**Quicksort**

El ordenamiento rápido (quicksort en inglés) es un algoritmo de ordenación creado por el científico británico en computación C. A. R. Hoare.

## **Descripción del algoritmo**

El algoritmo trabaja de la siguiente forma:

* Elegir un elemento del conjunto de elementos a ordenar, al que llamaremos pivote.
* 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.
* 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.
* 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.

Como se puede suponer, la eficiencia del algoritmo depende de la posición en la que termine el pivote elegido.

* En el mejor caso, el pivote termina en el centro de la lista, dividiéndola en dos sublistas de igual tamaño. En este caso, el orden de complejidad del algoritmo es O(n·log n).
* En el peor caso, el pivote termina en un extremo de la lista. El orden de complejidad del algoritmo es entonces de O(n²). El peor caso dependerá de la implementación del algoritmo, aunque habitualmente ocurre en listas que se encuentran ordenadas, o casi ordenadas. Pero principalmente depende del pivote, si por ejemplo el algoritmo implementado toma como pivote siempre el primer elemento del array, y el array que le pasamos está ordenado, siempre va a generar a su izquierda un array vacío, lo que es ineficiente.
* En el caso promedio, el orden es O(n·log n).

No es extraño, pues, que la mayoría de optimizaciones que se aplican al algoritmo se centren en la elección del pivote.

## **Técnicas de elección del pivote**

El algoritmo básico del método Quicksort consiste en tomar cualquier elemento de la lista al cual denominaremos como pivote, dependiendo de la partición en que se elija, el algoritmo será más o menos eficiente.

* Tomar un elemento cualquiera como pivote tiene la ventaja de no requerir ningún cálculo adicional, lo cual lo hace bastante rápido. Sin embargo, esta elección «a ciegas» siempre provoca que el algoritmo tenga un orden de O(n²) para ciertas permutaciones de los elementos en la lista.
* Otra opción puede ser recorrer la lista para saber de antemano qué elemento ocupará la posición central de la lista, para elegirlo como pivote. Esto puede hacerse en O(n) y asegura que hasta en el peor de los casos, el algoritmo sea O(n·log n). No obstante, el cálculo adicional rebaja bastante la eficiencia del algoritmo en el caso promedio.
* La opción a medio camino es tomar tres elementos de la lista - por ejemplo, el primero, el segundo, y el último - y compararlos, eligiendo el valor del medio como pivote.

# Bubble Sort

El ordenamiento de burbuja (Bubble Sort en inglés) es un sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada. Este algoritmo obtiene su nombre de la forma con la que suben por la lista los elementos durante los intercambios, como si fueran pequeñas "burbujas". 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.

## **Descripción**

Una manera simple de expresar el ordenamiento de burbuja en [pseudocódigo](https://es.wikipedia.org/wiki/Pseudoc%C3%B3digo) es la siguiente:

Este algoritmo realiza el ordenamiento o reordenamiento de una lista **a** de **n** valores, en este caso de **n** términos numerados del **0** al **n-1**; consta de dos bucles anidados, uno con el índice **i**, que da un tamaño menor al recorrido de la burbuja en sentido inverso de **2** a **n**, y un segundo bucle con el índice **j**, con un recorrido desde **0** hasta **n-i**, para cada iteración del primer bucle, que indica el lugar de la burbuja.

La burbuja son dos términos de la lista seguidos, **j** y **j+1**, que se comparan: si el primero es mayor que el segundo sus valores se intercambian.

Esta comparación se repite en el centro de los dos bucles, dando lugar a una lista ordenada. Puede verse que el número de repeticiones solo depende de **n** y no del orden de los términos, esto es, si pasamos al algoritmo una lista ya ordenada, realizará todas las comparaciones exactamente igual que para una lista no ordenada. Esta es una característica de este algoritmo. Luego veremos una variante que evita este inconveniente.

## **En la práctica**

A pesar de que el ordenamiento de burbuja es uno de los algoritmos más sencillos de implementar, su orden O (n2) lo hace muy ineficiente para usar en listas que tengan más que un número reducido de elementos. Incluso entre los algoritmos de ordenamiento de orden O (n2), otros procedimientos como el ordenamiento por inserción son considerados más eficientes.

Dada su simplicidad, el ordenamiento de burbuja es utilizado para introducir el concepto de algoritmo de ordenamiento para estudiantes de ciencias de la computación. A pesar de esto, algunos investigadores como Owen Astrachan han criticado su popularidad en la enseñanza de ciencias de la computación, llegando a recomendar su eliminación de los planes de estudio.1​

Sumado a esto, Jargon File, un libro ampliamente citado en la cultura hacker, lo denomina "el mal algoritmo genérico", y Donald Knuth, uno de los mayores expertos en ciencias de la computación, afirma que el ordenamiento de burbuja "no parece tener nada para recomendar su uso, a excepción de un nombre pegadizo y el hecho de que conlleva a problemas teóricos interesantes".2​

El ordenamiento de burbuja es asintóticamente equivalente en tiempos de ejecución con el ordenamiento por inserción en el peor de los casos, pero ambos algoritmos difieren principalmente en la cantidad de intercambios que son necesarios. Resultados experimentales como los descubiertos por Astrachan han demostrado que el ordenamiento por inserción funciona considerablemente mejor incluso con listas aleatorias. Por esta razón, muchos libros de algoritmos modernos evitan usar el ordenamiento de burbuja, reemplazándolo por el ordenamiento por inserción.

El ordenamiento de burbuja interactúa vagamente con el hardware de las CPU modernas. Requiere al menos el doble de escrituras que el ordenamiento por inserción, el doble de pérdidas de caché, y asintóticamente más predicción de saltos. Varios experimentos de ordenamiento de cadenas en Java hechos por Astrachan muestran que el ordenamiento de burbuja es 5 veces más lento que el ordenamiento por inserción, y 40% más lento que el ordenamiento por selección.1​