LA CRIBA REFINADA DE ERATÓSTENES

RESUMEN
En el presente artículo se describe y se comenta el algoritmo conocido como Criba de Eratóstenes para la obtención de todos los números primos menores que un número dado y se presenta una variante del mismo que permite además la descomposición en factores primos de cualquier entero positivo. Finalmente se demuestra que bajo ciertas condiciones tal descomposición puede efectuarse en un tiempo polinomial dependiendo de la cantidad de cifras decimales del entero.

PALABRAS CLAVES: Algoritmo, criba de Eratóstenes, números primos, descomposición en factores primos, tiempo polinomial.

LUIS ALEJANDRO GÓMEZ
Ingeniero de Sistemas
Profesor Universidad Autónoma de Manizales
zemogal@hotmail.com

ABSTRACT
In this article the algorithm known as Sieve of Erathostenes for obtaining all the prime numbers smaller than a given number is described and commented and a variation of it that allows the prime factorization of any positive integer is also introduced. Finally it is shown that under certain conditions this factorization may be carried out in polynomial time depending on the amount of decimal numerals of the integer.

KEYWORDS: *Algorithm, Sieve of Erathostenes, prime numbers, prime factorization, polynomial time.*

1. INTRODUCCIÓN

Para quienes no saben o no recuerdan qué es la Criba de Eratóstenes, se trata de un algoritmo inventado en el siglo tercero antes de Cristo para determinar cuáles de los primeros enteros positivos hasta un número dado, son primos. Su invención se atribuye a Eratóstenes, el mismo matemático, poeta, filósofo, astrónomo y geógrafo de Cirene que estimó ingeniosamente la longitud de la circunferencia terrestre con una precisión asombrosa, más de mil años antes de que en la Europa medieval se empezara a sospechar que la Tierra no es plana.

El nombre de Criba se debe sin duda a que el algoritmo actúa como una especie de filtro que en varias pasadas permite extraer todos los números primos que se hallan mezclados con el 1 y los números compuestos, entre los primeros enteros positivos. Y para conseguir esto se procede del modo siguiente:

Haga una lista de todos los naturales desde 1 hasta un número *n* dado:

1 2 3 4 5 6 7 8 9 10 11 12
13 14 15 16 17 18 19 20 21 ... *n*

Marque el 2 como primer primo y tache de ahí en adelante todos sus múltiplos:

1 [2] 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~
13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ... *n*

De los no tachados que siguen a 2 , el primero es el siguiente primo. Marque el 3 y tache todos los múltiplos de éste que no haya tachado en el paso anterior:

1 [2] [3] ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~
13 ~~4~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ... *n*

Continúe de esta manera hasta haber marcado un número mayor que $\sqrt{n}$ y entonces marque todos los números mayores que 1 que hasta ese momento hayan permanecido sin ser tachados:

1 [2] [3] ~~4~~ [5] ~~6~~ [7] ~~8~~ ~~9~~ ~~10~~ [11] ~~12~~
[13] ~~14~~ ~~15~~ ~~16~~ [17] ~~18~~ [19] ~~20~~ ~~21~~ ... *n*

Los números marcados son todos los primos entre 1 y *n*.

Dos cosas merecen principalmente una justificación en este procedimiento:

La primera es la afirmación implícita de que tras marcar un primo y tachar todos sus múltiplos, el siguiente número que queda sin tachar es el siguiente primo. Esto se debe a que cuando se hayan marcado digamos los primos 2, 3, ..., P_k y se hayan tachado todos sus múltiplos, los números compuestos que quedan sin tachar contienen exclusivamente factores mayores que P_k, y el menor de estos números compuestos es $(P_{k+1})^2$. Todos los demás números no tachados mayores que P_k y menores que $(P_{k+1})^2$ son necesariamente primos pues no son múltiplos de ningún primo menor que P_{k+1} ni son el producto de factores primos mayores que P_k. De modo que al marcar el

Fecha de Recepción: 27 Enero de 2006
Fecha de Aceptación: 01 Junio de 2006

2 como primo y tachar todos los pares, no sólo se sabe que 3 es el siguiente primo sino que también lo son todos los impares entre 3 y 9. Así mismo al marcar el 3 y tachar sus múltiplos, no sólo se sabe que 5 es el siguiente primo sino que también lo son todos los no tachados entre 5 y 25 y así sucesivamente.

La otra cosa que merece justificación es la afirmación, también implícita, de que después de marcar como primo un número mayor que $\sqrt{n}$ no es ya necesario tachar más números y los que quedan sin tachar (con excepción del 1) son primos. Esto es así porque si $P_k > \sqrt{n}$ para algún k, entonces $(P_k)^2 > n$, de modo que todos los números compuestos entre 2 y n inclusive, contienen factores primos menores que P_k y estarán ya tachados cuando éste se marque.

