Algoritmos

La palabra algoritmo, uno de los términos tecnológicos de moda, frecuentemente, es usado con cierto desconocimiento cuando hablamos de la **inteligencia artificial** o de conceptos cercanos.

Un **algoritmo** es, sencillamente, una secuencia ordenada y finita de operaciones sencillas a través de la cual podemos determinar la solución a un problema.

Los algoritmos nos permiten ejecutar una **acción** o **resolver un problema** mediante una serie de instrucciones definidas, ordenadas y finitas.

Basándonos en esta definición, tanto una receta de cocina como el cálculo de una raíz cuadrada de un número o un protocolo sanitario concreto son algoritmos.

Un algoritmo es una secuencia de instrucciones finitas que llevan a cabo un conjunto de procesos de forma lógica, sistemática y ordenada para dar respuesta a un determinado problema o propósito, con lo que un algoritmo resuelve un problema concreto a través de una serie finita de una o más instrucciones definidas, concisas y ordenadas.

1.1. Características de los algoritmos

Las **características** básicas de un algoritmo son:

- **Serie finita**: tiene inicio y fin, todo algoritmo comienza en un estado inicial con una serie de datos específicos, y culmina con una solución o salida.
- **Secuencialidad**: un algoritmo está compuesto por una serie de pasos ordenados.
- **Sin ambigüedad**: las secuencias son concretas, cada paso es claro y no deja lugar a ambigüedad.
- **Validez**. Un algoritmo es válido si carece de errores. Un algoritmo puede resolver el problema para el que se planteó y sin embargo no ser válido debido a que posee errores.
- **Eficiencia**. Un algoritmo es eficiente si obtiene la solución al problema en poco tiempo. No lo es si es lento en obtener el resultado.
- **Optimización**. Un algoritmo es óptimo si es el más eficiente posible y no contiene errores. La búsqueda de este algoritmo es el objetivo prioritario del

programador. No siempre podemos garantizar que el algoritmo hallado es el óptimo, a veces sí.
- Ser **independiente** de la computadora. Los algoritmos deben escribirse para poder ser utilizados por cualquier máquina.
- Ser **preciso**. Los resultados de los cálculos deben de ser exactos, no solo aproximarse a la solución.
- Debe poder **repetirse**. Los algoritmos deben permitir su ejecución tantas veces como sea necesario, además de resolver el problema cambiando los datos de entrada.

Condiciones que deben cumplir los algoritmos:

1. **Finitud** del número de acciones elementales.
2. **Definibilidad**. Cada paso debe efectuarse de forma precisa. Precisión en el leguaje: pseudocódigo.
3. **Entrada**. Deben estar especificados los datos a los que se puede aplicar.
4. **Salida**. Debe especificarse el conjunto de todas las salidas que pueden producirse.
5. **Efectividad**. Todos los procesos deben ejecutarse en un tiempo finito.

1.2. Algoritmia

La **algoritmia** es el tratamiento sistemático de técnicas para el diseño y análisis de algoritmos eficientes, que faciliten el desarrollo de programas de ordenador. Comprende:

- Estudio conceptual
- Criterios de evaluación

Estudio conceptual apoyándose en:

- La modelización de los datos (árboles, grafos, ...)
- Los modelos de máquinas (Turing, Von Neumann).
- Los métodos (divide y vencerás, recursión, ...)
- Estudio algoritmos clásicos en modelos discretos (palabras, grafos, ...)

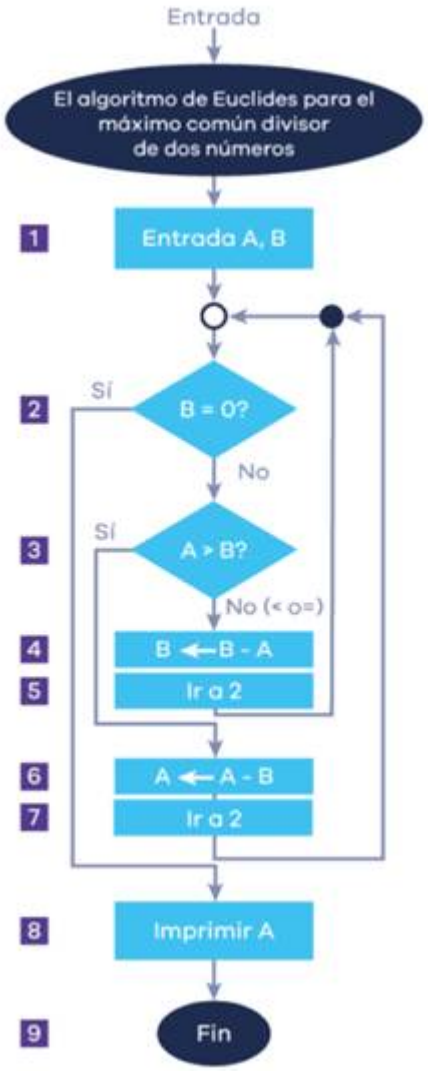
Conocimiento **criterios de evaluación**:

- Las prestaciones de los algoritmos (complejidad).

