# El Correcto y Completo Desarrollo de un Algoritmo.

por
M. en C. Eduardo René Rodríguez Ávila.
Sección de Estudios de Posgrado e Investigación
UPIICSA

A Patricia Prieto Corona,
*por permitirme compartir esta clase con ella.*

## Presentación.

*"En el Gran Templo de Benares, bajo el domo que marca el centro del mundo, descansa un plato de bronce en el que están fijas tres agujas de diamante, cada una de un cúbito de largo y tan delgadas como el cuerpo de una abeja. En una de estas agujas Dios colocó sesenta y cuatro discos de oro puro, el disco más largo descansa sobre el plato y sobre éste está uno más pequeño y así sucesivamente hasta la punta. Esta es la Torre de Brahma. Día y noche, incesantemente, los sacerdotes transfieren los discos de una aguja a otra, de acuerdo a fijas e inmutables leyes que requieren que el sacerdote en turno no mueva mas que un disco a la vez y que el disco se coloque en un aguja de tal forma que no haya un disco más pequeño bajo éste. Cuando los sesenta y cuatro discos hayan sido transferidos de la aguja donde Dios los colocó a una de las otras, los brahmanes se convertirán en polvo, y con un trueno el mundo se desvanecerá."*

Edouard Lucas, 1883.

Nadie sabe si la leyenda anterior es verdadera o no, pero ésta inspiró al francés Edouard Lucas a desarrollar el conocido juego "Las Torres de Hanoi". Esta historia será el vehículo que nos introducirá en uno de los más interesantes y apasionantes temas de la teoría de la computación: el desarrollo de algoritmos.

## Los algoritmos y su historia.

Los algoritmos no son sólo una curiosidad académica u objeto de estudio de una ciencia. Son la base de todo proceso metódico, formal o informal, y los encontramos por todas partes con múltiples nombres: receta, proceso, método, técnica, procedimiento, fórmula, rutina, conjuro. Un diccionario común [1] [2] [3] describirá como algoritmo al conjunto de símbolos y procedimientos usados en la realización de un cálculo. Descripciones similares se encuentran incluso en los diccionarios especializados [4] [5] El desarrollo de la informática y computación han hecho que el término adquiera cierto nivel de cotidianidad, sin que eso implique la pérdida de la rigurosidad requerida para dar a una serie de pasos el apelativo de algoritmo. Para muchos será una palabra más, producto de nuestra era tecnológica, pero su origen se remonta muy atrás en el tiempo.

Uno de los más grandes matemáticos árabes del siglo IX de nuestra era, Abu 'Abd Allah Muhhamad ibn Musa al-Khwarizmi (literalmente: Padre de Abdullah, Mohamed, hijo de Moisés,

nativo de Khwarizm —hoy Khiwa) con su obra "*Kitab al-jabr wa'l-muqabala*" (Reglas de Ecuaciones y Restauración), ayudó a difundir las matemáticas árabes por el mundo occidental a tal grado que del título de su obra se ha desprendido el término *álgebra* (*al-jabr*). Con el paso del tiempo y por defectos de pronunciación, su nombre se difundió simplemente como **Al-Juarismi** y de éste los términos *guarismo* y *algorismo* (usados para referirse a cualquier método de cómputo usando la notación arábiga de numeración). El término *algorismo* también fue corrompido en su pronunciación hasta derivar uno más difundido en latín como *algoritmus*, empleado desde el siglo XVII por los matemáticos para referirse a procedimientos de cálculo. Finalmente, la palabra que conocemos no apareció en un diccionario sino hasta la edición de 1957 del "*Webster´s New World Dictionary*".

Las matemáticas son, en cierto sentido, una colección de algoritmos. La aparición de las computadoras, propiciada por la necesidad y propósito de realizar cálculos y conteos, obligó a mucha gente a hablar de métodos y procedimientos para describir los programas que para ellas se elaboraban; programas que eran en sí la representación de un procedimiento matemático, de un *algoritmo*. La escritura de programas es entonces la elaboración o implementación de algoritmos, aunque claro no podemos afirmar que hacer matemáticas sea sólo hacer programas. Al final, programar bien es hacer matemáticas y, como todo buen matemático, todo buen programador debe contar con un buen método (formal o intuitivo) que le permita asegurarse de que sus programas son correctos. Aquí es donde empieza a verse la importancia de la definición y concepción del término ***algoritmo***.

Podemos ver que tanto las matemáticas como la computación se originan del acto de contar. Sin embargo, las primeras se encargan de descubrir las verdades acerca de las estructuras abstractas; la segunda, de estudiar el conjunto de acciones que obedecen a reglas formales estructuradas en el tiempo. A este respecto los trabajos de Alan M. Turing y Emil Post han sido decisivos y con profundas implicaciones en ambas disciplinas, mucho más allá del propósito del presente artículo para proceder a describirlas. Deberemos conformarnos por el momento únicamente con saber que sus modelos (la Máquina de Turing y la Máquina de Post) son las formalizaciones aceptadas de lo que debe considerarse como un ***procedimiento efectivo***, el cual devuelve una respuesta o indica la falta de ésta a un problema.

