Public Key Cryptography – Lab2

Algo 1: Euclidean algorithm (division based)

```
function gcd(a, b)
    while b ≠ 0
       t := b;
       b := a mod b;
       a := t;
    return a;
```

For implementing this algorithm I have used the GMP library in order to be able to work with arbitray length integer numbers. The documentation of the functions that handle big numbers is available at https://gmplib.org/manual/Integer-Functions.html

Algo 2: Stein's algorithm

1. If both and b are 0, gcd is zero $gcd(0,0) = 0$.
2. $gcd(a,0) = a$ and $gcd(0,b) = b$ because everything divides 0.
3. If a and b are both even, $gcd(a,b) = 2 * gcd\left(\frac{a}{2},\frac{b}{2}\right)$ because 2 is a common divisor. Multiplication with 2 can be done with bitwise shift operator.
4. If a is even and b is odd, $gcd(a,b) = gcd\left(\frac{a}{2},b\right)$. Similarly, if a is odd and b is even, then $gcd(a,b) = gcd\left(a,\frac{b}{2}\right)$. It is because 2 is not a common divisor.
5. If both a and b are odd, then $gcd(a,b) = gcd\left(\frac{|a-b|}{2},b\right)$. Note that difference of two odd numbers is even
6. Repeat steps 3–5 until $a = b$, or until $a = 0$. In either case, the GCD is $2^k b$, where k is the number of common factors of 2 found in step 2.

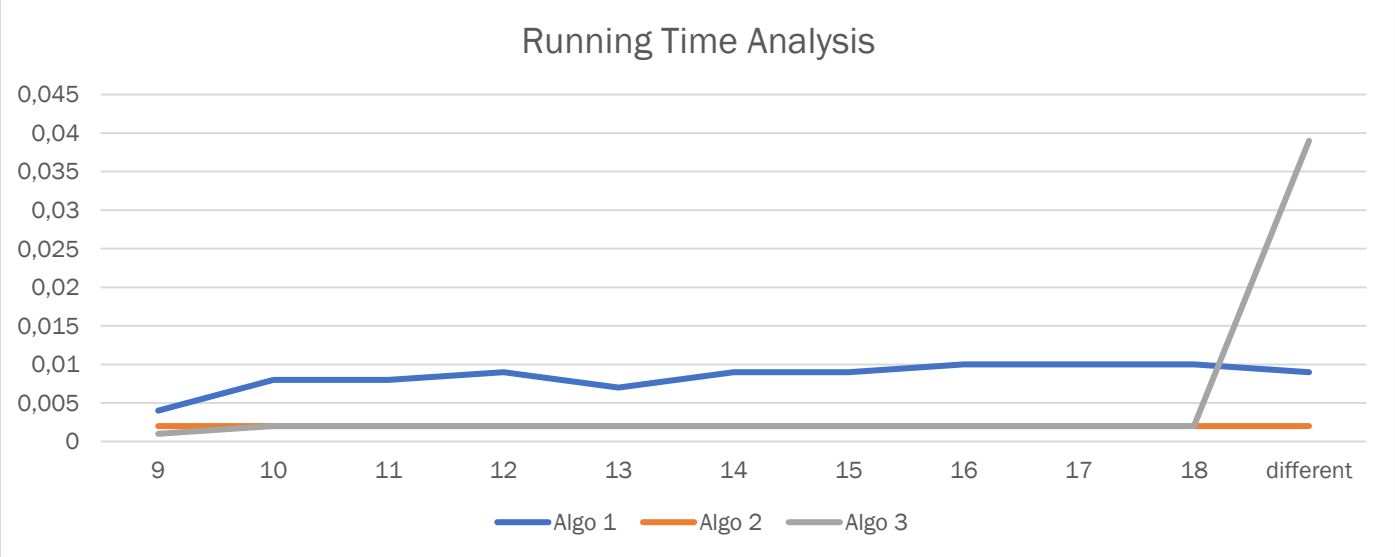
Complexity: $O(n^2)$, where n is the number of bits in the larger number

Algo 3: Euclidean algorithm (subtraction based)

```
function gcd(a, 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;
```

Running Time Analysis

Number of digits	Algo 1	Algo 2	Algo 3	inputs
9	0,004	0,002	0,001	139478521 952107352
10	0,008	0,002	0,002	4399960924 4287523996
11	0,008	0,002	0,002	52762160628 44353821646
12	0,009	0,002	0,002	338572204080 590412399540
13	0,007	0,002	0,002	1157853731689 3321920574455
14	0,009	0,002	0,002	68154387535207 95123251917189
15	0,009	0,002	0,002	319783614859866 110351283242007
16	0,01	0,002	0,002	3178618382627662 9448851122642290
17	0,01	0,002	0,002	69438758074573737 72396883010134320
18	0,01	0,002	0,002	494319003743625691 563038720333905941
different	0,009	0,002	0,039	626028641712083864 34003534629305
very different	0,009	0,001	37,272	6266417120564 337578



On the x-axis we input scenarios:

- same number of digits, with number of digits from 9 to 18
- different number of digits
- very different number of digits

On the y-axis we have the real execution time (in milliseconds).

Vlad Cîncean / 933
cvie1883@scs.ubbcluj.ro