Métodos de ordenamiento

La **ordenación** o **clasificación** de datos (*sort*) es una operación consistente en disponer un conjunto —estructura— de datos en algún orden determinado.

Un conjunto de datos (estructura) puede ser almacenado en un *archivo*, un *array*, un *array de registros*, una *lista enlazada* o un *árbol*. Cuando los datos están almacenados en un *array*, una lista enlazada o un árbol, se denomina **ordenación interna**. Si los datos están almacenados en un archivo, el proceso de ordenación se llama **ordenación externa**.

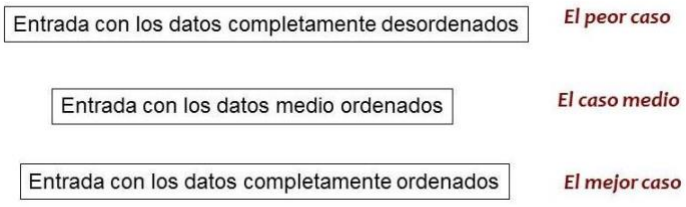
Los métodos de ordenación son numerosos y se basan en:

- La **comparación**: el algoritmo de ordenación tiene que ser capaz de disponer de una operación tal que, dados dos datos cualesquiera, la operación de comparación indique si están en orden o no.
- El **intercambio de datos**: el algoritmo debe ser capaz de intercambiar dos elementos de la lista.

Se dividen en dos grandes grupos:

- **Directos**: *burbuja, selección, inserción.*
- **Indirectos**: *Shell Sort, Merge Sort, Quick sort y Radix Sort.*

En todos estos casos, *T(n)* depende no sólo del tamaño de los datos de entrada, sino también de lo ordenados que éstos se encuentren.



El método de la burbuja

Es el más conocido, sencillo y fácil de implementar. El nombre **ordenamiento por burbuja** se deriva del hecho de que los valores más pequeños en el arreglo flotan o suben hacia la parte inicial (primeras posiciones) del arreglo, mientras que los valores más grandes caen

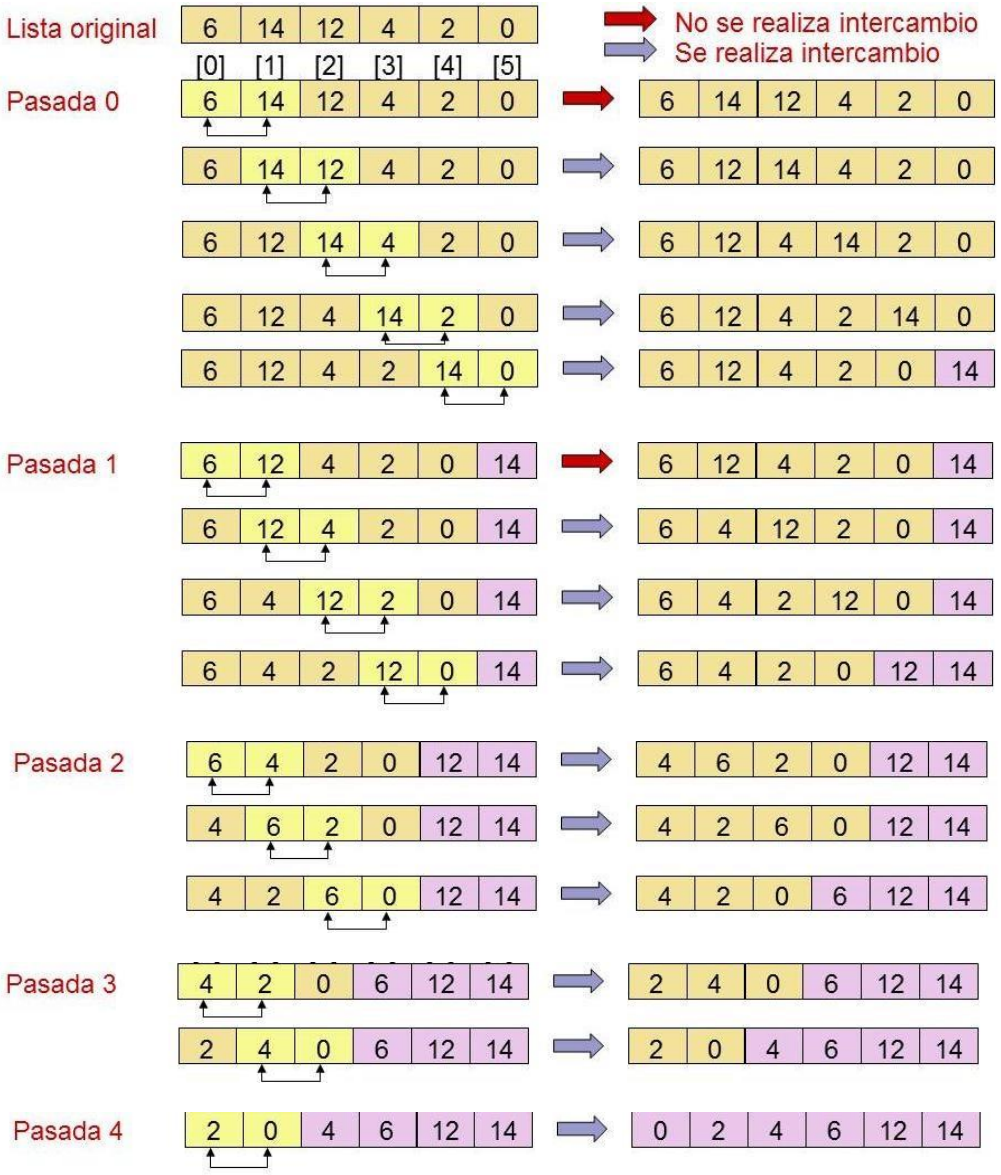
hacia la parte final (últimas posiciones) del arreglo. Si una pareja está en orden decreciente, se intercambian los valores. De lo contrario, quedan los valores iguales.

