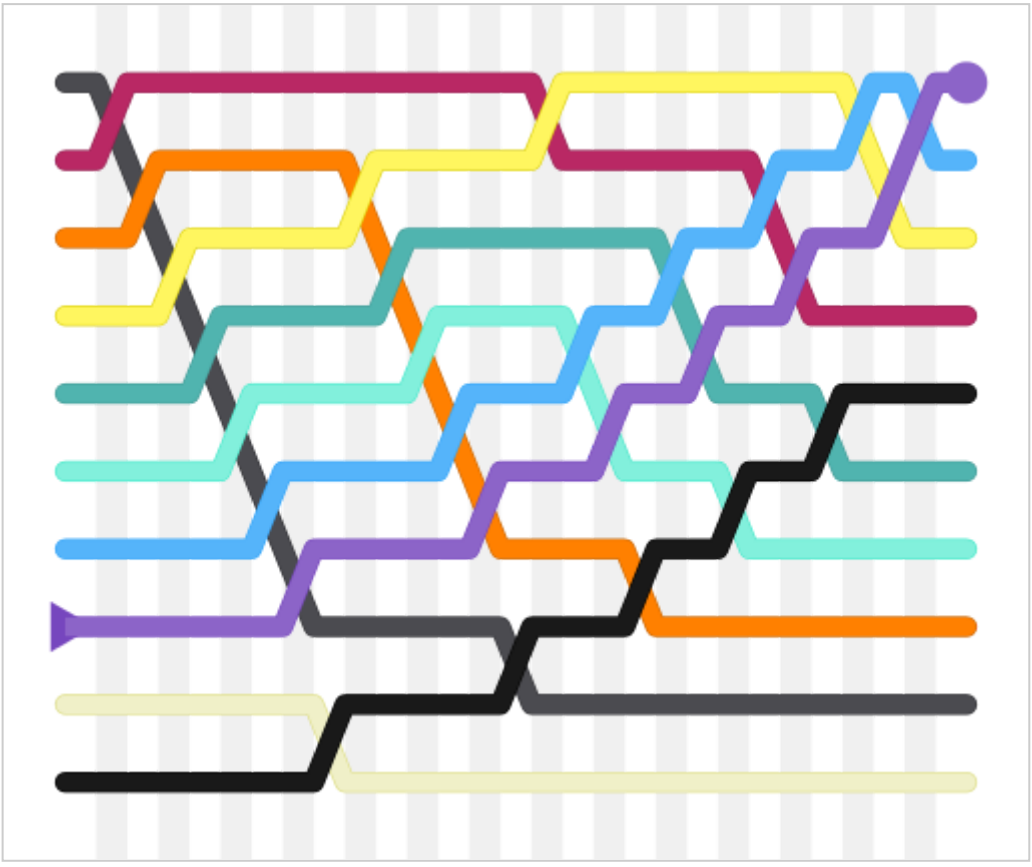
Ordenamiento de burbuja

De Wikipedia, la enciclopedia libre

La **Ordenación 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 el más sencillo de implementar.

Este algoritmo es esencialmente un algoritmo de fuerza bruta lógica.

Representación animada de ordenación de un conjunto de números mediante el algoritmo burbuja. Comenzando desde el inicio del vector, se compara cada par de elementos adyacentes. Si ambos no están ordenados (el segundo es menor que el primero), se intercambian sus posiciones. En cada iteración, un elemento menos necesita ser evaluados (el último), ya que no hay más elementos a su derecha que necesiten ser comparados, puesto que ya están ordenados.



Ordenamiento de burbuja

Índice

Descripción

Una manera simple de expresar el ordenamiento de burbuja en pseudocódigo es la siguiente:

```
::::< math > si a(j) > a(j+1) entonces
                aux ← a(j)
                a(j) ← a(j+1)
                a(j+1) ← aux
            fin si
        fin para
    fin para
fin procedimiento
```

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 la postre 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.

