Práctica 2: Métodos de Ordenamiento

Angel Bernardo Márquez Valdivia

22110348

Inteligencia Artificial

6E2

Métodos de Ordenamiento

Los métodos de ordenamiento son algoritmos diseñados para reorganizar los elementos de una lista, arreglo u otra estructura de datos en un orden específico, generalmente ascendente o descendente. Estos métodos son fundamentales en la informática y se utilizan para mejorar la eficiencia de búsquedas, análisis y otras operaciones en datos. Existen numerosos métodos de ordenamiento, cada uno con sus características, ventajas y desventajas.

Tipos de ordenamientos:

Los 2 tipos de ordenamientos que se pueden realizar son: los internos y los externos.

- **Ordenamientos internos:**

Son aquellos en los que los valores a ordenar están en memoria principal, por lo que se asume que el tiempo que se requiere para acceder cualquier elemento sea el mismo (a [1], a [500], etc.).

- **Ordenamientos externos:**

Son aquellos en los que los valores a ordenar están en memoria secundaria (disco, cinta, cilindro magnético, etc.), por lo que se asume que el tiempo que se requiere para acceder a cualquier elemento depende de la última posición accesada (posición 1, posición 500, etc.).

Algoritmos de ordenamiento.

1. Internos:

- Inserción (InsertionSort).
- Selección (SelectionSort).
- Intercambio.
- Ordenamiento de árbol.
- QuickSort.
- MergeSort.
- RadixSort.

2. Externos:

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

Clasificación de los algoritmos de ordenamiento de información.

El hecho de que la información está ordenada nos sirve para poder encontrarla y acceder de manera más eficiente, ya que de lo contrario se tendría que hacer de manera secuencial.

A continuación, se describirán cuatro grupos de algoritmos para ordenar información.

- **Algoritmos de inserción.**

En este tipo de algoritmo los elementos que van a ser ordenados son considerados uno a la vez. Cada elemento es "insertado" en la posición apropiada con respecto al resto de los elementos ya ordenados. Entre estos algoritmos se encuentran el de Inserción Directa, ShellSort, Inserción Binaria y Hashing.

- **Algoritmos de intercambio.**

En este tipo de algoritmos se toman los elementos de dos en dos, se comparan y se "intercambian" si no están en el orden adecuado. Este proceso se repite hasta que se ha analizado todo el conjunto de elementos y ya no hay intercambios. Entre estos algoritmos se encuentran el Burbuja (BubbleSort) y QuickSort.

- **Algoritmos de selección.**

En este tipo de algoritmos se "selecciona" o se busca el elemento más pequeño (o más grande) de todo el conjunto de elementos y se coloca en su posición adecuada. Este proceso se repite para el resto de los elementos hasta que todos son analizados. Entre estos algoritmos se encuentra el de Selección Directa.

- **Algoritmos de enumeración.**

En este tipo de algoritmos cada elemento es comparado contra los demás. En la comparación se cuenta cuántos elementos son más pequeños que el elemento que se está analizando, generando así una "enumeración". El número generado para cada elemento indicará su posición.

Los métodos de ordenamiento se dividen en simples y complejos.

- **Los métodos simples:** Inserción (o por inserción directa), Selección, Burbuja y ShellSort, en dónde el último es una extensión al método de inserción, siendo más rápido.
- **Los métodos complejos:** el QuickSort (ordenación rápida) y el HeapSort.

- **Método de ordenamiento Burbuja**

El método de ordenamiento Burbuja (Bubble Sort) es un algoritmo simple que organiza los elementos de una lista comparando y, si es necesario, intercambiando pares adyacentes. Este proceso se repite varias veces hasta que la lista esté completamente ordenada.

Características principales:

Estrategia: Compara elementos consecutivos y los intercambia si están en el orden incorrecto.

Eficiencia:

- **Mejor caso:** $O(n)$, cuando la lista ya está ordenada.
- **Peor caso y promedio:** $O(n^2)$, cuando la lista está completamente desordenada.

Estabilidad: Es estable porque no cambia el orden relativo de elementos iguales.