Para un *array* de *n* elementos, este método requiere n-1 pasadas:

Algoritmo Burbuja

Entrada: Vector V, Tamaño entrada n
Salida: Vector ordenado V

```
para i ← 0 hasta n − 1 hacer
    para j ← 0 hasta n − 1 hacer
        si V[j] > V[j + 1] entonces
            aux ← V[j + 1]
            V[j + 1] ← V[j]
            V[j] ← aux
        fin si
    fin para
fin para
```



```
Algoritmo Burbuja_Mejorado
Entrada: Vector V, Tamaño entrada n
Salida: Vector ordenado V
  hubo_cambio ← verdadero
  para i ← 0 hasta i < (n − 1) ∧ hubo_cambio hacer
    hubo_cambio ← falso
    para j ← 0 hasta j < (n − i) hacer
      si V[j] > V[j + 1] entonces
        hubo_cambio ← verdadero
        aux ← V[j + 1]
        V[j + 1] ← V[j]
        V[j] ← aux
      fin si
    fin para
  fin para
```

El orden de complejidad del método de la burbuja es $O(n^2)$. Este método es uno de los más sencillos; sin embargo, es de los más ineficientes, por lo que es de los menos utilizados.

Ventajas:

- Fácil implementación.
- No requiere memoria adicional.

Desventajas:

- Muy lento.
- Muchas comparaciones.
- Muchos intercambios.