- Su corrección (verificación).
- Su expresión en un lenguaje adecuado (Java, C, Python...)

1.3. Visualización de algoritmos

Los **diagramas de flujo** son representaciones gráficas que muestran los pasos de un proceso. Permiten representar procesos computacionales con el fin de comunicar y debatir sobre los mismos en las etapas previas al desarrollo de un código.



Los diagramas de flujo siguen unas **normas**, como el uso de rectángulos para las instrucciones y diamantes para las decisiones, utilizando flechas para conectar unos pasos con otros, etc.

Uno de los objetivos de los diagramas de flujo es ayudar a definir la entrada, el proceso y la salida de los algoritmos.

Los diagramas de flujo ayudan a los programadores a **visualizar los pasos** de un algoritmo. Al igual que hacer un esquema antes de escribir un ensayo, los diagramas de flujo ayudan a organizar las ideas y usan símbolos específicos para representar diferentes partes de un algoritmo.

Los diagramas de flujo ayudan a **organizar las ideas** definiendo la entrada, el proceso y la salida de los algoritmos, a la vez que siguen unas normas, para las que usan símbolos específicos que representan las diferentes partes del algoritmo. El significado de los símbolos es el siguiente:

- **Óvalo**: comienzo o final del programa.
- **Paralelogramo**: operación de entrada.
- **Rectángulo**: instrucción o proceso a realizar.
- **Diamante**: decisión a tomar (el programa debe continuar por una de las dos rutas).
- **Híbrido**: operación de salida.
- **Flechas**: conectan unos pasos con otros e indican la dirección del flujo.

1.4. Notación algorítmica

Debe recoger las reglas de descripción de los objetos manipulados (informaciones de léxico), las operaciones puestas en juego (acciones del léxico) y el modo de organizar estas acciones en el tiempo (primitivas de control).

Acciones posibles:

- Acción elemental
- Composición secuencial
- Composición condicional.
- Composición alternativa.
- Composición selectiva.
- Composición iterativa.
- Acciones con nombres

Notación para cada posible acción:

Acción elemental

acción elemental

Composición secuencial

acción primera

acción segunda

...

acción n-ésima

Composición condicional

si condición entonces

acción si

fin si

Composición alternativa

si condición entonces

acción si

si no

acción no

fin si

Composición selectiva

según indicador hacer

valor primero: acción primera

valor segundo: acción segunda

...

valor último: acción última

fin según

Composición iterativa

1. repetir ... hasta

repetir

acción

hasta condición.

2. mientras ... hacer

mientras condición hacer

acción

fin mientras

3. para ...

para índice = valor inicial hasta índice = valor final hacer

acción

fin para

Acciones con nombre

acción nombre es

acciónprimera

acciónsegunda

...

acciónúltima

finacciónnombre

1.5 Partes de un algoritmo

1 - La entrada de un algoritmo

Son los datos con los que el algoritmo opera. Ejemplos:

A. Si se trata de una receta, la entrada serían los ingredientes.

B. Si se trata de un algoritmo informático, la entrada sería, por ejemplo, un conjunto de datos.

2 - El proceso del algoritmo

Son los pasos y operaciones que sigue el algoritmo utilizando o no los datos de entrada:

A. Si se trata de una receta, los diferentes pasos definidos y en orden que se han establecido

B. Si se trata de un algoritmo informático, el proceso es la combinación ordenada de operaciones matemáticas, algebraicas, etc. que se han establecido por parte de los autores: sumas, restas, comparaciones, otros algoritmos, etc.

3 - La salida del algoritmo

Es el resultado que entrega el algoritmo:

