

Análisis y diseño de Algoritmos

1er Examen Parcial

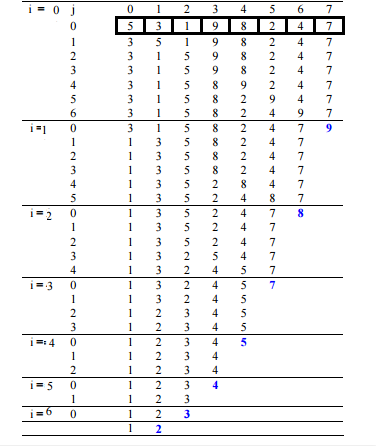
Maestro: Dr. Rodrigo Soule De Castro

Alumno: Daniel Alejandro Morales Castillo

1er algoritmo.

BubbleSort

La **Ordenación de burbuja** (**Bubble Sort** en inglés) es un sencillo [algoritmo de ordenamiento](https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento). Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiandolos 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](https://es.wikipedia.org/wiki/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.



¿Cuál es el mejor caso? - cuando los elementos están ordenados

¿Cuál es su complejidad? - O(n2)

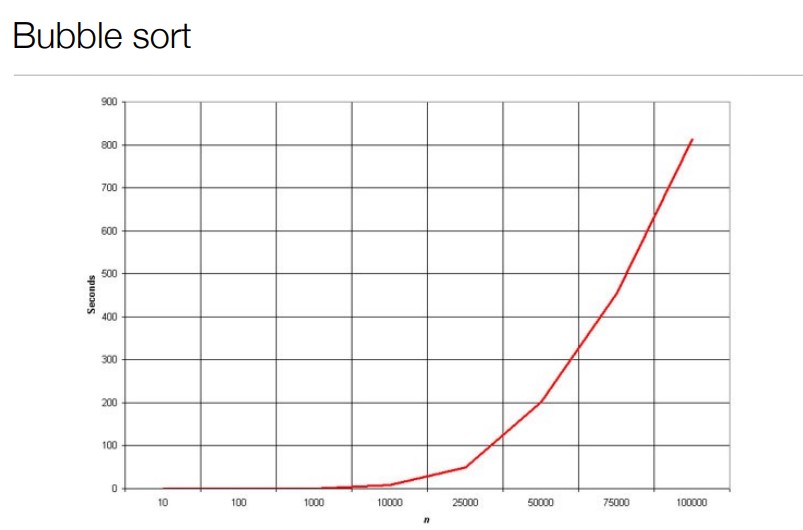
**Ventajas:**

Es muy simple y fácil de implementar

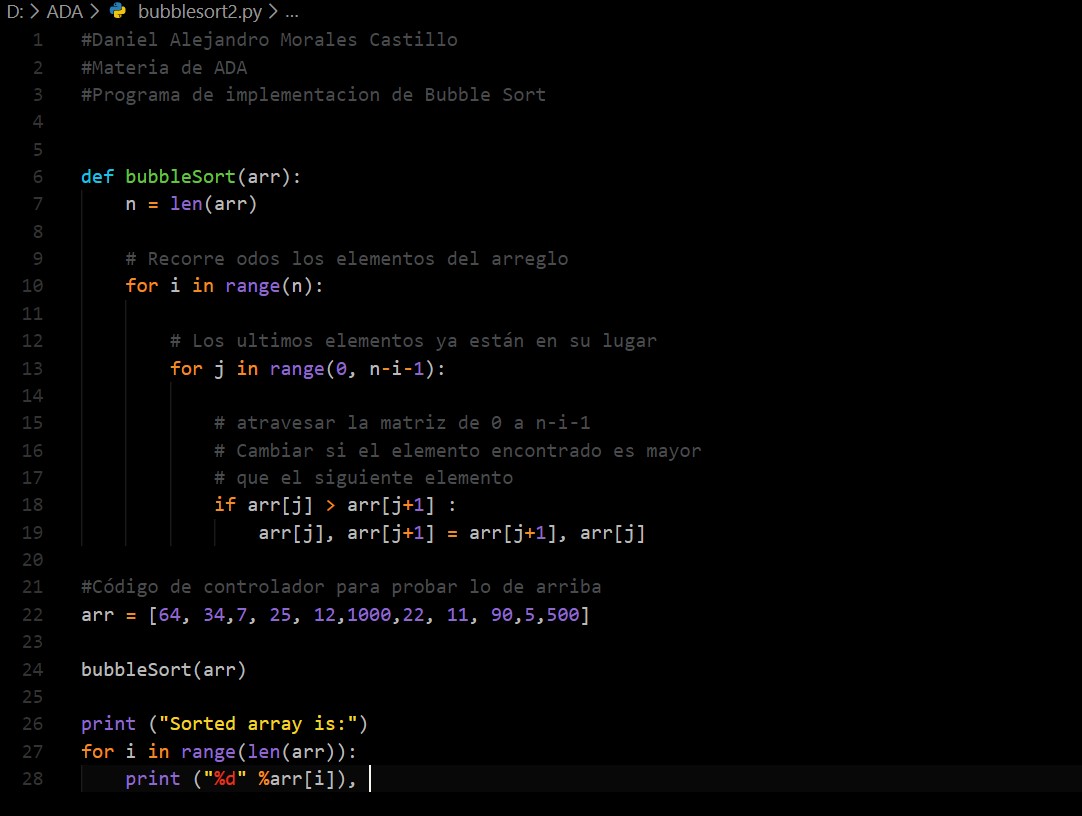
**Desventajas:**

Es el algoritmo más lento e ineficiente entre los algoritmos de ordenamiento.

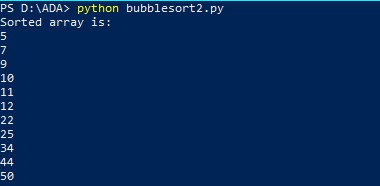
**Gráfica.**



\*Código realizado en Python



\*Salida por consola



Análisis:

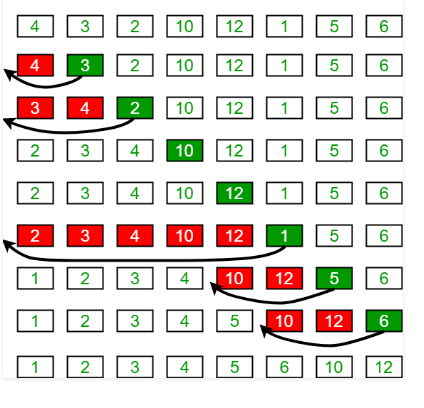
El ordenamiento de burbuja(Bubble Sort) tiene una complejidad O( n2 ).

