**Algoritmia**

**Presentado por:**

**Jesús Acosta Jaraba**

**Presentado a:**

**Moisés Rodríguez**

**Universidad Corporativa de Colombia**

**UCC**

**Santa Marta (Magdalena)**

**Introducción**

La ordenación es una utilización esencial en computación. La generalidad de los datos producidos por un programa están ordenados de alguna modo, y muchos de los cómputos que tiene que ejecutar un programa son más y más eficientes si los datos concerniente los que operan están ordenados Búsqueda y ordenamiento constituyen ambos tareas fundamentales en la gestión de datos, fundamentalmente si se presenta de grandes volúmenes. Las dos operaciones se desarrollan con cimiento en un dato de referencia al que usualmente se llama clave.

La  búsqueda permite hallar un elemento  característico en el conjunto, mientras que el ordenamiento consiste en situarlos datos atendiendo a un criterio de manera que sea más accesible encontrar el elemento que se requiere o identificar las relaciones entre los datos.

**Objetivos**

**Objetivo general:**

* Comprender y analizar los métodos de ordenamiento y búsqueda de un programa.

**Objetivo específico:**

* Conocer el algoritmo de búsqueda.
* Conocer el algoritmo de ordenamiento.
* Investigar los diferentes tipos de algoritmos de búsqueda.
* Investigar los diferentes tipos de algoritmos de ordenamiento.

**Métodos de ordenamiento y búsqueda de un programa:**

**Algoritmos de Búsqueda**

Los procesos de búsqueda involucran recorrer un arreglo completo con el fin de encontrar algo. Lo más común es buscar el menor o mayor elemento o buscar el índice de un elemento determinado.

Para buscar el menor o mayor elemento de un arreglo, podemos usar la estrategia, de suponer que el primero o el último es el menor (mayor), para luego ir comparando con cada uno de los elementos, e ir actualizando el menor (mayor). A esto se le llama Búsqueda Lineal.

La búsqueda es una operación de vital importancia cuando se manipulan grandes

Conjuntos de datos donde localizar un elemento no es tarea fácil, como afirman Lopez, Jeder y Vega (2009: 129).

Son importantes para encontrar un dato dentro de un arreglo, para ello existen diversos algoritmos que varían en complejidad, eficiencia, tamaño del dominio de búsqueda.

**Búsqueda Secuencial**

Consiste en ir comparando el elemento que se busca con cada elemento del arreglo hasta cuándo se encuentra.

**Búsqueda Binaria**

En el caso anterior de búsqueda se asume que los elementos están en cualquier orden. En el peor de los casos deben hacerse n operaciones de comparación. Una búsqueda más eficiente puede hacerse sobre un arreglo ordenado. Una de éstas es la Búsqueda Binaria. La Búsqueda Binaria, compara si el valor buscado está en la mitad superior o inferior. En la que esté, subdivido nuevamente, y así sucesivamente hasta encontrar el valor.

**Búsqueda Lineal**

Este método consiste en tomar un dato clave que identifica al elemento que se busca y hacer un recorrido a través de todo el arreglo comparando el dato de referencia con el dato de cada posición.

**Algoritmos de ordenamiento**

Los algoritmos de ordenamiento nos permiten, como su nombre lo dice, ordenar información de una manera especial basándonos en un criterio de ordenamiento.

En la computación el ordenamiento de datos cumple un rol muy importante, ya sea como un fin en sí o como parte de otros procedimientos más complejos. Se han desarrollado muchas técnicas en este ámbito, cada una con características específicas, y con ventajas y desventajas sobre las demás.

Para que os hagáis una idea de la dificultad del problema, propongo el siguiente mini juego. Se trata de unos barriles ordenar (entre 3 y 10) con el fin de aumentar de peso. El peso de cada barril fue asignado al azar. Utilice la opción "arrastrar y soltar" para mover los barriles.

Tienes una escala no calibrada que le permite comparar el peso de barriles y estantes que pueden servir para el almacenamiento intermedio. Estos son exactamente los mismos elementos que los que están disponibles a un ordenador: una función de comparación y áreas de almacenamiento. El objetivo es, obviamente, de ordenar los barriles con las menos comparaciones e intercambios posibles.

**Algoritmo de intercambio**

Este algoritmo no es el más eficiente, pero es muy didáctico y por ende es común su utilización en los primeros cursos programación. Consiste en tomar cada elemento y compararlo con los que están a su derecha, cada vez que se identifica un par de elementos que no cumplen el criterio de ordenamiento que se está aplicando se intercambian. En cada iteración se encuentra el elemento que corresponde a una posición.

**Algoritmo de Selección**

Este método es similar al anterior en cuanto a los recorridos del vector y las

Comparaciones, pero con un número menor de intercambios. En el método de intercambio cada vez que se comparan dos posiciones y éstas no están ordenadas se hace el intercambio de los datos, de manera que en un mismo recorrido puede haber varios intercambios antes de que el número ocupe el lugar que le corresponde en el arreglo ordenado. En el método de selección, en cada recorrido se identifica el elemento que pertenece a una posición y se hace un solo intercambio.