El definir a un algoritmo como una serie de pasos ordenados no nos dice más de lo que ya hemos hablado. No nos dice si debe o no haber una respuesta como resultado de su aplicación. No nos dice tampoco cuánto tiempo debe tomar llegar al resultado. Los modelos de Turing y Post contemplan estas dos últimas consideraciones: la declaración de contar o no con una respuesta y la implicación del tiempo invertido en el proceso, que es proporcional a las operaciones (pasos) a realizar con éste.

Un último punto igual de importante es apreciar que tanto el modelo de Post como el de Turing parten de un conjunto de reglas muy precisas del cual se construye el procedimiento a seguir. Así, y con base en la definición de estos modelos, ofrecemos la siguiente definición de lo que es un algoritmo:

*Un **algoritmo** es un conjunto finito de pasos definidos, estructurados en el tiempo y formulados con base en un conjunto finito de reglas no ambiguas, que proveen un procedimiento para dar la solución o indicar la falta de ésta a un problema en un tiempo determinado.*

## Las características de un algoritmo.

De la definición proporcionada y los cuestionamientos previos a ésta, podemos ver que un *algoritmo*, para ser catalogado como tal, debe exhibir ciertas propiedades:

- **Ser definido**.- Sin ambigüedad, cada paso del algoritmo debe indicar la acción a realizar sin criterios de interpretación.
- **Ser finito**.- Un número específico y numerable de pasos debe componer al algoritmo, el cual deberá finalizar al completarlos.
- **Tener cero o más entradas**.- Datos son proporcionados a un algoritmo como insumo (o estos son generados de alguna forma) para llevar a cabo las operaciones que comprende.
- **Tener una o más salidas**.- Debe siempre devolver un resultado; de nada sirve un algoritmo que hace algo y nunca sabemos que fue. El devolver un resultado no debe ser considerado como únicamente "verlos" en forma impresa o en pantalla, como ocurre con las computadoras. Existen muchos otros mecanismos susceptibles de programación que no cuentan con una salida de resultados de esta forma. Por *salida de resultados* debe entenderse todo medio o canal por el cual es posible apreciar los efectos de las acciones del algoritmo.
- **Efectividad**.- El tiempo y esfuerzo por cada paso realizado debe ser preciso, no usando nada más ni nada menos que aquello que se requiera para y en su ejecución.

## El desarrollo de algoritmos.

Aunque paradójicamente no exista un algoritmo para el desarrollo de algoritmos, es posible enumerar una serie de pasos que pueden considerarse en conjunto como una guía genérica en la correcta creación de un algoritmo. Estos pasos son:

1. Declaración del problema.
2. Desarrollo de un modelo.
3. Diseño del procedimiento de solución.
4. Validación.
5. Implementación.
6. Análisis y complejidad.
7. Pruebas.
8. Documentación.

## Un ejemplo

En lugar de describir áridamente cada uno de los pasos indicados, lo haremos mediante un simple pero muy interesante ejercicio. Un ejercicio en el que usaremos la historia con que dimos inicio al presente escrito.

**Declaración del problema.**

Por lo general, y salvo aquellos casos en que la descripción de un problema obedece a un formalismo, todo problema es planteado en prosa. Es de esta descripción que debemos identificar los distintos elementos involucrados, sus características, su conducta y relaciones. Diversas técnicas se han desarrollados al respecto: análisis de dominio, la descripción informal de Abbot o el análisis estructurado [12] Para nuestro ejercicio esta descripción ya ha sido dada, y podemos identificar los siguientes elementos:

- Las tres agujas de diamante
- Sesenta y cuatro discos de distinto tamaño
- Las reglas que establecen que:
    - Se deben mover todos los discos de una aguja a otra.
    - Sólo puede moverse un disco a la vez.
    - Ningún disco de tamaño mayor puede descansar sobre uno de menor tamaño.

**Desarrollo de un modelo.**

El modelo puede ser abstracto, esquemático, iconográfico o pictográfico. Identificados los elementos involucrados en nuestro problema podemos visualizar el sistema formado, quitando todo elemento decorativo y presentando una visión más sencilla. Así, sin sacerdotes, sin el domo que marca el centro del mundo, y sin importar los materiales empleados, podemos ilustrar a los objetos identificados con la imagen presentada en la figura 1.

