Computational complexity and one-way functions in cryptography

Fibonacci numbers

In mathematics, the **Fibonacci numbers** - commonly denoted F_n - form a sequence called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is

$F_0 = 0 \quad F_1 = 1 \quad F_n = F_{n-1} + F_{n-2}$ for $n > 1$.

The beginning of the sequence is thus:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

Two of the many different algorithms (and, so, computer programs) to compute Fibonacci numbers are the following (here written in Python)

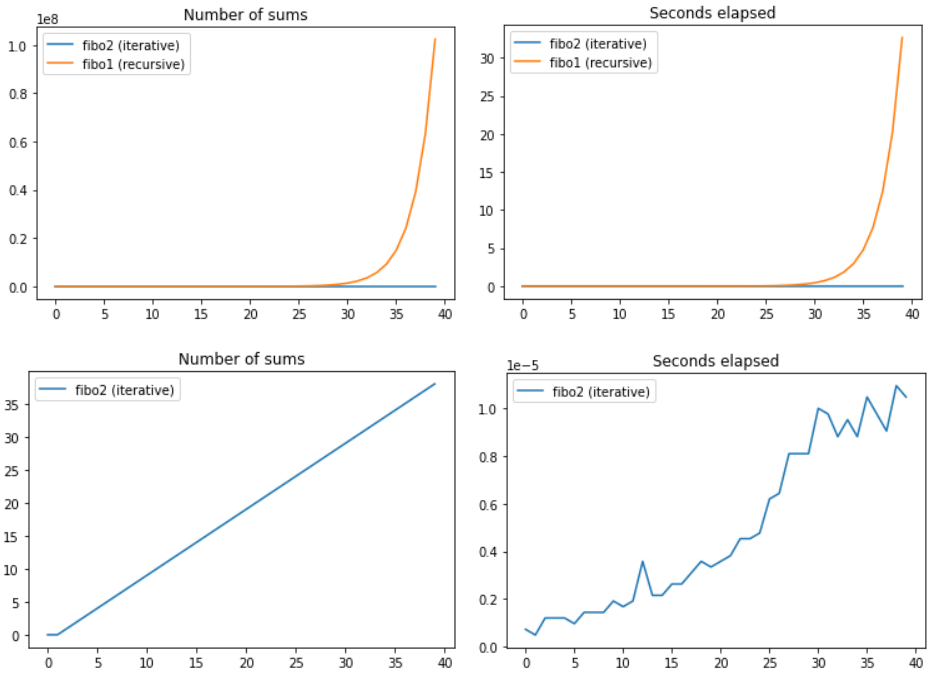
```
def fibo1(n):
    if n<2:
        return n
    return fibo1(n-1) + fibo1(n-2)

def fibo2(n):
    a, b = 1, 0
    while n > 1:
        a, b = a+b, a
        n = n - 1
    return a
```

In the code here
https://mybinder.org/v2/gh/michaellodi/RawInterdisciplinaryMaterial/HEAD?labpath=Fibonacci.ipynb
some instructions and functions have been added to compute how much time the two functions take.
You can `Run -> Run all cells` and wait around 90 seconds to see the results.

The program helps to experimentally measure and visualize what is called the “computational complexity” of the two algorithms/functions/programs.

In this case, the complexity is measured by counting *the number of steps* (we can consider just *the number of sums* without loss of generality) and by measuring the time spent executing (a trend should be visible, although time is heavily machine-dependent).
Complexity is measured with respect to the increase of number n.

IDENTITIES Enlightening Interdisciplinarity in STEM for Teaching
Co-funded by the Erasmus+ Programme of the European Union
Grant Agreement n°2019-1-IT02-KA203-063184



It is infeasible to use fibo1 for not-so-big numbers like 40 or greater.

Factorization

Only in February 2020, *with months of huge computing time*, the number
214032465024074496126442307283933356300861471514475501779775492088141802344714013664334551909580467961099285187247091458768739626192155736304745477052080511905649310668769159001975940569345745223058932597669747168173806936489469987157849497593749793 7

has been factored for the first time into its two prime factors:
641352894770715802787901901705773890848250147429434472081168596320245323446302386235987526683477087376619255856946397988533 67 *
333720275949781565562260106053551142279407603447675546667845209870238417292100370802574486732968818775657189862580369320627 11

However, it is relatively easy for a computer to perform this multiplication (at least it is *straightforward and almost instantaneous* in [Python3 (click to run)]).

Here
https://mybinder.org/v2/gh/michaellodi/RawInterdisciplinaryMaterial/HEAD?labpath=Factorization.ipynb
you can try both a naive algorithm and one of the fastest known algorithms for factoring numbers in primes.

Co-funded by the Erasmus+ Programme of the European Union
Grant Agreement n°2019-1-IT02-KA203-063184