3C TIC ***(Edición 24)*** *Vol.7 – Nº1*
Marzo – junio'18, 13 - 20
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529
DOI: http://dx.doi.org/10.17993/3ctic.2018.59.13-20

Recepción: 06 de noviembre de 2017
Aceptación: 22 de diciembre de 2017
Publicación: 29 de marzo de 2018

ALGORITMO DE KARATSUBA EN OPERACIONES DE EXPONENCIACIÓN

KARATSUBA ALGORITHM OPERATIONS EXPONENTIATION

Jesús Ayuso Pérez[1]

1. Compositor musical y desarrollador. Licenciado en Ingeniería Informática por la Universidad Carlos III de Madrid (UC3M). E-mail: ayusoperez@terra.com

Citación sugerida:

Ayuso Pérez, J. (2018). Algoritmo de Karatsuba en operaciones de exponenciación. *3C TIC: Cuadernos de desarrollo aplicados a las TIC,* 7(1), 13-20. DOI: <http://dx.doi.org/10.17993/3ctic.2018.59.13-20/>.

3C TIC ***(Edición 24)*** *Vol.7 – Nº1*
Marzo – junio'18, 13 - 20
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529
DOI: http://dx.doi.org/10.17993/3ctic.2018.59.13-20

RESUMEN

El algoritmo dado por Anatoly Alexeevitch Karatsuba en 1960 (Karatsuba, 1962) para la multiplicación no es únicamente aplicable a dicha operación, se puede aplicar a cualquier operación algebraica que se construya sobre una operación que cumpla la propiedad distributiva con respecto a otra que componga a la misma (Ayuso, 2013-2017). De ahí que en el presente documento propongamos un algoritmo de exponenciación entre enteros basado en dicho concepto.

ABSTRACT

The algorithm given by Anatoly Alexeevitch Karatsuba in 1960 (Karatsuba, 1962) for multiplication does not apply only to this operation, it can be applied to any algebraic operation that is built on an operation that fulfills the distributive property with respect to something else that compose one the same (Ayuso, 2013-2017). Hence, in this document we propose an algorithm of exponentiation between integers based on that concept.

PALABRAS CLAVE

Karatsuba, Algoritmo, Exponenciación, Binomio Newton, Potencias.

KEYWORDS

Karatsuba, Algorithm, Exponentiation, Binomial Newton, Powers.

3C TIC *(Edición 24)* Vol.7 – Nº1
Marzo – junio'18, 13 - 20
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529
DOI: http://dx.doi.org/10.17993/3ctic.2018.59.13-20

1. INTRODUCCIÓN

Partiendo de que la multiplicación entre enteros de longitud *n*, por ejemplo *u* por *v*, puede descomponerse (Karatsuba, 1962) siguiendo el criterio:

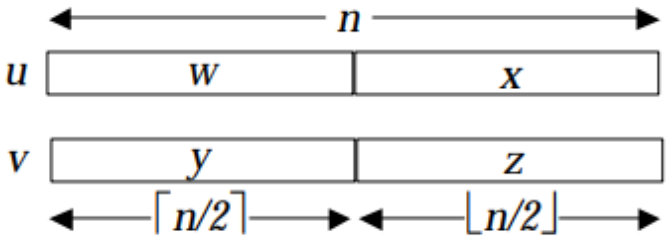

Ilustración 1. Ilustración de la descomposición de Karatsuba.
Fuente: elaboración propia.

Tirando de bibliografía, podemos resumir que el resultado de dicha operación, *u * v*, puede expresarse matemáticamente como:

Fórmula 1. Algoritmo de Karatsuba de multiplicación entre enteros.

$$u * v = (w\ y)\ 10^{n} + (w\ z + x\ y)\ 10^{n/2} + x\ z$$

Con la definición anterior, y teniendo en cuenta que la exponenciación de un entero *b* de longitud *n* elevado a otro entero *e* puede definirse como una sucesión de multiplicaciones (Booth, 1951), tal y como queda representado en la siguiente fórmula:

$$b^{e} = b * b * b * \underbrace{\ldots\ldots}_{\{e\ veces\}} * b * b$$

Fórmula 2. Algoritmo de exponenciación entre enteros.