A. Si se trata de una receta, la salida es el plato establecido por parte del autor.

B. Si se trata de un algoritmo informático, la salida es el dato o conjunto de datos calculados esperados o un error

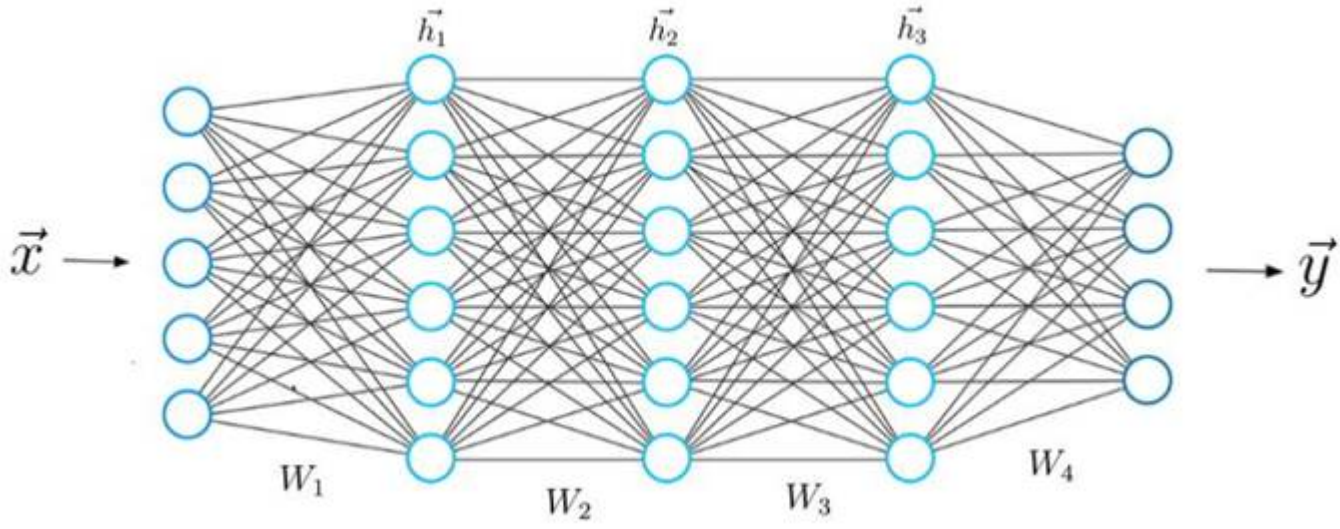
Los algoritmos recogen **operaciones** tan sencillas que pueden ser realizadas con éxito por cualquiera, incluso por máquinas: una máquina, ordenador, dispositivo, etc. puede ejecutar de forma automática y en tiempos ínfimos cualquier algoritmo sencillo (sumar, ordenar, repetir condiciones sencillas, etc.), y, por tanto, puede ejecutar cualquier algoritmo complejo siempre que pueda definirse como una combinación ordenada y concisa de algoritmos sencillos (divide y vencerás).

De esta manera las máquinas, ordenadores, etc., disponiendo de una mínima capacidad, pueden realizar casi cualquier operación por compleja que sea si nosotros la podemos **definir en términos** de otras **más simples**.

El trabajo de los programadores informáticos, el cual consiste en traducir determinados problemas del mundo, entorno, ciencia o empresa a un lenguaje o lenguajes que una máquina pueda entender, es decir, en algoritmos que una máquina pueda ejecutar: para ello hay que describir una realidad en pequeños problemas simples en sucesión y poner al ordenador/dispositivo a la tarea

En tanto en cuanto se disponga del talento para la programación de algoritmos, así como la capacidad de procesamiento y la capacidad para guardar y disponer de

datos, las posibilidades para resolver problemas se multiplican exponencialmente.



Tal es la capacidad que la sociedad está alcanzando en las últimas décadas, tanto de desarrollar algoritmos complejos como de lograr capacidades asombrosas de procesamiento y almacenamiento de datos (arquitecturas de ordenadores en la nube, grandes capacidades de almacenamiento en bases de datos cada vez más veloces y un largo etc.), que realmente se consiguen resolver problemas verdaderamente retadores y ambiciosos.

Problemas tan retadores y complejos que a veces es enormemente complicado poder describir lo que un algoritmo hace y cómo lo hace (interpretabilidad, explicabilidad) y si tienen un comportamiento ético.

Ejemplos de ello no dejan de surgir en los últimos tiempos en los ámbitos de las redes sociales y, en general, en cualquier ámbito donde implicamos evoluciones tecnológicas, entre ellos:

- Transporte (coches autónomos)
- Salud (genética, prevención, etc.)
- Finanzas
- Ciencia en general
- Periodismo

Todo ello también genera el gran reto de evolucionar el marco regulatorio o legal para dar alcance al uso de los algoritmos para dar equilibrio a la utilidad que ofrecen y al coste o impacto que puede suponer su uso, privacidad, sesgos, etc.

Existen diferentes tipos de problemas: ordenar datos, buscar patrones, encontrar la ruta óptima, etc. A su vez, existen múltiples tipos de algoritmos para resolver problemas en diferentes campos de estudio: procesamiento de imágenes, criptografía, inteligencia artificial.

Cuando una solución se lleva a cabo mediante un algoritmo, la solución se vuelve **reutilizable**. Uno de los objetivos de la programación es el desarrollo de software reusable, que pueda ser reutilizado en multitud de proyectos reduciendo así los costes operativos.