INSTITUTO POLITÉCNICO NACIONAL

Unidad Profesional Interdisciplinaria de Ingeniería
Campus Zacatecas.

Materia:
Análisis y Diseño de Algoritmos

Práctica 01: Análisis de Casos

Docente:
Erika Sánchez Femat

Nombre del alumno:
Dalia Naomi García Macías

Fecha de entrega:
18 de septiembre de 2023

Índice

1. Introducción

Durante esta práctica, trabajaremos con los algoritmos de ordenamiento burbuja y burbuja mejorada. El algoritmo de burbuja es un método de ordenamiento simple que se utiliza para ordenar listas o arreglos de elementos. Su nombre proviene de la forma en que los elementos más grandes "burbujean" hacia la parte superior de la lista o arreglo durante el proceso de ordenamiento. La idea básica detrás de este algoritmo es comparar pares de elementos adyacentes y, si están en el orden incorrecto, intercambiarlos. Este proceso se repite hasta que no se realicen intercambios en un pase completo a través de la lista, lo que indica que la lista está ordenada. Sin embargo, a pesar de ser tan simple, este algoritmo puede ser ineficiente para listas grandes debido a su complejidad cuadrática.

En respuesta a las limitaciones del algoritmo de burbuja, se desarrolló el algoritmo de burbuja mejorada, que es similar en su enfoque, pero con una optimización que reduce la cantidad de operaciones innecesarias, mejorando así su tiempo de ejecución.

2. Desarrollo de la Práctica

2.1. Implementación de los Algoritmos

2.1.1. Algoritmo de Burbuja

El algoritmo de burbuja, también conocido como Bubble Sort, es un método de ordenamiento simple que funciona revisando cada elemento de la lista con el siguiente, intercambiándolos si están en el orden incorrecto. Esto se repite hasta que la lista esté ordenada. A continuación, se presenta un ejemplo gráfico de su funcionamiento:

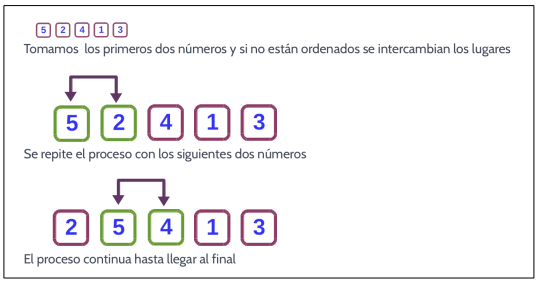

Figura 1: Ejemplo del algoritmo de burbuja (parte 1)

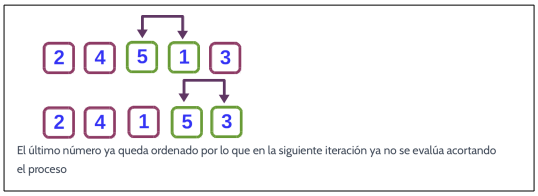

Figura 2: Ejemplo del algoritmo de burbuja (parte 2)

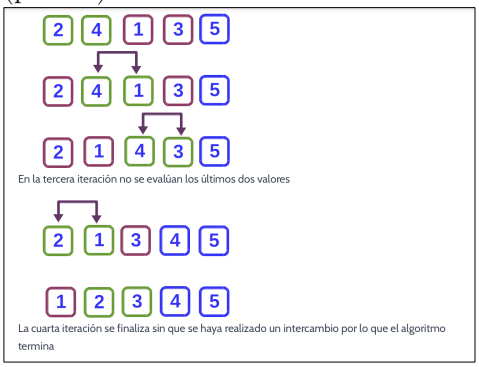

Figura 3: Ejemplo del algoritmo de burbuja (parte 3)

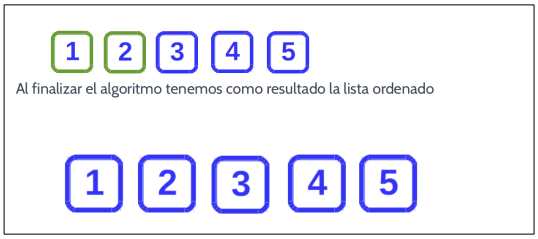

Figura 4: Ejemplo del algoritmo de burbuja (parte 4)

2.1.2. Algoritmo de Burbuja Mejorada

El algoritmo de burbuja mejorada es una variante que mejora el rendimiento del algoritmo de burbuja estándar. Cada vez que se recorre el arreglo, se registra si se realizan intercambios de elementos y dónde ocurren. Esto evita futuros intercambios innecesarios y mejora el tiempo de ejecución.

2.1.3. Código

Primero que nada antes de empezar a desarrollar el código se debían tomar en cuenta las especificaciones que incluiría, las cuales decían que nuestro código debía contener un menú para que así el usuario pudiera tener la facilidad de escoger qué método de ordenamiento quería usar, del mismo modo con el tamaño y los elementos que contendrían el arreglo.
Teniendo en cuenta esto, tomé la decisión de usar funciones, ya que así podría incluir ambos métodos de manera más sencilla dentro del menú. Creé cuatro funciones: primero, una que recopilara los datos del arreglo dados por el usuario y a su vez me imprimiera el arreglo sin ordenar; luego, la función "burbuja" que tendría el funcionamiento del primer método (el método de la burbuja); después, otra que contuviera el desarrollo del segundo método (el método de la burbuja mejorada); y finalmente, una que me mostrara cómo quedaría el arreglo ya ordenado.