Simples listas pequeñas: Adecuado solo para listas pequeñas, ya que no es eficiente con listas grandes.

Funcionamiento del método Burbuja

En cada pasada:

1. Se revisa toda la lista desde el inicio hasta el penúltimo elemento.
2. Si un elemento es mayor (o menor, dependiendo del orden deseado) que su vecino inmediato, se intercambian.
3. Al final de cada pasada, el elemento más grande "burbujea" hacia su posición correcta al final de la lista.
4. El proceso se repite hasta que no haya más intercambios.

Ejemplo:

1. **Inicialización**

- Bibliotecas Importadas: Se usa tkinter para crear la interfaz gráfica y messagebox para mostrar mensajes emergentes.
- Ventana Principal: Se crea una ventana con un título y un tamaño definido.

2. **Estructura de Entrada**

- Entrada del Usuario:
- El usuario ingresa una lista de números (pueden ser negativos o con punto decimal) separados por comas en un campo de texto.
- Botón de Acción:
- Al hacer clic en el botón "Ordenar", se ejecuta la función ordenar_lista.

3. **Proceso de Ordenamiento**

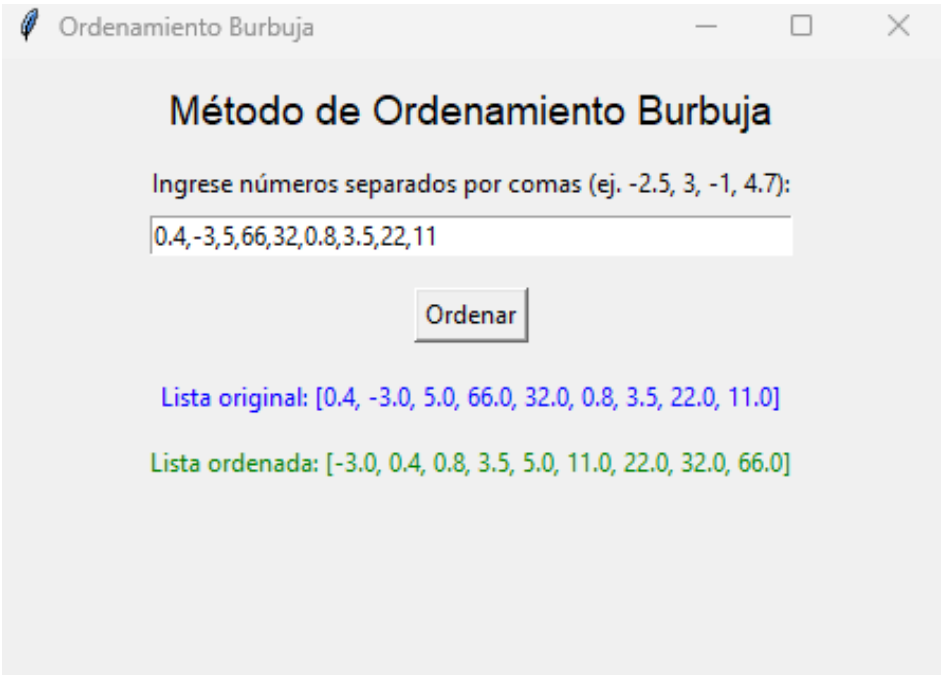
- La función ordenar_lista:
- Toma la Entrada: Captura el texto ingresado.
- Convierte a Lista: Intenta convertirlo en una lista de números flotantes.
- Si falla (entrada no válida), muestra un mensaje de error.
- Muestra la Lista Original: La lista ingresada se muestra en la interfaz.
- Ordena con Burbuja: Usa la función bubble_sort para ordenar la lista.
- Muestra el Resultado: La lista ordenada se muestra en la interfaz.

4. **Algoritmo de Ordenamiento:**

- Bubble Sort:
- Compara pares de elementos adyacentes.
- Intercambia si están en el orden incorrecto.
- Repite el proceso hasta que no haya más intercambios.

5. **Salida:**

- La interfaz gráfica muestra:
- La lista original ingresada.
- La lista ordenada tras aplicar el algoritmo.



