Grado en Ingeniería Informática

Curso 2020/2021

Teoría Avanzada de la Computación

Test de Primalidad - *AKS*

Hito 1

Autores:

Iván Miguelez García	*100383387*
Alba Reinders Sánchez	*100383444*
Alejandro Valverde Mahou	*100383383*

Índice

1. Hito 1: Heurísticas iniciales

Para esta primera parte de la práctica se pide realizar un estudio analítico y empírico de las dos primeras heurísticas del algoritmo AKS, utilizado para determinar la primalidad de un número natural.

Además, se ha replicado el código en el lenguaje de programación *Python* de manera alternativa, para facilitar la comprensión y análisis del código.

1.1. Heurística 1: comprobación potencia perfecta

Comprobar si un número es una potencia perfecta es comprobar lo siguiente:

$$a^b = n \mid a \in \mathbb{N},\ b > 1$$

Si esta propiedad se cumple, se dice que n no es un número primo.

1.1.1. Estudio analítico

Se pretende determinar la complejidad temporal de los pasos del algoritmo en los que se lleva a cabo esta primera tarea. A continuación, se realiza el estudio analítico para averiguar *T(n)* y *O(n)*.

Para ello se tiene que analizar la estructura del código. Se puede ver que está compuesto por dos bucles *do while*. El bucle exterior itera sobre a y el bucle interior itera sobre b.

Para el análisis del bucle exterior, es necesario determinar el valor máximo de a. Se puede afirmar que, dado que el valor mínimo de b es 2, el valor máximo de a será $\sqrt{n}$, porque:

$$a^2 = n \ \Rightarrow\ a = \sqrt{n}$$

El bucle interior requiere un desarrollo un poco mayor. Para conseguir el valor máximo de b es necesario despejarlo en la ecuación.

$$a^b = n \ \Rightarrow\ \log_a(a^b) = \log_a(n) \ \Rightarrow\ b * \log_a(a) = \log_a(n)$$

$$b = \log_a(n)$$

Por tanto, el número de ciclos del bucle interior depende tanto de n como de a.

Si se unen ambas complejidades, se puede ver que para esta primera heurística, la complejidad temporal es:

$$T(n) = \sum_{a=2}^{\sqrt{n}} \log_a(n)$$

$$O(n) = \sqrt{n} * \log(n)$$

1.1.2. Estudio empírico

Como se puede ver en la Figura 1, los tiempos obtenidos no coinciden con la complejidad esperada. Esto puede deberse a diversos factores.

- Puede ser que no se hayan probado con números suficientemente grandes como para apreciar la curva esperada.
- Puede que el resultado esperado no esté escalado correctamente, y por tanto los valores no coincidan.
- Debido a que se ha ejecutado un código en *Java* en una máquina *Windows*, puede que los tiempos no estén bien medidos y tengan mucho ruido,

Para próximos hitos se volverá a probar, pero con una cantidad de números mayor.

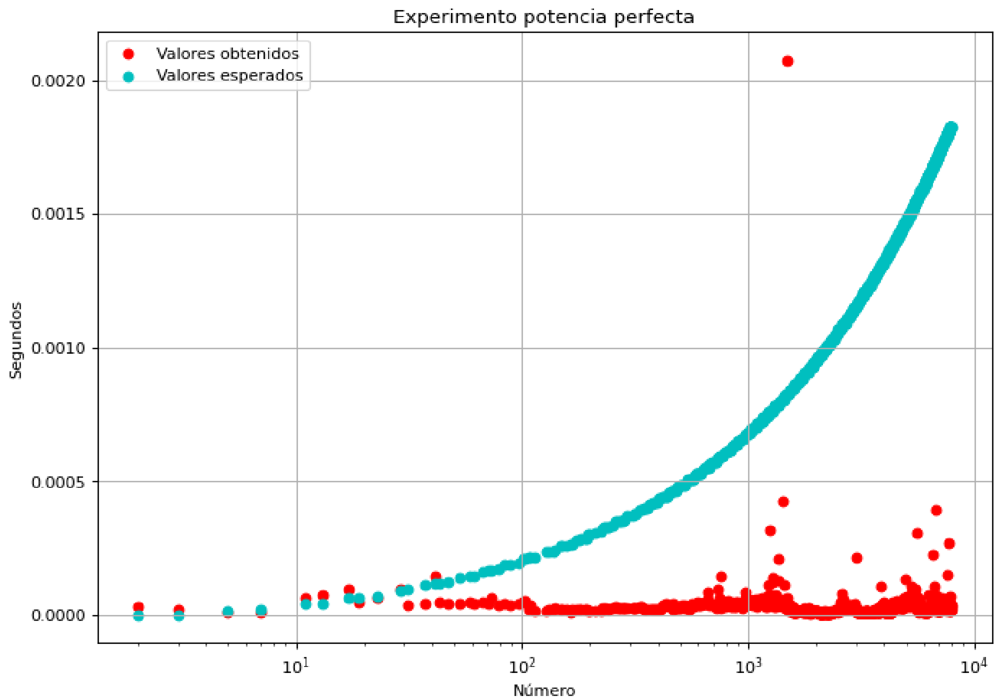

Figura 1: Gráfica de tiempo de la Heurística 1

1.2. Heurística 2: cálculo de *r* y *mcd*

Se parte de la siguiente premisa:

$$\exists a \leq r \mid 1 < mcd(n, a) < n$$

Si se cumple, se dice que n es un número compuesto.

