ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

FACULTAD DE INFORMATICA Y ELECTRONICA

CARRERA DE SOFTWARE

TEMA

MÉTODOS ITERATIVOS DE ORDENAMIENTO DE ARREGLOS UNIDIMENSIONALES (VECTORES)

ESTUDIANTE	CÓDIGO
JHOSTIN QUISPE	7365

ASIGNATURA

ESTRUCTURAS DE DATOS

PROFESOR

JULIO ROBERTO SANTILLAN CASTILLO

FECHA DE ENTREGA

15/04/2024

RIOBAMBA - ECUADOR

INTRODUCCION

Los métodos de ordenamiento son procedimientos fundamentales en el ámbito de la computación que permiten organizar listas o arreglos de datos de acuerdo con criterios específicos, generalmente numéricos o alfabéticos. Estos métodos son de vital importancia para el manejo eficiente de datos, ya que facilitan su búsqueda, análisis y procesamiento los algoritmos de ordenamiento se pueden clasificar en dos grandes categorías: métodos iterativos y métodos recursivos.

En este informe, se aborda el estudio de los métodos de ordenamiento iterativos, que se caracterizan por utilizar bucles para repetir una serie de pasos hasta que los datos estén completamente ordenados. A lo largo del informe, se describen los métodos de ordenamiento de burbuja, por inserción, por selección, por mezcla, por montón, y rápido, resaltando sus características, principios de funcionamiento y complejidades de tiempo.

El objetivo de este informe es proporcionar una visión general de estos métodos de ordenamiento iterativos, destacando sus aplicaciones y consideraciones importantes en términos de eficiencia y rendimiento. Además, se identifican las ventajas y desventajas de cada método para orientar a los lectores en la selección de la técnica de ordenamiento más adecuada según las necesidades específicas de cada situación.

DESARROLLO

Los métodos iterativos son una clase de algoritmos que operan mediante la repetición de una serie de pasos hasta que se alcance un objetivo específico. En el contexto de la programación y la computación, los métodos iterativos se utilizan para resolver problemas, procesar datos o realizar cálculos de manera sistemática.

Características Fundamentales

Los métodos de ordenamiento iterativos comparten características distintivas que los diferencian de otros enfoques:

Iteración: La base de estos métodos es la repetición de un conjunto de instrucciones hasta que se cumpla una condición de terminación. Esta estructura cíclica permite procesar cada elemento de la colección de manera individual y progresiva.

Comparaciones: Se realizan comparaciones entre pares de elementos para determinar su orden relativo. Estas comparaciones se basan en operadores de comparación como mayor que, menor que o igual que, aplicados a las propiedades o valores de los elementos.

Intercambios: En función de las comparaciones realizadas, se intercambian los elementos de la colección para colocarlos en el orden correcto. Los intercambios se llevan a cabo mediante asignaciones temporales o manipulaciones directas de la estructura de datos.

Sencillez: Suelen ser más fáciles de entender e implementar que los métodos recursivos, ya que no requieren de un razonamiento recursivo complejo. Esto los hace accesibles para programadores con diferentes niveles de experiencia.

Eficiencia: Algunos métodos iterativos, como el método de mezcla, pueden ser muy eficientes en cuanto a tiempo y espacio de memoria, especialmente para colecciones de gran tamaño.

Ejemplos de métodos iterativos en la computación incluyen los algoritmos de ordenamiento (como el ordenamiento de burbuja, por inserción, y por selección), la búsqueda secuencial, y los algoritmos de optimización. En general, los métodos iterativos son una herramienta esencial para los desarrolladores de software y los científicos de datos, ya que permiten realizar tareas complejas de manera controlada y eficiente.

Tipos de Métodos Iterativos

Los métodos iterativos son algoritmos que resuelven problemas de manera progresiva mediante la repetición de pasos controlados. En el contexto de la computación, estos métodos se aplican comúnmente en algoritmos de ordenamiento para organizar listas o arreglos de datos de acuerdo con criterios específicos. A continuación, se describen en detalle los principales tipos de métodos iterativos utilizados para ordenar datos:

Método de Ordenamiento de Burbuja (Bubble Sort):

El ordenamiento de burbuja es un algoritmo simple que compara pares de elementos adyacentes en una lista y los intercambia si están en el orden incorrecto. Este proceso se repite para cada elemento hasta que la lista esté completamente ordenada.

Como Funciona: El método de ordenamiento de burbuja es un algoritmo sencillo que recorre repetidamente el arreglo y compara elementos adyacentes, intercambiándolos si están en el orden incorrecto. Este proceso se repite hasta que no se realizan más intercambios, lo que indica que el arreglo está ordenado. A pesar de su simplicidad, el algoritmo puede ser ineficiente para arreglos grandes, ya que tiene una complejidad de tiempo de O(n^2).

