Proyecto Programado: Factorizando Números - Paolo Induni Ocampo

1. Resumen Ejecutivo

El teorema fundamental de la aritmética dice que cualquier número entero mayor a 1 puede ser representado como un producto de números primos, esto puede ser relacionado directamente con la factorización, la cual es una técnica que se puede utilizar para descomponer un número en la multiplicación de varios valores primos.

Si bien existen varios tipos de factorización, la mayoría de algoritmos que se utilizan coinciden en algunos principios, por lo que es crucial poder identificar y entender la variedad de procedimientos utilizados si se desea poder aprovechar las fortalezas de cada uno y reconocer en cuáles escenarios podrían brindar su máximo potencial, para ello, en este proyecto se van a desarrollar las diferentes funciones relacionadas con cada uno de los tipos de factorización.

Para poder alcanzar el objetivo, primero se tiene que entender una de las principales características compartidas entre los métodos a utilizar, la cual es división entre números primos, esta será la base de las funciones y facilitará mucho más trabajar las mismas, para todos los métodos, se realizaron 10 pruebas con 100 números diferentes, pudiendo variar desde 1000 hasta 10 x10^{15} y no mayores a un límite previamente establecido, todos los números son cercanos a este límite para asegurar de que las pruebas se ejecuten en los escenarios más exigentes posibles para su rango.

El modo 1 logró factorizar números no muy grandes sin demasiada dificultad, de hecho, no existió una diferencia de tiempo significativa con respecto a los demás, aunque, al aumentar los números, este sistema deja de funcionar correctamente. Distinto al anterior, el modo 2 pudo resolver casi todos los casos, lamentablemente, el tiempo incrementa fuertemente conforme crecen los números. El tercer método fue rápido para obtener los factores, aunque se rompe cuando alcanza el límite de recursión. El modo 4 logra resolver las operaciones sin mucha dificultad, incluso alcanzó el límite establecido para las pruebas en un margen de tiempo aceptable. Finalmente, el modo 5 es el único que utiliza una metodología diferente y más desarrollada, lo que le permitió conseguir los tiempos más bajos.

Los modos 1 y 3 se vieron directamente afectados por el uso de la recursión (Aunque este no fue el principal problema del modo 1, sino su simplicidad, ya que requería repetir los algoritmos muchas más veces que los demás), la cual limita la cantidad de ejercicios que podemos realizar, de hecho, esto se evidencia al analizar los modos 2 y 4, los cuales a pesar de tener prácticamente los mismos algoritmos, fueron capaces de realizar operaciones mucho más grandes por el simple hecho de utilizar iteración. Finalmente, el modo 5 logró aprovechar al máximo el uso de la Criba de Eratóstenes para simplificar su algoritmo.

Se podría decir que los resultados son los anticipados y van mejorando conforme aumenta la complejidad del algoritmo, pero definitivamente, este balance parece ser lo que permite que este problema pueda tener tantas soluciones, desde

un algoritmo simple como el modo 1, hasta una función compuesta y eficiente como lo es la del modo 5, aunque si se busca determinar cuál de todas tiene mayor potencial, se debe destacar el planteamiento del modo 5.

2. Introducción

La factorización de números consiste en poder expresar el valor de un número con una multiplicación de números primos, para poder hacer esto, debemos encontrar cuáles valores pueden formar parte de la operación de manera que la división sea exacta, esto abre paso a múltiples procedimientos que nos pueden guiar a obtener el resultado correcto de una factorización.

En este caso, se desea desarrollar 5 métodos de factorización programables, a los cuales llamaremos "modos", utilizando las estrategias de programación de "recursión" e "iteración", para obtener los resultados de manera satisfactoria, cada modo cuenta con sus propias operaciones.

Con el objetivo de llegar a una solución, se estarán desarrollando las funciones respectivas para cada modo, siguiendo el principio de división entre números primos, aunque, en el quinto modo, se acompañará ese mismo principio utilizando una "Criba de Eratóstenes" para factorizar los números, también, se espera poder demostrar si el uso de este método podría potencialmente acelerar el proceso.

Así mismo, buscamos probar cuáles métodos son más rápidos o eficaces que otros, midiendo el número más grande con el que funcionan de forma correcta y el tiempo con el que logran realizar la factorización exitosamente, así también se puede determinar cuál tiene una mejor relación entre complejidad y eficiencia.