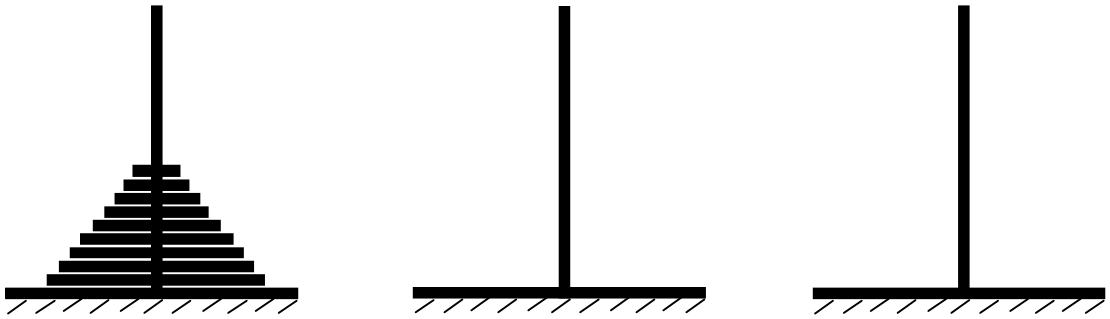

**Figura 1**. Los postes y discos que forman el sistema de las Torres de Hanoi.

**Diseño del procedimiento de solución.**

El siguiente paso es la elaboración sistemática de la serie de acciones a ejecutar para representar los movimientos que se llevan a cabo en la realidad y las reglas o restricciones que apliquen. En otras palabras, *la formulación de nuestro algoritmo*. Para nuestro problema se presenta el siguiente, y muy elegante, algoritmo recursivo.

```
Algoritmo Torres de Hanoi
Rutina Principal
Inicio
     Anillos←Pide la cantidad de anillos a mover
     Mueve(Anillos, origen, intermedio, destino)
Fin

Rutina Mueve (a, o, i, d)
Inicio
     Si a = 1
          Entonces Muestra o "--->" d
          En otro caso Mueve(a-1, o, d, i)
                       Muestra o "--->" d
                       Mueve(a-1, i, o, d)
     Fin Si
Fin
```

**Figura 2**.- Algoritmo de solución al problema de las Torres de Hanoi.

**Validación.**

Una vez formulado el algoritmo es necesario llevar a cabo su verificación. Para sorpresa de muchos, no existe un procedimiento para llevar a cabo dicha verificación, esto es, no existe un algoritmo capaz de indicar si nuestro algoritmo es correcto o no, de la misma forma que no hay un algoritmo para la elaboración de algoritmos. Esta es una de las consecuencias de los trabajos de Turing y Post.

Al inicio de este artículo mencionamos que "todo programador debe contar con un método que le permita determinar si sus programas son correctos o no", ¿como es posible esto si ya hemos dicho que tal procedimiento no existe? Primero, es necesario entender que aquí nos referimos a un método, no propiamente a un algoritmo. La determinación de la validez de un programa se logra con una elaboración metódica, que es la que se ofrece con el seguimiento de estos ocho pasos. Segundo, cada algoritmo debe ser verificado de manera individual acorde a sus parámetros, acorde a las reglas que le han dado forma.

La validación de un algoritmo, esencialmente, se realiza a través de lo que se conoce como *prueba de escritorio*; en esta prueba, paso a paso, el verificador observa las acciones del algoritmo, los valores de cada variable, cada cambio en estos valores y el flujo de las acciones a lo largo de todo el conjunto. El fin de esta etapa es verificar que el algoritmo realiza el cálculo o proceso para el cual fue elaborado. ¿Cuántas pruebas deben hacerse? ¿Cuántos valores probarse? Estas y otras preguntas sólo puede responderlas quien lleve a cabo la validación del algoritmo pero es importante que cubran un espectro que va desde el caso trivial hasta el caso extremo, pasando por uno o más casos intermedios. Esto, junto con el paso de **Análisis y Complejidad** constituyen lo que se conoce como *análisis de algoritmos*, en el que se busca determinar las características de desempeño y comportamiento de un

algoritmo. Dejamos al lector que realice las respectivas pruebas de escritorio para el caso que estamos desarrollando.

**Implementación.**

La implementación de un algoritmo consiste en la escritura de un programa empleando un lenguaje de programación. Si este paso debe ser posterior al de **Análisis y Complejidad** o viceversa, es discutible. Para muchos debe ser posterior. Un algoritmo debe implementarse una vez que se ha probado su valía y resultados esperados. Para otros debe ser anterior, la dificultad del análisis puede ser minimizada a través de una primera implementación que arroje datos sobre el comportamiento del algoritmo. Esta última actividad es conocida como *profiling*, el perfilar su funcionamiento[13] Existen varias herramientas (*profilers*) que pueden ayudar a este respecto, aunque esto también puede ser llevado a mano. La correspondiente implementación del algoritmo es presentada a continuación en Lisp.

```
(DEFUN TORRESHANOI
  (LAMBDA (N O I D)
       (SETQ MOVIMIENTOS 0)
       (COND
            (EQ N 1)(MUEVEDISCO O D))
       (T (TORRESHANOI
                 (DIFFERENCE N 1) O D I)
          (MUEVEDISCO O D)
          (TORRESHANOI
                 (DIFFERENCE N 1) O I D))
       (PRINT MOVIMIENTOS)))

(DEFUN MUEVEDISCO
    (LAMBDA (I F)
       (SETQ MOVIMIENTOS (PLUS MOVIMIENTOS 1))
       (LIST (CONCATEN (I F)))))
```

**Figura 3**.- Implementación del algoritmo en Lisp. Incluye contadores para determinar las veces que se llama a la función encargada se mover discos (*profiling*).

