Euclid’s Algorithm for Greatest Common Divisor

Under The Supervision of
Dr. Swapnil Lokhande

Group Members:

Name: PRASAD KAPURE ID:202151115

Name: SNEHAL NALAWADE ID: 202151160

Name: BARAIYA POOJAN ID:202151185

Name: PATEL UTKARSH ID:202152329

Name: SHIVANI MEENA ID: 202152340

Contents:

Introduction

In this Project, we discuss about greatest common Divisor of two numbers and what are the efficient techniques to calculate the GCD of two numbers . One of those techniques is Euclid’s GCD algorithm, which was invented by Euclid of Alexandria , who was a Greek mathematician born in 300 BC. We also compute the time complexity of Euclid’s GCD algorithm . We will compare the time complexity of brute force algorithm vs Euclid’s GCD algorithm.

Divisor

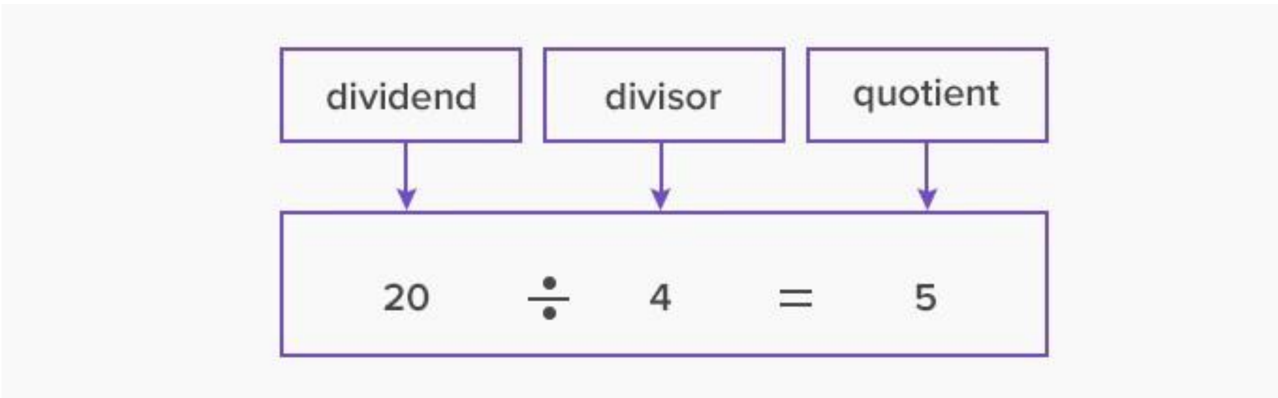
What is Divisor?

A divisor is a number that divides another number either completely or with a remainder.

A divisor is represented in a division equation as:

Dividend ÷ Divisor = Quotient.

On dividing 20 by 4, we get 5. Here 4 is the number that divides 20 completely into 5 parts and is known as the divisor. Its division equation is



Similarly, if we divide 20 by 5, we get 4. Thus, both 4 and 5 are divisors of 20.

For an integer P, we say set S is a set of all divisors of N if

S={ x | P mod x = 0 && x<=P && x ∈ N}

For examples

divisors(144) = { 1, 2, 3, 4, 6, 8, 9, 12, 16, 18, 24, 36, 48, 72, 144}

Code 1: Print all divisors of given number(Brute force)

```
int n;
cin>>n;
for(int i=1;i<=n;i++)
{
      if(n%i==0)
      printf("%d ",i);
}
```

Input: 36

Output: 1 2 3 4 6 9 12 18 36

Time complexity: **O(n)**

Code 2: Print all divisors of given number

```
 int n;
 cin>>n;
for(int i=1;i<=sqrt(n);i++)
{
	if(n%i==0)
	{
		if(i*i != n)
		printf(“%d ”,i);
		else
		printf(“%d %d ”,i,n/i);
	}
}
```

Input : 36

Output: 1 36 2 18 3 12 4 9 6

Time complexity : **O(n^(1/2)**

	Code 1	Code 2
Advantage	Give a all divisors in sorted order	It takes only sqrt(n) steps. Suppose n=36 steps=6.
Disadvantage	It take n steps. Suppose n=36 steps=36.	Give a all divisors in non-sorted order

Code 3: Print number of divisor

```
int a;
cin>>a;
int answer=0;
for(int i=1;i<=sqrt(a);i++)
{
   if(a%i==0)
   {
      if(i*i!=a)
      answer=answer+2;
      else
      answer++;
   }
}
```

```
printf("%d",answer);
```

