

# Ordenamiento de Burbuja

Asier Amigorena Díez  
79049057N  
24/11/2025

## REFERENCIAS

- [1] Owen Astrachan, "Bubble sort: An archaeological algorithmic analysis." [Online]. Available: <https://users.cs.duke.edu/~ola/bubble/bubble.html>
- [2] GeeksforGeeks, "C program for bubble sort." [Online]. Available: <https://www.geeksforgeeks.org/c/c-bubble-sort/>
- [3] Jay Wengrow, *A Common-Sense Guide to Data Structures and Algorithms, Second Edition*. La Vergne: Pragmatic Programmers, LLC, The, 2020.
- [4] GeeksforGeeks, "Halstead's software metrics - software engineering." [Online]. Available: <https://www.geeksforgeeks.org/software-engineering/software-engineering-halsteads-software-metrics/>
- [5] Schneider Electric, "Métrica: Halstead complejidad." [Online]. Available: <https://product-help.schneider-electric.com/Machine%20Expert/V1.2/es/CodeAnly/CodeAnly/Metrics/Metrics-9.htm>

## I. L<sup>A</sup>T<sub>E</sub>X

El sistema utilizado ha sido Overleaf, se trata de una herramienta basada en TeXLive. El compilador con el que se ha trabajado para elaborar este archivo de L<sup>A</sup>T<sub>E</sub>X ha sido pdfLaTeX. En caso de tener ya instalados los paquetes nombrados en 1 que se incluyen en el paquete 'texlive-full' no sería necesario realizar la parte de la instalación de paquetes. Acto seguido se compila utilizando pdflatex para generar el archivo .aux, mediante bibtex se preparan las citas con el fichero .aux como destino, y por último se compila 2 veces más para que aparezcan correctamente todas las citas.

Código 1. Cómo compilar usando pdflatex

```
sudo apt install texlive texlive-latex-extra texlive-publishers texlive-lang-spanish
pdflatex BubbleSort.tex
bibtex BubbleSort
pdflatex BubbleSort.tex
pdflatex BubbleSort.tex
```

## II. HISTORIA Y JUSTIFICACIÓN DEL DISEÑO

Una de las primeras referencias que se hacen a este algoritmo de ordenamiento es la realizada por Edward Friend en 1956. Pero no se menciona el término 'ordenamiento de burbuja', sino 'ordenamiento por intercambio' ('*sorting by exchange*'). No es hasta el año 1962 cuando, en un artículo (presentado en 1959 y revisado en 1961), Kenneth Eugene Iverson hace mención al término '*Bubble Sort*'. A pesar de que previamente se presentaron artículos que incluían un algoritmo parecido al de burbuja, es la primera vez que se utiliza el término. [1]

## III. FUNCIONAMIENTO

### III-A. Explicación

Este método de ordenamiento consiste en comparar valores adyacentes e intercambiarlos en caso de que no estén en el orden correcto. Esto hace que los valores más pequeños se desplacen hacia el comienzo de la lista, mientras que los más grandes 'floten' hacia el final, lo que se asemeja al comportamiento de burbujas. [2] [3]

### III-B. Código

Este algoritmo no es conocido por su gran rendimiento, sino por su sencillez a la hora de implementarlo en un lenguaje de programación convencional