El perfilamiento del algoritmo se lleva a cabo mediante la inserción de un contador que nos permite saber cuántas veces se ha ejecutado la *instrucción* o *sección* que resulta crítica o definitiva para describir el comportamiento del algoritmo (aquella que tomará más tiempo, la que se ejecutará más veces, la responsable de demandar recursos). En nuestro caso es el llamado a la función encargada de mover los discos. Así, en este caso lo que estamos midiendo son los movimientos realizados por cantidad de discos a desplazar, lo que lleva asociado un tiempo por cada movimiento y por ende un tiempo total.

**Análisis y complejidad.**

El análisis y determinación de la complejidad de un algoritmo es una actividad que raya en arte. Requiere un elevado conocimiento del problema, así como de herramientas y métodos de análisis. El análisis de un algoritmo, iniciado con su *validación*, continua con el estudio para determinar su *función de trabajo* (análisis) y culmina con la determinación de su *eficiencia* (complejidad). Veremos cada parte paso a paso.

Supongamos que de la implementación (o la realización de varias pruebas de escritorio) de nuestro algoritmo de ejemplo, obtenemos los siguientes datos[1]:

| Discos | Movimientos |
|---|---|
| 1 | 1 |
| 2 | 3 |
| 3 | 7 |
| 4 | 15 |
| 5 | 31 |
| 6 | 63 |
| 7 | 127 |
| 8 | 255 |
| 9 | 511 |
| 10 | 1023 |

**Tabla 1**.- Datos obtenidos sobre la ejecución de la rutina de movimientos de discos.

La búsqueda de una función de trabajo pretende obtener una ecuación que nos permita calcular la cantidad de movimientos a realizar con un número determinado de discos; una ecuación que describa al conjunto de datos que hemos colectado. Por supuesto, esto puede lograrse con un estudio cuidadoso del algoritmo, pero en muchos casos esto requiere habilidades, conocimientos y práctica que no todos poseemos. Una alternativa es tratar de obtener esta ecuación mediante el análisis estadístico de los datos colectados (de aquí la necesidad y comentarios sobre la implementación) ya que es un proceso mecanizable.

Una *curva de ajuste* es el resultado de lo que se conoce como *análisis de regresión*, una técnica estadística que busca establecer una relación entre una variable dependiente $\boldsymbol{y}$, junto con una o más variables independientes $\boldsymbol{x_1}$, $\boldsymbol{x_2}$, $\boldsymbol{x_3}$, … que es escrita en forma de una ecuación que describe a una curva. El número de curvas que es posible calcular es infinito, para algunas pueden usarse métodos sencillos como es el de *mínimos cuadrados*, para otras es necesario recurrir a otra técnicas más elaboradas. Mediante mínimos cuadrados podemos determinar curvas como las presentadas en las siguientes figuras:

[1] Para efectos de este ejercicio sólo se ha generado una muestra de 10 valores, recuérdese que en la práctica, y acorde a los principios estadísticos, la muestra deberá contener de 25 a 30 elementos como mínimo.