Índice

3. Marco teórico

Python, sucesor del lenguaje ABC, es un lenguaje de código abierto creado por Guido van Rossum a finales de los 80 mientras trabajaba en el sistema operativo Amoeba. Su versión 2.0 se lanza en octubre del año 2000, la cual contaba con algunas características nuevas para los lenguajes de su tipo, aunque esto no sería lo que impulsaría su avance para que llegara a ser el lenguaje que conocemos hoy en día, ya que este comenzó a ser desarrollado por la comunidad, abriendo paso a la versión 3.0. Esta versión es muy diferente a las anteriores lanzadas 8 años antes, aunque, sus desarrolladores han realizado un gran esfuerzo para compatibilizar varias características con la versión 2.6 de python.

Este lenguaje, a pesar de ser "sencillo", es bastante completo y cuenta con muchas herramientas para ayudar al usuario, lo que ha vuelto a Python uno de los favoritos en la programación moderna, haciendo que aún más personas se sumen al desarrollo de bibliotecas que faciliten procesos y aumenten la eficacia del mismo. Mencionando algunos ejemplos, este proyecto utiliza la librería SYS, la cual es una de las más básicas, esta permite acceder a variables y funciones contenidas por el intérprete para poder manipular varias funciones y algunas limitaciones del IDE.

En este caso, Pycharm es el entorno de programación utilizado para este proyecto, es una herramienta que permite al usuario trabajar con Python 3.0 e implementar tanto librerías como herramientas que facilitan el trabajo a la hora de programar, es uno de los IDEs más recomendados para windows, ya que logra aprovechar de la mayoría de recursos que python tiene a su disposición, sin llegar a ser demasiado complejo o consumir recursos excesivos.

4. Descripción de la solución

4.1. Modo 1 y 2

Estos métodos intentan dividir el número que se desea factorizar con un posible factor menor a la mitad del mismo (número//factor), si es divisible, se guardará ese factor y repetirá el proceso con el resultado de esta división como el número a factorizar, en cambio, si el número no es divisible entre el factor actual, aumenta el posible factor en 1 (factor+1) hasta que se llegue a sobrepasar el límite antes establecido (factor > número//2), cuando esto pase, la función retornará el "residuo" como el último de sus factores.

Ambos modos utilizan la misma fórmula, aunque la forma de trabajar los valores cambia, ya que el modo 1 utiliza recursión (Funciones que se llaman a sí mismas), mientras que por otro lado, el modo 2 usa Iteración (Contadores dentro de un límite establecido).

4.2. Modo 3 y 4

Ambas formas revisan si el número que se desea factorizar es par o impar (número % 2), si es par lo divide entre 2 las veces que sea posible (Agrega 2 como factor tantas veces hayamos dividido) hasta llegar a un valor impar o primo (cuyo caso sería el último factor), cuando el número sea impar, entonces se verifica si es divisible por el factor actual (número % factor), luego tomará como posible factor inicial el 3, y en caso de que no sea posible realizar la división probará con el siguiente número impar (factor+2) hasta dar con un factor del número en cuestión, este aumenta hasta que el posible factor sea mayor a la raíz del mismo ($\text{factor}^2 > \text{número}$), cuando se presente este caso, se habrá obtenido el último factor.

Igual a como ocurre con los modos 1 y 2, ambos utilizan la misma fórmula, con la diferencia de que el modo 3 utiliza recursión (Funciones que se llaman a sí mismas), mientras que el modo 4 usa Iteración (Contadores dentro de un límite establecido).

