INSTITUTO POLITÉCNICO NACIONAL

Unidad Profesional Interdisciplinaria de Ingeniería
Campus Zacatecas.

MATERIA:

- Analisis y diseño de algoritmos

MAESTRA:

- Erika Sánchez Femat

ALUMNA:

- Naylin Yusseth Martínez Mireles

PRACTICA 01

(Reporte de analisis de casos)

Índice

1. INTRODUCIÓN

La finalidad de realizar este reporte, es analizar y comprender como es el funcionamiento de los códigos de burbuja y burbuja mejorada, también hacer que quede más claro cuáles serían los mejores casos, los peores y el caso promedio al implementar alguno de estos dos códigos.
En la materia de análisis y diseño de algoritmos, estamos tratando de comprender como es el funcionamiento de estos y cuáles son las maneras de llevarlo a cabo sin que implique tanto tiempo y esfuerzo.
Por ello tomamos como herramienta, el examinar ejemplos de los casos posibles que se nos puedan presentar, para así poderlos comparar y obtener la mejor opción de programarlos.
El hacer esto, tambien nos ayuda para darnos cuenta de cual es el nivel de complejidad que tienen cada uno de ellos y asi poder tener mas conocimiento de como resolver cada caso dependiendo las necesidades que este conlleve.
La complejidad de un algoritmo es una medida de cuán eficiente es el algoritmo para resolver el problema. En otras palabras, es una medida de cuánto tiempo y espacio (memoria) requiere el algoritmo para producir una solución y asi poder ver cual nos combiene más.
En programación, se utilizan distintas formas de programar, cada persona tiene características a la hora de plasmarlas en un código, como por ejemplo el nombre las variables, la estructura que precenta y los metodos computacionales que utiliza, pero la mayoría, si no es que todos, siempre buscaremos las alternativas más rápidas y sencillas de poder realizar un código.
Claro que también hay que tomar en consideración los lenguajes de programación que utilizamos.
En este caso el que tomaremos en cuenta será el lenguaje de Python, tanto los dos códigos de burbuja, como los ejemplos que presente en este reporte, estarán elaborados en este lenguaje de programación, ya que es que él estamos aprendiendo en la materia.

2. DESARROLLO

Aquí es donde explicaré como es que lleve a cabo la investigación de la información realmente relevante de este reporte, voy a plasmar los ejemplos de casos que encontré para poder entender de manera más clara cuáles serían las mejores y peores situaciones a la hora de programar.
Pero primero hay que comprender que es el análisis de complejidad y para qué nos sirve.
Como lo dije anteriormente, la complejidad computacional o simplemente complejidad de un algoritmo es la cantidad de recursos necesarios para ejecutarlo. Se presta especial atención al tiempo de cálculo y los requisitos de almacenamiento de memoria.
Hay varias formas de medir la complejidad de un algoritmo.
Uno de los más comunes es contar el número de operaciones básicas (como sumas o multiplicaciones) que realiza el algoritmo. Esto se conoce como la complejidad temporal del algoritmo.

Otra forma de medir la complejidad es contar la cantidad de memoria (en bytes o bits) que requiere el algoritmo. Esto se conoce como la complejidad espacial del algoritmo.

Las complejidades de tiempo y espacio de un algoritmo se pueden expresar utilizando la notación O grande. Esta notación proporciona una forma de comparar la complejidad de diferentes algoritmos. Por ejemplo, un algoritmo con una complejidad temporal de O(n) se considera más eficiente que un algoritmo con una complejidad temporal de O(n^2), porque crece a un ritmo más lento a medida que aumenta el tamaño de entrada.

En general, el objetivo del diseño de algoritmos es encontrar algoritmos que sean eficientes y fáciles de implementar. Esto implica un compromiso entre la complejidad del tiempo y el espacio, así como otros factores como la simplicidad y la mantenibilidad.