Código 2. Bubble Sort en C

```
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

## IV. EFICIENCIA

Este algoritmo tiene 2 fases, una de comparación y otra de intercambio. En la fase inicial se compara si los valores deben ser cambiados; esto se realiza con todos los valores de la lista hasta que el más alto se encuentra al final. Una vez finalizada la primera iteración, se hace lo mismo omitiendo el último elemento que ya ha sido ordenado, lo que sería  $\sum_{i=1}^{N-1} (N-i)$  siendo N el número de valores en la lista. Durante la fase relativa a los cambios de posición, puede ser que no sea necesario que sean realizados. En el peor caso se tienen que realizar N cambios. Si por ejemplo tuviéramos 5 valores a comparar, se realizarían 10 comparaciones y en caso de tener que cambiar todas las posiciones tendríamos un cambio por operación, es decir 20 operaciones en total. Si incrementamos el número de posiciones de la lista a comparar, como se puede ver en la tabla I, el número de operaciones aumenta casi en  $N^2$  por lo que se diría que tiene una complejidad de  $O(N^2)$  [3] 1

# BUBBLE SORTING

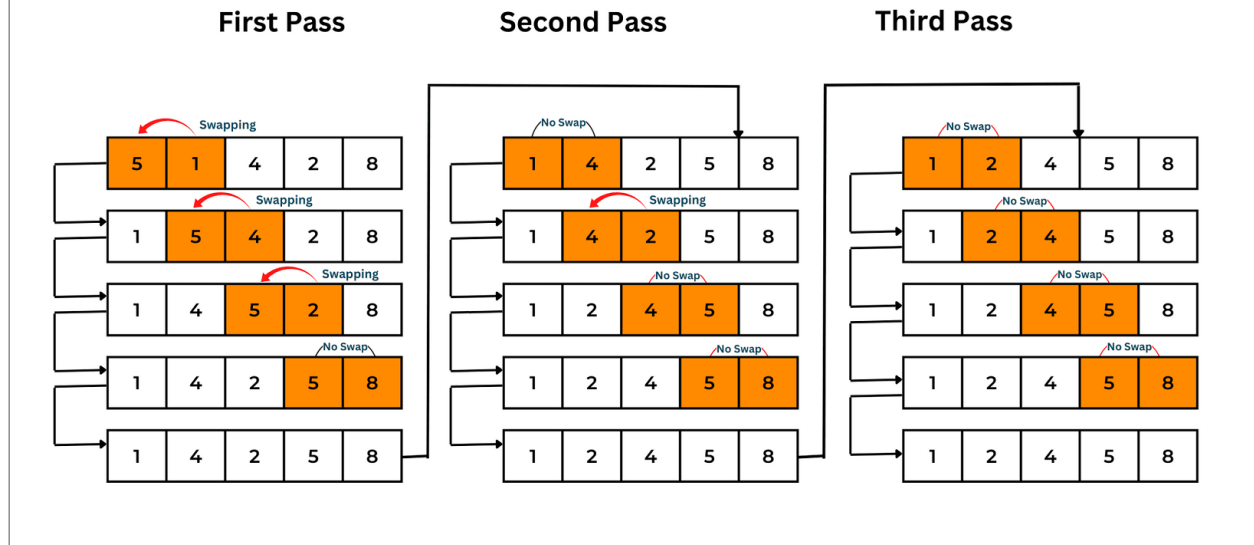


Figura 1. Explicación visual del funcionamiento del ordenamiento de burbuja

Número de valores	Número de operaciones	$N^2$
5	20	25
10	90	100
20	380	400

TABLA I  
OPERACIONES DE BUBBLE SORT

Algoritmo	Dificultad	Esfuerzo
Burbuja	17,25	4165
Selección	15,95	4242
Inserción	23,20	6652
Rápido	7,88	3157

TABLA II  
MÉTRICA DE HALSTEAD

## V. OTROS ALGORITMOS CONOCIDOS

El algoritmo de ordenamiento de burbuja no es, ni mucho menos, el único utilizado ya que existen múltiples soluciones a la hora de reestructurar una lista. A continuación se muestra una lista con algunos de los más conocidos.

### V-A. Lista de algunos algoritmos conocidos

- Ordenamiento por selección ('Selection Sort')
- Ordenamiento por inserción ('Insertion Sort')
- Ordenamiento rápido ('Quick Sort')
- Ordenamiento por mezcla ('Merge Sort')
- Ordenamiento por montículos ('Heap Sort')

## VI. COMPARACIONES CON OTROS ALGORITMOS

La manera óptima de comparar el rendimiento de un método de estas características es mediante la métrica de Halstead

### VI-A. Métrica de Halstead

La métrica se utiliza para medir la complejidad de un programa sin necesidad de ejecutarlo. Como paso inicial se desglosa el método en tokens, estos tokens serán los operadores ( $n_1$  el número de operadores distintos y  $N_1$  el número total de operadores) y operandos ( $n_2$  el número de

operandos distintos y  $N_2$  el número total de operandos). Con estos datos se pueden representar distintos aspectos. [4] [5]

- Dificultad  $D = (n_1/2) * (N_2/n_2)$
- Longitud  $L = N_1 + N_2$
- Longitud Calculada  $N_x = n_1 * \log_2(n_1) + n_2 * \log_2(n_2)$
- Volumen  $V = N \log_2(n)$
- Esfuerzo  $E = V * D$
- Vocabulario  $n = n_1 + n_2$

Para realizar la comparación, únicamente serán empleadas las métricas de esfuerzo (E) y dificultad (D).II [1]

## VII. CONCLUSIONES

Tras repasar el funcionamiento del ordenamiento de burbuja, así como su rendimiento y comparación con otros algoritmos, podemos aceptar lo que dijo Donald Knuth, 'El ordenamiento por burbuja no parece tener nada para recomendarlo' ('bubble sort seems to have nothing to recommend it') ya que pese a ser conocido por ser un método eficiente en cuanto a dificultad para ser programado, existen algoritmos más eficientes por ese aspecto. [1] Lo que lo hizo eficiente en líneas de código, ha supuesto que será ineficiente para el tratamiento de datos como se ha podido ver con su complejidad de  $O(N^2)$  I