Características: Este método es fácil de entender e implementar, y no requiere estructuras de datos adicionales. Es un algoritmo estable y puede manejar listas parcialmente ordenadas de forma eficiente.

Ventajas: Es intuitivo y fácil de codificar, además de ser adecuado para listas pequeñas o casi ordenadas.

Desventajas: Su complejidad de tiempo es alta $(O(n^2))$, lo que lo hace ineficiente para listas grandes. No es adecuado para aplicaciones que requieran un rendimiento rápido.

Para revisar el código de este método, este como archivo con el nombre metodo_burbuja.cp.

Método de Ordenamiento por Inserción (Insertion Sort)

El ordenamiento por inserción organiza los elementos de una lista de uno en uno, insertándolos en su posición correcta dentro de la lista ordenada.

Como Funciona: El método de ordenamiento por inserción es un algoritmo que funciona dividiendo el arreglo en dos partes: una parte ordenada y otra desordenada. El algoritmo inserta cada elemento del segmento desordenado en su posición correcta dentro del segmento ordenado. Es eficaz para arreglos pequeños o parcialmente ordenados y tiene una complejidad de tiempo promedio de O(n^2).

Características: Es un método simple y eficiente para listas pequeñas o parcialmente ordenadas. El algoritmo es estable y puede manejar listas con duplicados.

Ventajas: Tiene un rendimiento aceptable en listas pequeñas o casi ordenadas y es fácil de implementar.

Desventajas: Su complejidad de tiempo puede ser alta $(O(n^2))$, para listas grandes, lo que lo hace menos eficiente que otros métodos.

Puede revisar el código, este como archivo con el nombre metodo_insercion.cp.

Método de Ordenamiento por Selección (Selection Sort)

El ordenamiento por selección selecciona el elemento mínimo (o máximo) de la lista y lo coloca en la primera posición. Repite el proceso para los elementos restantes hasta que la lista esté ordenada.
Como Funciona: El método de ordenamiento por selección es un algoritmo que selecciona el elemento mínimo de un segmento de la lista y luego lo intercambia con el primer elemento del segmento. El algoritmo luego procede a encontrar el siguiente elemento mínimo en los elementos restantes y lo intercambia con el segundo elemento del segmento, y así sucesivamente hasta que toda la lista esté ordenada. La complejidad de tiempo de este algoritmo es O(n^2).

Características: Es un método sencillo que no requiere estructuras de datos adicionales. No es un algoritmo estable.

Ventajas: Es fácil de implementar y entender, y puede ser útil para listas pequeñas.

Desventajas: Tiene una complejidad de tiempo de $O(n^2)$, lo que lo hace ineficiente para listas grandes. No es un algoritmo estable, por lo que el orden de los elementos iguales puede cambiar.

El código se puede observar el archivo con el nombre en metodo_seleccion.cp.

Método de Ordenamiento por Mezcla (Merge Sort)

El ordenamiento por mezcla es un algoritmo basado en la estrategia de "divide y vencerás". Divide la lista en dos mitades, ordena cada mitad y luego las combina en una lista ordenada.

Como Funciona: El método de ordenamiento por mezcla es un algoritmo basado en el paradigma de divide y vencerás. El algoritmo divide el arreglo en subarreglos más pequeños, los ordena de forma recursiva y luego los fusiona para obtener un arreglo completamente ordenado. Es un algoritmo eficiente con una complejidad de tiempo de O(n log n) tanto en el peor como en el mejor caso. Es especialmente adecuado para arreglos grandes.

Características: Este método es estable y tiene una complejidad de tiempo de *O(nlogn)* en todos los casos, lo que lo hace eficiente para grandes conjuntos de datos.

Ventajas: Es eficiente y estable, y maneja listas grandes con eficacia. Además, puede adaptarse fácilmente a estructuras de datos como listas enlazadas.

Desventajas: Puede requerir memoria adicional para la mezcla de sublistas. Su implementación puede ser más compleja que otros algoritmos.

El código se encuentra en con el nombre metodo_mezcla.cp.

Método de Ordenamiento Shell (Shell Sort)

El ordenamiento Shell, también conocido como Shell Sort, es una mejora sobre el método de ordenamiento por inserción. Introduce la idea de un incremento o gap para dividir la lista en sublistas más pequeñas, que luego son ordenadas de forma independiente utilizando el método de inserción.

Como Funciona: El método Shell es una mejora del algoritmo de ordenamiento por inserción. Funciona utilizando un intervalo o "gap" que se va reduciendo progresivamente a medida que el algoritmo avanza. Al utilizar un intervalo más grande en las primeras etapas, el algoritmo puede mover

elementos de manera más eficiente que en un ordenamiento por inserción estándar. Tiene una complejidad de tiempo promedio de O(n log n) y es una opción eficaz para arreglos de tamaño mediano.

