Algorítmica

Introducción.

La ciencia de la computación es el estudio de los algoritmos, incluyendo sus propiedades, hardware, aspectos lingüísticos y aplicaciones.

Podemos definir un **Algoritmo** como una secuencia ordenada y finita de pasos no ambiguos tales que al llevarse a cabo, dará como resultado que se realice la tarea para la que se ha diseñado en un tiempo finito y con recursos limitados.

De esta forma, no podemos por ejemplo obtener la sucesión de **Fibonacci** mediante un algoritmo pues esta es infinita.

Nota : Debemos tener claro que hay una diferencia entre un programa y un algoritmo. Un programa es una serie de instrucciones ordenadas codificadas en un lenguaje de programación que *expresa* un algoritmo.

Los algoritmos.

Un algoritmo es una secuencia finita y ordenada de pasos, exentos de ambigüedad, tal que al llevarse a cabo dará el resultado esperado a la tarea esperada. Se resuelve con recursos limitados y tiempo finito.

- *Definición de Bazaara* : algoritmo como proceso iterativo.
- *Definición de Knuth* :

Un método computacional es una cuaterna (Q, I, Ω, f) en la que Q es un conjunto que contiene a I y a Ω como subconjuntos y f es una función de Q en Q tal que:

$$f(q) = q \quad \forall q \in \Omega$$

y donde Q es el conjunto de estados del cálculo, I la entrada , Ω la salida y f la regla de cálculo aplicada. Cada entrada $x \in I$ define una sucesión computacionel $x_0, ..., x_n$ tal que $x_0 = x$ y $x_{k+1} = f(x_k)$ si $k \geq 0$.

Se dice que la sucesión computacional termina en k etapas si k es el menor entero para el cual x_k está en Ω y en esa caso, a partir de x se obtiene como salida x_k.

Ejemplo: Algoritmo de Euclídes

Dados dos enteros m y n, calcular su m.c.d:

- *E1* : Calculo del resto de m entre n.
- *E2* : Si el resto es 0, el algoritmo termina y la solución es n.
- *E3* : Tomo m=n y n=r, vuelvo a *E1* y repito el proceso.

Propiedades de los algoritmos

Los algoritmos tienen un conjunto de propiedades comunes a todos ellos:

- *Finitud.* Se puede completar en un número finito de pasos.
- *Especifidad.* Cada etapa del algoritmo está definida y descrita correctamente para dar respuesta a una parte determinada del problema.
- *Input.* El algoritmo recibe una entrada de datos.
- *Output.* El algoritmo devuelve un resultado en base a estos datos.
- *Efectividad.* Se completa en un tiempo determinado, que viene dado por el número de operaciones que se requieren para afrontar el problema.

Además los algoritmos dependen tanto de sus propiedades de forma y matemáticas, como del hardware en el que sean ejecutadas. Según el uso que se le quiera dar al algoritmo se implementará en un lenguaje de programación o en otro.

Etapas en la elaboración de un algoritmo

1. *Construcción:* Acto de crear un algoritmo.
2. *Expresión de algoritmos:* Cada paso debe de describirse de manera clara.
3. *Validación:* Que calcule los resultados esperados al problema dado. Es una frase previa a la escritura del programa.
4. *Análisis:* Comprobar el tiempo que tardaría, así como el tamaño que ocuparía. Permite realizar comparaciones teóricas entre algoritmos.
5. *Test:* Corrección de errores que se detecten y comparación de resultados.

La elección final de un algoritmo dependerá, como ya hemos dicho antes, tanto del número de operaciones que se requieran como de los recursos de hardware disponibles. También son otros factores a considerar la adaptación a cualquier tipo de computador, la simplicidad y elegancia del algoritmo y del coste económico de su realización, así como su eficiencia, rapidez y facilidades de programación.

Para probar un algoritmo y validarlo, interesa probar su comportamiento y funcionamiento en los casos extremos, que prevemos que podría causar errores en su funcionamiento.

Comportamiento: Algoritmos Selección y Ordenación

Un mismo algoritmo puede dar diversos tiempos en función de los datos del problema que desee resolver.

Caso inicial: U y V arrays de n elementos. U ordenado de forma ascendente y V de forma descendente.

- *Selección:* Indiferentes U y V. Menos de 15% de diferencia en sus ejecuciones.

- *Inserción:* Para U y 5000 elementos tarda $\frac{1}{5}$ segundos. Para V y 5000 elementos tarda 3 minutos y medio.

Tema 1: La Eficiencia de los Algoritmos

Cálculo de la eficiencia de un algoritmo

Podemos destacar tres métodos para calcular la eficiencia e un algoritmo:

1. *Enfoque empírico o (a posteri):* Programar los diferentes algoritmos candidatos y ejecutarlos sobre diferentes casos con ayuda de un ordenador.
2. *Enfoque teórico o (a priori):* Trata de determinar matemáticamente la cantidad de recursos necesarios para cada algoritmo como una función del tamaño de los casos considerados.
3. *Enfoque Híbrido:* Determina teóricamente la forma de la función que describe la eficiencia del algoritmo y cualquier parámetro numérico que se necesite se determina empíricamente con un ordenador.

El tiempo de ejecución de un programa depende de:

1. *Input del programa.*
2. *Calidad del código generado por computador.*
3. *Naturaleza y velocidad de instrucciones máquina.*
4. *Complejidad del algoritmo.*

No siempre un algoritmo es mejor cuanto menor sea su tiempo de ejecución por ejemplo:

Sean dos algoritmos que consumen uno n^3 días y otro n^2 segundos, el primero es mejor asintóticamente hablando y el segundo mejor que el primero desde un punto de vista práctico, ya que la constante *oculta* lo hace mejor.

Notaciones O y Ω

Para poder comparar los algoritmos empleando los tiempos de ejecución, se emplea una notación asintótica, según la cual un algoritmo de ejecución T(n) se dice que es de orden O(f(n)), si existe una constante positiva c y un número entero n_0 tales que:

$$\forall\ n \geq n_0 \longrightarrow T(n) \leq cf(n)$$

Así queda claro que cuando T(n) es O(f(n)), lo que estamos dando es una cota superior para el tiempo de ejecución, que siempre referiremos al peor caso del problema en cuestión.

De manera análoga, se introduce la notación $\Omega(n)$. Decimos que un algoritmo es $\Omega(g(n))$ si existen dos constantes positivas k y m_0 tales que:

$$\forall\ n \geq m_0 \longrightarrow T(n) \geq kg(n)$$

Es de destacar la simetría de ambas notaciones, mientras que una acota superiormente, la otra lo hace inferiormente. Una de las razones por lo que esto es útil, es porque hay veces en las que un algoritmo es rápido, pero no lo es para los *inputs* por lo que debemos de estar dispuestos a saber lo menos que estamos dispuestos en consumir tiempo para resolver cualquier caso de un problema.

Notación Θ