Pontifica Universidad Católica del Perú

Trabajo Académico del curso Arquitectura de Computadoras

Tiempo de ejecución y optimización

Alumno

Alejandro Macedo Pereira

Profesor

Jorge Benavides Aspiazu

24 de noviembre de 2017

1. Introducción

Python es un lenguaje de programación de alto nivel muy valorado y usado en la actualidad por la sencillez de su sintaxis y lo poderosa y flexible que puede ser para la creación de programas en cualquier ámbito de la humanidad. No obstante, al ser un lenguaje interpretado no llega a ser tan rápido que un programa implementado con un lenguaje compilado como C o C++. Sin embargo, el factor que más afecta al tiempo de ejecución es la lógica y el algoritmo usado para resolver el problema ya que sin una implementación y uso de los recursos eficiente, el programa será lento sin importar el lenguaje utilizado. Por esta razón se presentan dos algoritmos usados para la resolución de un problema propuesto, donde se explicara la lógica seguida en cada caso y su correspondiente tiempo de ejecución en Python y C++ para realizar la comparación entre ambos lenguajes. Adicionalmente se realizará un proceso de optimización al código escrito en Python usando el módulo para observar la mejora que proporciona.

2. Primer algoritmo

El primer algoritmo usado para resolver el problema fue tratar de mantener la mínima distancia entre los 3 números, lo cual se consigue avanzando al término siguiente del menor de los 3.

Triangulares	Pentagonales	Hexagonales
3	5	6
6	5	6
10	**5**	6
10	12	**6**
	.	
	.	
	.	

Cuadro 1: Primer algoritmo seleccionando el menor de los 3

Como se sabe que estos 3 números en algún momento serán iguales, este algoritmo terminará cuando se cumpla esta condición. Adicionalmete se ha desarrollado el mismo algoritmo pero en C++ con el motivo de servir de comparación entre estos dos lenguajes (*ambos códigos se encuentran en la carpeta PrimeraForma*. Los resultados de ambos códigos se presentan a continuación:

Cuadro 2: Tiempos de ejecución para ambos lenguajes[1]

	Python	C++
Tiempo	90ms - 100ms	1ms - 3ms

Como se observa, el tiempo que demora Python en ejecutar el código no es muy alto, lo que nos indica que el algoritmo usado es eficiente ya que este tiempo está en el orden de los milisegundos; sin embargo no se compara a C++ ya que toma aproximadamente $\frac{1}{10}$ del tiempo.

3. Segundo algoritmo

Luego de investigar en Internet, ví que este problema había sido planteado en la página de Project Euler, en la cual se presentan desafíos de matemática y programación [2], siendo este el problema número 45. De la misma forma también encontré una solución a este desafío hecha por Kristian Edlund en su página Mathblog [1] donde presenta una solución aprovechando la propiedad de que el conjunto de los números hexagonales es un subconjunto de los números triangulares, lo cual se prueba a continuación.

$$\begin{gathered} T(n) = n * (n+1)/2 \\ reemplazando\ n = 2m-1 \\ T(2m-1) = (2m-1) * ((2m-1)+1)/2 \\ T(2m-1) = (2m-1) * 2m/2 \\ T(2m-1) = m * (2m-1) = H(m) \end{gathered} \tag{1}$$

Esto facilita mucho el cálculo del número pedido ya que solo se necesitaría saber si un número hexagonal es al mismo tiempo pentagonal. Esto se consigue con la inversa de la fórmula de los números pentagonales; si n es un número entero, entonces también es pentagonal.

$$n = \frac{\sqrt{24x+1}+1}{6} \tag{2}$$

Esto reduce considerablemente la complejidad de algoritmo ya que no se necesita hacer una verificación, además de que se avanza considerablemente más rápido que el algoritmo anterior ya que la tasa de crecimiento de los números hexagonales es mucho mayor que de los otros dos. De la misma forma que el algoritmo anterior, se implementó este algoritmo en C++ a manera de comparar ambos lenguajes en cuando a tiempo de ejecución.

Cuadro 3: Tiempos de ejecución para ambos lenguajes[1]

	Python	C++
Tiempo	10ms - 15ms	0.6ms - 0.8ms

Se observa que el tiempo ha disminuído mucho en comparación al algoritmo anterior, casi alcanzando el tiempo que demoraba C++ con el algoritmo anterior, claro que con este algoritmo el código en C++ se ejecuta aproximadamente $\frac{1}{10}$ del tiempo que toma Python.

4. Optimización al segundo algoritmo

Al ser Python un lenguaje interpretado el tiempo que tarda el programa en ejecutarse es mayor ya que cada línea tiene que ser enviada al intérprete y luego convertida a código maquina para su posterior ejecución. Por lo tanto, si fuera posible compilar parte de las funciones usadas en el programa, el tiempo de ejecución mejoraría ya que son muchas menos líneas que tienen que enviarse para su interpretación y ejecución. Para este fin es que se usa el módulo Cython, el cual permite compilar y luego importar estas porciones de código para que sean llamadas en otros programas. Este módulo fue usado para compilar las dos funciones matemáticas auxiliares en el segundo algoritmo y los resultados de esta

aceleración serán presentadas a continuación (*el código utilizado se encuentra en la carpeta OptimizacionSegundaForma*).

Cuadro 4: Tiempos de ejecución sin y con optimización [1]

	Python sin optimización	Python con optimización
Tiempo	10ms - 15ms	2ms - 3ms

5. Conclusiones

Con los resultados obtenidos se observa la importancia de el desarrollo de algoritmos eficientes para mejorar los tiempos de ejecución de los programas escritos. Esto se vuelve aún más importante en los problemas donde la cantidad de datos se vuelve mucho más grande, como aplicaciones de Big Data, porque sino podrían tardar años en ejecutarse si no se implementa eficientemente. También se observa la ayuda que pueden bridar módulos como Cython para agilizar la ejecución de programas escritos en lenguajes interpretados como Python, aumentado su flexibilidad y ámbitos donde pueda ser usado. Finalmente, aunque hubiera querido medir el consumo de memoria de los programas escritos en Python, los módulos para este fin no funcionaban correctamente ya que entregaban información sin sentido por lo que no se pudo hacer el respectivo análisis.

Referencias

[1] EDLUND, Kristian
2011, *Project Euler 45: After 40755, what is the next triangle number that is also pentagonal and hexagonal?*. Consulta: 22 de noviembre del 2017.
http://www.mathblog.dk/project-euler-45-next-triangle-pentagonal-hexagonal-number/

[2] PROJECT EULER
Project Euler. Consulta: 23 de noviembre del 2017.
https://projecteuler.net/

[1] Ambos programas ejecutados en una computadora con procesador i7-4710HQ y 16 GB de RAM