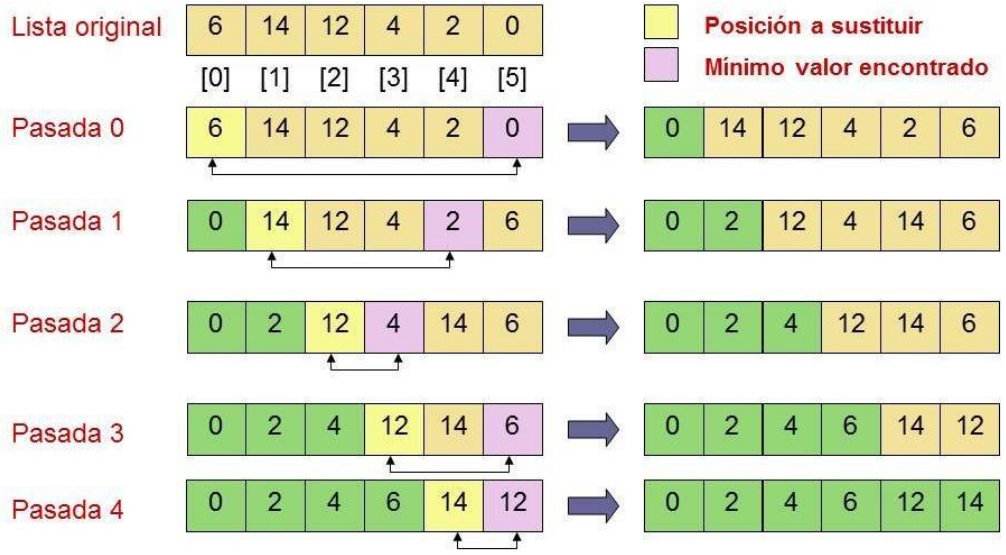
El método de selección

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, y así excluirlo de la lista. Luego, el segundo más pequeño; y así sucesivamente hasta ordenar todo el arreglo.

Algoritmo Seleccion

Entrada: Vector V, Tamaño entrada n
Salida: Vector ordenado V

```
min ← V[0]
index ← 0
para i ← 0 hasta i < n hacer
    para j ← i + 1 hasta j < n hacer
        si V[j] < min entonces
            min ← V[j]
            index ← i
        fin si
    fin para
    V[index] ← V[i]
    V[i] ← min
fin para
```



El algoritmo de selección es $O(n^2)$.

Ventajas:

- Fácil implementación.
- No requiere memoria adicional.
- Realiza pocos intercambios.
- Rendimiento constante: poca diferencia entre el peor y el mejor caso.

Desventajas:

- Lento.
- Realiza numerosas comparaciones.

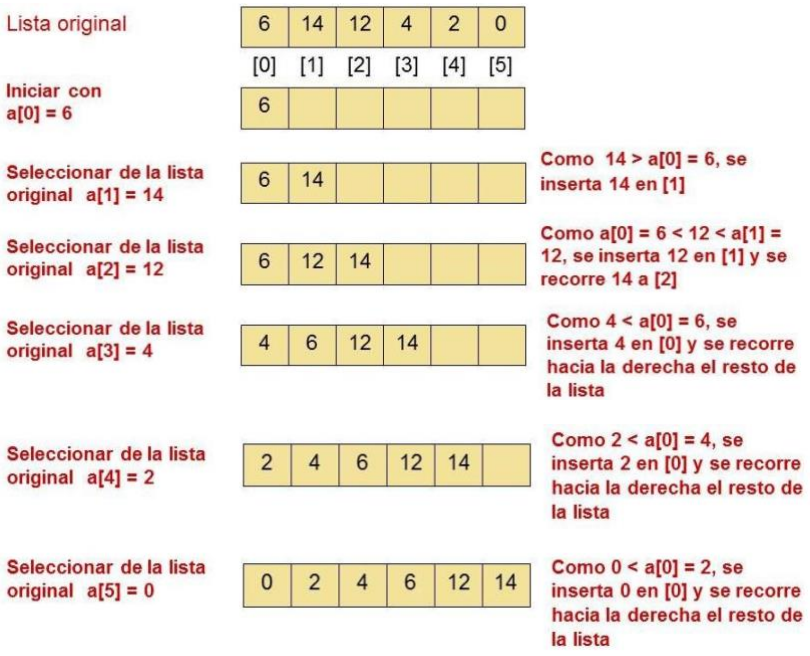
El método de inserción

El algoritmo de ordenamiento por inserción ordena un arreglo de *n* elementos en orden ascendente, insertando directamente cada elemento de la lista en la posición adecuada y recorriendo hacia la derecha (de [i] a [i+1]) los elementos restantes de la lista que son mayores al elemento insertado.

Algoritmo Insercion

Entrada: Vector V, Tamaño entrada n
Salida: Vector ordenado V

```
para k ← 1 hasta n hacer
    y ← V[k]
    i ← k − 1
    mientras i >= 0 ∧ y < V[i] hacer
        V[i + 1] ← V[i]
        i ← i − 1
    fin mientras
    V[i + 1] ← y
fin para
```



El algoritmo de inserción es $O(n^2)$.

Ventajas:

- Fácil implementación.
- Requerimientos mínimos de memoria.

Desventajas:

- Lento.
- Numerosas comparaciones.