**Algoritmo de la burbuja**

Por la forma como se hacen los recorridos y las comparaciones éste es uno de los

Algoritmos de ordenamiento más fáciles de comprender y de programar, pero como explica Joyanes (2000: 252) no es recomendable su utilización en el desarrollo de software, por ser el de menor eficiencia. La técnica de ordenamiento conocida como burbuja o burbujeo consiste en comparar el primer elemento con el segundo y si no cumplen el criterio de orden que se está aplicando se intercambian, acto seguido se pasa a comparar el segundo elemento con el tercero y si no están ordenados se intercambian también, luego se compara el tercero con el cuarto y después el cuarto con el quinto y así sucesivamente hasta comparar los elementos

**Algoritmo de Inserción**

Este método consiste en ordenar los elementos del arreglo progresivamente, comenzando por los primeros y avanzando hasta ordenarlos todos.

**Algoritmo de Donald Shell**

Es un algoritmo de inserción con saltos decrecientes diseñado por Donald Shell† y

Reconocido generalmente por el nombre de su creador. Funciona de forma similar algoritmo de inserción, pero a diferencia de éste no mueve los elementos una posición, sino varias posiciones a la vez, de manera que los elementos llegan más rápido a su destino. Este método es muy adecuado para vectores con gran cantidad de datos.

**Algoritmo de Ordenamiento Rápido**

Este algoritmo es considerado el más rápido de los estudiados en este capítulo. Fue diseñado por Tony Hoare y se basa en la técnica *dividir para vencer* de donde se desprende que ordenar dos listas pequeñas es más rápido y más fácil que ordenar una grande (Joyanes, 2000: 405).

El fundamento del algoritmo *quicksort* es la partición del vector o la lista en tres de menor tamaño distribuidas a la izquierda, al centro y a la derecha. La del centro contiene el valor utilizado como referencia para particional el vector y por tanto un solo elemento.

**Fusión de vectores ordenados**

Esta operación recibe otros nombres como intercalación (Joyanes, 1996: 343) y mezcla (López et al, 2009: 135). Se trata de unir los elementos de dos vectores o listas ordenadas y formar un tercer vector o lista también ordenados, en donde se aprovecha el orden previo delas entradas para conseguir un conjunto ordenado en menor tiempo.

**Otros algoritmos de Ordenamiento**

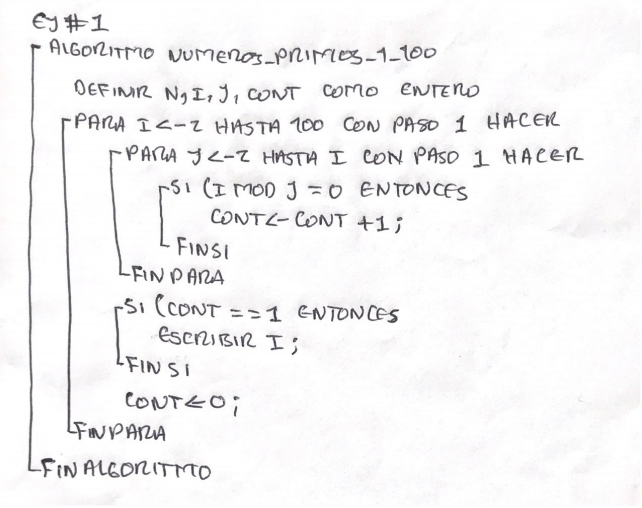
Además de los seis algoritmos de ordenamiento presentados en este trabajo existen muchos más, algunos más eficientes que los aquí estudiados y por tanto más complejos, por lo que exigen conocimientos más avanzados en ciencias de la computación. A continuación se mencionan algunos pensando en los lectores que desean profundizar en este tema.

Mergesort: es un algoritmo recursivo que aplica el principio *divide y vencerás* de forma similar a *Quicksort*. Consiste en particionar el vector o la lista en dos partes iguales, se ordena recursivamente cada una de ellas y luego se unen mediante el proceso conocido como fusión de sucesiones ordenadas o mezcla de vectores ordenados.

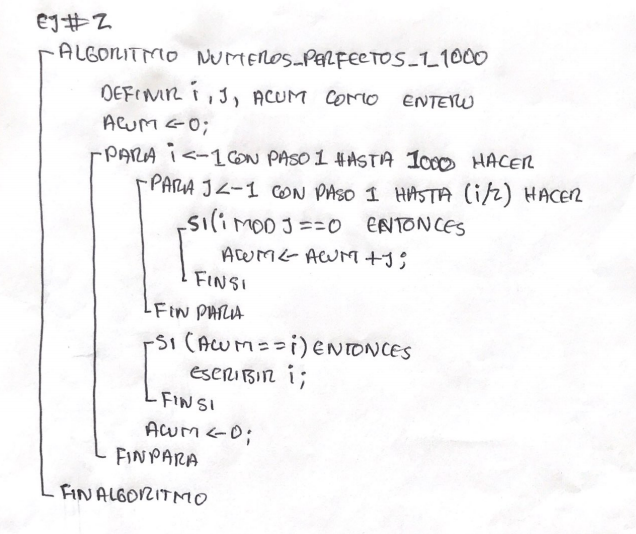
Heapsort: este algoritmo combina las ventajas de Quicksort y *Mergesort*. Utiliza una estructura de datos denominada *heap* (montón) que es un árbol binario completo del que se eliminan algunas hojas de extrema derecha.