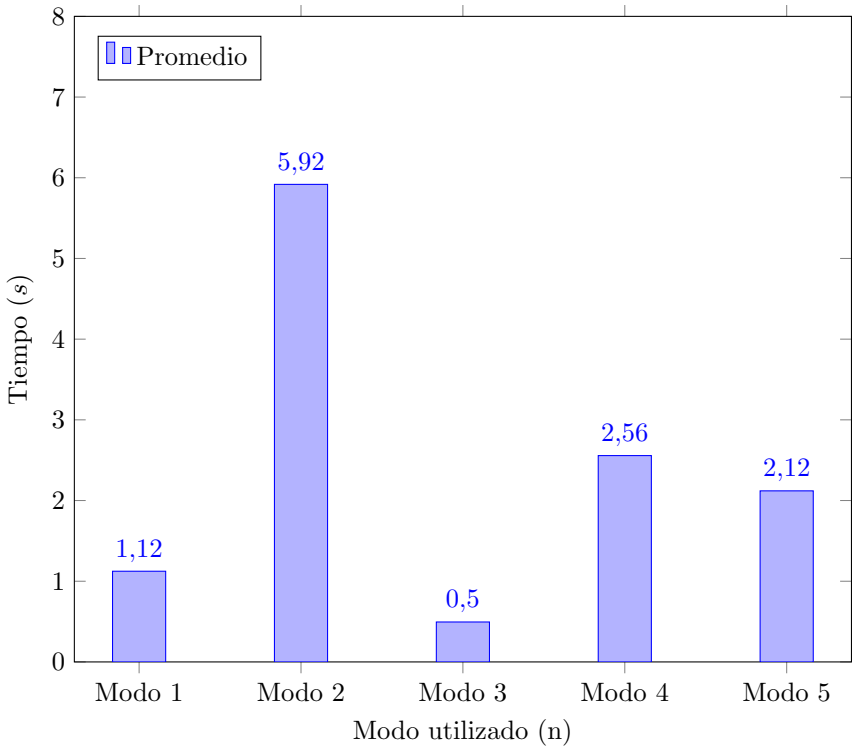
4.3. Modo 5

Este es probablemente el modo con el procedimiento más diferente de todos los que se pusieron a prueba en este proyecto, ya que inicia generando una "Criba de Erastótenes" (Una Criba de Erastótenes es una lista con todos los números primos dentro de un rango establecido que no sean mayores a la raíz del número máximo) ($\text{primo}^2 < \text{máximo}$), con esta criba, la función únicamente tiene que revisar si el número es divisible por alguno de los números pertenecientes a la criba (número % criba[i]), en caso de serlo, realiza la división y prueba a dividir el resultado por otro número perteneciente a la criba, sino, entonces prueba con el siguiente número en la criba (número % criba[i+1]) hasta completar los factores.

5. Resultados de las pruebas

5.1. Gráfica

Esta gráfica muestra en segundos el promedio de tiempo obtenido por cada modo tras ejecutar los 10 casos de prueba. Cada caso utilizó 100 números, variando desde 1000 hasta 10 x10^{15}.



5.2. Tabla con resultados

Método:	Resultado:	Promedio (s):	Máximo alcanzado:
Modo 1	CRASH	1.124	5 x10^{3}
Modo 2	COMPLETADO	5.918	10 x10^{10}
Modo 3	CRASH	0.495	10 x10^{7}
Modo 4	COMPLETADO	2.439	10 x10^{15}
Modo 5	COMPLETADO	2.12	10 x10^{15}

5.3. Hardware

A continuación se especifica el hardware con el que se ejecutaron las pruebas para todos los modos, el programa fue desarrollado con el IDE Pycharm y se corrieron las pruebas desde la consola CMD.

- Procesador i-3 9100f de Intel
- 16 Gigabytes de RAM a 2666 Ghz
- Sistema operativo Windows 10 en 64 bits de Microsoft

6. Conclusiones

6.1. Modo 1

Este modo fue capaz de factorizar números menores a 1000 sin problema, de hecho, obtiene la factorización de números pequeños con facilidad, aunque, muestra debiliad cuando se tratan números mayores a 5000, ya que termina forzosamente cuando se ingresa un número primo mayor a esta cantidad.

6.2. Modo 2

A diferencia de su antecesor, fue capaz de factorizar prácticamente todas las pruebas, aunque presentó un fuerte incremento de tiempo con números cercanos o mayores a 10 x10^8, fue bastante estable a lo largo de la mayoría de pruebas gracias a que no le afecta el límite de recursión, pero definitivamente no es el más eficiente de todos los métodos.

6.3. Modo 3

En este caso, logra mostrar buenos resultados a pesar de no ser una función tan compleja, aunque su principal debilidad es el uso de recursión, ya que esta presenta un límite establecido dentro del IDE y será un impedimento en caso de querer factorizar números mayores a 10^7.

6.4. Modo 4

Este modo fue el segundo con mejores resultados, al no tener problemas con el límite de recursión puede aprovechar al máximo su fórmula y continuar sin problema hasta los exponentes más grandes, de hecho, al no ser muy complejo, podría considerarse como el más equilibrado en la relación eficiencia-dificultad, ya que no requiere la programación de una criba.

6.5. Modo 5

Finalmente, con el modo 5 se puede notar una mejoría significativa, principalmente cuando se trabaja con números grandes. Dentro de su composición, la criba es lo que más tiempo consume, aunque facilita tanto el procedimiento de la función que aún así logra reducir los tiempos en comparación a los demás.

7. Bibliografía

[1] I. Challenger-Pérez, Y. Díaz-Ricardo and R.A. Becerra-García, "El lenguaje de programación Python", Ciencias Holguín, vol. 20, no. 2, pp. 1-13.

[2] Ramirez, E.(2017). Fundamentos de Programación. Sin editorial