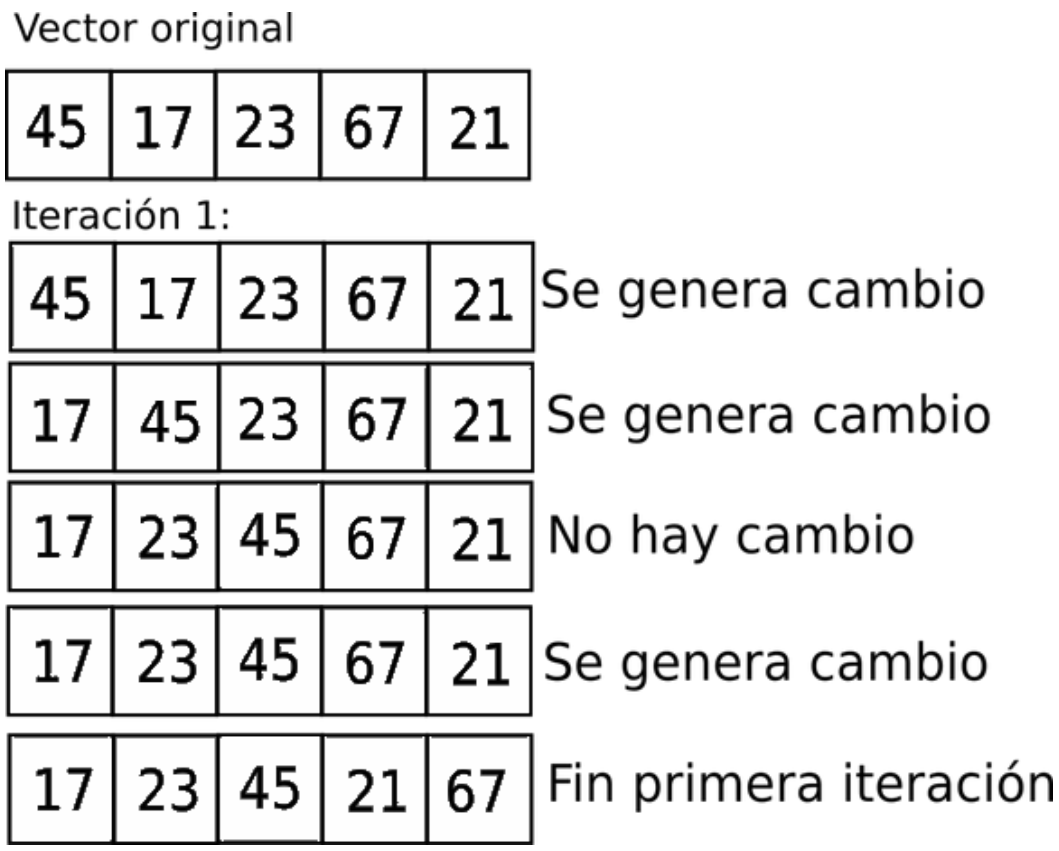
En general, el estudio de la complejidad de los algoritmos es una parte importante tanto de las matemáticas como de la informática. Nos ayuda a comprender las limitaciones de los algoritmos y las compensaciones involucradas en la resolución de problemas complejos.
Una vez aclarando esto, falta conocer como es el funcionamiento de los metodos que estoy utilizando, el de burbuja y burbuja mejorada.

Método burbuja:

El método de ordenamiento burbuja consiste en comparar cada elemento de la estructura con el siguiente e intercambiándolos si corresponde. El proceso se repite hasta que la estructura esté ordenada. El orden se establece de acuerdo a la clave y la estructura tiene que tener acceso directo a sus componentes.

Método burbuja mejorada

Este método recorre todo el arreglo comparando cada uno de los elementos con el elemento siguiente e intercambiándolo de ser necesario. Al finalizar la iteración el elemento mayor queda ubicado en la última posición, mientras los elementos menores ascienden una posición.