2.2. Análisis de Casos

2.2.1. Mejor Caso

El mejor caso para ambos algoritmos ocurre cuando la lista ya está completamente ordenada. En este caso, no se realizarán intercambios, y el algoritmo simplemente recorrerá la lista una vez para verificar que está ordenada. El número de comparaciones y movimientos será mínimo. Por ejemplo, si tenemos una lista con n elementos y ya está ordenada, el algoritmo solo realizará $n-1$ comparaciones en el peor de los casos.

```
1  Vector original:
2  [1, 2, 3, 4, 6, 5]
3
4  [1 > 2, 3, 4, 6, 5]
5  [1, 2 > 3, 4, 6, 5]
6  [1, 2, 3 > 4, 6, 5]
7  [1, 2, 3, 4 > 6, 5]
8  [1, 2, 3, 4, 6 > 5]
9
10 Vector ordenado:
11 [1, 2, 3, 4, 5, 6]
```

Figura 5: Ejemplo del mejor caso.

2.2.2. Caso Promedio

El caso promedio para estos algoritmos es similar al peor caso, ya que en cada pasada se comparan y, en su caso, intercambian todos los elementos. En el caso promedio, se esperaría que se realicen aproximadamente $n^2/2$ comparaciones en total para una lista de n elementos.

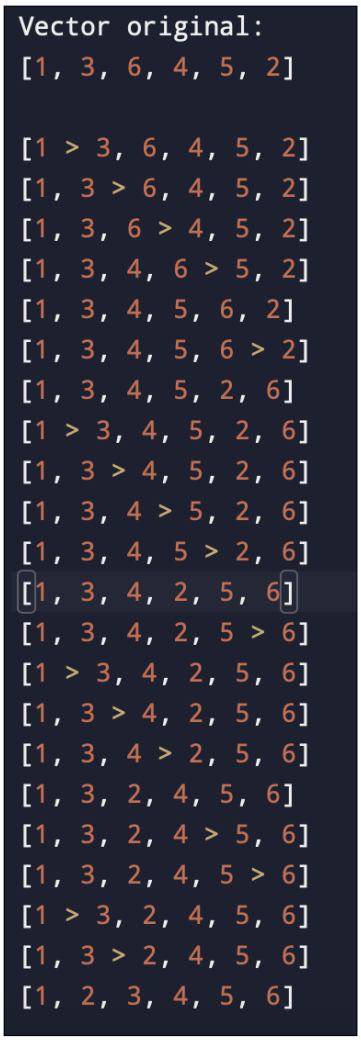

Figura 6: Ejemplo del caso promedio (parte 1)

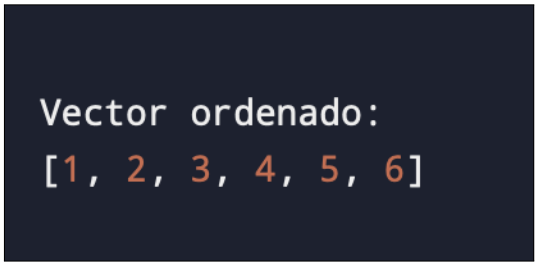

Figura 7: Ejemplo del caso promedio (parte 2)

2.2.3. Peor Caso

El peor caso ocurre cuando la lista está ordenada en orden inverso, de manera que el algoritmo debe realizar el máximo número de comparaciones y movimientos. En este caso, ambos algoritmos requerirán aproximadamente $n^2/2$ comparaciones y movimientos para ordenar la lista de n elementos.

2.3. Complejidad

Ambos algoritmos tienen una complejidad de tiempo en el peor caso de $O(n^2)$, donde n es el número de elementos en la lista. Esto significa que el tiempo de ejecución aumenta cuadráticamente con el tamaño de la lista. A pesar de la optimización en el algoritmo de burbuja mejorada, sigue compartiendo la misma complejidad en el peor caso debido a su estructura fundamental de comparación y intercambio de elementos.

2.4. Comparación de Resultados

Se realizó una comparación de los tiempos de ejecución de ambos algoritmos utilizando la biblioteca de tiempo. En general, se observó que el algoritmo de burbuja mejorada tuvo un tiempo de ejecución ligeramente más rápido en comparación con el algoritmo de burbuja estándar. Sin embargo, la diferencia en el tiempo de ejecución no fue significativa para tamaños pequeños de lista. La diferencia se hizo más evidente a medida que el tamaño de la lista aumentó, lo que respalda la eficiencia de la burbuja mejorada.

3. Conclusiones

En conclusión, los algoritmos de ordenamiento burbuja y burbuja mejorada son métodos simples pero ineficientes para ordenar listas. Aunque son fáciles de entender e implementar, su complejidad cuadrática hace que sean inadecuados para listas grandes. La burbuja mejorada ofrece una pequeña mejora en términos de tiempo de ejecución, pero aún así no es una opción eficiente para grandes conjuntos de datos.

Es importante comprender los principios de estos algoritmos, pero en la práctica, se recomienda utilizar algoritmos de ordenamiento más eficientes, como QuickSort o MergeSort, para manejar listas de tamaño considerable.

4. Referencias

- https://juncotic.com/ordenamiento-de-burbuja-algoritmos-de-ordenamiento/
- https://runestone.academy/ns/books/published/pythoned/SortSearch/ElOrdenamientoBurbuja.html
- https://tutospoo.jimdofree.com/tutoriales-java/m%C3%A9todos-de-ordenaci%C3%B3n/burbuja-optimizad