Estudiante: Brian Ramirez Arias
Carnet:2024109557
Parte teórica de ordenamientos:

- Burbuja: El ordenamiento burbuja hace múltiples pasadas a lo largo de una lista. Compara los ítems adyacentes e intercambia los que no están en orden. Cada pasada a lo largo de la lista ubica el siguiente valor más grande en su lugar apropiado. Un ordenamiento burbuja **se considera frecuentemente como el método de ordenamiento más ineficiente** ya que debe intercambiar ítems antes de que se conozca su ubicación final.
  -Posible autor: Edward H. Friend
  -Posible año de creación: 1950
  -Tiempo: Es el mas lento.
  -Ventajas:
    *Eficaz
    *Sencillo
    *Código reducido para realiza el ordenamiento
  -Desventajas:
    *Consume bastante tiempo de computadora
    *Requiere muchas lecturas/escrituras en memoria
  -Referencia:
  https://runestone.academy/ns/books/published/pythoned/SortSearch/ElOrdenamientoBurbuja.html
  https://ed.team/comunidad/ventajas-y-desventajas-del-metodo-de-ordenacion-por-burbuja
- Selección: El ordenamiento por **selección mejora el ordenamiento burbuja** haciendo un sólo intercambio por cada pasada a través de la lista. Para hacer esto, un ordenamiento por selección busca el valor mayor a medida que hace una pasada y, después de completar la pasada, lo pone en la ubicación correcta.
  -Posible autor: Vojtěch Jarník
  -Posible año de creacion: 1960
  -Tiempo: Lento, mas rápido que burbuja.
  -Ventajas: Es fácil su implementación. No requiere memoria adicional. Realiza pocos intercambios. Tiene un rendimiento constante, pues existe poca diferencia entre el peor y el mejor caso.
  -Desventajas: Es lento y poco eficiente cuando se usa en listas grandes o medianas. Realiza numerosas comparaciones.

  -Referencia:
  https://runestone.academy/ns/books/published/pythoned/SortSearch/ElOrdenamientoPorSeleccion.html
  https://www.ecured.cu/Algoritmo_de_ordenamiento_por_selecci%C3%B3n

- Inserción: Funciona de una manera ligeramente diferente. Siempre mantiene una sublista ordenada en las posiciones inferiores de la lista. Cada ítem nuevo se "inserta" de vuelta en la sublista previa de manera que la sublista ordenada sea un ítem más larga.
  -Posible autor: Donald Knuth
  -Posible año de creacion: 1968
  -Tiempo: Intermedio
  -Ventajas: Es fácil su implementación y requerimientos mínimos de memoria.
  -Desventajas: Es lento y también realiza numerosas comparaciones.
  -Referencia:
  https://runestone.academy/ns/books/published/pythoned/SortSearch/ElOrdenamientoPorInsercion.html
  https://www.unlam.edu.ar/descargas/Metodos.doc
- Shell: **Mejora el ordenamiento por inserción al romper la lista original** en varias sublistas más pequeñas, cada una de las cuales se ordena mediante un ordenamiento por inserción. La manera única en que se eligen estas sublistas es la clave del ordenamiento de Shell. En lugar de dividir la lista en sublistas de ítems contiguos, el ordenamiento de Shell usa un incremento i, a veces denominado brecha, para crear una sublista eligiendo todos los ítems que están separados por i ítems.
  -Posible autor: Donald Shell
  -Posible año de creación: 1959
  -Tiempo: Intermedio. Mas rápido que inserción
  -Ventajas: Es fácil su implementación, requerimientos mínimos de memoria y este también es un algoritmo lento, pero puede ser de utilidad para listas que están ordenadas o semiordenadas, porque en ese caso realiza muy pocos desplazamientos.
  -Desventajas: Es lento y también realiza numerosas comparaciones.
  -Referencia:
  https://runestone.academy/ns/books/published/pythoned/SortSearch/ElOrdenamientoDeShell.html
  https://toaz.info/doc-view-3

- Quicksort: **Un ordenamiento rápido** primero selecciona un valor, que se denomina el valor pivote. Aunque hay muchas formas diferentes de elegir el valor pivote, simplemente usaremos el primer ítem de la lista. El papel del valor pivote es ayudar a dividir la lista. La posición real a la que pertenece el valor pivote en la lista final ordenada, comúnmente denominado punto de división, se utilizará para dividir la lista para las llamadas posteriores a la función de ordenamiento rápido.
  -Posible autor: Tony Hoare
  -Posible año de creación: 1962
  -Tiempo: Es el algoritmo más rápido hasta la fecha
  -Ventajas:
    *Muy rápido
    *No requiere memoria adicional.
    *Este es un algoritmo que puedes utilizar en la vida real.
    *Es muy eficiente.