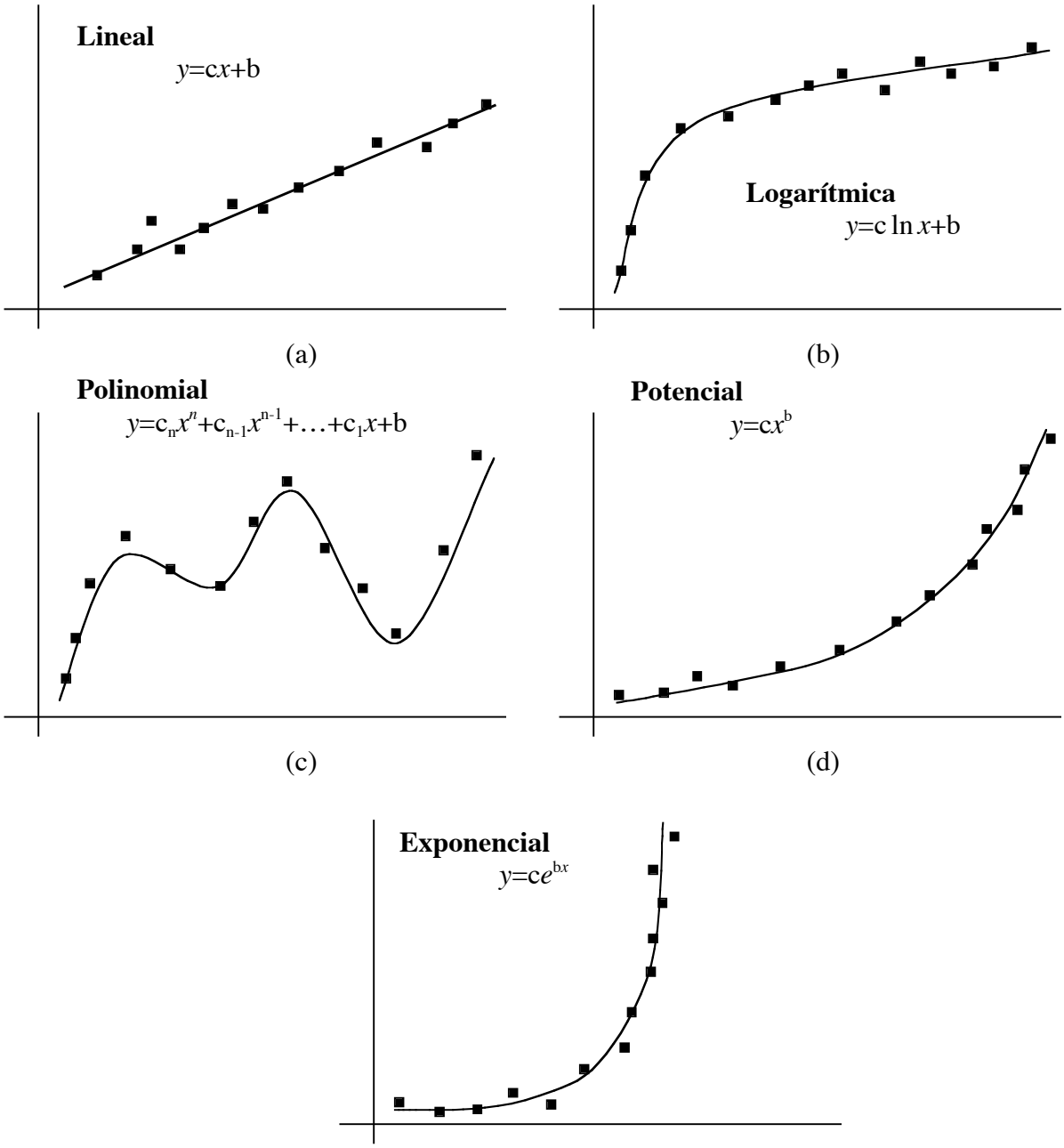


**Figura 4**. Cinco modelos de curvas de ajuste y sus ecuaciones representativas.

La *recta de regresión* (como también es llamado el caso lineal) es el ejemplo más sencillo de la aplicación del método de mínimos cuadrados. Para obtenerla sólo debe resolverse el sistema de ecuaciones

$$\begin{aligned} c\sum x + bn &= \sum y \\ c\sum x^2 + b\sum x &= \sum xy \end{aligned} \tag{1}$$

donde $n$ es el número de datos en la muestra. Para nuestro conjunto de datos, tenemos

$$\begin{aligned} 55c + 10b &= 2036 \\ 385c + 55b &= 18379 \end{aligned} \tag{2}$$

y resolviendo el sistema de ecuaciones obtenemos $c$=87.0424 y $b$=-275.1333, por lo que la ecuación buscada es

$$\mathrm{y} = 87.0424x - 275.1333\,. \tag{3}$$

¿Qué tan bien explica la ecuación obtenida al conjunto de datos? ¿Qué tanto se acerca lo que esta ecuación produce a los datos reales? Esto es algo que puede determinarse mediante el *factor de correlación*, $r$, un valor en el rango [-1, 1], que podemos determinar por la fórmula:

$$r = \frac{\sum\left(y - \bar{y}\right)\left(\hat{y} - \bar{\hat{y}}\right)}{\sqrt{\sum\left(y - \bar{y}^2\right)\sum\left(\hat{y} - \bar{\hat{y}}^2\right)}}\,. \tag{4}$$

donde $y$ corresponde a los valores originales, $\hat{y}$ a los valores calculados, y la barra sobre cada variable denota el promedio de cada conjunto de datos.

Mientras más cercana sea $r$ a ±1 más precisamente nuestra ecuación podrá reproducir los datos reales[2]. En la práctica se toma como aceptable un valor superior a **±0.8** pero todo depende del problema y precisión buscada. Para la recta de regresión calculada tenemos $r$=0.79883. Con base en el criterio indicado lo consideraremos no apropiado y buscaremos uno mejor.

Es fácil generalizar el modelo de mínimos cuadrados de forma que curvas polinomiales de grado mayor a 1 puedan calcularse. Generalizando el modelo planteado tenemos

$$\begin{aligned} \mathrm{c}_g\sum x^g + c_{g-1}\sum x^{g-1} + \ldots + c_0 n &= \sum y \\ \mathrm{c}_g\sum x^{g+1} + c_{g-1}\sum x^{g} + \ldots + c_0 \sum x &= \sum xy \\ \mathrm{c}_g\sum x^{g+2} + c_{g-1}\sum x^{g+1} + \ldots + c_0 \sum x^2 &= \sum x^2 y \\ &\vdots \\ \mathrm{c}_g\sum x^{2g} + c_{g-1}\sum x^{2g-1} + \ldots + c_0 \sum x^{g} &= \sum x^g y \end{aligned} \tag{5}$$

donde $g$ indica el grado del polinomio a calcular. Así, para una curva de segundo grado ($g$=2), cuya ecuación característica es

$$\mathrm{y} = \mathrm{c}_2 x^2 + c_1 x + c_0 \tag{6}$$

el sistema de ecuaciones a resolver sería

[2] El valor positivo o negativo de $r$ se explica por la raíz cuadrada involucrada. Un valor positivo indica que los valores altos de la ecuación explican a los valores altos de la muestra de datos, y una correlación similar explica a los valores bajos. Un valor negativo se interpreta como valores altos de la ecuación explicando a los bajos de la muestra de datos y viceversa.

