***TRABAJO GRUPAL PRIMER CUATRIMESTRE***

***MONOGRAFIA***

***INTEGRANTES: Beltrán Maira Melina COMICION: 1K11***

***García Olga María***

***García Celia Elizabeth***

1. ***ANÁLISIS ALGORITMICO***

***Concepto***

*Luis Joyanes Aguilar, define al algoritmo como un método para resolver un problema. Oberon, profesor quien tituló uno de sus más famosos libros, Algoritmos + Estructuras de datos=Programas, señalándonos que sólo se puede llegar a realizar un buen programa con el diseño de un algoritmo y una correcta estructura de datos.*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del análisis algorítmico permite generalizar el número de operaciones que*

*requiere un algoritmo para encontrar la solución a un problema. Dicho en otras*

*palabras, su implementación nos lleva a*

*El uso del* ***análisis algorítmico*** *permite generalizar el número de operaciones que requiere un algoritmo para encontrar la solución a un problema. Dicho en otras palabras, su implementación nos lleva a:*

* *Determinar tiempos de respuesta.*
* *Determinar recursos computacionales.*

*La característica básica que debe tener un algoritmo es que sea correcto, es decir, que produzca el resultado deseado en tiempo finito. Adicionalmente puede interesarnos que* ***sea claro****, que este* ***bien estructurado,******que sea fácil de usar, que sea fácil de implementar y que sea eficiente****; es decir, cuál es su* ***tiempo de ejecución*** *y qué* ***cantidad de memoria utiliza****. A la cantidad de tiempo que requiere la ejecución de un cierto algoritmo se le suele llamar coste en tiempo mientras que a la cantidad de memoria que requiere se le suele llamar coste en espacio. Para realizar el análisis de un algoritmo, es necesario:*

* *Conocer la complejidad del problema que resuelve el algoritmo*
* *Conocer la dimensión de la entrada (número de elementos)*
* *Determinar el número de operaciones a realizar.*

*La complejidad de un algoritmo se representa a través de una función matemática, ya sean polinomios, logaritmos, exponentes, etc. A través de un análisis teórico, se pueden obtener funciones que representen el número de operaciones, independientemente de cómo se implementaron. Conviene buscar algoritmos correctos que mantengan tan bajo como sea posible el consumo de recursos que hacen del sistema, es decir, que sean lo más eficientes posible. Cabe hacer notar que el concepto de eficiencia de un algoritmo es un concepto relativo, esto quiere decir que ante dos algoritmos correctos que resuelven el mismo problema, uno es más eficiente que otro si consume menos recursos. Por tanto, podemos observar que el concepto de eficiencia y en consecuencia el concepto de coste nos permitirá comparar distintos algoritmos entre ellos.*

*Ventajas:*

*Elección de algoritmos eficientes para resolver problemas específicos.*

*No depende de lenguajes de programación ni de hardware*

*Desventajas:*

*Para muchos casos, en análisis no es trivial*

**Tipos de Algoritmo**

**Algoritmo Sensible:** Modifica su tiempo de ejecución según el tipo de entrada.

**Algoritmo No Sensible**: Su tiempo de ejecución es independiente al tipo de entrada.

**Algoritmo Estable:** Aquellos que teniendo clave repetida, mantiene su posición inicial igual a la final.

**Algoritmo No Estable:** Aquello que no respetan la posición inicial igual que la final teniendo claves repetidas

***2) METODOS DE ORDENAMIENTO***

***CONCEPTO:***

*Es la operación de arreglar los registros de una tabla en algún orden secuencial de acuerdo a un criterio de ordenamiento. El ordenamiento se efectúa con base en el valor de algún campo en un registro.*

*El propósito principal de un ordenamiento es el de facilitar las búsquedas de los miembros del conjunto ordenado.*

*El ordenar un grupo de datos significa* ***mover los datos o sus referencias*** *para que queden en una secuencia tal que represente* ***un orden, el cual puede ser numérico, alfabético o incluso alfanumérico, ascendente o descendente.***

*Un algoritmo de ordenamiento es un algoritmo que pone elementos de una lista o un vector en una secuencia dada por una relación de orden, es decir, el resultado de salida ha de ser* ***una permutación —o******reordenamient****o****— de la entrada que satisfaga la relación de orden dada.*** *Las relaciones de orden más usadas son el* ***orden numérico*** *y el* ***orden lexicográfico.*** *Ordenamientos eficientes son importantes para optimizar el uso de otros algoritmos (como los de búsqueda y fusión) que requieren listas ordenadas para una ejecución rápida. También es útil para poner datos en forma canónica y para generar resultados legibles por humanos.*

*Desde los comienzos de la computación, el problema del ordenamiento ha atraído gran cantidad de investigación, tal vez debido a la complejidad de resolverlo eficientemente a pesar de su planteamiento simple. Por ejemplo,* ***BubbleSort (método de la burbuja)*** *fue analizado desde 1956. Aunque muchos puedan considerarlo un problema resuelto, nuevos y útiles algoritmos de ordenamiento se siguen inventando hasta el día de hoy.*