- **<u>Método de ordenamiento por Selección</u>**

El método de ordenamiento por selección (Selection Sort) es un algoritmo simple que funciona dividiendo una lista en dos partes: una parte ordenada y otra desordenada. La idea principal es encontrar el elemento más pequeño (o más grande, según el orden deseado) en la parte desordenada y moverlo a la parte ordenada.

Pasos del Método de Selección

- Encuentra el elemento más pequeño en la parte desordenada de la lista.
- Intercambia este elemento con el primer elemento de la parte desordenada.
- Repite el proceso para el resto de la lista hasta que esté completamente ordenada.

Características del Selection Sort

Complejidad temporal:

- Peor caso: $O(n^2)$
- Mejor caso: O (n^2)
- Promedio: $O(n^2)$

Espacio: Es un algoritmo en el lugar, lo que significa que no requiere memoria adicional significativa ($O(1)$.

Es fácil de implementar, pero no es eficiente para listas grandes.

Ejemplo:

Proceso del Algoritmo:

El algoritmo de Selección (Selection Sort) sigue estos pasos:

1. **Iterar sobre la lista:**

- En cada iteración, se busca el número más pequeño en la parte no ordenada de la lista.

2. **Buscar el elemento mínimo en la sublista no ordenada:**

- Comparar los elementos de la sublista (desde el índice actual hasta el final de la lista) para identificar el número más pequeño.

3. **Intercambiar el número más pequeño con el primer elemento de la sublista no ordenada:**

- Intercambiar el número encontrado con el primer elemento de la sublista no ordenada.

4. Repetir el proceso hasta que toda la lista esté ordenada.

5. **Ordenamiento por Selección:**

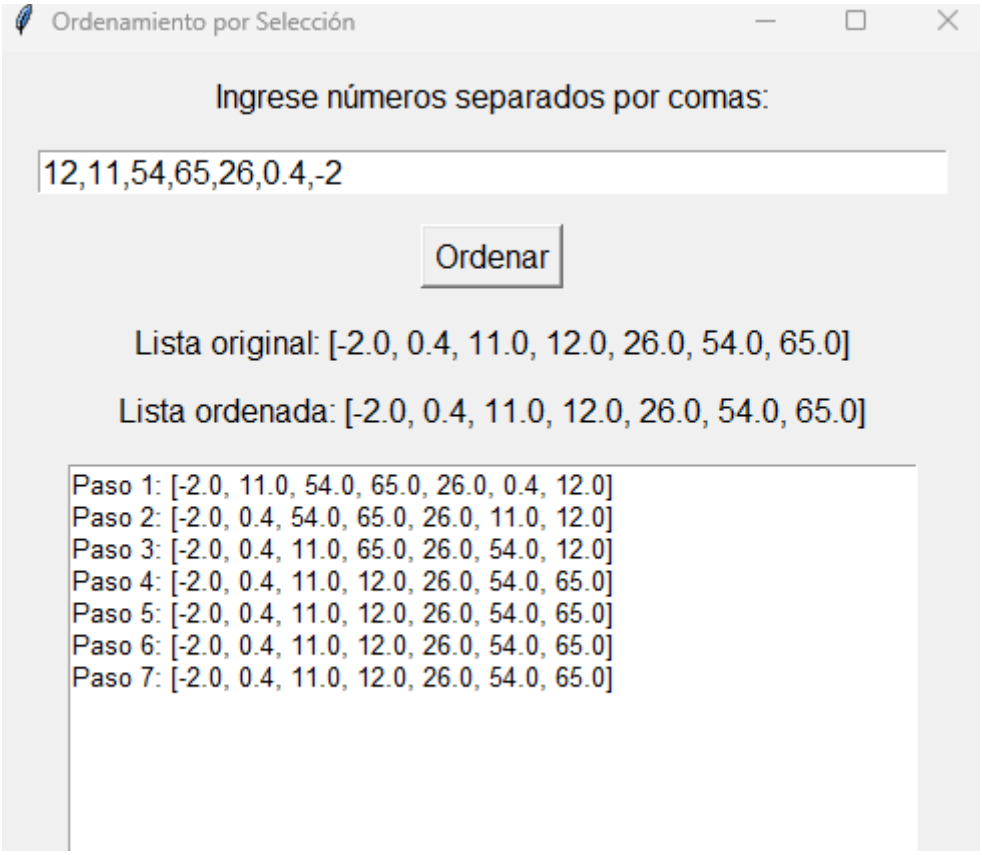
- En cada iteración, se encuentra el índice del número más pequeño en la parte no ordenada de la lista.
- Este número se intercambia con el primer elemento de la parte no ordenada.
- Cada paso se registra en una lista llamada steps.