2. UNA VARIANTE

A continuación expondré una forma de utilizar el esquema de la criba de Eratóstenes no sólo para encontrar todos los primos entre 1 y n sino también para descomponer en factores primos cualquier número en ese mismo rango:

Haga una lista de todos los naturales desde 1 hasta un número n dado:

1 2 3 4 5 6 7 8 9 10 11 12

13 14 15 16 17 18 19 20 21 ... n

Ahora los primos se marcarán escribiendo un 1 debajo de ellos. Marque el 2 como primer primo y escriba un 2 debajo de cada uno de sus múltiplos.

1	2	3	4	5	6	7	8	9	10	11	12
	1		2		2		2		2		2

13	14	15	16	17	18	19	20	21	...	n
	2		2		2		2			

De los números mayores que 2 que tengan debajo un espacio en blanco, el primero es el siguiente primo. Marque el 3 y escriba un 3 debajo de cada uno de sus múltiplos que tenga debajo un espacio en blanco:

1	2	3	4	5	6	7	8	9	10	11	12
	1	1	2		2		2	3	2		2

13	14	15	16	17	18	19	20	21	...	n
	2	3	2		2		2	3		

Continúe de esta manera hasta haber marcado un número mayor que $\sqrt{n}$ y entonces marque todos los números mayores que 1 que tengan debajo un espacio en blanco.

1	2	3	4	5	6	7	8	9	10	11	12
	1	1	2	1	2	1	2	3	2	1	2

13	14	15	16	17	18	19	20	21	...	n
1	2	3	2	1	2	1	2	3		

Al terminar de ejecutar esta versión refinada de la Criba de Eratóstenes, se tendrá un arreglo en el cual cada número primo tendrá debajo un 1 y cada compuesto su menor factor primo. Llamaré a este arreglo E.

3. UN ALGORITMO DE TIEMPO POLINOMIAL PARA LA DESCOMPOSICION EN FACTORES PRIMOS

Si llamamos F a la función que asigna a cada natural m en el rango de 2 a n, la cadena de símbolos que representa su descomposición en factores primos y llamamos E_m al número que se encuentra debajo de m en el arreglo E, entonces tendremos el siguiente

ALGORITMO PARA LA DESCOMPOSICIÓN EN FACTORES PRIMOS

I. Si $E_m = 1$ entonces $F(m) = m$

II. Si no $F(m) = E_m \mathbf{x} F(m \div E_m)$

En el paso **II.** el signo **x** se ha escrito en negrilla para dar a entender que no se requiere efectuar aquí una multiplicación, sino simplemente escribir dicho signo.

Un ejemplo ayudará a comprender mejor el funcionamiento de este algoritmo:

Sea $m = 20$. Puesto que $E_{20} = 2$ y $20 \div 2 = 10$ tenemos:

$$F(20) = 2 \mathbf{x} F(10)$$

Pero $E_{10} = 2$ y $10 \div 2 = 5$, con lo cual:

$$F(20) = 2 \mathbf{x} 2 \mathbf{x} F(5)$$

Finalmente $F(5) = 5$, pues $E_5 = 1$, de donde:

$$F(20) = 2 \mathbf{x} 2 \mathbf{x} 5$$

La demostración de que la función F produce en efecto la descomposición en factores primos de cualquier natural m en el rango de 2 a n puede hacerse por inducción sobre la cantidad de factores de m:

I. Paso básico:

Si m posee sólo un factor primo, (es decir si m es primo) entonces $E_m = 1$ y $F(m) = m$.

II. Paso inductivo:

Supóngase que F produce la descomposición en factores primos de cualquier natural en el rango de 2 a n que posea k factores (hipótesis de inducción).

Si $m = p_{i1} p_{i2} p_{i3} \ldots p_{ik+1}$, es decir si m posee k+1 factores primos siendo p_{i1} el menor de ellos, entonces $E_m = p_{i1}$ y por lo tanto $F(m) = p_{i1} \mathbf{x} F(p_{i2} p_{i3} \ldots p_{ik+1})$. Pero de acuerdo con la hipótesis de inducción $F(p_{i2} p_{i3} \ldots p_{ik+1}) = p_{i2} \mathbf{x} p_{i3} \mathbf{x} \ldots \mathbf{x} p_{ik+1}$, con lo cual $F(m) = p_{i1} \mathbf{x} p_{i2} \mathbf{x} p_{i3} \mathbf{x} \ldots \mathbf{x} p_{ik+1}$. □∀

Nótese que la cantidad de divisiones necesarias para descomponer un número *m* por medio de este algoritmo es exactamente igual a la cantidad de factores primos de *m* menos uno. Enseguida daré un argumento para probar que el tiempo de ejecución del algoritmo está acotado por una función polinomial que depende del número de cifras decimales de *m*:

Sea *d* la cantidad de cifras decimales del número *m*. Entonces $m < 10^d$ y $\log_2 m < (\log_2 10)d$. Pero dado que $\log_2 10 < 4$, puede asegurarse que $\log_2 m < 4d$, y por consiguiente la parte entera de $\log_2 m$ es también menor que $4d$. Si utilizamos la notación $[x]$ para referirnos a la parte entera de *x*, lo anterior puede resumirse así:

$$[\log_2 m] < 4d \qquad (1)$$

Por otra parte, si hacemos $j = [\log_2 m]$ tendremos, de acuerdo con la definición de parte entera, que $j \leq [\log_2 m] < j+1$, lo cual equivale a decir que $2^j \leq m < 2^{j+1}$. Esto implica que *m* no posee más de *j* factores primos, pues si fuese el producto de digamos $j+h$ factores con $h \geq 1$, se tendría que $m \geq 2^{j+h}$, ya que cada factor primo es mayor o igual a 2. Y dado que $2^{j+h} = 2^{j+1+h-1} = 2^{j+1} \cdot 2^{h-1} \geq 2^{j+1}$, resultaría que $m \geq 2^{j+1}$, y esto contradice la desigualdad: $m < 2^{j+1}$. De acuerdo con todo lo anterior, si llamamos *P(m)* a la cantidad de factores primos de *m*, puede asegurarse que:

$$P(m) \leq [\log_2 m] \qquad (2)$$

Combinando esta desigualdad con la (1) se obtiene:

$$P(m) < 4d \qquad (3)$$

Restando 1 a los dos miembros:

$$P(m) - 1 < 4d - 1 \qquad (4)$$

La expresión del lado izquierdo de esta nueva desigualdad representa la cantidad de divisiones necesarias para descomponer en factores primos el número *m* aplicando el algoritmo ya expuesto. Dado que cada una de estas divisiones puede efectuarse en un tiempo proporcional a la cantidad de cifras del dividendo, puede afirmarse que la división que requiere más tiempo es la primera, de modo que si ésta requiere un tiempo *Td* para alguna constante *T*, podemos hallar una cota superior para el tiempo requerido por el total de las divisiones multiplicando los dos miembros de la desigualdad anterior por *Td*:

$$Td(P(m) - 1) < 4Td^2 - Td \qquad (5)$$

Si observamos que para la factorización de un número *m* es necesario consultar el arreglo E tantas veces como factores primos posee dicho número, y si asumimos que el tiempo necesario para cada una de estas consultas está acotado superiormente por una constante *C*, podemos hallar a partir de la desigualdad (3) una cota superior para el tiempo requerido por el total de las consultas multiplicando ambos miembros de dicha desigualdad por C:

$$CP(m) < 4Cd \qquad (6)$$

Finalmente al sumar miembro a miembro las desigualdades (5) y (6) se tiene una cota superior para el tiempo total necesario para la ejecución del algoritmo:

$$Td(P(m) - 1) + CP(m) < 4Td^2 + (4C-T)d \qquad (7)$$

El miembro izquierdo de esta desigualdad es una estimación algo exagerada del tiempo necesario para la factorización del número m, y la misma desigualdad muestra que tal estimación exagerada está acotada superiormente por un polinomio de segundo grado en la variable *d,* que es precisamente lo que se quería probar.

4. CONCLUSION

Todo lo anterior parece contradecir el hecho de que en los largos años de historia del pensamiento matemático, no se ha logrado desarrollar un algoritmo eficiente para la descomposición en factores primos, hecho que ha cobrado en la actualidad una importancia capital debido a que el método más utilizado actualmente para garantizar alguna seguridad en la transmisión de información a través de Internet (el sistema RSA) se basa en la dificultad de factorizar, en un tiempo razonable, números cuya expresión en binario posee al rededor de 128 dígitos o bits. Debido a esto, el descubrimiento de un algoritmo eficiente para la factorización sería una verdadera amenaza para la seguridad de nuestra comunicación.

Lo que yo he probado en estas páginas de manera tan minuciosa es de hecho casi una falacia, pues a pesar de que el algoritmo expuesto es muy eficiente, antes de poderlo ejecutar es necesario contar con el arreglo E que se obtiene por medio de la Criba Refinada de Eratóstenes, y quisiera que mis distinguidos lectores se tomaran el tiempo necesario para determinar cuántos milisegundos y cuántos bytes se requieren para construír un arreglo como ese para un número de 128 bits. Puedo asegurarles que yo he pensado en el asunto y he hallado algunas conclusiones fascinantes al respecto, pero como diría el gran Pierre de Fermat, el resto de esta página es demasiado estrecho para contenerlas. Quizás en un próximo artículo....

5. BIBLIOGRAFÍA

[1] HOPCROFT, MOTWANI, ULLMAN. Introducción a la Teoría de Autómatas, Lenguajes y Computación, Segunda edición, 584 páginas, Addison Wesley, Madrid, 2000.

[2] VINOGRADOV, Iván. Fundamentos de la Teoría de los Números, Primera edición, 210 páginas, Editorial MIR, Moscú, 1977.