Cuando una lista ya está ordenada, el método de ordenación por burbuja está forzado a pasar por toda la lista y hacer comparaciones, lo que hace que su complejidad sea cuadrática en el mejor de los casos, esto lo cataloga como algoritmo más ineficiente que existe aunque funciona bien cuando son pocos elementos.

**Ordenamiento por Inserción(Insertion Sort).**

El ordenamiento por inserción (insertion sort en [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s)) es una manera muy natural de ordenar para un ser humano, y puede usarse fácilmente para ordenar un mazo de cartas numeradas en forma arbitraria. Requiere O(n²) operaciones para ordenar una lista de n elementos.

Inicialmente se tiene un solo elemento, que obviamente es un conjunto ordenado. Después, cuando hay *k* elementos ordenados de menor a mayor, se toma el elemento *k+1* y se compara con todos los elementos ya ordenados, deteniéndose cuando se encuentra un elemento menor (todos los elementos mayores han sido desplazados una posición a la derecha) o cuando ya no se encuentran elementos (todos los elementos fueron desplazados y este es el más pequeño). En este punto se *inserta* el elemento *k+1* debiendo desplazarse los demás elementos.



**Ventajas:**

Es muy simple y fácil de implementar.

Eficiente en conjuntos de datos pequeños Eficiente en conjuntos de datos casi ordenados u ordenados.

