

0.1 GCD and Bezout's identity

- *Algorithm*: Euclidean (algo. 1), Extended Euclidean (algo. 2)
- *Input*: Two integers a and b
- *Complexity*: $\mathcal{O}(\log(\min(a, b)))$
- *Data structure compatibility*: N/A
- *Common applications*: Modular arithmetic, such as RSA encryption

Problem. GCD and Bezout's identity

Given two integers a and b , find out the greatest common divisor d , and the Bezout's identity x and y such that $ax + by = d$.

Description

GCD is the abbreviation of the greatest common divisor, which is important in cryptography because it can decide whether two integers are coprime or not [1]. Assume that $a > b$, the trivial way to calculate the GCD is to use a loop, and perform a modular calculation at each step from 1 to b to find it. However, when the integer is very large, the running time of this method can be very low, since it has a time complexity of $\mathcal{O}(b)$. Therefore, the Euclidean algorithm is designed to solve GCD in a faster way, which has a time complexity of $\mathcal{O}(\log(\min(a, b)))$. Also assume that $a > b$, first calculate $r = a \bmod b$, then repeat the process for b and r and so on, until the remainder reaches 0, and the previous divisor b is the result.

Euclidean algorithm

Suppose that it takes N steps to use Euclidean algorithm to calculate the GCD. Denote f_N as the N_{th} number of Fibonacci series, and we can prove that $a \geq f_{N+2}$ and $b \geq f_{N+1}$ using mathematical induction. Since The N_{th} Fibonacci number has the expression

$$f_N = \frac{1}{\sqrt{5}} \left[\left(\frac{1 + \sqrt{5}}{2} \right)^N - \left(\frac{1 - \sqrt{5}}{2} \right)^N \right] \approx \phi^N$$

where $\phi \approx 1.618$, then golden ratio. So we can get $N \approx \log_{\phi}(f_N)$. Assume that $a > b$, then we can deduce that $f_{N+1} \approx b$, and get $N + 1 \approx \log_{\phi}(b)$, and we finally get to the point that the time complexity of Euclidean algorithm is $\mathcal{O}(\log(\min(a, b)))$.

Algorithm 1: Euclidean

Input : Two integers a, b

Output: The greatest common divisor d

```
1 Function GCD( $a, b$ ):  
2   if  $b = 0$  then  
3     return  $a$ ;  
4   end if  
5   return GCD( $b, a \bmod b$ )  
6 end
```

Bezout's identity and Extended Euclidean algorithm

Bezout's identity claims that given two integers a, b and their greatest common divisor d , we can find a unique pair of coefficients x, y such that $ax + by = d$. Extended Euclidean algorithm is the extension of the traditional Euclidean Algorithm, which is created to get the Bezout's coefficient of a and b . It shares the same time complexity with the traditional Euclidean algorithm, which is $\mathcal{O}(\log(\min(a, b)))$. Moreover, it calculates the two coefficients x and y at the same time of the GCD d during recursions.

Algorithm 2: Extended Euclidean

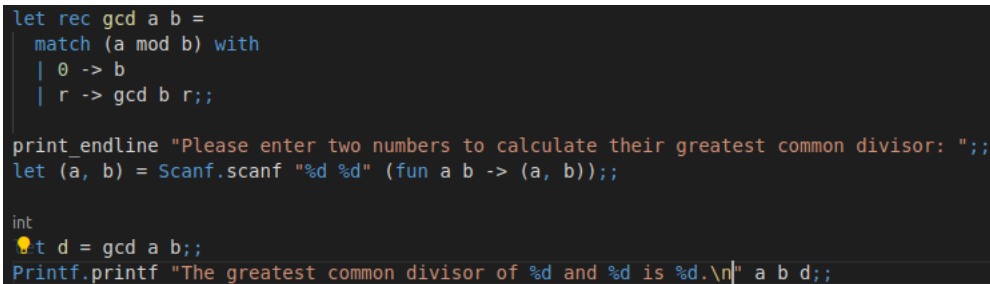
Input : Two integers a, b

Output: a tuple d, x, y , The greatest common divisor d , and the Bezout's identity x, y

```
1 Function extendedGCD( $a, b$ ):
2   if  $b=0$  then
3     | return ( $a, 1, 0$ );
4   end if
5   ( $d1, x1, y1$ )  $\leftarrow$  extendedGCD( $b, a \bmod b$ );
6    $d \leftarrow d1$ ;
7    $x \leftarrow y1$ ;
8    $y \leftarrow x1 - a/b * y1$ ;
9   return ( $d, x, y$ )
10 end
```

OCaml implementation

Below is the OCaml code screenshot, and the code will be attached in the compressed file.

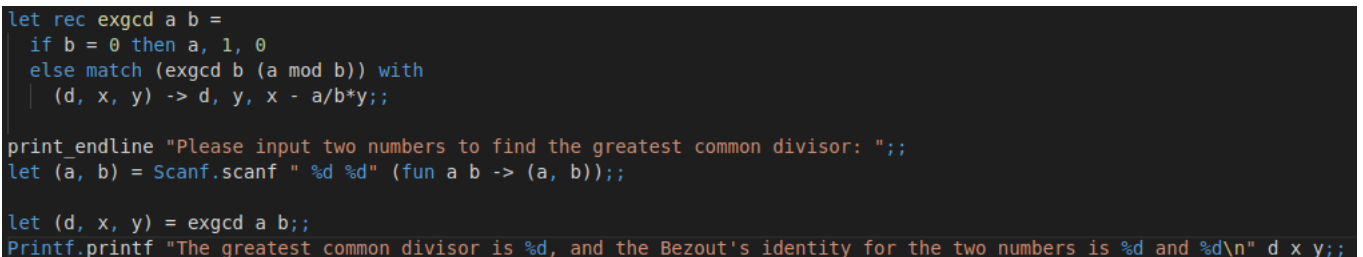


```
let rec gcd a b =
  match (a mod b) with
  | 0 -> b
  | r -> gcd b r;;

print_endline "Please enter two numbers to calculate their greatest common divisor: ";
let (a, b) = Scanf.scanf "%d %d" (fun a b -> (a, b));;

int
let d = gcd a b;;
Printf.printf "The greatest common divisor of %d and %d is %d.\n" a b d;;
```

Figure 1: Implementation of Euclidean algorithm



```
let rec exgcd a b =
  if b = 0 then a, 1, 0
  else match (exgcd b (a mod b)) with
  | (d, x, y) -> d, y, x - a/b*y;;

print_endline "Please input two numbers to find the greatest common divisor: ";
let (a, b) = Scanf.scanf " %d %d" (fun a b -> (a, b));;

let (d, x, y) = exgcd a b;;
Printf.printf "The greatest common divisor is %d, and the Bezout's identity for the two numbers is %d and %d\n" d x y;;
```

Figure 2: Implementation of Extended Euclidean algorithm

References.

- [1] Manuel. VE477 – Introduction to Algorithms (lecture slides). 2019 (cit. on p. 1).