Curiosamente, indiferentemente de que sea requisito o no, la operación será siempre entre enteros de la misma longitud, ya que se trata de productos de un número por él mismo. Además, la simplificación de ahorrar 1 multiplicación de las 4 documentadas (Karatsuba, 1962) en la descomposición, tal y como advirtió Karatsuba a raíz de:

$$w\ v + w\ z + x\ y + x\ z - w\ y - x\ z = w\ z + x\ y$$
$$w\ y + w\ z + x\ y + x\ z = (w + x)\ (y + z)$$

Fórmula 3. Técnica de Karatsuba de simplificación de multiplicaciones por sustracciones.

no tendría demasiado sentido en este caso, ya que encontraríamos que: *w * z = x * y*.

Por ello, se observa que es posible describir la potencia de un entero con el enfoque utilizado por Karatsuba, similar a lo mostrado en la **Ilustración 1**, de la siguiente manera:

3C TIC ***(Edición 24)*** *Vol.7 – Nº1*
Marzo – junio'18, 13 - 20
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529
DOI: http://dx.doi.org/10.17993/3ctic.2018.59.13-20

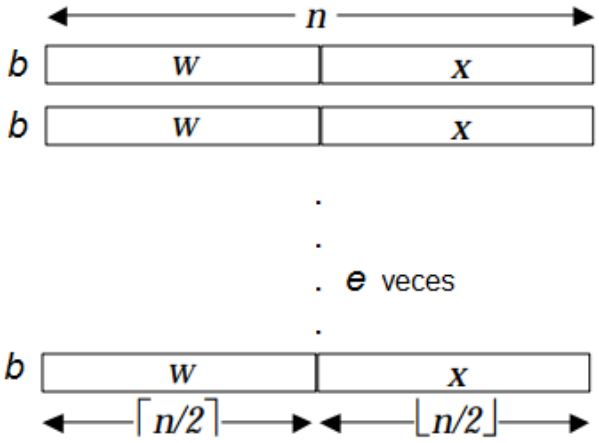

Ilustración 2. Ilustración de la exponenciación por Karatsuba.
Fuente: elaboración propia.

Con lo que se tiene que, razonando de la misma forma que en la operación de multiplicación, para la operación de exponenciación se llega a:

$$(w+x) * (w+x) * \overbrace{\ldots\ldots}^{\{e \text{ veces }\}} * (w+x) = (w+x)^e$$

Fórmula 4. Algoritmo de Karatsuba de exponenciación entre enteros.

Por otra parte, es sabido por el teorema del Binomio (Newton, 1669, 206 ff.) que el resultado de la expresión anterior es calculable sin más desarrollo directamente con la fórmula:

$$(w+x)^e = \binom{e}{0} w^e x^0 10^{e\,n/2} + \binom{e}{1} w^{e-1} x^1 10^{e-1\,n/2} + \binom{e}{2} w^{e-2} x^2 10^{e-2\,n/2} + \cdots + \binom{e}{e-1} w^1 x^{e-1} 10^{n/2} + \binom{e}{e} w^0 x^e 10^0$$

Ilustración 3. Fórmula del binomio de Newton aplicada a Karatsuba.
Fuente: elaboración propia.

Dicho esto, se entenderá la exponenciación de un entero como la potencia de un binomio teniendo en cuenta ciertas potencias por la base en la que se está trabajando, en nuestro caso: **base 10;** las cuales suelen significar simples desplazamientos. Entendidos estos últimos como una acción de trasladar dígitos, *d*, en determinada base, *B*, cambiando su peso significativo:

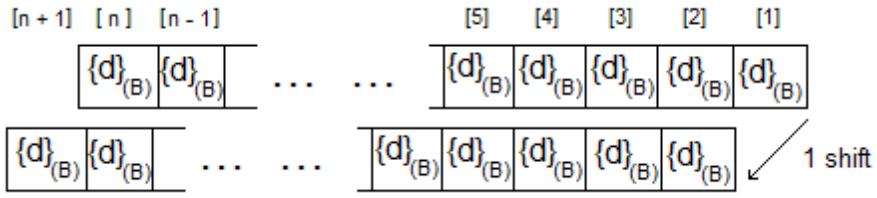

Ilustración 4. Técnica de desplazamiento de dígitos en determinada Base.
Fuente: elaboración propia.