6. **Interfaz Gráfica:**

- Se utiliza tkinter para crear la ventana.
- Contiene un campo de texto para la entrada, un botón para ejecutar el algoritmo, y etiquetas para mostrar los resultados.
- También incluye un cuadro de texto para mostrar los pasos detallados del algoritmo.

7. **Salida:**

- La lista original, la lista ordenada y los pasos del algoritmo se muestran en la interfaz gráfica.



- **Método de ordenamiento por Inserción**

El método de ordenamiento por inserción (Insertion Sort) es un algoritmo de ordenamiento sencillo que funciona de la siguiente manera:

El algoritmo construye la lista ordenada de manera incremental.

Divide la lista en dos partes:

- La **sublista ordenada**: Al principio, contiene el primer elemento.
- La **sublista no ordenada**: Contiene el resto de la lista.

Proceso:

- En cada paso, selecciona el primer elemento de la sublista no ordenada.
- Lo inserta en la posición correcta dentro de la sublista ordenada.
- Continúa hasta que toda la lista queda ordenada.

¿Cómo funciona el algoritmo de inserción paso a paso?

Supongamos que tenemos una lista:

[5, 2, 4, 6, 1, 3]

1. **Primer paso:** La sublista ordenada tiene un solo elemento: [5]. La sublista no ordenada es [2, 4, 6, 1, 3].
- Tomamos el siguiente elemento 2 de la sublista no ordenada.
- Lo insertamos en la posición correcta dentro de la sublista ordenada: [2, 5].

2. **Segundo paso:** Ahora la sublista ordenada es [2, 5]. Tomamos el siguiente elemento 4.
- Se inserta en la posición correcta dentro de [2, 5], resultando en [2, 4, 5].
- Continuamos el proceso hasta que la lista completa queda ordenada.

Ejemplo:

Explicación del Código

- Método de Inserción (insertion_sort):

La función insertion_sort(lista) implementa el algoritmo de ordenamiento por método de inserción que ya explicamos anteriormente.

- **Interfaz Gráfica con Tkinter:**

Se utiliza la librería Tkinter para crear una ventana donde el usuario puede ingresar números, presionar un botón para ordenar, y visualizar el resultado.

- **ordenar_lista:**

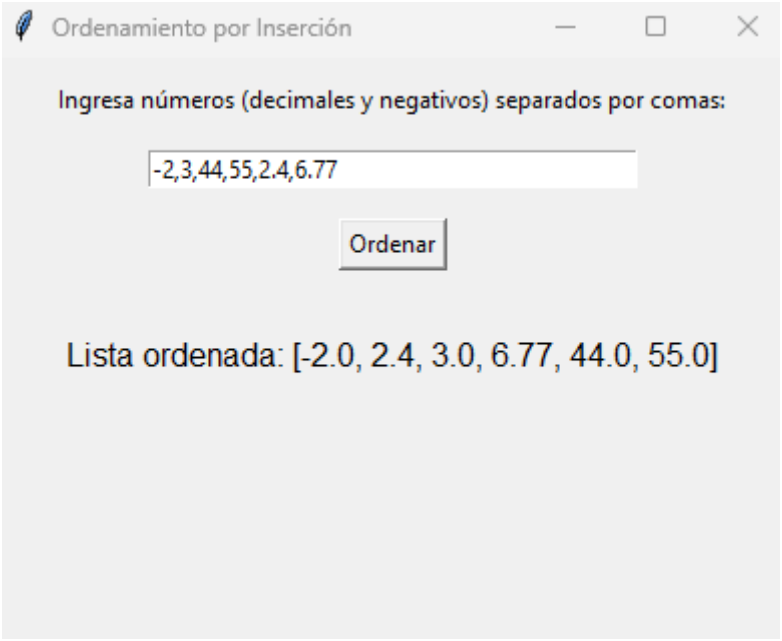
Esta función toma la entrada del usuario, convierte el texto en una lista de enteros, ordena la lista con el método de inserción y muestra el resultado en la interfaz.