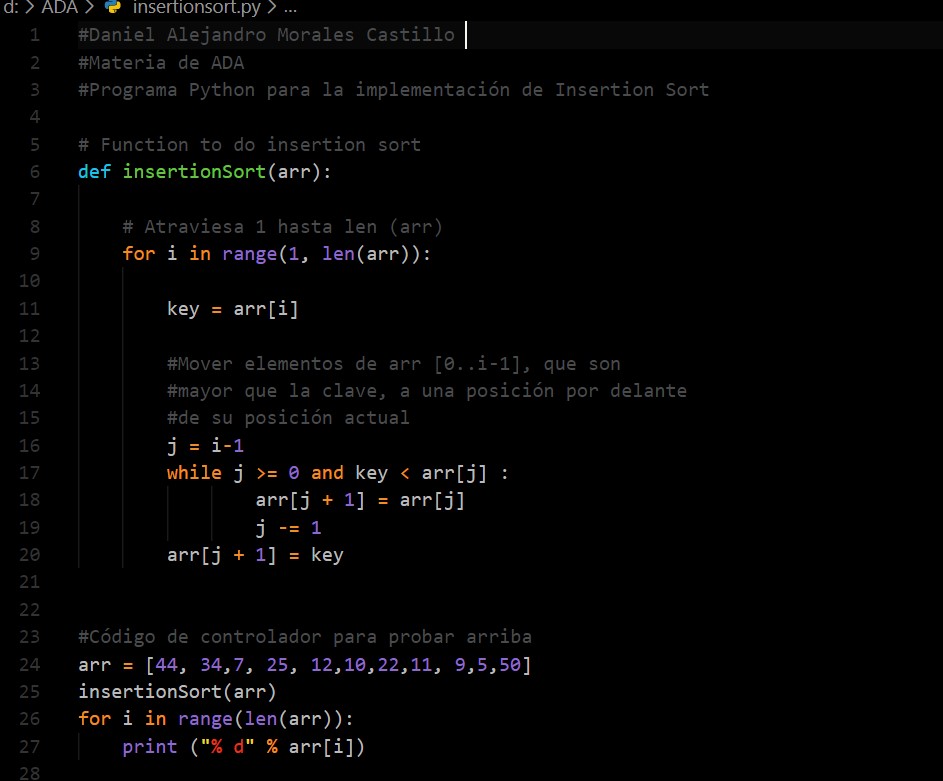
Estable

El tiempo de ejecución depende de la entrada: una secuencia que ya está altamente ordenada tomará menos tiempo

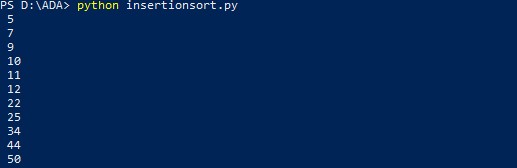
**Desventajas**:

Es ineficiente para conjuntos grandes de elementos.

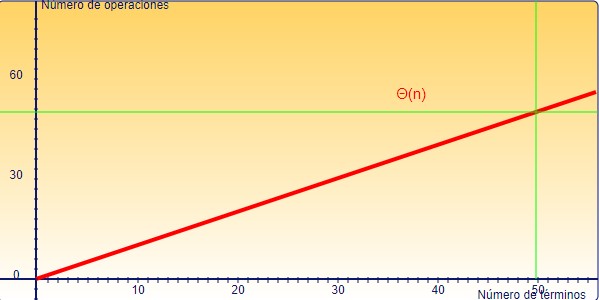
\*Código en Python de Insertion Sort

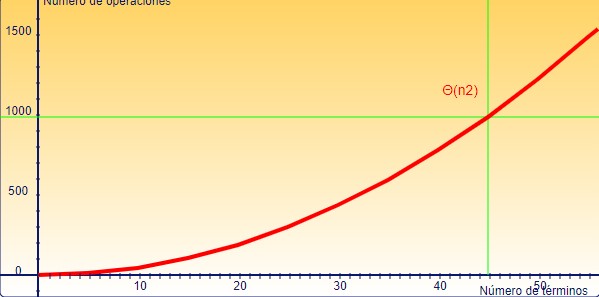


\*Salida por consola



Gráfica:





**Complejidad computacional**:

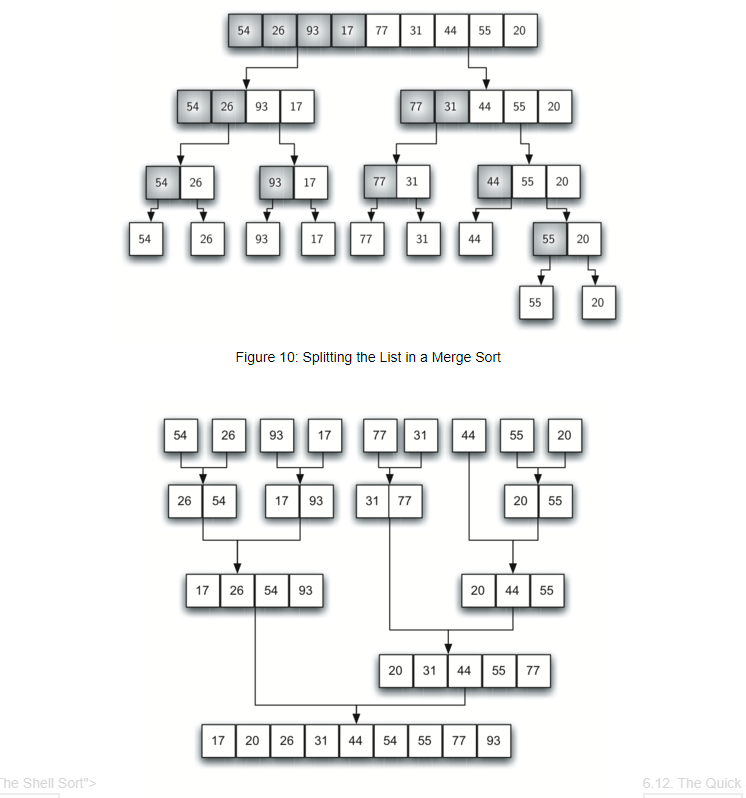
Mejor caso: *O(n)*

Caso promedio: *O(n²)*

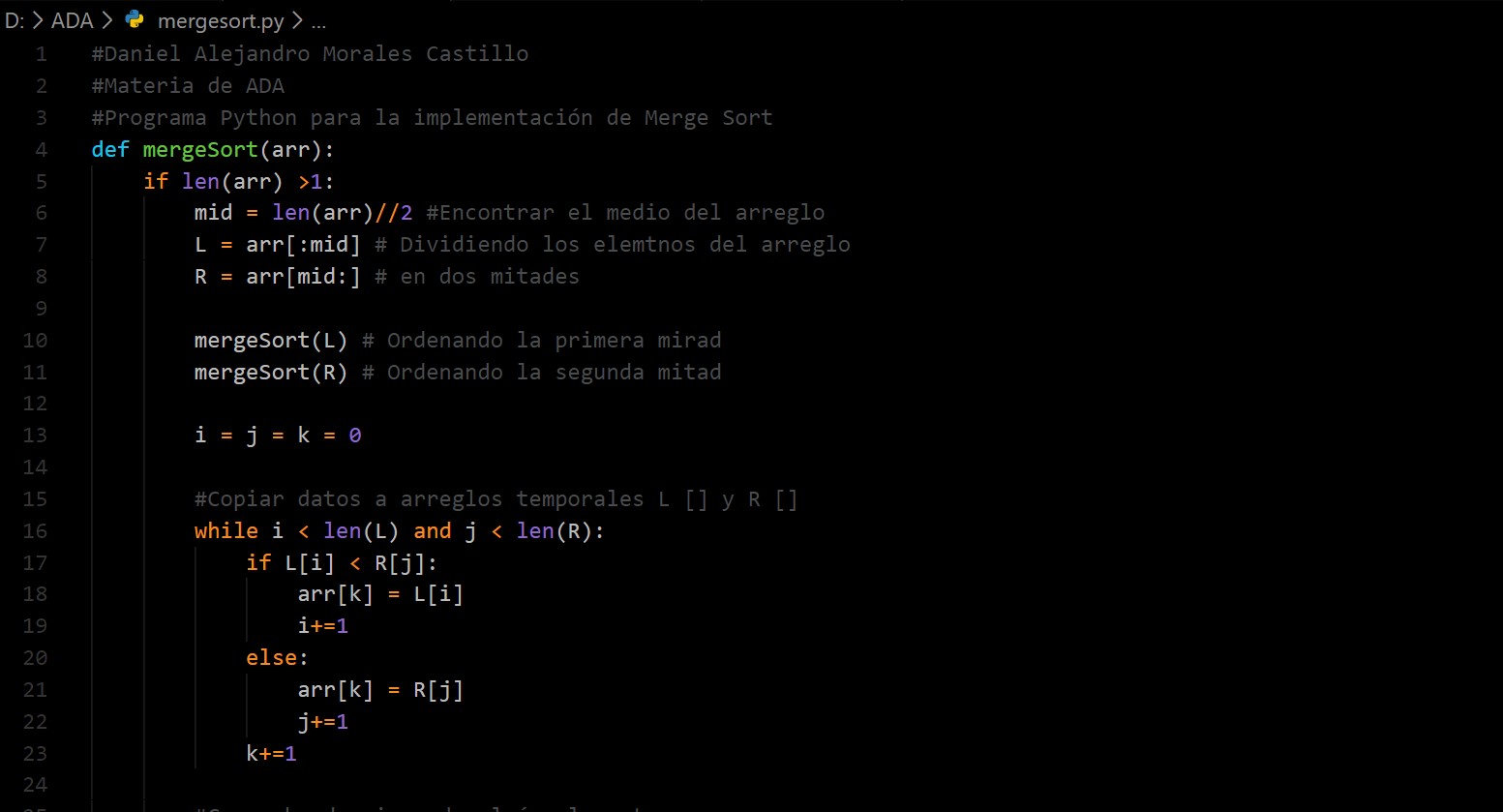
Peor caso: *O(n²)*

**Ordenamiento por mezcla(Merge Sort).**

El ordenamiento por mezcla es un algoritmo recursivo que divide continuamente una lista por la mitad. Si la lista está vacía o tiene un solo ítem, se ordena por definición (el caso base). Si la lista tiene más de un ítem, dividimos la lista e invocamos recursivamente un ordenamiento por mezcla para ambas mitades. Una vez que las dos mitades están ordenadas, se realiza la operación fundamental, denominada mezcla. La mezcla es el proceso de tomar dos listas ordenadas más pequeñas y combinarlas en una sola lista nueva y ordenada.

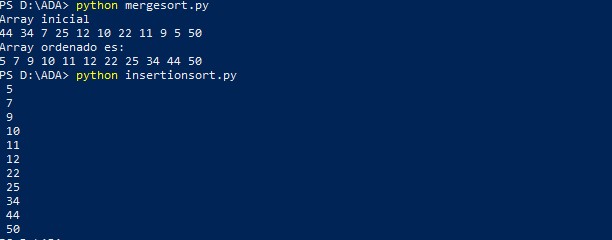


\*Código en Python