$$\begin{aligned} c_2\sum x^2 + c_1\sum x + c_0 n &= \sum y \\ c_2\sum x^3 + c_1\sum x^2 + c_0\sum x &= \sum xy \\ c_2\sum x^4 + c_1\sum x^3 + c_0\sum x^2 &= \sum x^2 y \end{aligned} \tag{7}$$

Substituyendo valores

$$\begin{aligned} 385c_2 + 55c_1 + 10c_0 &= 2036 \\ 3025c_2 + 385c_1 + 55c_0 &= 18379 \\ 25333c_2 + 3025c_1 + 385c_0 &= 169593 \end{aligned} \tag{8}$$

y resolviendo el sistema de ecuaciones, obtenemos

$$y = 23.136363x^2 - 167.457575x + 23.8666 \tag{9}$$

con un factor de correlación $r$=0.962649.

Por supuesto, nuestra búsqueda no se limitaría a sólo dos polinomios y habría que probar los suficientes para determinar si es o no una ecuación lineal la que buscamos. Por motivos de espacio no buscaremos más en esta dirección y nos avocaremos a revisar el mundo no lineal.

La primera curva no polinomial que exploraremos será una curva logarítmica de forma

$$y = c \ln x + b. \tag{10}$$

El modelo de mínimos cuadrados puede ser fácilmente adaptado a este caso; el sistema de ecuaciones sería

$$\begin{aligned} c\sum \ln x + bn &= \sum y \\ c\sum (\ln x)^2 + b\sum \ln x &= \sum (\ln x) y \end{aligned} \tag{11}$$

y llevando a cabo los cálculos necesarios obtendríamos

$$\begin{aligned} 15.104413c + 10b &= 2036 \\ 27.65024c + 15.104413b &= 4449.05206 \end{aligned} \tag{12}$$

y tras resolver el sistema de ecuaciones determinaríamos que

$$y = 284.0813 \ln x - 225.48823 \tag{13}$$

con un factor de correlación $r$=0.631225.

Otras curvas de este tipo deben ser probadas antes de determinar que el problema no exhibe características de naturaleza logarítmica. Para efectos de este ejemplo asumiremos que así es y exploraremos curvas de tipo exponencial.

Para probar una curva con la forma

$$y = b\, e^{cx} \tag{14}$$

es necesario recurrir a una pequeña transformación previa a fin de adecuarla al modelo que hemos venido trabajando. La transformación consiste en aplicar el logaritmo natural a ambos lados de la ecuación, re-expresándola como[3]

$$y = b\, e^{cx} \xrightarrow{\ln} \ln y = cx \ln \mathrm{e} + \ln b \tag{15}$$

para formar el sistema

$$\begin{aligned} c\sum x + \ln b\, n &= \sum \ln y \\ c\sum x^2 + \ln b \sum x &= \sum x \ln y \end{aligned} \tag{16}$$

y así el sistema de ecuaciones a resolver es

$$\begin{aligned} 55c + 10 \ln b &= 36.88201 \\ 385c + 55 \ln b &= 264.5676 \end{aligned}. \tag{17}$$

Los valores obtenidos para las incógnitas tras la resolución del sistema son $c$=0.74808 y $ln\ b$=-0.42624. Debe tenerse presente que se tratan de valores para un sistema transformado. Por lo que debe revertirse el proceso para llegar a la ecuación que originalmente estábamos buscando. Así, mediante la operación inversa (aplicar el exponente $e$ a ambos lados de la ecuación), obtenemos:

$$\ln y = cx + \ln b \xrightarrow{e^x} y = b\, e^{cx} \tag{18}$$

siendo la ecuación buscada:

$$\mathrm{y} = 0.652963\, e^{0.74808x} \tag{19}$$

con $r$=0.9980178.

Un valor de correlación tan alto nos da un buen indicio de la naturaleza de la ecuación que estamos buscando. Veamos el caso potencial, a través de una ecuación de la forma:

$$y = b\, x^c \tag{20}$$

que mediante la transformación

$$y = b\, x^c \xrightarrow{\ln} \ln y = c \ln x + \ln b \tag{21}$$

nos generará el sistema de ecuaciones

---

[3] Algunas propiedades de los logaritmos: ln($e$)=1; ln (a·b)=ln(a)+ln(b); ln($a^b$)=b*ln(a).

$$\begin{aligned} c\sum \ln x + \ln b\, n &= \sum \ln y \\ c\sum (\ln x)^2 + \ln b \sum \ln x &= \sum \ln x \ln y \end{aligned} \tag{22}$$

cuyos valores son

$$\begin{aligned} 15.104413c + 10\ln b &= 36.88201 \\ 27.65024c + 15.104413\ln b &= 70.21359 \end{aligned} \tag{23}$$

y que al ser resuelto nos producirá los resultados $c$=2.99953 y $ln\ b$=-0.84242. Similarmente al caso anterior es necesario considerar los resultados como parciales debido a que se requiere aplicar la retransformación de la ecuación

$$\ln y = c\ln x + \ln b \xrightarrow{e^x} \mathrm{y} = \mathrm{b}\, x^c \tag{24}$$

para así obtener

$$y = 0.43067\, x^{2.99953} \tag{25}$$

con $r$=0.96885.

