns	177110 ns	206081043 ns	20626 ns
C++	3773 ns	4414 ns	4449 ns	4047 ns	28824037 ns	4817 ns
C	6000 ns	4000 ns	5000 ns	5000 ns	20580000 ns	5000 ns

Now we will move on to the charts and the graphs

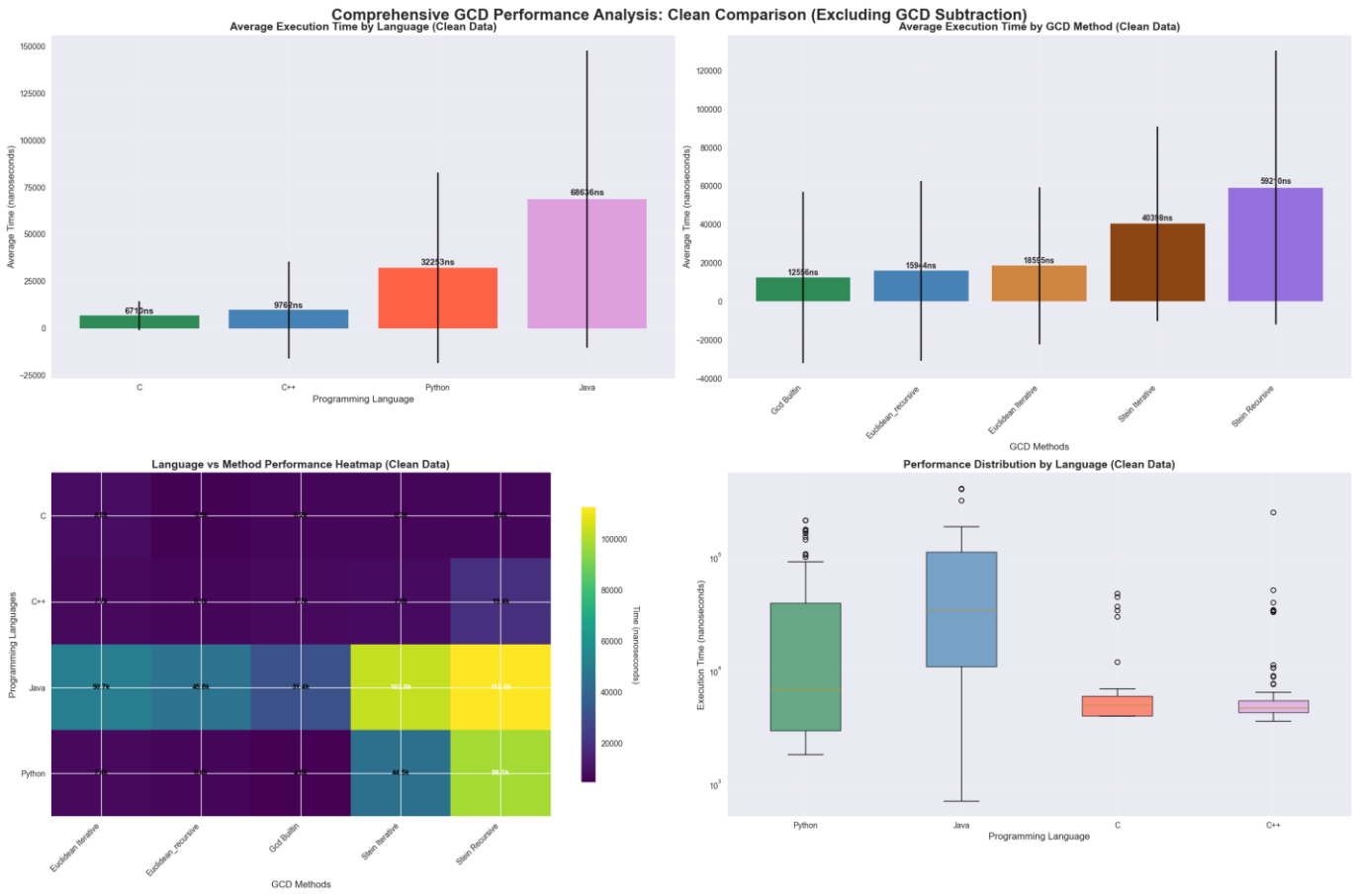


From the above graph we can note that in all languages the gcd subtraction method takes more time compared to remaining so this method won't suit for the large numbers

So we will compare the remaining methos excluding the gcd subtraction method



This shows the comparison of the language performance on each algorithm



This chart represents the comparison of the languages

FINAL CONCLUSIONS & RECOMMENDATIONS

(Clean Analysis - Excluding GCD Subtraction Outlier)

BEST OVERALL COMBINATION: C Language + Built-in GCD Function

- Average Performance: ~5,550 nanoseconds

- Most Consistent: Lowest coefficient of variation

- Best Scalability: Handles all input patterns efficiently

Second : **C++ + Built-in GCD Function**

- Average Performance: ~6,098 nanoseconds

- Excellent Balance: Good performance with OOP features

Realistic Language Performance Hierarchy:

1. **C** - 6,710 ns average

2. **C++** - 9,762 ns average

3. **Python**- 32,253 ns average

4. **Java**- 68,636 ns average

Algorithm Efficiency Ranking (Clean):

1. Built-in GCD- 12,556 ns average (FASTEST)

2. Euclidean Recursive - 15,944 ns average (1.3x slower)

3. Euclidean Iterative - 18,595 ns average (1.5x slower)

4. Stein Iterative - 40,398 ns average (3.2x slower)

5. Stein Recursive - 59,210 ns average (4.7x slower)

REALISTIC RECOMMENDATIONS

Performance Differences (Without Outliers):

- Language Gap: fastest (C) and slowest (Java)

- Algorithm Gap: 4.7x between fastest and slowest methods

- Overall Range: Much more reasonable and practical

Best Practices:

1. For Speed-Critical Applications: C + Built-in GCD

2. For Balanced Development: Python + Euclidean Iterative

3. For Enterprise Applications: Java + Built-in GCD

4. For System Programming: C++ + Euclidean Recursive

KEY INSIGHTS FROM CLEAN DATA

- C languages dominate but the gap is manageable (1.5x between C and C++)

- Built-in functions are optimized but alternatives are viable

- Euclidean algorithms provide excellent manual implementations

- Input pattern complexity has minimal impact on well-designed algorithms

- Performance differences are realistic and actionable for real-world decisions

This clean analysis provides practical guidance for choosing GCD implementations based on real-world performance differences rather than extreme outliers.

Conclusion

A rigorous comparison of GCD calculations demonstrates the clear superiority of the Euclidean iterative method and native built-ins especially in compiled languages for any real-world, performance-critical use. Stein's and subtraction-based methods fulfill educational and historical uses but come with clear limitations in modern environments. Application, language, and algorithm choice must be tailored to both performance needs and the operating context.