1.2.1. Estudio analítico

Para determinar la complejidad temporal de esta segunda tarea se tiene que dividir el estudio en dos partes. En primer lugar, se debe calcular r y después calcular el mcd.

Cálculo de r

Analizando la estructura del código se ve que está compuesto por un bucle *do while* que itera sobre r y que dentro se llama a la función *multiplicativeOrder()*. Esta función tiene a su vez un bucle *do while* que itera sobre k.

Se tiene que r es el mínimo r tal que:

$$O_r(n) > log_2^2(n)$$

Donde $O_r(n)$ es el orden de n módulo r y representa el menor k tal que:

$$O_r(n) = k \;\Leftrightarrow\; n^k \equiv 1 \; mod \; r$$

Se sabe cuál es el máximo r por el lema 4.3 de *Primes is in P*[1]:

$$r \leq \lceil \log^5(n) \rceil$$

Por lo tanto se concluye que la complejidad temporal del cálculo de r es la unión de las complejidades de ambos bucles:

$$O(n) = log^5(n) * log^2(n) = log^7(n)$$

Cálculo del mcd

Por último, para la complejidad de calcular el máximo común divisor de dos número a y b, se tiene en cuenta el peor de los casos: cuando a y b son números consecutivos en la sucesión de *Fibonacci*.

En este caso, el número de iteraciones del bucle es el índice del término de la sucesión, el cual se saca con la fórmula de E.Lucas:

$$f_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}$$

cuya complejidad es $log(n)$.

Por tanto, la complejidad total del mcd:

$$O(n) = log(n) * log^5(n) = \log^6(n)$$

y la fórmula de la complejidad total de la heurística 2 es de:

$$O(n) = \log^7(n) + \log^6(n)$$

1.2.2. Estudio empírico

En este caso, tal como se puede ver en la Figura 2, la diferencia entre el tiempo propuesto en el estudio analítico y el empírico es mucho mayor. Esto puede indicar que los cálculos del estudio analítico no sean correctos, o puede deberse a las mismas causas que se han descrito en el anterior estudio empírico 1.1.2.

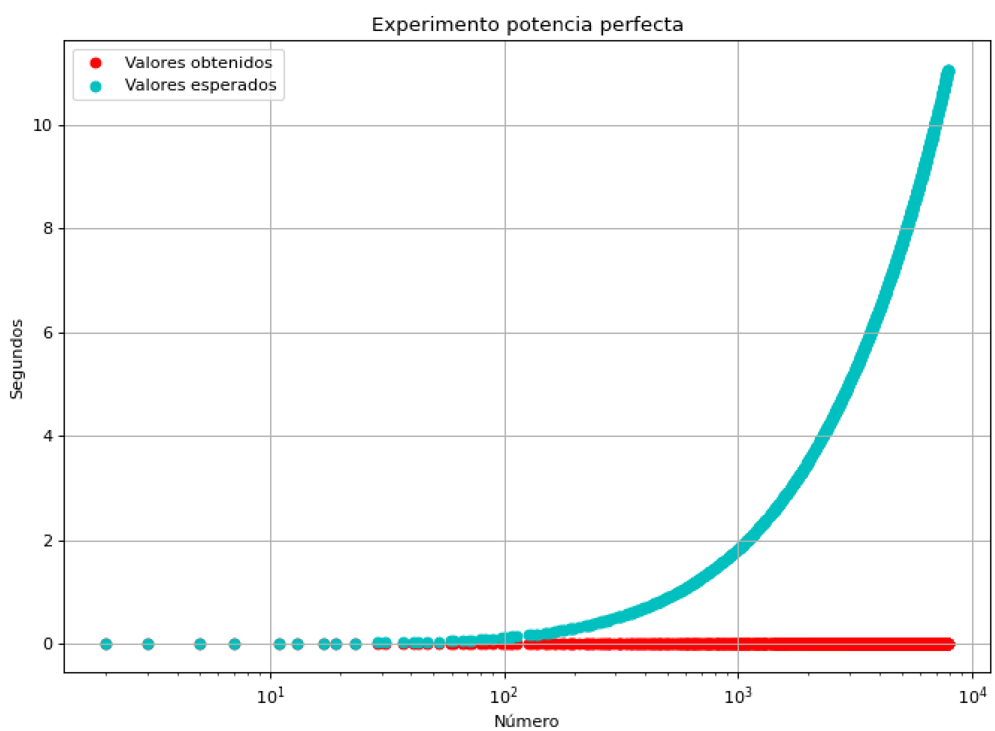

Figura 2: Gráfica de tiempo de la Heurística 2

Referencias

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Ann. of Math*, 2:781–793, 2002.