Así, sin lugar a dudas una ecuación exponencial es la que describe el comportamiento de nuestro algoritmo y vale la pena explorar algunas otras posibilidades, por ejemplo:

$$\mathrm{y} = c2^x + b \tag{26}$$

cuyo sistema de ecuaciones es

$$\begin{aligned} c\sum 2^x + b\, n &= \sum y \\ c\sum 2^{2x} + b\sum 2^x &= \sum 2^x y \end{aligned} \tag{27}$$

y cuyos valores son

$$\begin{aligned} 2046c + 10b &= 2036 \\ 1398100c + 2046b &= 1396054 \end{aligned} \tag{28}$$

cuya resolución da los valores $c$=1 y $b$=-1, que corresponden a la ecuación:

$$y = 2^x - 1 \tag{29}$$

con un factor de correlación r=1, que nos indica que ésta es la ecuación buscada. Con esto hemos determinado el comportamiento del algoritmo, sólo nos resta indicar su costo.

El costo de ejecución de un algoritmo está directamente relacionado con los recursos que éste emplea. La ejecución de más pasos (o su repetición) para llegar a una solución, implican el uso del

procesador por más tiempo; mantener, recuperar y organizar datos requiere memoria o espacio en disco; presentarlos consume tiempo, material de impresión o espacio de almacenamiento no volátil. La medición del costo se da en términos de espacio o tiempo, el cálculo del costo se da en términos monetarios una vez que se ha medido la complejidad del algoritmo.

La complejidad computacional del algoritmo es indicada por el *orden* de su función de trabajo. Existen dos formas de cómo denotar este orden y que son conocidas como "gran *O*" y "pequeña *o*". En la práctica se suele identificar a la *gran O* como sinónimo del orden de la función de trabajo del algoritmo, sin embargo ambas medidas son válidas e identifican la forma en que una función varía en magnitud a medida que sus parámetros tienden a un límite. Su definición es como sigue. Una función $f(n)$ se dice que es de orden $O(n)$ si cumple la siguiente condición

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \text{constante} \neq 0 \tag{30}$$

y se dice que es de orden $o(n)$ si

$$\lim_{n\to\infty}\frac{f(n)}{h(n)} = 0 \tag{31}$$

en ambas expresiones $f(n)$ denota la función de trabajo del algoritmo, $g(n)$ y $h(n)$ son funciones propuestas que se obtienen, respectivamente, del término de la función con mayor crecimiento (el que define propiamente el comportamiento del algoritmo) y de cualquier otra función con un crecimiento mayor. En nuestro caso

$$\begin{aligned}\lim_{x\to\infty}\frac{f(x)}{g(x)} &= \frac{2^x-1}{2^x}\\ &= \lim_{x\to\infty}\frac{2^x}{2^x} - \lim_{x\to\infty}\frac{1}{2^x}\\ &= 1-0=1\end{aligned} \tag{32}$$

con lo que queda establecido que $O(f(x))=2^x$. Mientras que, por ejemplo,

$$\begin{aligned}\lim_{x\to\infty}\frac{f(x)}{h(x)} &= \frac{2^x-1}{e^x}\\ &= \lim_{x\to\infty}\frac{2^x}{e^x} - \lim_{x\to\infty}\frac{1}{e^x}\\ &= 0-0=0\end{aligned} \tag{33}$$

nos indica que $o(f(x))=e^x$; de hecho, cualquier función que crezca más rápido que $2^x$ nos indicará un orden $o(f(x))$. Es con base en esta determinación que se cataloga a un algoritmo como ***P*** (polinomial) o ***NP*** (no polinomial). Los algoritmos ***P*** son preferidos a los ***NP***, debido a su naturaleza gradual y suave en crecimiento con respecto a los parámetros de entrada.

**Pruebas.**

La verificación y análisis del algoritmo están enfocados a la revisión de los resultados que éste provee y cómo los provee. La implementación del algoritmo generalmente incurre en controles adicionales que están más allá de la naturaleza propia del algoritmo pero que son necesarios. Estos controles tienen que ver con mecanismos de seguridad para la validación de entrada de datos, así como su salida, control de recursos y acciones sobre excepciones. La adición de toda esta programación extra debe validarse mediante pruebas de ejecución destinadas a validar la calidad del programa creado y garantizar su funcionamiento antes de ofrecerlo al usuario.

