Universidad de Málaga

Escuela Técnica Superior de Ingeniería Informática

Grado en Ingeniería de la Salud

205 - Estructura de Datos y Algoritmos

Factorización de enteros

Informe de laboratorio. Práctica 1.

Autora:
Arrabalí Cañete, Carmen Lucía
2.º Grado en Ingeniería de la Salud
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga
carmenarrabali@uma.es

Supervisor:
D. Carlos Cotta Porras
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga
ccotta@uma.es

6 de octubre de 2021

1. Introducción y Objetivos

En esta práctica se va a estudiar el algoritmo de fuerza bruta para calcular los factores primos de un número entero, el cual se va a escribir en un entorno Java utilizando IntelliJ IDEA como IDE.

El problema parte de tener un número $n \in \mathbb{N}^+$, encontrar una lista de números $P =< p_1, p_2, ..., p_t >$, cuya longitud es desconocida a *priori*, que cumpla lo siguiente:

- $p_i \in \mathbb{N}^+$ es primo para todo $1 \leqslant i \leqslant t$.
- $p_1 \leqslant ... \leqslant p_t$
- $\prod_{i=1}^{t} p_i = n$

2. Aplicación y Funcionamiento

A grandes rasgos, lo que hace la fuerza bruta en este problema es, en primer lugar comprobar todos los divisores potenciales del número que se analiza que esté entre 2 y el máximo divisor el cuál se calcula como $maxdiv = \lfloor \sqrt{n} \rfloor$.

En segundo lugar, si encuentra el divisor p, será primo, lo agrega a la lista, simplifica n y actualiza el valor que se encuentra en $maxdiv$.

Por último, una vez que el algoritmo ha terminado de comprobar los requisitos, si nos queda un número $n > 1$, ese también sería primo y deberá ser agregado a la lista.

Cabe mencionar que la lista tendrá un orden ascendente, es decir, los valores primos más pequeños deben aparecer al principio de la misma.

Por lo tanto, el método que hace todas esas comprobaciones se puede observar en el apartado 3.1.

Por otro lado, entre los recursos se tiene un archivo llamado *analyze.R* que contiene un script el cual se encarga de procesar los ficheros que se generan con uno de los modos del archivo *TestBruteForceFactorization*, el cual también se proporciona en el Campus Virtual.

3. Resultados y Conclusiones

Al ejecutar el archivo *analyze.R* en el entorno R, crea un gráfico, el cuál se muestra a continuación y en el que se observa que si el valor analizado tiene un número alto de dígitos, como puede 15, tarda más tiempo en hallar la lista de números primos que lo conforman que un número de 6 dígitos.

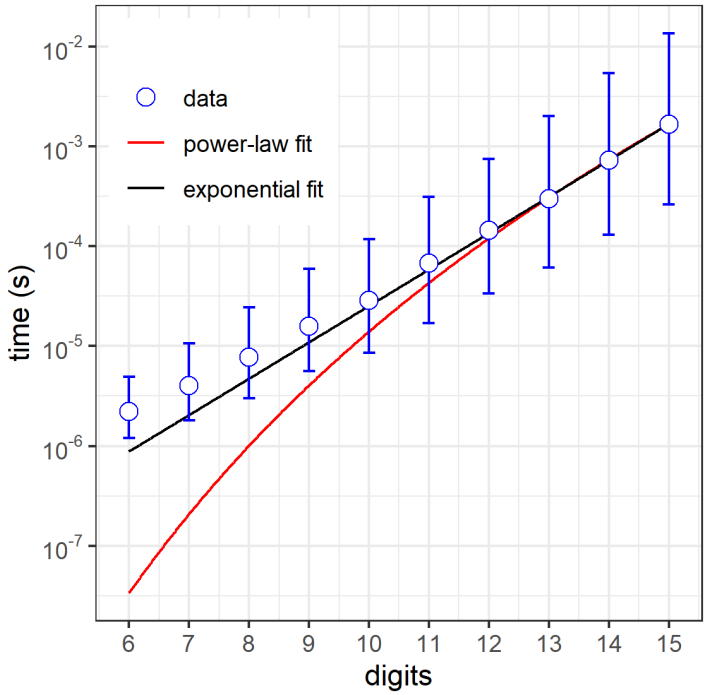

Figura 1: Gráfica que evalúa el tiempo de ejecución en función del número de dígitos que tenga el valor que se analiza.

Para comprobar realmente que el algoritmo funciona de manera correcta, se van a probar los siguientes números aleatorios 15, 21, 34 y 128445, cuya factorización teórica es $[3, 5]$, $[3, 7]$, $[2, 17]$ y $[3, 5, 8563]$, respectivamente. Al ejecutar el programa introduciendo dichos números en un archivo de texto y pasándoselo como parámetro, da el siguiente resultado, el cual cumple con los resultados teóricos.

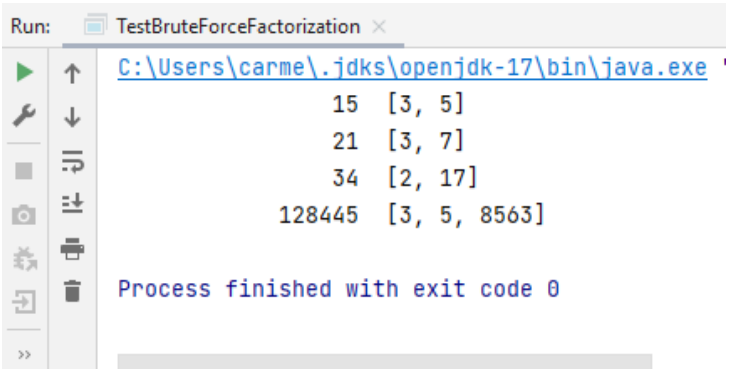

Figura 2: Resultado de la ejecución del algoritmo con el archivo de prueba.

3.1. Método *_factorize* de la clase *BruteForzeFactorization*

```
public List<Long> _factorize(long n) {
        List<Long> factors = new LinkedList<Long>();
        maxdiv = (int) Math.floor(Math.sqrt(n));
        i = 2;
        while(i <= maxdiv) {
                if(n % i == 0){
                        do {
                                factors.add((long) i);
                                n = n/i;
                        } while (n % i <= 0);
                        maxdiv = (int) Math.floor(Math.sqrt(n));
                } i++;
        } if (n > 1) {
                factors.add(n);
        } return factors;
}
```