Algoritmos de Búsqueda y Ordenamiento en Python.

**Alumnos:** Sebastián Rapetti y Rosas Alejo.

**Materia:** Programación 1.

**Profesor:** Sebastián Bruselario.

**Fecha de entrega:** 9 / 06 / 2025

**Introducción:**

Los algoritmos de búsqueda y ordenamientos de datos se eligió porque se considera una rama esencial para la programación, el desarrollo y rendimiento de los programas.

El tema de Algoritmos de búsqueda y ordenamientos de datos cobra relevancia en el mundo de la programación ya que estos facilitan la búsqueda de datos que se requieran, también son fundamentales para el procesamiento y manipulación de estos mismos en el área del desarrollo de aplicaciones o el desarrollo web.

Estos reducen la complejidad de un problema, dando como ventajas, la eficiencia, precisión, flexibilidad y ahorro de tiempo.

Como objetivos a cumplir en el desarrollo de este trabajo se tienen principalmente en cuenta los siguientes puntos:

* Mejor entendimiento y comprensión del tema.
* Poder ser capaces de aplicar estos algoritmos en un ámbito profesional.
* Fortalecer habilidades de programación y resolución de problemas.
* Aplicar los conocimientos adquiridos en la misma carrera.
* Analizar la eficiencia de distintos tipos de algoritmos.

**Marco Teórico**

**Algoritmos de ordenamiento:**

Estos mismos desarrollan un papel fundamental en el desarrollo de Software ya que permiten mayor eficiencia, manipulación y control en grandes volúmenes de datos, si se implementan de forma correcta y adecuada causan una mejora significativa en el rendimiento de aplicaciones y procesamiento de datos.

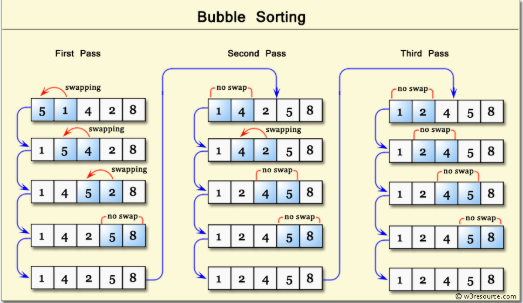
Los algoritmos o métodos de ordenamiento se definen como la disposición de datos en un orden determinado, los cuales, mediante diferentes técnicas de ordenanza, pueden reorganizar una lista de elementos de forma secuencial ya sea ascendente o descendente. Este método se utiliza principalmente para poder representar datos de una forma más comprensible.

Estos son fundamentales para la optimización de tareas y permiten un mayor acceso y eficiencia a los datos.

**Tipos de algoritmos de ordenamiento:**

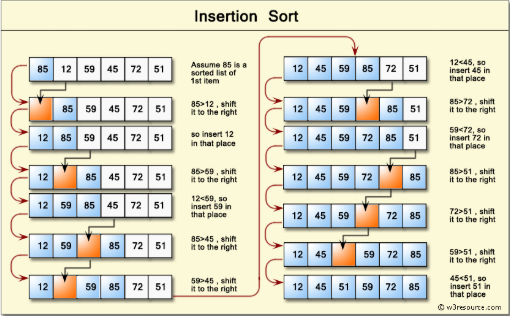
**1 – Ordenamiento de burbujas o BubbleSort:**

Este tipo de ordenamiento es un algoritmo básico en la ordenación, compara las entradas adyacentes de una lista y las intercambia hasta que estén en el orden correcto. Esto lo hace de manera que el algoritmo recorre las secciones sin ordenar de las listas, hasta llegar al final de esta para luego volver a empezar y que se repita hasta el penúltimo elemento.



**2 – Ordenamiento por inserción:**

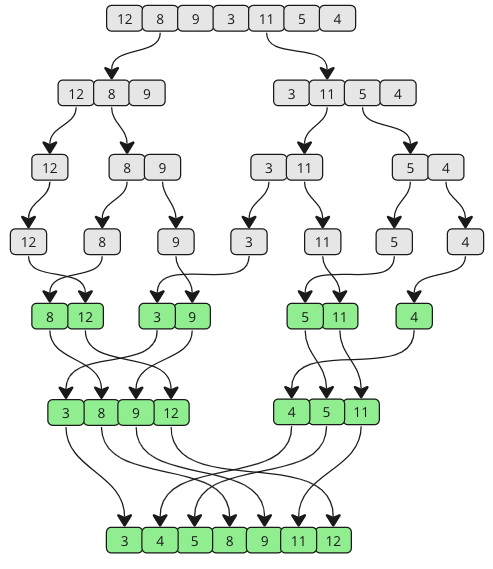
El ordenamiento por inserción se segmenta en dos secciones las cuales son: ordenada y desordenada, la intención es recorrer la sección desordenada e ir colocando cada elemento de la lista en el lugar que le corresponde dentro de la parte ordenada.