**Documentación.**

La mayoría de la gente piensa en la documentación de un programa como la creación de su manual de usuario. Ciertamente, junto con el producto es necesario entregar la documentación necesaria para su operación pero, adicionalmente a lo que se requiera para su distribución, es necesario la elaboración de la documentación necesaria sobre el algoritmo implementado, que en esencia es todo lo realizado aquí. Toda esta documentación no es de interés para el usuario final del programa, pero sí para el desarrollador o equipo de desarrollo. Por supuesto, esta documentación es lo que respalda la construcción del algoritmo y su correspondiente implementación, es la propiedad intelectual de su creador y hasta puede considerarse confidencial. Esta será requerida y de mucha utilidad ante una solicitud de patente o cualquier disputa legal sobre su origen.

## Epilogo.

¿Cuánto tiempo deberá pasara para que se transfieran los 64 discos de una aguja a otra? ¿Cuánto tiempo tomará a los brahmanes convertirse en polvo y al mundo desaparecer? Supongamos que cada sacerdote es capaz de ejecutar el movimiento de un disco, sin incurrir en movimientos equivocados, en un segundo. Con la función de trabajo que hemos obtenido podemos ahora saber que

$$y = 2^{64} - 1 = 1.84467440737 \text{x} 10^{19} \text{ segundos} \tag{34}$$

dividiendo entre 60 para convertir a minutos, dividiendo nuevamente entre 60 para convertir el resultado en horas, dividiendo éste entre 24 para convertirlo a días y, finalmente, dividiendo entre 365 sabremos que la tarea requiere un periodo de tiempo de 584,942,417,355 años. Con lo que podemos estar tranquilos. El universo tiene apenas unos $20 \text{x} 10^9$ años y la tierra no apareció sino mucho después de su formación. Así que los brahmanes tardarán un poco en completar su tarea.

## Conclusiones.

Se ha expuesto el desarrollo completo de un algoritmo. *El deber ser*. Todo programador o investigador enfocado al desarrollo de algoritmos es libre de aplicar los pasos que considere necesarios. Ciertamente no se requiere seguir todos estos para programar una computadora, pero sí serán necesarios desarrollarlos en mayor o menor medida cuando tratemos de explicar por qué se tarda tanto alguno de nuestros programas y, más aún, cuando tratemos de dar a conocer el resultado de un trabajo con el que ofrecemos un novedoso y mejorado método de solución no contemplado antes.

Para el desarrollo del cálculo de la función de trabajo existe software muy útil (hojas de cálculo, paquetes estadísticos, software matemático) que puede ahorrarnos este tedioso procedimiento. Aquí se han desarrollado paso a paso para ilustrar de forma más tangible su aplicación.

## Referencias.


[1] **Standard Dictionary International Edition**; Funk & Wagnalls Co.; New York, USA, 1967.

[2] **Merriam-Webster Collegiate Dictionary.** Palm Edition; Franklin Electronic Publishers, Inc.; 2001 USA.

[3] García-Pelayo y Gross, Ramón; **Pequeño Larousse en Color;** Ediciones Larousse, España, 1981. ISBN 2-03-020551-6.

[4] De Galiana Mingot, Tomás; **Pequeño Larousse de Ciencias y Técnicas**; México, 1984. ISBN 968-6042-22-9.

[5] Valerie Illingworth, Gen. Ed.; **Diccionario de Informática**; Ediciones Díaz de Santos S.A.; Madrid, España, 1993. ISBN 84-7978-068-1.

[6] Baldor, Aurelio; **Álgebra**; Ediciones y Distribuciones Códice, S.A., Madrid, España; 1982. ISBN 84-357-0062-3.

[7] Knuth, Donald E; **The Art of Computer Programming. Volume 1: Fundamentals Algorithms**; Addison-Wesley, 3rd Ed.; USA, 1999. ISBN 0-201-89683-4.

[8] Bracho, Felipe; *Matemáticas o Computación ¿el huevo, la gallina o la lógica combinatoria?*; **Ciencia y Desarrollo**; Núm. 75, Año XIII; Julio-Agosto 1987; CONACyT, México.

[9] Hopcroft, John E. & Ullman, Jeffrey D.; **Introduction to Automata Theory, Languages, and Computation**, Addison-Wesley; USA, 1979. ISBN 0-201-02988-X.

[10] Uspenski, Victor A.; **Máquina de Post**, Editorial Mir; Moscú, 1983.

[11] Goodman, S. E. & Hedetniemi, S. T.; **Introduction to the Design and Analysis of Algorithm**; 5th Ed., McGraw-Hill International Editions; Singapore, 1988. ISBN 0-07-Y66300-9.

[12] Booch, Grady; **Object-Oriented Designs with Applications**; The Benjamin/Cummings Publishing Company, Inc.; California, USA, 1991. ISBN 0-8053-0091-0.

[13] Aho, Alfred V., Kernighan, Brian W. & Weinberger, Peter J.; **The AWK Programming Language**; Addison-Wesley; USA, 1988. ISBN 0-201-07981-X.

[14] Spiegel, Murray R. **Probabilidad y Estadística**. **Serie Shaum**; McGraw-Hill; México, 1984. ISBN 968-451-102-7.


**Versión electrónica del documento.**

Una copia de este texto puede ser obtenido en formato PDF del sitio http://homepage.mac.com/eravila, así como un archivo de Excel ejemplificando y complementando los cálculos efectuados.