ORDENAMIENTOS

EL ORDENAMIENTO DE ELEMENTOS PUEDE SER:

Ordenación Interna.- En memoria principal (arrays, listas).

Ordenación Externa.- En memoria secundaria, dispositivos de almacenamiento externo (archivos y bases de datos).

TIPOS DE ORDENAMIENTOS:

Los más usuales son:

Directos:

- **POR BURBUJEO (Compara e intercambia elementos)**
- **POR SELECCIÓN (Selecciona el más pequeño y lo intercambia)**
- **POR INSERCIÓN (Inserta los elementos en una sublista ordenada)**

Indirectos:

- **ORDENACIÓN RÁPIDA (o Quick Sort - divide una lista en dos partes)**

Un aspecto a considerar es la diferencia entre el peor y el mejor caso. Cada algoritmo se comporta de modo diferente de acuerdo a cómo se le entregue la información; por eso es conveniente estudiar su comportamiento en casos extremos, como cuando los datos están prácticamente ordenados o muy desordenados.

COMPLEJIDAD:

La complejidad del algoritmo tiene que ver el rendimiento. Para ello tenemos que identificar una operación fundamental que realice nuestro algoritmo, que en este caso es **comparar**. Luego contamos cuántas veces el algoritmo necesita comparar. Si en una lista de **n** términos realiza **n** comparaciones la complejidad es O(n). Algunos ejemplos de complejidades comunes son:

- **O(1)** : Complejidad constante.
- **$O(n^2)$** : Complejidad cuadrática.
- **O(n log(n))** : Complejidad logarítmica.

Puede decirse que un algoritmo de complejidad O(n) es más rápido que uno de complejidad $O(n^2)$.