Características: El método Shell utiliza una serie de incrementos decrecientes para dividir la lista en sublistas. Al ordenar las sublistas, se reduce progresivamente el tamaño del incremento hasta llegar a 1, momento en el cual la lista completa se ordena con un último paso de inserción.

Ventajas: Este método es más eficiente que el ordenamiento por inserción tradicional, especialmente para listas grandes. Permite la personalización a través de la elección de la secuencia de incrementos, lo que puede mejorar su rendimiento.

Desventajas: No es un algoritmo estable, ya que el orden relativo de los elementos iguales puede cambiar. Además, su eficiencia depende en gran medida de la elección de la secuencia de incrementos, que puede ser compleja de determinar para un rendimiento óptimo.

El código correspondiente se puede encontrar con el nombre metodo_shell.cp.

Estos métodos iterativos ofrecen una variedad de enfoques para ordenar datos, cada uno con sus propias características, ventajas y desventajas. La elección del método más adecuado dependerá de factores como el tamaño y la estructura de los datos, así como de los requisitos de eficiencia y estabilidad. Para revisar el código de cada método, consulte los archivos mencionados.

CONCLUSION

En conclusión, los métodos iterativos de ordenamiento son técnicas esenciales en el campo de la computación y la manipulación de datos estos algoritmos permiten organizar listas y arreglos de manera sistemática según criterios numéricos o alfabéticos cada método iterativo tiene sus propias características, ventajas y desventajas, lo que los hace adecuados para diferentes contextos y tipos de datos.

El ordenamiento de burbuja es sencillo de implementar, pero ineficiente para listas grandes el ordenamiento por inserción es eficaz para listas pequeñas o parcialmente ordenadas, pero puede ser lento en listas grandes el ordenamiento por selección es fácil de entender, aunque su rendimiento es menor en comparación con otros métodos.

El ordenamiento por mezcla destaca por su eficiencia y estabilidad, especialmente para grandes cantidades de datos el ordenamiento por montón es rápido y eficiente, aunque puede no ser estable. Por último, el ordenamiento rápido es uno de los más utilizados por su velocidad, aunque su rendimiento puede variar según la elección del pivote.

En general, los métodos iterativos ofrecen soluciones flexibles y prácticas para el ordenamiento de datos es importante comprender las características y limitaciones de cada método para elegir la mejor opción en función de las necesidades específicas de cada situación. Para profundizar en el estudio de estos algoritmos, se recomienda revisar los archivos de código correspondientes en cada método se mencionó con el nombre, estos contienen las implementaciones prácticas de los métodos mencionados, lo que facilita su análisis y aplicación en diversos escenarios.

BIBLIOGRAFIA

Morera, J. D. D. M., & Polini, S. C. (2013). Comparación entre algoritmos recursivos e iterativos y su medición en términos de eficiencia. *Uniciencia*, *27*(1), 341-350. https://www.revistas.una.ac.cr/index.php/uniciencia/article/view/4959

Guijo Moreno, M. (2023). Métodos iterativos de resolución de sistemas lineales basados en subespacios de Krylov. Aplicaciones a la resolución numérica de EDP y en matrices test. https://idus.us.es/handle/11441/155990

5.11. El ordenamiento por mezcla — Solución de problemas con algoritmos y estructuras de datos. (s/f). Runestone.academy. Recuperado el 13 de abril de 2024, de https://runestone.academy/ns/books/published/pythoned/SortSearch/ElOrdenamientoPorMezcla.html

González, G. I. C. (s/f). *Ordenamiento por Método Shell*. Unam.mx. Recuperado el 13 de abril de 2024, de https://repositorio-uapa.cuaieed.unam.mx/repositorio/moodle/pluginfile.php/1472/mod_resource/content/1/contenido/index.html

METODOS DE ORDENAMIENTO DE VECTORES/ARREGLOS. (s/f). Blogspot.com. Recuperado el 13 de abril de 2024, de https://naslyuribe0507ita.blogspot.com/2011/03/metodos-de-ordenamiento-de.html

Ordenamiento de vectores: métodos iterativos y recursivos. (s/f). Scribd. Recuperado el 13 de abril de 2024, de https://es.scribd.com/document/414903567/METODOS-DE-ORDENAMIENTO-DE-VECTORES-docx

Paul, J., & Perfil, V. T. mi. (s/f). *Programación Java*. Blogspot.com. Recuperado el 13 de abril de 2024, de https://javajhon.blogspot.com/2020/11/ordenamiento.html

Szpiniak, A. F., & Rojo, G. A. (s/f). *Enseñanza de la programación*. Edu.ar. Recuperado el 13 de abril de 2024, de https://sedici.unlp.edu.ar/bitstream/handle/10915/14157/Documento_completo.pdf?sequence=1&isAllowed=y