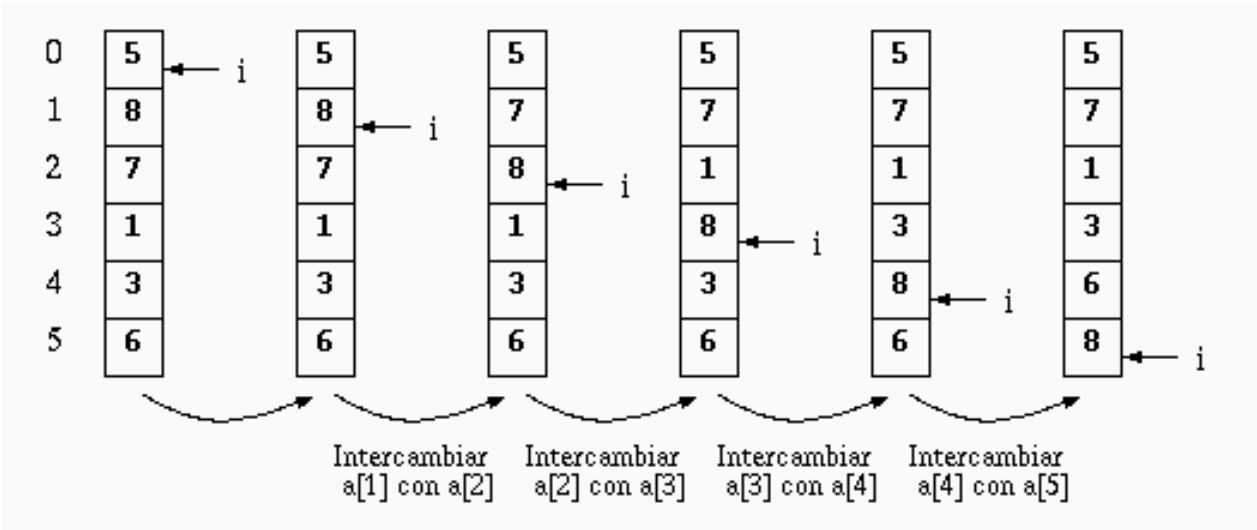
2.1. ANÁLISIS DE CASOS

Ya una vez conociendo el funcionamiento de los métodos, ahora si ya puedo pasar a analizar los casos

Método burbuja:

Ejemplo 1.

La idea consiste en realizar varios recorridos secuenciales en el arreglo intercambiando los elementos adyacentes que estén desordenados.
Como se muestra en la siguiente figura, en la primera iteración se lleva el máximo a su posición definitiva al final del arreglo:



Mejor caso:
El mejor caso, seria la ultima columna, cuando el arreglo ya este ordenado, esto significaria que no tendriamos que hacer más movimientos por que el arreglo ya estaría ordenado.

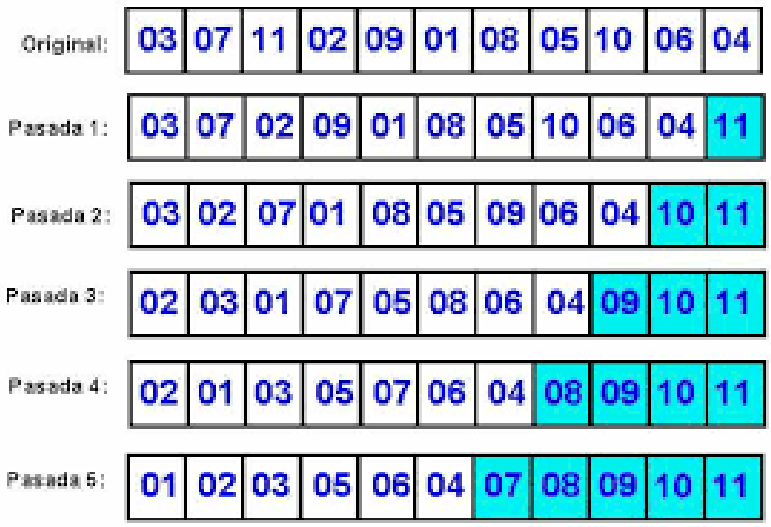
Peor caso:
El peor caso, seria en la primera columna, ya que el número más grande se

encuentra en la segunda posición, debido a ello se tendrian que hacer más movimientos para lograr ordenar el arreglo, lo cúal implica más tiempo y más memoria.

Caso Promedio:

El caso promedio en general es, cuando el arreglo está semiordenado, ¿a qué me refiero con esto?, pues a que el número más grande esta en el centro o en un lugar más cercano al que le pertenece, debido a esto los movimientos serán menores y más fáciles de ejecutar.

Ejemplo 2.