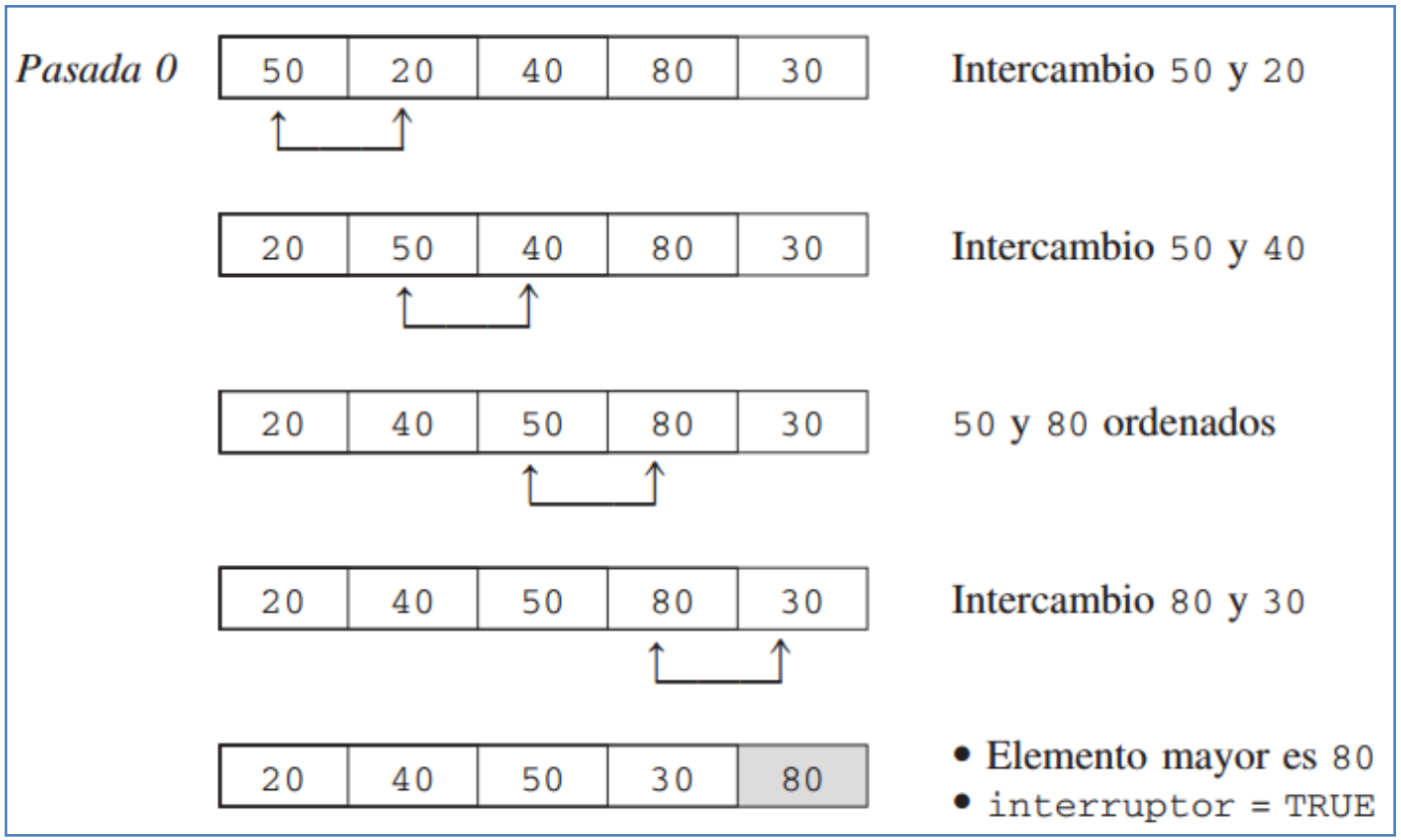
ORDENAMIENTO POR BURBUJEO

El **burbujeo** (bubble sort), funciona de la siguiente manera: Se va comparando cada elemento del array con el siguiente. Si un elemento es mayor que el que le sigue, entonces se intercambian. Esto produce que quede en el array como último elemento, el más grande.

Este proceso debe repetirse recorriendo todo el array hasta que no ocurra ningún intercambio. Los elementos que van quedando ordenados ya no se comparan.

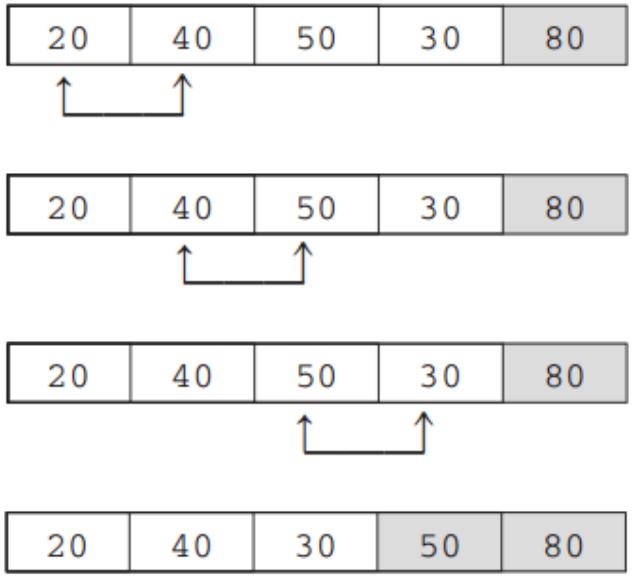
ORDENAMIENTO POR BURBUJEO MEJORADO

Constituye una mejora ya que el algoritmo termina inmediatamente cuando los datos ya están ordenados. Detecta que los datos ya están ordenados porque no se producen intercambios (bandera =0 al terminar el ciclo interno). Siendo $O(n^2)$ (Peor caso). Ejemplo:



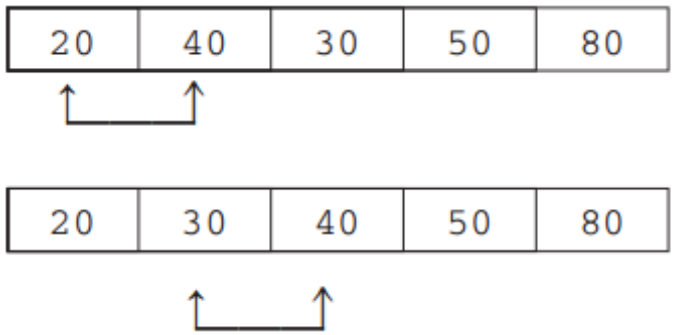
ORDENAMIENTO POR BURBUJEO MEJORADO

En la pasada 1:

