Project Euler – Problem 3 Solution

Largest Prime Factor

This problem can be solved in many different ways. Suppose that we want to find the largest Prime Factor for a number N. One possible solution of the problem would be to list all the prime numbers until $\sqrt{N}$ and find thus the largest prime number that divides the number N.

However the solution I implemented in java, is based on finding all the prime factors of number N. Starting from a number *a = 2*, we divide repeatedly the number N until, it can no longer be divided with *a*. Then we increment *a* by 1 and do the same procedure with the following number and so forth, until N reaches the number 1 by the divisions.

The largest number that divided N in this procedure is prime. It can be easily proved that if this number is not prime, then we are lead to a contradiction. Suppose that the largest number is ***a'***, then ***a'*** consists of at least two smaller factors **b,c,** which are prime and apparently are also divisors of N. However, the algorithm in a previous step divides repeatedly N by **b** until N is no longer divided by this number. This is not possible, therefore our claim that ***a'*** is prime, is proven correctly.

Evidently, this implementation is not optimal, since we do several unnecessary computations. For example once N can no longer be divided by 2, there is no point of trying to divide N with 4, 6, 8… There are techniques based on the idea of **Eratosthenis Sieve**, which could accelerate the algorithm. However this implementation seems to solve quite fast the problem.