3C TIC (Edición 24) Vol.7 – Nº1
Marzo – junio'18, 13 - 20
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529
DOI: http://dx.doi.org/10.17993/3ctic.2018.59.13-20

2. METODOLOGÍA

En este apartado trabajamos con nomenclatura en forma de pseudocódigo, abstrayéndonos de lenguajes de programación. Lo primero que haremos será dar una implementación recursiva para el cálculo de la parte central de la fórmula del binomio, viendo así más simple el algoritmo de Karatsuba adaptado a la exponenciación. Partimos de que tenemos una operación llamada *combinatorial(m ,n)* capaz de calcular el número combinatorio *m* sobre *n*, con ello definimos:

```
c = combinatorial(e, e - i);
y = expKaratsuba(w, i);
z = expKaratsuba(x, e - i);

if(i == 1)
      return (c * y * z) * 10^size; // desplazamiento base 10

return ((c * y * z) * 10^(i*size) )
                  + binomialNewton(w, x, e, size, i - 1);
```

Fórmula 5. Algoritmo recursivo (binomialNewton).

Con la anterior operación ya podemos describir el algoritmo de exponenciación mediante la técnica descrita por Karatsuba para la multiplicación. Tendríamos que el resultado de elevar un entero *b* de longitud *n* al entero *e* sería:

```
if(e == 0)
      return 1;
if(n == 1) // CASO BASE
      return b^e;

w = b / 10^(n / 2);
x = b % 10^(n / 2);

return expKaratsuba(w, e) * 10^(e*(n/2)) + // desplz. base 10
            binomialNewton(w, x, e, n/2, e - 1) +
            expKaratsuba(x, e);
```

Fórmula 6. Algoritmo recursivo de exponenciación por Karatsuba (expKaratsuba).

3C TIC ***(Edición 24)*** *Vol.7 – Nº1*
Marzo – junio'18, 13 - 20
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529
DOI: http://dx.doi.org/10.17993/3ctic.2018.59.13-20

La anterior implementación, aunque pueda aparentar elegancia, peca de ser extremadamente ineficiente debido al constante cómputo de cálculos repetidos. Para corregir dicho problema, es muy normal recurrir a una solución más basada en un paradigma de *Programación Dinámica* (Bellman, 1954) en lugar del enfoque usado por Karatsuba valiéndose de un *Divide y Vencerás* (Karatsuba, 1962).

Para ello, se puede evaluar el definir una serie de datos precalculados al algoritmo de exponenciación como tal; siendo el primero de estos datos una estructura con los resultados de la exponenciación de los elementos de los que consta la base en la que estemos trabajando: **base 10** en nuestro caso. Es decir, teniendo que *e* es la potencia a la que estamos elevando nuestro número entero, definimos para los diferentes valores de *e*:

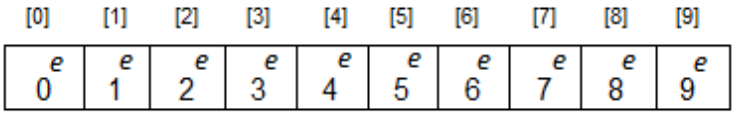

Ilustración 5. Estructura que contiene los resultados de las potencias del 0 al 9.
Fuente: elaboración propia.