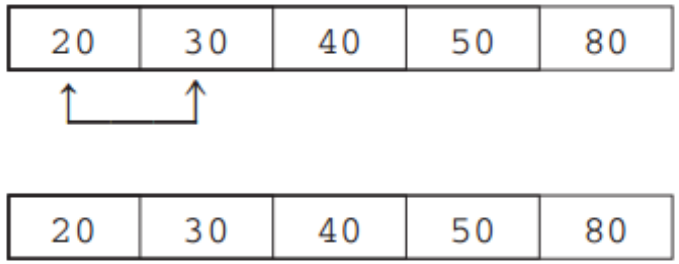


ORDENAMIENTO POR BURBUJEO MEJORADO

En la pasada 2, sólo se hacen dos comparaciones:



En la pasada 3, se hace una única comparación de 20 y 30, y no se produce intercambio:



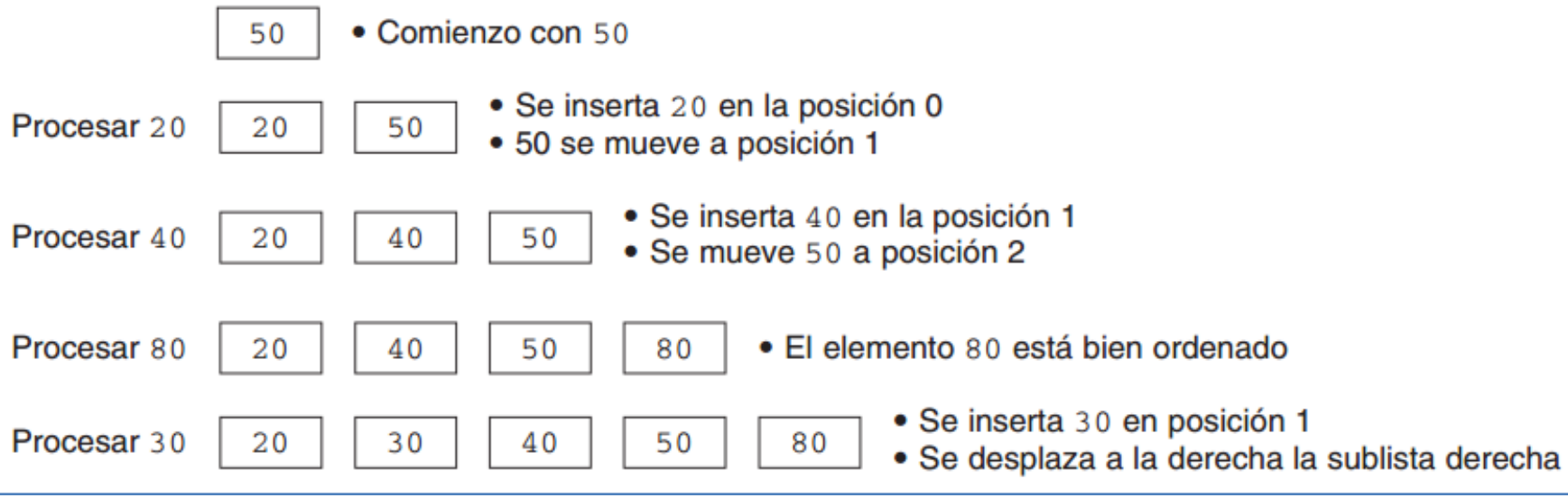
Animación: http://lwh.free.fr/pages/algo/tri/tri_bulle_es.html

ORDENAMIENTO POR INSERCIÓN

Inserción (insertion sort) es una manera muy natural de ordenar. Consiste en tomar uno por uno los elementos de un array y lo coloca en su posición con respecto a los anteriormente ordenados. Así empieza con el segundo elemento y lo ordena con respecto al primero. Luego sigue con el tercero y lo coloca en su posición ordenada con respecto a los dos anteriores, así sucesivamente hasta recorrer todas las posiciones del array.

