1 **El** **análisis de algoritmos** es una parte importante de la Teoría de [complejidad computacional](https://es.wikipedia.org/wiki/Complejidad_computacional) más amplia, que provee estimaciones teóricas para los recursos que necesita cualquier [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) que resuelva un problema computacional dado. Estas estimaciones resultan ser bastante útiles en la búsqueda de algoritmos eficientes.

A la hora de realizar un análisis teórico de algoritmos es común calcular su complejidad en un **sentido asintótico**, es decir, para un tamaño de entrada suficientemente grande.

La medida exacta (no [asintótica](https://es.wikipedia.org/wiki/As%C3%ADntota)) de la eficiencia a veces puede ser computada pero para ello suele hacer falta aceptar supuestos acerca de la implementación concreta del algoritmo, llamada [modelo de computación](https://es.wikipedia.org/wiki/Modelo_de_computaci%C3%B3n). Un modelo de computación puede definirse en términos de un [ordenador abstracto](https://es.wikipedia.org/wiki/M%C3%A1quina_abstracta), como la [Máquina de Turing](https://es.wikipedia.org/wiki/M%C3%A1quina_de_Turing), y/o postulando que ciertas operaciones se ejecutan en una unidad de tiempo. Por ejemplo, si al conjunto ordenado al que aplicamos una [búsqueda binaria](https://es.wikipedia.org/wiki/B%C3%BAsqueda_binaria) tiene 'n' elementos, y podemos garantizar que una única búsqueda binaria puede realizarse en un tiempo unitario, entonces se requieren como mucho log2 N + 1 unidades de tiempo para devolver una respuesta.

Las medidas exactas de eficiencia son útiles para quienes verdaderamente implementan y usan algoritmos, porque tienen más precisión y así les permite saber cuánto tiempo pueden suponer que tomará la ejecución. Para algunas personas, como los desarrolladores de [videojuegos](https://es.wikipedia.org/wiki/Videojuego), una constante oculta puede significar la diferencia entre éxito y fracaso.

Las estimaciones de tiempo dependen de cómo definamos un paso. Para que el análisis tenga sentido, debemos garantizar que el tiempo

**Donald Ervin Knuth**

Se le conoce principalmente por ser el autor de la obra The Art of Computer Programming (El arte de programar computadoras), una de las más respetadas referencias en el campo de las ciencias de la computación. Sentó las bases y dio nombre al análisis de algoritmos, y ha realizado numerosos aportes a varias ramas teóricas de la informática. Es el creador de TEX, del sistema de diseño de tipos METAFONT y del estilo de programación conocido como programación literaria (Literate programming).3​ Knuth es conocido como el "padre del análisis de algoritmos".4​

Knuth es un programador conocido por su humor geek: ofrece una recompensa de 2,56 dólares a quien encuentre errores conceptuales o tipográficos en sus libros (la razón detrás de la extraña cifra es que «256 centavos son 1 dólar hexadecimal»), y por otro lado ofrecía 3,16 por errores en 3:16 Bible Texts Illuminated.

2 **ORDEN DE UN ALGOR ITMO**

Aunque a lo largo de los años se han desarrollado muchos algoritmos de ordenamiento, los usados con más frecuencia son: método de la burbuja, de selección, de la burbuja mejorada, shell. Todos ellos realizan el ordenamiento en memoria RAM, por ello se denominan métodos de ordenamiento interno.

**PRINCIPIOS GENERALES DEL ORDENAMIENTO**

Los algoritmos que se presentan comparan elementos de un mismo vector, y, si los dos elementos están desordenados se realiza una transferencia bidireccional entre estos valores.