-Desventajas

*Implementación un poco más complicada.

*Recursividad (utiliza muchos recursos).

*Mucha diferencia entre el peor y el mejor caso.

-Referencia:
https://runestone.academy/ns/books/published/pythoned/SortSearch/ElOrdenamientoRapido.html
https://toaz.info/doc-view-3

- Radix: Consiste en ordenar los números tomando en cuenta el valor relativo que tienen las cifras o dígitos de un número en un determinado sistema de numeración. La característica de este algoritmo está en que no hace comparaciones para ordenar las listas, simplemente se encarga de ir contando o agrupando los números que tengan el mismo valor relativo en determinada cifra.
  -Posible autor: Herman Hollerith
  -Posible año de creación: 1929
  -Tiempo: Intermedio.
  -Ventajas: Si las máquinas tienen la ventaja de ordenar los dígitos (sobre todo si están en binario) lo ejecutarían con mucho mayor rapidez de lo que ejecutan una comparación de dos llaves completas.
  -Desventajas: El ordenamiento es razonablemente eficiente si el número de dígitos en las llaves no es demasiado grande.
  -Referencia:
  https://toaz.info/doc-view-3
- Shake: es una mejora del método de la burbuja en la cual el proceso se realiza tanto desde la primera posición a la última del arreglo como en sentido inverso, evitando así que los elementos más pequeños tarden un mayor tiempo en "ascender" a las posiciones superiores.
  -Posible autor: No definido
  -Posible año de creación: 1956
  -Tiempo: Lento, más rápido que burbuja.
  -Ventajas: Es un algoritmo simple de entender e implementar y puede ser ligeramente más eficiente que el ordenamiento burbuja tradicional para conjuntos de datos casi ordenados
  -Desventajas: No es tan eficiente como otros algoritmos de ordenamiento más avanzados, como Quicksort o Merge Sort.
  -Referencia:
  https://codemyn.blogspot.com/2019/07/metodos-de-ordenamiento-shakersort-en-c.html
- Merge: 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.

-Posible autor: No definido
-Posible año de creación: 1948
-Tiempo: Rapido
-Ventajas:

* 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.)

-Desventajas: 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.
-Referencia:
https://runestone.academy/ns/books/published/pythoned/SortSearch/ElOrdenamientoPorMezcla.html
https://themergesort.blogspot.com/2012/05/ventajas-y-desventajas.html

- Busqueda Binaria: En lugar de buscar secuencialmente en la lista, una búsqueda binaria comenzará examinando el ítem central. Si ese ítem es el que estamos buscando, hemos terminado. Si no es el ítem correcto, podemos utilizar la naturaleza ordenada de la lista para eliminar la mitad de los ítems restantes. Si el ítem que buscamos es mayor que el ítem central, sabemos que toda la mitad inferior de la lista, así como el ítem central, se pueden ignorar de la consideración posterior.
  -Posible autor: John Mauchly y J. Presper Eckert
  -Posible año de creación: 1946
  -Tiempo: Rápido
  -Ventajas:
  *Se puede aplicar tanto a datos en listas lineales como en árboles binarios de búsqueda.
  *Es el método más eficiente para encontrar elementos en un arreglo ordenado.
  -Desventaja:
  *Este método funciona solamente con arreglos ordenados, por lo cual si nos encontramos con arreglos que no están en orden, este método, no nos ayudaría en nada.
  -Referencia:
  https://ed.team/comunidad/ventajas-y-desventajas-de-la-busqueda-binaria

- Busqueda Secuencial: Consiste en buscar un elemento en una colección desde el primer dato hasta encontrar en dato que se necesita, se pasa de uno a uno.
  -Posible autor: No definido
  -Posible año de creación: 1968
  -Tiempo: Lento
  -Ventajas:
    *Es un método sumamente simple que resulta útil cuando se tiene un conjunto de datos pequeños.
    *Si los datos buscados no están en orden es el único método que puede emplearse para hacer dichas búsquedas.
  -Desventaja:
    *Este método tiende hacer muy lento.
    *Se requiere buscar en todo el arreglo, lo que hace el proceso muy largo.
  -Referencia:
  https://ed.team/comunidad/ventajas-y-desventajas-de-la-busqueda-lineal