**3 – Ordenación por combinación:**

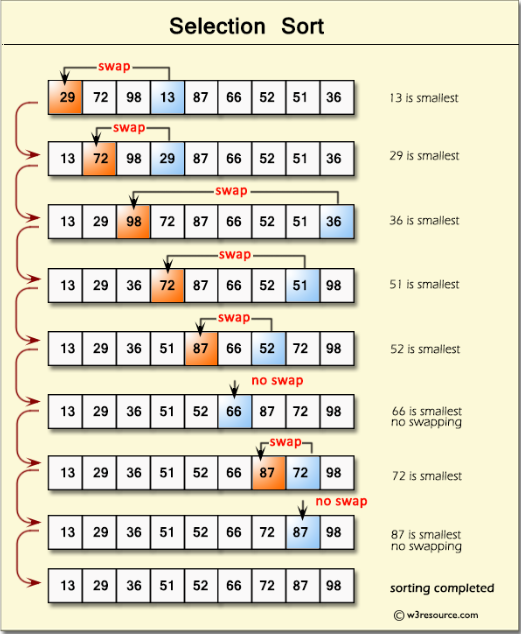
Este tipo de ordenamiento es un poco complejo cuando se quiere dividir las listas, este consta de dos pasos principales los cuales son:

* Imaginemos que la matriz a ordenada contiene N elementos. El paso inicial del algoritmo consiste en fragmentar la lista desordenada hasta obtener N sublistas, cada una compuesta por un único elemento.
* Como consecuencia el algoritmo fusiona constantemente todas las sublistas, dos a la vez. Este proceso se repite hasta que todos los elementos se hayan fusionado para formar una única matriz ordenada.



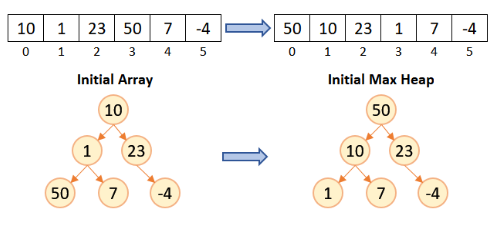
**4 – Ordenamiento por selección:**

El tipo de ordenamiento por selección básicamente busca el valor mínimo en la matriz de entrada y lo mueve a la matriz ordenada. Este proceso se repite hasta que toda la matriz este ordenada la cual se logra dividiendo la lista o matriz de entrada en dos partes y después moviendo constantemente elementos de la lista desordenada a la ordenada.



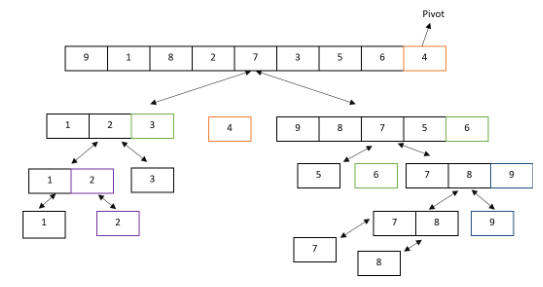
**5 – Ordenamiento por montón:**

El ordenamiento por Heap es otro tipo de algoritmo que opera dividiendo matrices o listas en dos secciones, una ordenada y otra sin ordenar. El objetivo de esta es identificar el elemento mayor transformando la porción desordenada en una estructura de datos tipo montículo.



**6 – Ordenación rápida:**

El algoritmo de ordenamiento rápido o Quicksort es básicamente la estrategia de dividir la lista o matriz de entrada. Es muy utilizado debido a su alta eficiencia y a que no requiere espacio adicional significativo a diferencia del ordenamiento por combinación. El procedimiento consiste en seleccionar un elemento de referencia, ubicarlo en su posición ordenada y reorganizar los elementos menores a su izquierda y los mayores a su derecha. Esta operación se aplica recursivamente hasta alcanzar el orden total de la secuencia.



**¿Qué necesidades resuelven?**

La búsqueda permite encontrar rápidamente datos específicos dentro de una estructura, tiene mucha utilidad cuando por ejemplo:

* Tenes grandes volúmenes de datos y se necesita acceder a uno en particular.
* Querés corroborar si un elemento existe en una colección.
* Necesitas recuperar información asociada a una clave.

Todo esto por parte de la Búsqueda, en la parte de **Ordenamiento** se buscan cubrir otras necesidades como, por ejemplo:

* Mejorar la legibilidad de los datos
* **Acelerar búsquedas,** ya que muchos algoritmos de búsqueda son más rápidos en listas ordenadas.
* **Detectar patrones o valores extremos**.
* **Realizar análisis más precisos**, como encontrar tendencias o agrupar datos similares.

**Conclusiones:**

Como reflexión o conclusiones finales podemos hacernos las siguientes preguntas:

**¿Qué salió bien?**

* La selección del tema fue una buena decisión ya que los algoritmos de búsqueda y ordenamiento en Python son una rama fundamental en la programación, permiten trabajar la lógica y la eficiencia del código.
* El marco teórico expone con claridad los tipos de algoritmos que hay, su importancia y cómo funcionan cada uno de ellos.
* Se aplico de forma correcta la parte practica tomando como ejemplos algunos de los algoritmos clásicos

**¿Qué pudimos aprender?**

* Al comparar diferentes algoritmos se puede ver como algunos son más óptimos según el contexto.
* También se pudo fortalecer la capacidad de estructurar funciones claras, reutilizables y bien organizadas.
* También se pudo avanzar en el cómo presentar y poder comunicar un proyecto técnico de forma clara.

**¿Qué posibilidades futuras tiene el proyecto?**

* Una de las posibilidades futuras que tiene este proyecto podría ser el ampliarlo a estructuras de datos más complejas.
* Se podría utilizar o aplicar una interfaz grafica como Tkinter para poder visualizar de mejor manera como se ordenan los elementos paso a paso.

Este proyecto cumple con los objetivos planteados: poder comprender y aplicar algoritmos de búsqueda y ordenamiento, también el mejorar habilidades de programación. Si bien hay un sinfín de cosas que se pueden mejorar, se considera que es un buen punto de partida para poder realizar proyectos técnicos más ambiciosos a futuro.

**Bibliografía:**

* <https://www.geeksforgeeks.org/sorting-algorithms-in-python/>
* <https://www.xccelerate.co/blog/6-types-of-sorting-algorithms-to-use-in-python>
* <https://medium.com/@mise/algoritmos-de-b%C3%BAsqueda-y-ordenamiento-7116bcea03d0>