En [computación](https://es.wikipedia.org/wiki/Computaci%C3%B3n) y [matemáticas](https://es.wikipedia.org/wiki/Matem%C3%A1ticas) un **algoritmo de ordenamiento** es un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) que pone elementos de una [lista](https://es.wikipedia.org/wiki/Lista_(estructura_de_datos)) o un [vector](https://es.wikipedia.org/wiki/Vector_(programaci%C3%B3n)) en una secuencia dada por una [relación de orden](https://es.wikipedia.org/wiki/Orden_total), es decir, el resultado de salida ha de ser una [permutación](https://es.wikipedia.org/wiki/Permutaci%C3%B3n) —o reordenamiento— de la entrada que satisfaga la relación de orden dada. Las relaciones de orden más usadas son el orden numérico y el [orden lexicográfico](https://es.wikipedia.org/wiki/Orden_lexicogr%C3%A1fico). Ordenamientos eficientes son importantes para optimizar el uso de otros algoritmos (como los de [búsqueda](https://es.wikipedia.org/wiki/Algoritmo_de_b%C3%BAsqueda) y fusión) que requieren listas ordenadas para una ejecución rápida. También es útil para poner datos en forma canónica y para generar resultados legibles por humanos.

Desde los comienzos de la computación, el problema del ordenamiento ha atraído gran cantidad de investigación, tal vez debido a la complejidad de resolverlo eficientemente a pesar de su planteamiento simple y familiar. Por ejemplo, [BubbleSort](https://es.wikipedia.org/wiki/Bubblesort) fue analizado desde 1956.[1](https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento#cite_note-1)​ Aunque muchos puedan considerarlo un problema resuelto, nuevos y útiles algoritmos de ordenamiento se siguen inventado hasta el día de hoy (por ejemplo, el [ordenamiento de biblioteca](https://es.wikipedia.org/wiki/Library_sort) se publicó por primera vez en el 2004). Los algoritmos de ordenamiento son comunes en las clases introductorias a la computación, donde la abundancia de algoritmos para el problema proporciona una gentil introducción a la variedad de conceptos núcleo de los algoritmos, como [notación de O mayúscula](https://es.wikipedia.org/wiki/Cota_superior_asint%C3%B3tica), [algoritmos divide y vencerás](https://es.wikipedia.org/wiki/Algoritmo_divide_y_vencer%C3%A1s), [estructuras de datos](https://es.wikipedia.org/wiki/Estructura_de_datos), análisis de los [casos peor, mejor, y promedio](https://es.wikipedia.org/wiki/Casos_peor,_mejor,_y_promedio), y límites inferiores.

Los algoritmos de ordenamiento se pueden clasificar en las siguientes maneras[2](https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento#cite_note-2)​:

* La más común es clasificar según el lugar donde se realice la ordenación
  + [Algoritmos de ordenamiento interno](https://es.wikipedia.org/w/index.php?title=Ordenamiento_interno&action=edit&redlink=1): en la [memoria](https://es.wikipedia.org/wiki/Memoria_de_ordenador) del [ordenador](https://es.wikipedia.org/wiki/Ordenador).
  + [Algoritmos de ordenamiento externo](https://es.wikipedia.org/wiki/Ordenamiento_externo): en un lugar externo como un [disco duro](https://es.wikipedia.org/wiki/Disco_duro).
* Por el tiempo que tardan en realizar la ordenación, dadas entradas ya ordenadas o inversamente ordenadas:

3 **ORDENAMIENTO**

Analizar los siguientes métodos de ordenamiento:

a**.**  El **método de ordenación por burbuja** se basa en comparaciones sucesivas de dos elementos consecutivos y realizar un intercambio entre los elementos hasta que queden ordenados.

* **Método de ordenación de burbuja mejorada**

Sin embargo el algoritmo de burbuja analizado realiza todas las comparaciones hasta ( n-1)

Se puede detectar que el vector se ordenado e interrumpir las operaciones, de esta forma mejoramos el rendimiento del algoritmo. Cuando se realiza una ordenación **Si en todas las operaciones no se a realizado un intercambio de elementos es debido a que el vector se encuentra ordenado** se puede utilizar un indicador, bandera o variable booleana para detectar el fin de la ordenación

 También es conocido como el **método del intercambio directo**. Dado que solo usa comparaciones para operar elementos, se lo considera un algoritmo de comparación, siendo uno de los más sencillos de implementar.