Por otro lado, es necesario declarar otra estructura en la que se tienen precalculadas las *e*-ésimas filas del triángulo de Pascal (Pascal, 1654), la cual nos dará los distintos coeficientes para la exponenciación:

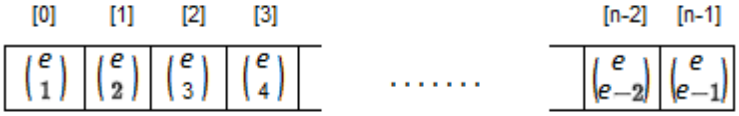

Ilustración 6. Estructura con la fila e+1 del triángulo de Pascal/Tartaglia.
Fuente: elaboración propia.

Con las estructuras de datos anteriores, se entiende que la implementación alternativa basa su eficiencia en haber calculado previamente el resultado de elevar los números 2, 3, 4, 5, 6, 7, 8, y 9 a las diferentes potencias de la operación de exponenciación que se está realizando, y además del cálculo de las filas de números combinatorios para dicha potencia. Con todo ello, el algoritmo que se describe en el presente artículo resultaría atractivo en entornos donde se requiriera realizar una gran cantidad de exponenciaciones para una misma potencia. En esos casos, el coste de los mencionados cómputos previos estaría sobradamente justificado, y ya podríamos realizar multitud de exponenciaciones contra un mismo exponente.

3C TIC *(Edición 24)* *Vol.7 – Nº1*
Marzo – junio'18, 13 - 20
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529
DOI: http://dx.doi.org/10.17993/3ctic.2018.59.13-20

3. CONCLUSIONES

Aplicar el algoritmo de Karatsuba a la operación de exponenciación nos provee de una solución elegante para el cómputo de la misma que demuestra que el concepto en que se apoyó el citado autor es más poderoso y portable de lo que aparenta a priori. Nos proporciona una técnica de resolución potente y un paradigma para abordar otro tipo de operaciones siempre que éstas cumplan como mínimo la propiedad distributiva sobre la operación que la compone.

En conclusión, el concepto propuesto por Karatsuba es extensible a distintas operaciones algebraicas, ofreciendo siempre la posibilidad de dividir el cómputo de la misma en cálculos más ligeros y reducir el tamaño de los operandos implicados. Además de resultar una solución mucho más elegante, derivando en soluciones bastante limpias desde un punto de vista algorítmico.

Como posible futura línea de investigación, se pueden tratar de aplicar los conceptos descritos a entornos donde haya cálculos pesados de operaciones de exponenciación donde la potencia implicada en los cómputos sea siempre la misma, o donde haya un conjunto concreto de potencias a las que sean elevados una gran cantidad de números distintos.

3C TIC *(Edición 24)* Vol.7 – Nº1
Marzo – junio'18, 13 - 20
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529
DOI: http://dx.doi.org/10.17993/3ctic.2018.59.13-20

4. REFERENCIAS BIBLIOGRÁFICAS

Ayuso, J. (2015). Booth algorithm operations addition and subtraction. *3C TIC,* 4(2), pp. 113-119.

Ayuso, J. (2015). Booth algorithm modular arithmetic operations of addition and subtraction. *3C TIC,* 4(3), pp. 222-229.

Ayuso, J. (2015). Booth algorithm modular arithmetic operations of multiplication. *3C TIC,* 4(4), pp. 255-221.

Ayuso, J. (2016). Booth algorithm in signed-digit representation. *3C TIC,* 5(3), pp. 33-43.

Ayuso, J. (2017). Booth algorithm hardware operations addition and subtraction. *3C TIC*, 6(3), pp. 1-9.

Booth, A.D. (1945). A method of calculating reciprocal spacings for X-ray reflections from a monoclinic crystal. *J. Sci. Instr,* 22, p. 74.

Burks, A., Goldstein, H. and Von Neumann, J. (1946). Logical Design of an Electronic Computing Instrument.

Booth, A.D. and Britten, K. H. V. (1947). General Considerations in the Design of an Electronic Computer.

Booth, A.D. (1951). A signed binary multiplication technique. *Q. J. Mech. and Appl. Math,* 4(2), pp.236-240.

W. Reitwiesner, G. (1960). Binary Arithmetic, 231-308.