0.1 GCD and Bezout's identity

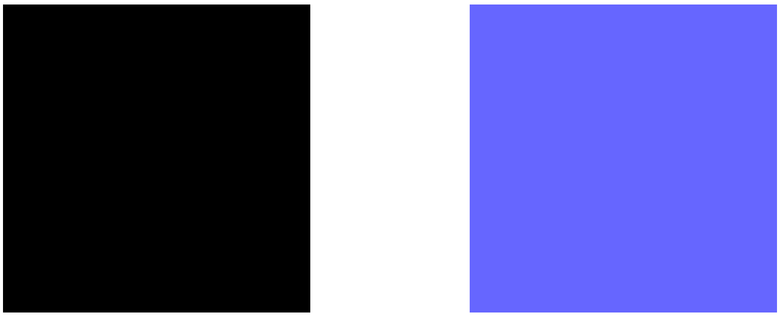
- *Algorithm:* Euclidean (algo. **??**), ExtendedEuclidean (algo. **??**)
- *Input:* Two integers a and b
- *Complexity:* $\mathcal{O}(\log(\min(a, b)))$
- *Data structure compatibility:* N/A
- *Common applications:* Modular arithmetic, such as RSA encryption

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. 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 $r = a \mod b$, then repeat the process for b and r and so on, until the remainder reaches 0, and the previous divisor b is the result.



(a) Pic. 1

(b) Pic. 2

Figure 1: Group of pictures

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

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

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

```
Function GCD(a, b):
    if b = 0 then
        return a;
    end if
    return GCD(b, a mod b)
end
```