ORDENAMIENTO POR INSERCIÓN

El método de ordenación por inserción es similar al proceso típico de ordenar tarjetas de nombres (cartas de una baraja) por orden alfabético, que consiste en insertar un nombre en su posición correcta dentro de una lista o archivo que ya está ordenado. Así el proceso en el caso de la lista de enteros `A = 50, 20, 40, 80, 30`.



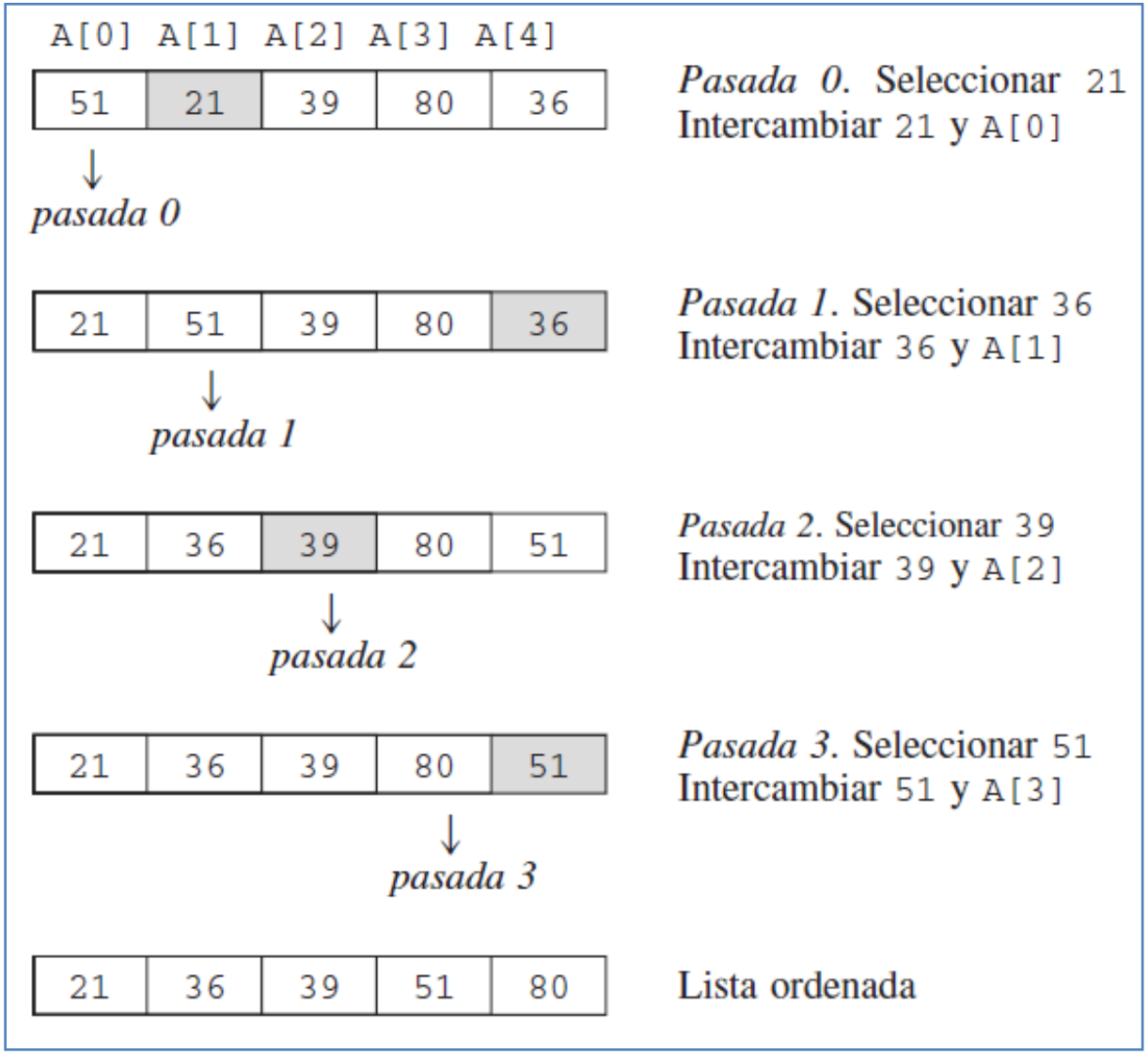
Animación: http://lwh.free.fr/pages/algo/tri/tri_insertion_es.html

ORDENAMIENTO POR SELECCIÓN

Selección (Selection sort) consiste en encontrar el menor de todos los elementos del array e intercambiarlo con el que está en la primera posición. Luego el segundo mas pequeño, y así sucesivamente hasta ordenarlo todo.