Una nueva versión del método de la burbuja seria limitando el número de comparaciones, dijimos que era inútil que se compare consigo misma. Si tenemos una lista de 10.000 elementos, entonces son 10.000comparaciones que están sobrando Imaginemos si tenemos

1.000.000 de elementos. El método sería mucho más optimo con “n” comparaciones menos (n = total de elementos

b. Inserción o método de la baraja

* El **ordenamiento por inserción** (**insertion sort** en [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s)) es una manera muy natural de ordenar para un ser humano, y puede usarse fácilmente para ordenar un mazo de cartas numeradas en forma arbitraria. Requiere {\displaystyle O(n^{2})}operaciones para ordenar una lista de {\displaystyle n}elementos.

Inicialmente se tiene un solo elemento, que obviamente es un conjunto ordenado. Después cuando hay {\displaystyle k}elementos ordenados de menor a mayor, se toma el elemento {\displaystyle k+1} y se compara con todos los elementos ya ordenados, deteniéndose cuando se encuentra un elemento menor (todos los elementos mayores han sido desplazados una posición a la derecha) o cuando ya no se encuentran elementos (todos los elementos fueron desplazados y este es el más pequeño). En este punto se *inserta* el elemento {\displaystyle k+1}debiendo desplazarse los demás elementos.

c. Selección o método sencillo

* El **método de ordenamiento por selección** consiste en encontrar el menor de todos los elementos del arreglo e intercambiarlo con el que está en la primera posición. Luego el segundo mas pequeño, y así sucesivamente hasta ordenar todo el arreglo.

Su funcionamiento es el siguiente:

* 1. Buscar el mínimo elemento de la lista
  2. Intercambiarlo con el primero
  3. Buscar el siguiente mínimo en el resto de la lista
  4. Intercambiarlo con el segundo
  5. Y en general:
  6. Buscar el mínimo elemento entre una posición *i* y el final de la lista
  7. Intercambiar el mínimo con el elemento de la posición *i*

d. Rápido o QuickSort

* **QuickSort** (en inglés, ordenamiento rápido). Es un [algoritmo](https://www.ecured.cu/Algoritmo) basado en la técnica de divide y vencerás, que permite, en promedio, ordenar n elementos en un tiempo proporcional a n log n.

El algoritmo consta de los siguientes pasos:

1. Elegir un elemento de la lista de elementos a ordenar, al que llamaremos [pivote](https://www.ecured.cu/index.php?title=Pivote&action=edit&redlink=1).
2. Resituar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. Los elementos iguales al pivote pueden ser colocados tanto a su derecha como a su izquierda, dependiendo de la implementación deseada. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.
3. La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.
4. Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.

.

3 ( e ) **Método de ordenamiento por Mezclar o MargeSort**

El algoritmo de ordenamiento por mezcla (merge sort en inglés) es un algoritmo de ordenamiento externo estable basado en la técnica divide y vencerás. Es un ordenamiento con una complejidad computacional logarítmica O(nlog{n})O(nlogn), que permite ordenar un listado de elementos de manera muy rápida.

La idea de los algoritmos de ordenación por mezcla es dividir la matriz por la mitad una y otra vez hasta que cada pieza tenga solo un elemento de longitud. Luego esos elementos se vuelven a juntar (mezclados) en orden de clasificación.

Comenzamos dividiendo la matriz:

**[31,4,88,2,4,2,42]**

**[31,4,88,1][4,2,42] *Dividimos en 2 partes***

**[31,4][88,1][4,2][42] *Dividimos en 4 partes***

**[31][4][88][1][4][2][42] *Piezas individuales***

Ahora tenemos que unirlos de nuevo en orden de mezcla:

Primero fusianamos elementos individuales en pares. Cada par se fusiona en orden de mezcla.

**[4,31][1,88][2,4][42]**

Luego fusionamos los pares en orden de mezcla:

**[1,4,31,88][2,4,42]**

Y luego fusionamos los dos últimos grupos.

**[1,2,4,4,31,42,88]**