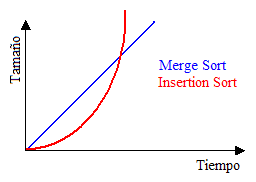




\*Salida por consola



**Ejemplo de Gráfica:**

****

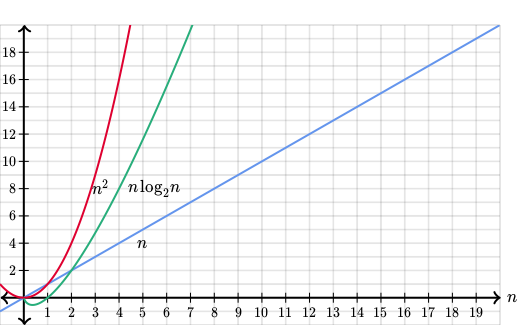
**Complejidad computacional:**

**O(n\*log(n))**

### **Ventajas y Desventajas**

* Método de ordenamiento estable mientras la función de mezcla sea implementada correctamente.
* Muy estable cuando la cantidad de registros a acomodar es de índice bajo, en caso contrario gasta el doble del espacio que ocupan inicialmente los datos.
* Efectivo para conjunto de datos a los que se puede acceder secuencialmente (arreglos, vectores, etc.)
* Principal desventaja: está definido recursivamente. Si se deseara implementarla no recursivamente se tendría que emplear una pila y se requeriría un espacio adicional de memoria para almacenarla.

**Gráfica de los tres algoritmos:**

****