Manejo de Errores:

Si el usuario ingresa datos no válidos, se muestra un mensaje de error con messagebox.showerror.

El código recorrió la lista desde el segundo elemento hasta el final, comparando cada elemento con los elementos anteriores en la lista. Durante este proceso:

- Compara cada elemento con los elementos anteriores (ya ordenados) de la lista.
- Movió los elementos mayores hacia la derecha para crear espacio.
- Insertó el elemento actual en su posición correcta en la parte ordenada.



- **Método de ordenamiento QuickSor**

QuickSort es un algoritmo de ordenamiento rápido basado en el enfoque "Divide y vencerás". Es uno de los algoritmos más eficientes para ordenar listas, especialmente para grandes cantidades de datos.

Funcionamiento:

El funcionamiento de QuickSort se basa en los siguientes pasos:

1. **Elegir un pivote:**

Seleccionar un elemento de la lista como "pivote". Puede ser el primer elemento, el último, un elemento aleatorio, o el que elijas según una estrategia.

2. **Particionar la lista:**

Reorganizar la lista para que todos los elementos menores que el pivote estén a su izquierda y los mayores estén a su derecha. Esto se conoce como "partición".

3. **Aplicar la misma lógica a las sublistas:**

Recursivamente, aplicar QuickSort a las sublistas (elementos a la izquierda y derecha del pivote) hasta que la lista esté completamente ordenada.

Complejidad de QuickSort

- **Complejidad promedio:** O(n log n)
- **Complejidad en el peor caso:** $O(n^2)$
- **Espacio adicional:** O(log n) en el caso óptimo, ya que se implementa de manera recursiva.

Ejemplo:

Funcionamiento:

1. **Implementación de QuickSort:**

La función quicksort(arr) implementa el método QuickSort que ordena una lista de números de manera recursiva.

2. **Procesar la entrada del usuario:**

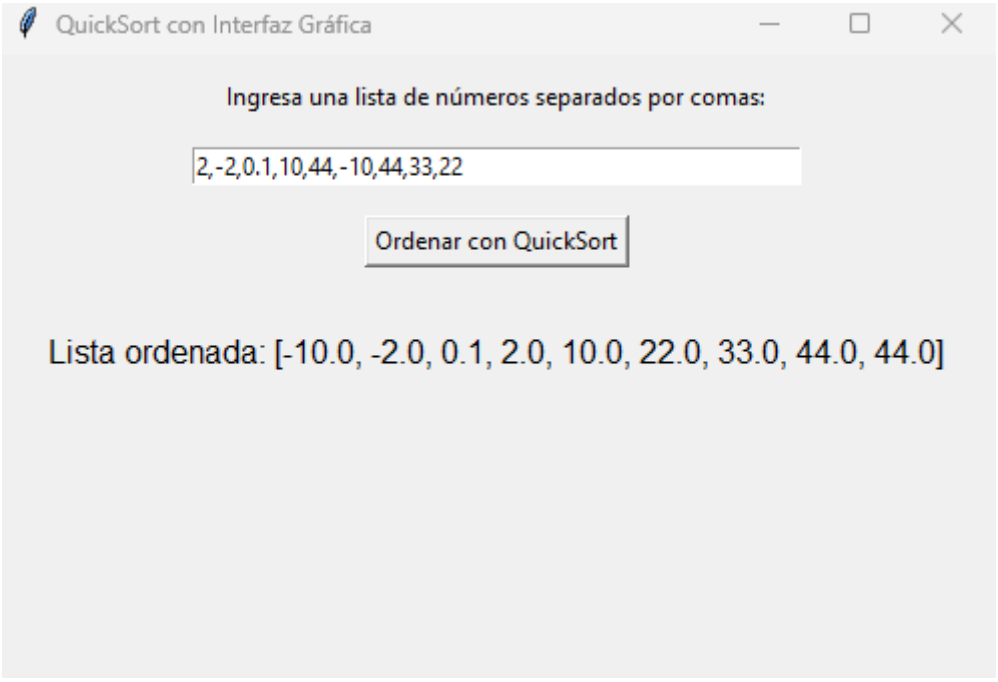
La función ejecutar_quicksort() toma la cadena ingresada por el usuario, la convierte en una lista de números con la función float() para aceptar decimales, y ejecuta la función quicksort().