Para comprender el funcionamiento, veamos un ejemplo sencillo:

Tenemos una lista de números que hay que ordenar:

$$a = \{55, 86, 48, 16, 82\}$$

Podemos ver que la lista que tiene cinco términos, luego:

$$n = 5$$

$$a_4 = 82$$
$$a_3 = 16$$
$$a_2 = 48$$
$$a_1 = 86$$
$$a_0 = 55$$

El índice **i** hará un recorrido de **2** hasta **n**:

para $i \leftarrow 2$ *hasta* n *hacer*

que en este caso será de 2 a 5. Para cada uno de los valores de **i**, **j** tomará sucesivamente los valores de **0** hasta **n-i**:

para $j \leftarrow 0$ *hasta* $n - i$ *hacer*

Para cada valor de **j**, obtenido en ese orden, se compara el valor del índice **j** con el siguiente:

si $a_{(j)} > a_{(j+1)}$ *entonces*

Si el término **j** es mayor que el término **j+1**, los valores se permutan, en caso contrario se continúa con la iteración.

Para el caso del ejemplo, tenemos que:

$n = 5$

Para la primera iteración del primer bucle:

$i = 2$

y **j** tomará los valores de **0** hasta **3**:

para $j \leftarrow$ *0 hasta 3 hacer*

	$j=0$	$j=1$	$j=2$	$j=3$	
a_4	82	82	82	$\rightarrow$ 82	86
a_3	16	16	$\rightarrow$ 16	$\rightarrow$ 86	82
a_2	48	$\rightarrow$ 48	$\rightarrow$ 86	16	16
a_1	$\rightarrow$ 86	$\rightarrow$ 86	48	48	48
a_0	$\rightarrow$ 55	55	55	55	55

Cuando **j** vale **0**, se comparan a_0 a_1, el 55 y el 86, dado que 55 < 86, no se permuta el orden.

Ahora **j** vale **1** y se comparan a_1 a_2 el 86 y el 48. Como 86 > 48, se permutan, dando lugar a una nueva lista.

Se repite el proceso hasta que **j** valga **3**, dando lugar a una lista parcialmente ordenada. Podemos ver que el término de mayor valor está en el lugar más alto.

Ahora **i** vale **3**, y **j** hará un recorrido de **0** a **2**.

Primero **j** vale **0**, se comparan a_0 a_1, el 55 y el 48. Como 55 > 48 se permutan dando lugar a la nueva lista.

Para j = 1 se compara el 55 con el 16 y se cambian de orden.

Para j = 2 se compara el 55 y el 82 y se dejan como están, finalizando el bucle con una lista mejor ordenada. Puede verse que los dos valores más altos ya ocupan su lugar. No se ha realizado ninguna comparación con el término cuarto, dado que ya se sabe que después del primer ciclo es el mayor de la lista.

	$j=0$	$j=1$	$j=2$	
a_4	86	86	86	86
a_3	82	82	$\rightarrow$ 82	82
a_2	16	$\rightarrow$ 16	$\rightarrow$ 55	55
a_1	$\rightarrow$ 48	$\rightarrow$ 55	16	16
a_0	$\rightarrow$ 55	48	48	48

El algoritmo consiste en comparaciones sucesivas de dos términos consecutivos ascendiendo de abajo a arriba en cada iteración, como la ascensión de las burbujas de aire en el agua, de ahí el nombre del procedimiento. En la primera iteración el recorrido ha sido completo, en el segundo se ha dejado él último término, al tener ya el mayor de los valores, en los sucesivos sé ira dejando de realizar las últimas comparaciones, como se puede ver.

Ahora ya **i** vale 4 y **j** recorrerá los valores de 0 a 1.

Cuando **j** vale 0, se comparan a_0 a_1, esto es, el 48 y el 16. Dado que 48 es mayor que 16 se permutan los valores, dando lugar a una lista algo más ordenada que la anterior. Desde esta nueva ordenación, **j** pasa a valer 1, con lo que se comparan los términos a_1 a_2 el 48 y el 55 que quedan en el mismo orden.