Mejor caso:

El mejor caso en este ejemplo sería la pasada 4 o la pasada 5, por ejemplo en la pasada 4, el arreglo esta casi ordenado, seria cuestión de hacer un procedimiento casi nulo para poder ordenarlo a la totalidad.

Peor caso:

El peor caso aquí, es desde el arreglo original, si nos damos cuenta esta todo desordenado, lo cúal conlleva a más trabajo.

Caso Promedio:

El caso promedio se encuentra en el centro, entre la pasada 2 y 3, los números se encuentran cerca de su posición correcta, sera cuestión de pocos pasos para ordenarlo completo.

Ejemplo 3.

j=0 [8, 3, 1, 19, 14]	8 > 3 (Intercambio)	[3, 8, 1, 19, 14]
j=1 [3, 8, 1, 19, 14]	8 > 1 (Intercambio)	[3, 1, 8, 19, 14]
j=2 [3, 1, 8, 19, 14]	8 > 19 (No intercambio)	[3, 1, 8, 19, 14]
j=3 [3, 1, 8, 19, 14]	19 > 14 (Intercambio)	[3, 1, 8, 14, 19]

Mejor caso:

En este ejemplo el mejor caso sería en el último arreglo de la primera columna, en donde j=3, podemos notar que en la columna 2 nos muestra que hace el interambio del número 19 por el 14, y de esa manera el arreglo queda ordenado.

Peor caso:

Aquí tenemos una situación interesante, si notamos el arreglo no esta tan desordenado, el número menor y el número mayor, se encuentran en la misma posición, por lo tanto el peor caso se encontraría cuando j=0, que nos dice que el arreglo esta completamente desordenado.

Caso promedio:
Pues realmente en este caso no esta tan marcado el caso promedio, también por que el arreglo es un pequeño, pero sería en la tercera posición de la primera columna cuando j=2.

Método burbuja mejorado:

Consiste en ciclar repetidamente a través de la lista, comparando elementos adyacentes de dos en dos.
Si un elemento es mayor que el que está en la siguiente posición se intercambian.
Vamos a ver un ejemplo.
Esta es nuestro arreglo:
$[4, 3, 5, 2, 1]$
Tenemos 5 elementos es decir, el centro toma el valor 5, comenzamos comparando el primero con el segundo elemento, 4 es mayor que 3, así que intercambiamos.
Ahora tenemos: [3–4–5–2–]
Ahora comparamos el segundo con el tercero: 4 es menor que 5, así que no hacemos nada.
Continuamos con el tercero y el cuarto: 5 es mayor que 2.
Intercambiamos y obtenemos: [3–4–2–5–1]
Comparamos el cuarto y el quinto: 5 es mayor que 1.
Intercambiamos nuevamente: [3–4–2–1–5]
Repitiendo este proceso vamos obteniendo los siguientes resultados:

$$3 - 2 - 1 - 4 - 5$$
$$2 - 1 - 3 - 4 - 5$$
$$1 - 2 - 3 - 4 - 5$$

OPTIMIZADO

Se pueden realizar algunos cambios en este algoritmo que pueden mejorar su rendimiento. Si observas bien, te darás cuenta que en cada pasada a través de la lista un elemento va quedando en su posición final. Si no te queda claro mira el ejemplo de arriba. En la primera pasada el 5 (elemento mayor) quedó en la última posición, en la segunda el 4 (el segundo mayor elemento) quedó en la penúltima posición. Podemos evitar hacer comparaciones innecesarias si disminuimos el número de éstas en cada pasada. Tan sólo hay que cambiar el ciclo interno de esta manera: for (j=0; j¡TAM- i; j++).
La clave es sustituir el 1 por i. Puede ser que los datos queden ordenados antes de completar el ciclo externo.