3. **Interfaz Gráfica con Tkinter:**

- Campo de entrada: El usuario ingresa una lista de números separados por comas en el campo de texto.
- Botón "Ordenar con QuickSort": Al hacer clic, se ejecuta la función ejecutar_quicksort().
- Resultado: La lista ordenada se muestra en la etiqueta de la interfaz.

4. **Mensajes de error:**

Si el usuario ingresa datos inválidos, se muestra una ventana de error con un mensaje claro.



- **<u>Método de ordenamiento Merge Sort</u>**

El ordenamiento por fusión, conocido como Merge Sort, es un algoritmo de ordenamiento eficiente basado en el enfoque Divide y Vencerás. Su funcionamiento se basa en dividir un arreglo o lista en subarreglos más pequeños, ordenarlos de forma recursiva y luego fusionarlos de manera ordenada.

Características de Merge Sort

1. **Enfoque Divide y Vencerás**:

   El arreglo se divide en mitades hasta que cada subarreglo tiene un solo elemento (o está vacío), y después se fusionan de forma ordenada.}

2. **Complejidad:**

- Complejidad en el peor, mejor y caso promedio: O(n log n).
- Esto lo hace muy eficiente para arreglos grandes.

3. **Es estable:** Un algoritmo estable mantiene el orden relativo de los elementos iguales.

4. No es un método en el lugar (in-place): Requiere espacio adicional para fusionar los elementos.

¿Cómo Funciona el Merge Sort?

El algoritmo Merge Sort sigue estos pasos:

- Dividir: Divide el arreglo en dos mitades aproximadamente iguales.
- Llamadas Recursivas: Ordena cada mitad de manera recursiva utilizando el mismo algoritmo.
- Fusionar: Combina (fusiona) las dos mitades ordenadas para crear una única lista ordenada.

Algoritmo Merge Sort

El proceso básico de Merge Sort implica lo siguiente:

- Dividir el arreglo: Dividir el arreglo en dos mitades.
- Aplicar Merge Sort de forma recursiva: Llamar a Merge Sort en cada una de las dos mitades.
- Fusionar las dos mitades ordenadas: Combinar las dos mitades ordenadas en una lista completamente ordenada.

Ejemplo:

1. **Interfaz Gráfica con tkinter:**

- Se crea una ventana utilizando tkinter.
- El usuario puede ingresar una lista de números separados por comas.

2. **Botón para Ejecutar Merge Sort:**
   - Cuando el usuario presiona el botón "Ordenar con Merge Sort", se ejecuta la función execute_merge_sort.

3. **Procesamiento de Datos:**
   - Se toma la cadena ingresada por el usuario, y se convierte en una lista de números (float para aceptar números decimales y negativos).

4. **Aplicar Merge Sort:**
   - La lista ingresada es pasada a la función merge_sort.
   - El método Merge Sort se aplica a esa lista.
5. **Mostrar el Resultado:**

   - Después de procesar los datos, la lista ordenada se muestra en la interfaz gráfica en una etiqueta.

6. **Mensajes de error:**
   - Si el usuario ingresa una entrada no válida, se muestra un cuadro de mensaje con un error.