En este caso la burbuja ha ascendido menos que en los casos anteriores, y la lista está ya ordenada, pero el algoritmo tendrá que completarse, realizando una última iteración.

	$j=0$	$j=1$	
a_4	86	86	86
a_3	82	82	82
a_2	55	$\rightarrow$ 55	55
a_1	$\rightarrow$ 16	$\rightarrow$ 48	48
a_0	$\rightarrow$ 48	16	16

Hay que tener en cuenta que el bucle realiza un número fijo de repeticiones y para finalizar tendrán que completarse, aun en el caso extremo, de que la lista estuviera previamente ordenada.

Por último **i** vale 5 y **j** solo puede vale 0, con lo que sólo se realizará una comparación de a_0 a_1 el 16 y el 48, que ya están ordenados y se dejan igual.

Los bucles finalizan y también el procedimiento, dejando la lista ordenada.

Una variante que finaliza en caso de que la lista esté ordenada, puede ser la siguiente: como en el ejemplo anterior, empleando un centinela **ordenado**, que detecta que no se ha modificado la lista en un recorrido de la burbuja, y que por tanto la lista ya está ordenada, finalizando inmediatamente.

	$j=0$	
a_4	86	86
a_3	82	82
a_2	55	55
a_1	$\rightarrow$ 48	48
a_0	$\rightarrow$ 16	16

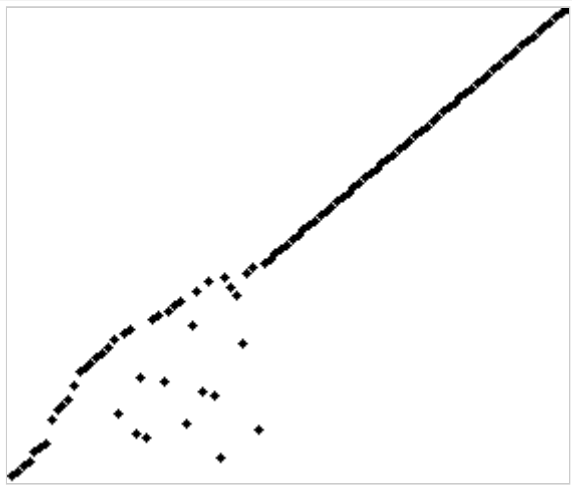
```
procedimiento DeLaBurbuja2 (a(0), a(1), a(2), ..., a(n-1))

    i ← 1
    ordenado ← no
    mientras (i < n) ∧ (ordenado = no) hacer

        i ← i + 1
        ordenado ← si
        para j ← 0 hasta n − i hacer

            si a(j) > a(j+1) entonces

                ordenado ← no
                aux ← a(j)
                a(j) ← a(j+1)
                a(j+1) ← aux

            fin si

        fin para

    fin mientras

fin procedimiento
```

```
procedimiento DeLaBurbuja3 (a(0), a(1), a(2), ..., a(n−1))
    i ← 1
    repetir
        i ← i + 1
        ordenado ← si
        para j ← 0 hasta n − i hacer
            si a(j) > a(j+1) entonces
                ordenado ← no
                aux ← a(j)
                a(j) ← a(j+1)
                a(j+1) ← aux
            fin si
        fin para
    hasta que ¬(i < n) ∨ (ordenado = si)
fin procedimiento
```

1.

Análisis

Rendimiento del algoritmo



Ejemplo del ordenamiento de burbuja ordenando una lista de números aleatorios.

Al algoritmo de la burbuja, para ordenar un vector de n términos, tiene que realizar siempre el mismo número de comparaciones:

$$c(n) = \frac{n^2 - n}{2}$$

Esto es, el número de comparaciones **c(n)** no depende del orden de los términos, si no del número de términos:

$$\Theta(c(n)) = n^2$$

Por lo tanto la cota ajustada asintótica del número de comparaciones pertenece al orden de **n** cuadrado.