Input : 12

Output : 6

Time complexity : **O(n^(1/2))**

Common divisors

For two number a and b we say x is a common divisor of a and b if

a mod x=0 and b mod x=0

For example

Divisor(12)={1, 2, 3, 4, 6, 12}

Divisor(18)={1, 2, 3, 6, 9, 18}

Common Divisor(12,18)={1, 2, 3, 6}

Code 1: Print all common divisors of given number (Brute force)

```
 int a,b;
cin>>a>>b;
 for(int i=1;i<=min(a,b);i++)
{
        if(a%i==0 && b%i==0)
        printf("%d ",i);
}
```

Input : 12 and 18

Output: 1 2 3 6

Time complexity : **O(n)**

Code 2: Print all common divisors of given number

```
int a,b;
cin>>a>>b;
map<int,bool>mp1;
```

```
map<int,bool>mp2;
if(a>b)
swap(a,b);
for(int i=1;i<=sqrt(a);i++)
    {
        if(a%i==0)
        {
            mp1[i]=true;
            mp1[a/i]=true;
        }
    }
    for(int i=1;i<=sqrt(b);i++)
    {
        if(b%i==0)
        {
            mp2[i]=true;
            mp2[b/i]=true;
```

```
    }
  }
  for(int i=1;i<=sqrt(a);i++)
  {
    if(mp1[i])
    {
      if(mp2[i])
      printf("%d ",i);
      if(i*i!=a && mp1[a/i] && mp2[a/i])
      printf("%d ",a/i);
    }
  }
```

Input : 12 and 18

Output: 1 2 6 3

Time complexity : **O(n^(1/2))**

	Code 1	Code 2
Advantage	Give a all common divisors in sorted order	It takes only sqrt(max(a,b)) steps.
Disadvantage	It take min(a,b) steps.	Give a all common divisors in non-sorted order

Code 3: Print number of common divisor

```
int     a,b;

cin>>a>>b;

map<int,bool>mp1;

map<int,bool>mp2;

if(a>b)

swap(a,b);

for(int i=1;i<=sqrt(a);i++)

  {

     if(a%i==0)
```

```
    {
        mp1[i]=true;
        mp1[a/i]=true;
    }
}
for(int i=1;i<=sqrt(b);i++)
{
    if(b%i==0)
    {
        mp2[i]=true;
        mp2[b/i]=true;
    }
}
int answer=0;
for(int i=1;i<=sqrt(a);i++)
{
    if(mp1[i])
```

```
        {
            if(mp2[i])
            answer++;
            if(i*i!=a && mp1[a/i] && mp2[a/i])
            answer++;
        }
    }
Printf("%d",answer);
```

Input : 12 and 18

Output: 4

Time complexity : **O(n^(1/2))**

Greatest Common divisors

For two numbers a and b let S be the set of common divisors of a and b then

- G = maximum integer in set S is called the GCD of a and b .
- G = max(s) , where max(s) is the greatest integer in set S.
- In short, the greatest among common divisors of a and be is called a Greatest common divisors of a and b.

If a = 12 and b = 8, then GCD(12,8) = 4.

Code 1: Find GCD (divisor method)

```
int a,b;
cin>>a>>b;
if(a>b)
swap(a,b);
```

```
map<int,bool>mp1;
map<int,bool>mp2;
for(int i=1;i<=sqrt(a);i++)
{
    if(a%i==0)
    {
        mp1[i]=true;
        mp1[a/i]=true;
    }
}
for(int i=1;i<=sqrt(b);i++)
{
    if(b%i==0)
    {
        mp2[i]=true;
        mp2[b/i]=true;
    }
```

```
}
int GCD=1;
for(int i=1;i<=sqrt(a);i++)
{
    if(mp1[i])
    {
        if(mp2[i])
        {
            if(i>GCD)
            GCD=i;
        }
        if(i*i!=a && mp1[a/i] && mp2[a/i])
        {
            if(a/i>GCD)
            GCD=a/i;
        }
    }
```

```
    }
    printf("%d",GCD);
```

Input: 45 and 30

Output: 15

Time complexity : **O(n^(1/2))**

Code 2 : Brute Force

```
int a,b;
  cin>>a>>b;
  int GCD=1;
  for(int i=1;i<=min(a,b);i++)
  {
    if(a%i==0 && b%i==0)
    GCD=i;
  }
  printf("%d",GCD);
```

Input: 45 and 30

Output: 15

Time complexity : **O(n)**

Code 3 : Find GCD (using prime Foctors)

