1 Problem statement

Create a program than contains an implementation of 3 different algorithms for computing the greatest common divisor of 2 natural numbers.

Show a graph of the running time of these algorithms.

2 Algorithms

2.1 Prime factorization

Ensure: $bc = gcd(a, b)$
  procedure PRIME_FACTORIZATION(a,b) ▷ Complexity: $\theta(2^n)$
    $div_a \leftarrow$ divisors(a)
    $div_b \leftarrow$ divisors(b)
    $div_c \leftarrow div_a \cap div_b$
    prime_factorization $\leftarrow \prod div_c$
  end procedure

2.2 Euclid's algorithm

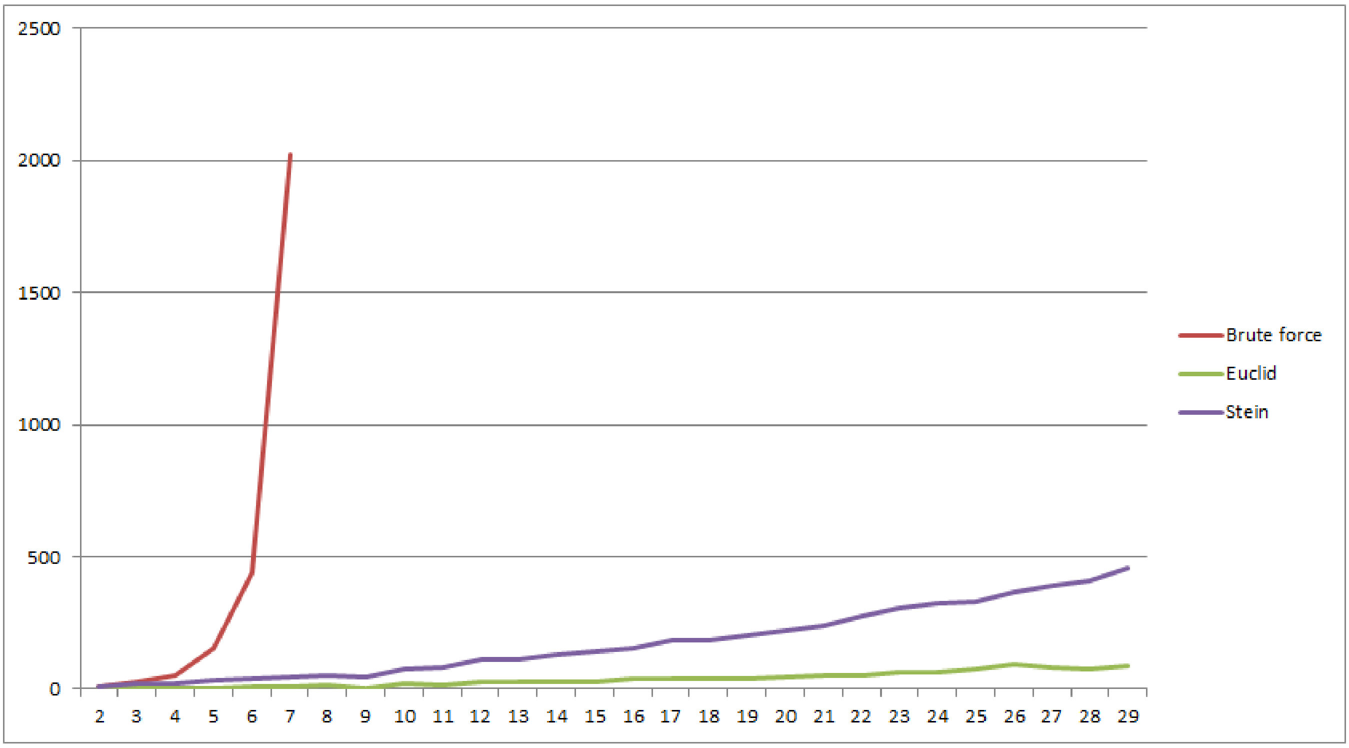
Ensure: $bc = gcd(a, b)$
  procedure EUCLID(a,b) ▷ Complexity: $\theta(n)$
    if $a == 0$ **then**
      euclid $\leftarrow$ b
    end if
    if $b == 0$ **then**
      euclid $\leftarrow$ a
    end if
    while $a > 0$ **do**
      temp $\leftarrow$ a
      a $\leftarrow b \mod a$
      b $\leftarrow$ temp
    end while
    euclid $\leftarrow$ a
  end procedure

2.3 Stein

Ensure: $bc = gcd(a, b)$

```
procedure EUCLID(a,b)                                   ▷ Complexity: θ(n)
    if b == a then
        euclid ← a
    end if
    if a == 0 then
        euclid ← b
    end if
    if b == 0 then
        euclid ← a
    end if
    if a mod 2 = 0 then
        if b mod 2 = 1 then
            stein ← stein(a/2, b)
        else
            stein ← stein(a/2, b/2) * 2
        end if
    end if
    if b mod 2 = 0 then
        stein ← stein(a, b/2)
    end if
    if a > b then
        stein ← stein((u − v)/2, v)
    else
        stein ← stein((v − u)/2, u)
    end if
end procedure
```

3 Runtime analysis



As can be seen from the graph, the naive, brute-force algorithm has an exponential complexity, and can't be used with numbers with more than 7-8 digits. Stein's and Euclid's algorithms are linear in time, but Euclid's has a smaller overhead, because it doesn't involve doing so many comparisons and divisions.