El número de intercambios **i(n)**, que hay que realizar depende del orden de los términos y podemos diferenciar, el caso mejor, si el vector está previamente ordenado, y el caso peor, si el vector está ordenado en orden inverso:

$$\Theta(i(n)) = ?$$

Por lo que no se puede determinar una cota ajustada asintótica del número de intercambios, dado que éste dependerá del orden del vector en cuestión.

Rendimiento en el caso desfavorable

Si pasamos al algoritmo un vector ordenado en orden inverso realizará un número de comparaciones:

$$c(n) = \frac{n^2 - n}{2}$$

Como ya hemos dicho anteriormente, y tendrá que realizar un número igual de intercambios entre los términos del vector, dado que en cada comparación los términos estarán desordenados, y se realizará el intercambio.

$$i(n) = \frac{n^2 - n}{2}$$

Por lo tanto en el caso más desfavorable tanto el número de comparaciones como el de intercambios coinciden:

$$O(c(n)) = O(i(n)) = n^2$$

El número de comparaciones o de intercambios en el caso más desfavorable pertenece al orden de **n** cuadrado.

Rendimiento en casos óptimos

En el caso óptimo, el más favorable, es la ordenación de un vector ya ordenado. En este caso el número de comparaciones será el mismo que en cualquier otro caso:

$$\Omega(c(n)) = n^2$$

La cota inferior asintótica del número de comparaciones pertenece al orden de **n** cuadrado, como en los demás casos, pero en todas las comparaciones el orden es el correcto y por tanto no se realiza ningún intercambio:

$$i(n) = 0$$

Por lo tanto el coste de intercambios no depende de **n**, y es constante:

$$\Omega(i(n)) = 1$$

El ordenamiento de burbuja tiene una complejidad **Ω**(n^2) igual que ordenamiento por selección. Cuando una lista ya está ordenada, a diferencia del ordenamiento por inserción que pasará por la lista una vez y encontrará que no hay necesidad de intercambiar las posiciones de los elementos, el método de ordenación por burbuja está forzado a pasar por dichas comparaciones, lo que hace que su complejidad sea cuadrática en el mejor de los casos. Esto lo cataloga como el **algoritmo más ineficiente** que existe, aunque para muchos programadores sea el más sencillo de implementar.

Conejos y Tortugas

La posición de los elementos en el ordenamiento de burbuja juegan un papel muy importante en la determinación del rendimiento. Los elementos mayores al principio de la lista son rápidamente movidos hacia abajo, mientras los elementos menores en el fondo de la lista se mueven a la parte superior muy lentamente. Esto llevó a nombrar estos elementos *conejos* y *tortugas*, respectivamente.

Se han realizado varios esfuerzos para eliminar las tortugas *(véase Exterminación)* y mejorar la velocidad del ordenamiento de burbuja, la cual será más redonda que nunca. El Ordenamiento por sacudida es un buen ejemplo, aunque aún mantiene, en el peor de los casos, una complejidad *O* (n^2). El ordenamiento por

combinación compara los elementos primero en pedazos grandes de la lista, moviendo tortugas extremadamente rápido, antes de proceder a pedazos cada vez más pequeños para alisar la lista. Su velocidad promedio es comparable a algoritmos rápidos (y complejos) como el ordenamiento rápido.

En la práctica

A pesar de que el ordenamiento de burbuja es uno de los algoritmos más sencillos de implementar, su orden $O(n^2)$ 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 $O(n^2)$, 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]

Ejemplo paso a paso

Tomemos como ejemplo los números: "9 6 5 8 2 1", que serán ordenados de menor a mayor valor usando el método burbuja. Los elementos siguientes resaltados están siendo comparados.