Realmente, la burbuja optimizada es un método igual al de la burbuja, lo único que las diferencia es que implementa una mejora al método de ordenamiento burbuja. Este consiste en recorrer todo el arreglo comparando cada uno de los elementos con el elemento siguiente e intercambiándolo de ser necesario. Al finalizar la iteración, el elemento mayor queda ubicado en la última posición, mientras los elementos menores ascienden una posición. La mejora es que, como al final de cada iteración el elemento mayor queda situado en su posición, ya no es necesario volverlo a comparar con ningún otro número, reduciendo así el número de comparaciones por iteración. Esto quiere decir que su mejor caso, peor caso y caso promedio serán iguales en el procedimiento, siempre y cuando el arreglo este desordenado tardara más y requerirá más trabajo, por ende será el peor caso, el mejor será cuando el tiempo y el esfuerzo sean menores.

```
Vector original
45 17 23 67 21

iteracion: 1: 17 23 45 21 67
iteracion: 2: 17 23 21 45 67
iteracion: 3: 17 21 23 45 67
iteracion: 4: 17 21 23 45 67
17 21 23 45 67
```

2.2. COMPLEJIDAD DE LOS CASOS

Ahora a determinar la complejidad de los casos, basandome en lo que pude recolectar de información hacer de cómo funciona y para qué sirve la complejidad.
Todos los casos y ejemplos estan relacionados, entonces será la misma complejidad para todos los ejemplos, pero no para los casos.

Mejor de los casos:
La complejidad de este caso es: $O(n)$

Peor de los casos:
La complejidad de este caso es: $O(n^2)$

Caso promedio:
La complejidad de este caso es: $O(n^2)$

3. CONCLUSIÓN

Finalmente, podría decir que es muy interesante entender como es el funcionamiento de los distintos métodos de programar y es a un más interesante el saber que herramientas hacen más factible la implementación de estos.

El método de ordenamiento por burbuja es uno de los métodos menos eficientes que hay, ya que tiene un tiempo más extenso al momento de ejecutarlo, por ellos se implementó el mejoramiento a este, y así fue como se conoció el método de burbuja optimizada, como su nombre lo dice, hace que el desarrollo del código sea más eficiente y eficaz a la vez.

En el tema de complejidad algorítmica, es totalmente necesario comprender cuál es el nivel de dificultad que presentan para buscar la manera de que sean más accesibles al entendimiento de otros programadores y que cumplan con las indicaciones de manera más rápida.

No es necesario ser un experto en lenguajes de programación o en temas relacionados a métodos, con el hecho de tener una idea de como estructurarlo y que funciones queremos que ejecute nuestro programa, con eso es necesario para poder crear algo que sea considerablemente óptimo para utilizar.

Estos tipos de reportes o de información, son de gran utilidad, para ayudarnos a comprender como es el funcionamiento de las herramientas que tenemos como programadores y a su vez conocer cuáles de ellas nos pueden servir más y en que momento es necesario usarlas. Talvez en un futuro podremos implementar un método basado en nuestros conocimientos y sobre todo en las necesidades que queramos cubrir con este.

4. REFERENCIAS

https://tutospoo.jimdofree.com/tutoriales-java/m%C3%A9todos-de-ordenaci
C3%B3n/burbuja-optimizado/
https://runestone.academy/ns/books/published/pythoned/SortSearch/
ElOrdenamientoBurbuja.html
https://www.infor.uva.es/~mserrano/EDI/cap4.pdf
http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro9/mtodo_de_
burbuja.html
https://dcodingames.com/ordenamiento-de-datos-por-el-metodo-burbuja/
https://www.europeanvalley.es/noticias/la-complejidad-de-los-algoritmos
https://eiposgrados.com/blog-python/tipos-de-algoritmos-de-ordenacion-e
#:~:text=Ordenamiento%20de%20burbuja%20(Bubble%20Sort),-
%20algoritmo%20de&text=Comienza%20comparando%20los%20dos%20primeros,
e%20intercambiamos%20si%20fuera%20necesario.
https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_complejidad_
computacional
https://numerentur.org/complejidad-computacional/
https://www.europeanvalley.es/noticias/la-complejidad-de-los-algoritmos
#:~:text=La%20complejidad%20de%20un%20algoritmo,el%20contexto%
20de%20los%20algoritmos.
http://puntocomnoesunlenguaje.blogspot.com/2012/07/metodo-de-ordenacion-
html
http://aniei.org.mx/paginas/uam/CursoAA/curso_aa_02.html
https://www.freecodecamp.org/espanol/news/introduccion-a-la-complejidad-
https://juncotic.com/ordenamiento-de-burbuja-algoritmos-de-ordenamiento