18=2*3*3

12=2*2*3

GCD(12,18)=2*3=6

```
int a,b;
    cin>>a>>b;
    vector<int>v1;
    vector<int>v2;
    while(a%2==0)
    {
        v1.push_back(2);
        a=a/2;
```

```
}
for(int i=3;i<=sqrt(a);i=i+2)
{
    while(a%i==0)
    {
        v1.push_back(i);
        a=a/i;
    }
}
if(a>2)
v1.push_back(a);
while(b%2==0)
{
    v2.push_back(2);
    b=b/2;
}
for(int i=3;i<=sqrt(b);i=i+2)
```

```
{
    while(b%i==0)
    {
        v2.push_back(i);
        b=b/i;
    }
}
if(b>2)
v2.push_back(b);
int GCD=1;
int pointer1=0;
int pointer2=0;
int size1=v1.size();
int size2=v2.size();
while(pointer1<size1 && pointer2<size2)
{
    if(v1[pointer1]==v2[pointer2])
```

```
        {
            GCD=GCD*v1[pointer1];
            pointer1++;
            pointer2++;
        }
        else if(v1[pointer1]<v2[pointer2])
        pointer1++;
        else
        pointer2++;
    }
    printf("%d",GCD);
```

Input: 96 and 144

Output: 48

Time complexity : **O(n)**

GCD property1 : GCD(a,b)=GCD(a-b,b) (a>=b)

: GCD(a,b)=GCD(a,b-a) (b>=a)

Code 4 : Find GCD (using property1)

If a and b both are same stop using property1

This number is GCD

GCD(12,18)=GCD(12,6)

GCD(12,6)=GCD(6,6) (STOP)

GCD=6

```
 int a,b;
cin>>a>>b;
while(a!=b)
{
   if(a>b)
```

```
        a=a-b;
        else
        b=b-a;
    }
    printf("%d",a);
```

GCD property2 : GCD(a,b)=GCD(a%b,b) (a>=b)

: GCD(a,b)=GCD(a,b%a) (b>=a)

Code 5 : Find GCD (using property2)(Euclid's algorithm)

<u>Euclid's GCD algorithm :</u>

• Input : two integers x and y.

• Output : GCD(x,y)

1. let x > y and if x < y then swap x and y .
2. while b is not zero do as follows –

a. Temporary_variable = x mod y;

b. x = y;

c. y = Temporary_variable;

d. finally x is nothing but a GCD of input values of x and y

CODE:

```
int a,b;
cin>>a>>b;
while(a!=0 && b!=0)
{
   if(a>b)
   a=a%b;
   else
   b=b%a;
}
if(a==0)
```

```
printf("%d",b);
else
printf("%d",a);
```

Input: 96 and 144

Output: 48

Time complexity : **O(log(min(a,b))**

<u>Proof of Euclid’s GCD Algorithm :</u>

For two integers a and b Euclid’s algorithms works

as follows

a > b

a = b*q + r0 (by division algorithm)

b = r0*q1 + r1

r0 = r1*q2 + r2

..

rn = (rn+1*qn+2)+ 0 (algorithm terminates)

• First we show that the algorithm terminates.

Since $r_{i+2} < r_{i+1}$, we have

- $r_0 > r_1 > r_2 > \cdots > r_n > r_{n+1} = 0$.
- This shows that the remainders are monotonically strictly decreasing positive integers until the last one, which is $r_{n+1} = 0$. Therefore the algorithm stops after no more than b divisions.
- We prove by induction the claim that for each i in $0 \le i \le n$ we have $\gcd(a, b) = \gcd(r_i , r_{i+1})$.
- For the base step $i = 0$, we have $\gcd(a, b) = \gcd(r_0, r_1)$ by definition of $r_0 = a$ and $r_1 = b$. For each i in $0 \le i < n$ we have $\gcd(r_i , r_{i+1}) = \gcd(r_{i+1}, r_{i+2})$.
- This shows that if $\gcd(a, b) = \gcd(r_i , r_{i+1})$, then $\gcd(a, b) = \gcd(r_{i+1}, r_{i+2})$, which is the induction step.
- This ends the proof of the claim. Now use the

claim with i = n: gcd(a, b) = gcd(rn, rn+1). But rn+1 = 0 and rn is a positive integer by the way the Euclidean algorithm terminates. Every positive integer divides 0. If rn is a positive integer, then the greatest common divisor of rn and 0 is rn. Thus, the Euclidean algorithm correctly computes the greatest common divisor of its input a and b as gcd(a, b) = rn.

<u>Time Complexity of Euclid's algorithm :</u>

The time complexity of this algorithm is O(log(min(a,b));

The time complexity of brute force algorithm is O(min(a,b));

a	b	Brute force algorithm (steps)	Euclid’s algorithm (steps)
10	20	10	2
100	16563	100	6
1000	165156	1000	9
10000	1561627	10000	14
100000	56167466	100000	17

This table clearly shows how efficient Euclid’s algorithm is !