Primera vuelta: (**9 6** 5 8 2 1) → (**6 9** 5 8 2 1), el algoritmo compara los primeros dos elementos y los cambia porque 9 > 6 (6 **9 5** 8 2 1) → (6 **5 9** 8 2 1) (6 5 **9 8** 2 1) → (6 5 **8 9** 2 1) (6 5 8 **9 2** 1) → (6 5 8 **2 9** 1) (6 5 8 2 **9 1**) → (6 5 8 2 **1 9**)

Segunda vuelta: (**6 5** 8 2 1 9) → (**5 6** 8 2 1 9) (5 **6 8** 2 1 9) → (5 **6 8** 2 1 9), como estos elementos ya están en orden, el algoritmo no hace cambios. (5 6 **8 2** 1 9) → (5 6 **2 8** 1 9) (5 6 2 **8 1** 9) → (5 6 2 **1 8** 9) (5 6 2 1 **8 9**) → (5 6 2 1 **8 9**)

Tercera vuelta: (**5 6** 2 1 8 9) → (**5 6** 2 1 8 9) (5 **6 2** 1 8 9) → (5 **2 6** 1 8 9) (5 2 **6 1** 8 9) → (5 2 **1 6** 8 9) (5 2 1 **6 8** 9) → (5 2 1 **6 8** 9) (5 2 1 6 **8 9**) → (5 2 1 6 **8 9**)

Cuarta vuelta: (**5 2** 1 6 8 9) → (**2 5** 1 6 8 9) (2 **5 1** 6 8 9) → (2 **1 5** 6 8 9) (2 1 **5 6** 8 9) → (2 1 **5 6** 8 9) (2 1 5 **6 8** 9) → (2 1 5 **6 8** 9) (2 1 5 6 **8 9**) → (2 1 5 6 **8 9**)

Quinta vuelta: (**2 1** 5 6 8 9) → (**1 2** 5 6 8 9) (1 **2 5** 6 8 9) → (1 **2 5** 6 8 9) (1 2 **5 6** 8 9) → (1 2 **5 6** 8 9) (1 2 5 **6 8** 9) → (1 2 5 **6 8** 9) (1 2 5 6 **8 9**) → (1 2 5 6 **8 9**)

Ejemplo en C++

Este es un ejemplo de un programa que captura números y los ordena de menor a mayor en lenguaje C++, compilado en Dev C++:

```
#include<iostream>

using namespace std;

main(){

int num,aux;

cout<<"Cuantos numeros seran: ";

cin>>num;

int arreglo[num];

cout<<endl<<"***CAPTURA DE NUMEROS***"<<endl;

for(int x=1;x<=num;x++){

cout<<"Ingresa el numero "<<x<<" de la serie: ";

cin>>arreglo[x];

cout<<endl;

}

cout<<"***MUESTRA DE NUMEROS***"<<endl;

for(int y=1;y<=num;y++){

cout<<"Numero "<<y<<".- "<<arreglo[y]<<endl;

}

for(int z=1;z<=num;z++){

for(int v=0;v<num;v++){

if(arreglo[v]>arreglo[v+1]){

aux = arreglo[v];

arreglo[v] = arreglo [v+1];

arreglo[v+1] = aux;

}

}
```

```
}

cout<<"***NUMEROS ARREGLADOS***"<<endl;

for(int w=1;w<=num;w++){

cout<<"Numero "<<w<<".- "<<arreglo[w]<<endl;

}

}
```

Referencias

1. Astrachan, Owen (2003). «Ordenamiento de burbuja: Un anális arqueológico de un algoritmo» (http://www.cs.duke.edu/~ola/papers/bubble.pdf). *SIGCSE* (en inglés). Consultado el 9 de marzo de 2011.
2. Knuth, Donald (1998). «5.2.2: Ordenamiento por intercambio». *El arte de programar ordenadores, Volumen 3* (en inglés) (segunda edición). Addison-Wesley. pp. 106-110. ISBN 0-201-89685-0.

Véase también

- Algoritmo de ordenamiento

Enlaces externos

- Implementaciones del algoritmo en Wikibooks (inglés) (https://en.wikibooks.org/wiki/Algorithm_implementation/Sorting/Bubble_sort)
- Distintas implementaciones del algoritmo en RosettaCode.org (http://rosettacode.org/wiki/Sorting_algorithms/Bubble_sort)