Su implementación requiere $O(n^2)$ comparaciones e intercambios para ordenar una secuencia de elementos.

ORDENAMIENTO POR SELECCIÓN



Animación: http://lwh.free.fr/pages/algo/tri/tri_selection_es.html

ORDENAMIENTO RÁPIDO (QUICKSORT)

- El método se basa en dividir los *n* elementos del array a ordenar en dos partes o particiones separadas por un elemento: una *partición izquierda*, un elemento *central* denominado *pivote* o elemento de partición, y una *partición derecha*.

- La partición o división se hace de tal forma que todos los elementos de la primera sublista (partición izquierda) son menores que todos los elementos de la segunda sublista (partición derecha).

- Las dos sublistas se ordenan entonces independientemente.

- Para dividir la lista en particiones (*sublistas*) se elige uno de los elementos de la lista y se utiliza como *pivote* o *elemento de partición*.

- Si se elige una lista cualquiera con los elementos en orden aleatorio, se puede seleccionar cualquier elemento de la lista como pivote, por ejemplo, el primer elemento de la lista.

ORDENAMIENTO RÁPIDO (QUICKSORT)

- Si la lista tiene algún orden parcial conocido, se puede tomar otra decisión para el pivote.
- Idealmente, el pivote se debe elegir de modo que se divida la lista exactamente por la mitad, de acuerdo al tamaño relativo de las claves. Por ejemplo, si se tiene una lista de enteros de 1 a 10, 5 o 6 serían pivotes ideales, mientras que 1 o 10 serían elecciones «pobres» de pivotes.
- Estas dos listas parciales se ordenan recursivamente utilizando el mismo algoritmo; es decir, se llama sucesivamente al propio algoritmo *quicksort*.
- La lista final ordenada se consigue concatenando la primera sublista, el pivote y la segunda lista, en ese orden, en una única lista.
- La primera etapa de *quicksort* es la división o «particionado» recursivo de la lista, hasta que todas las sublistas constan de sólo un elemento.

ORDENAMIENTO RÁPIDO (QUICKSORT)

1. *Lista original* | 5 | 2 | 1 | 7 | 9 | 3 | 8 | 7 |

pivote elegido | 5 |

↑

sublista izquierda1 (elementos menores que 5) | 2 | 1 | 3 |

sublista derecha1 (elementos mayores o iguales a 5) | 9 | 8 | 7 |

2. El algoritmo se aplica a la sublista izquierda

Sublista Izda1 2 1 3

↑ *pivote*

sublista Izda 1
sublista Dcha 3

Sublista Izda1 *Izda* *pivote* *Dcha*

| 1 | 2 | 3 |

ORDENAMIENTO RÁPIDO (QUICKSORT)

3. El algoritmo se aplica a la sublista derecha

Sublista Dcha1 9 8 7

sublista Izda 7 8

↑

pivote

sublista Dcha

Sublista Dcha1 *Izda* *pivote* *Dcha*

7	8	9

4. *Lista ordenada final*

Sublista izquierda *pivote* *Sublista derecha*

1 2 3 5 7 8 9

Animación: http://lwh.free.fr/pages/algo/tri/tri_rapide_es.html

Algoritmos de ordenamiento:

Internos:

- Inserción directa.
- Inserción binaria.
- Selección directa.
- Burbuja.
- Shake.
- Intercambio directo.
- Shell.
- Inserción disminución incremental.
- Heap.
- Tournament.
- Ordenamiento de árbol.
- Quick sort.
- Sort particionado.
- Merge sort.
- Radix sort.
- Cálculo de dirección.

Algoritmos de ordenamiento:

Externos:

- Straight merging.
- Natural merging.
- Balanced multiway merging.
- Polyphase sort.
- Distribution of initial runs.

FIN