Ordenamiento por base: este método consiste en distribuir los elementos de conjunto a ordenar en varios subconjuntos atendiendo algún criterio, se ordena cada subconjunto aplicando cualquier método de ordenamiento y luego se vuelven a unir conservando el orden. Por ejemplo, supóngase que se tiene un conjunto grande de tarjetas de invitación y se desea organizarlas alfabéticamente por el nombre de su destinatario. Se puede formar 26 subconjuntos según la letra con que comienza el nombre, luego se ordena cada montón utilizando.

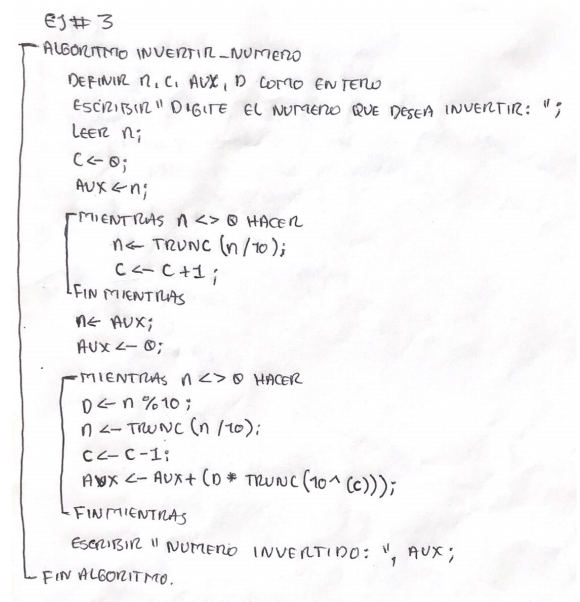
**a.**



**b.**



**c.**

****

**d. Palabra palíndroma**

El algoritmo que determinaría si una palabra o no es palíndroma en PSintseria de la siguiente forma.

Luego de que el usuario ingrese la palabra y se haya leído correctamente en el programa en la variable “palabra” se procedería a crear un siclo repetitivo, el cual tendría un variable contadora “ i ” que tomaría como valor inicial el tamaño de la palabra ingresada lo cual se hace posible con la utilización del comando “Longitud”. Este mismo siclo decrementaría con -1 la variable contadora “ i ” por cada interacción y llegaría hasta 0.

Dentro del ciclo anteriormente implementado tendíamos una variable que correspondería a la palabra invertida “palabraInvertida”, que iría acumulando letra por letra por lo que tiene la palabra original “palabra”. Esto seria posible con la implementación de el comando “Subcadena”, el cual descompone la palabra original en caracteres independientes.

Se lograría invertir la palabra indicando que la variable “palabraInvertida” se le pasaría

El carácter de la palabra original “palabra” que se encuentre en la posición “ i ”.

Al “palabraInvertida” estar acumulando lo que tiene ella misma más el carácter que está en la posición “ i ” de la palabra original “palabra” y debido a que el ciclo va decrementado se lograría tener la palabra invertida en la variable “palabra invertida”.

Por último, se compararía en la variable “palabraInvertida” que contiene la palabra invertida con la variable “palabra” que contiene la palabra original en una estructura de decisión “SI”.

Si las palabras sin iguales se arrojaría un mensaje donde diga que la palabra es palíndroma o por el contrario las palabras son diferentes se arrojaría un mensaje donde diga que la palabra no es palíndroma.

**Conclusión**

Es importante el conocimiento de ambas operaciones dada a que son empleadas profusamente en computación, de ahí su gran importancia, y se apoyan la una es la otra, por lo que las hemos agrupado y estudiando conjuntamente para el estudio de ambos algoritmos.

Concluimos que la ordenación es una utilización esencial en computación. Dado a que es importante en la producción de los datos producidos ya sea de ordenación o búsqueda.

El algoritmo de búsqueda u ordenación pueden resultar complejos a manera computacional, por lo que estudiarlo e investigarlos permite su mayor entendimiento y análisis.

**Bibliografía**

* [Bubble Sort: An archaeological algorithm analysis](http://www.cs.duke.edu/~ola/papers/bubble.pdf). Owen Astrachan. [Tipos, Estructura, Complejidad»](http://numerentur.org/ordenacion-de-vectores-i/). Consultado el 18 de mayo de 2020.
* Mezzadri, Daniels. [«¿Que es un algoritmo?»](https://dmezzadri.com/que-es-un-algoritmo/). Consultado el 20 de mayo de 2020.