***Tipos de Ordenamiento:***

***Ordenamiento Interno*** 🡪 Ordenamiento de datos en Memoria Principal.

**Ordenamiento Externo** 🡪 Ordenamiento de datos en Disco.

***3) ANALISIS DE LOS METODOS DE ORDENAMIENTO***

*1) METODO DE ORDEANMIENTO POR BURBUJA MEJORADA*

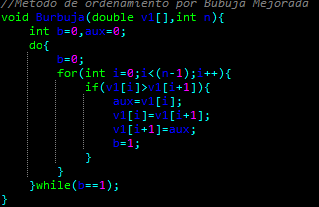
***DESCRIPCIÓN***

*La idea básica del ordenamiento de la burbuja es recorrer el conjunto de elementos en forma secuencial varias veces. Cada paso compara un elemento del conjunto con su sucesor (x[i] con x[i+i]), e intercambia los dos elementos si no están en el orden adecuado.*

***Método de ordenamiento******“Burbuja mejorada”:***

*Es una condición donde pregunta ¿el elemento que está en la posición 0 del array es mayor al elemento que se encuentra en la posición que le sigue? (en este caso en la posición 1) en caso de ser verdadera entra y continua ejecutando lo que se encuentra dentro del if, en caso de ser falso no ejecuta lo que está dentro de if, y el for recorre un vuelta más y analiza el elemento de la siguiente posición del arreglo y así hasta el final*

*El bucle do while, recorre siempre y cuando la bandera este en 1, y la bandera se pondrá en 1 solo cuando se termina de recorrer el for*

**

*Tenemos una variable llamada* ***aux*** *en el cual se guarda el elemento que se encuentra en la posición 0 del arreglo (como dijimos anteriormente*)

En la posición 1 del vector ahora se guarda el número que estaba en el aux

*El elemento que se encuentra en la posición 0 ahora es remplazado por el elemento que esta en la posición 1*

***El método de Burbuja mejorada es el más sencillo probablemente, consiste en comparar elementos de dos en dos. Si un elemento es mayor que el que está en la siguiente posición se van intercambiando hasta que se ordenan completamente***

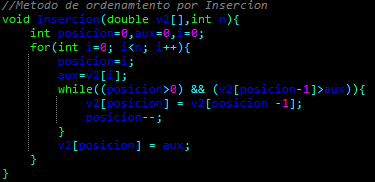
*2) METODO DE ORDENAMIENTO POR INSERCION*

***DESCRIPCIÓN***

*Este algoritmo se basa en hacer comparaciones, así que para que realice su trabajo de ordenación son imprescindibles dos cosas: un array o estructura similar de elementos comparables y un criterio claro de comparación, tal que dados dos elementos nos diga si están en orden o no.*

**Método de ordenamiento** **“Por inserción”:**

*A la variable posición se le asigna el número de posición en el que se va analizando en este primer caso 0*



*El while recorrerá si el número de posición es mayor a 0 y si el elemento de la posición anterior es mayor a lo que se guardó en mi aux*

*El elemento que se encuentra en la posición 0 se guarda en la variable aux*

*El elemento de la posición por ejem 1 se guarda ahora en la posición 0*

*Al no cumplirse el While nuevamente Sale y hace el cambio en el vector de la posición que quedo con el que está en el aux*

*El contador posición disminuye para verificar si se cambió el número de la posición anterior con el de la posición actual y vuelve a preguntar si se cumple el while nuevamente Si no se cumple no entra de nuevo*

**“El método por inserción consiste en ir comparando los elementos de dos en dos, si un elemento es mayor que el que está en la siguiente posición se intercambian”**

*3) METODO DE ORDENAMIENTO POR SELECCION*

***DESCRIPCIÓN***

*Buscas el elemento más pequeño de la lista.*

*Lo intercambias con el elemento ubicado en la primera posición de la lista.*

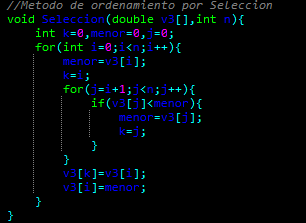
*Buscas el segundo elemento más pequeño de la lista.*

*Lo intercambias con el elemento que ocupa la segunda posición en la lista.*

*Repites este proceso hasta que hayas ordenado toda la lista.*

*Indica que el elemento de la posición 0 por ejemplo se guarda en la variable menor*

**Método de ordenamiento** **“Por selección”**



*Cuando lo encuentra lo guarda en la variable menor*

*Busca el número menor desde la posición 1 en adelante*

*El for recorrerá desde la posición 1*

*En este caso la posición 0 se guarda en k.*

*Aquí está guardando la posición en la que se encuentra dicho valor que es el menor*

En la posición 0 se guarda el número que está en la variable menor

*Usamos el parámetro k el cual indica la posición en la que está el número menor y en esa posición guardando el número que está en la posición* 0

**“El método por inserción consiste en ir comparando los elementos de dos en dos, si un elemento es mayor que el que está en la siguiente posición se intercambian”**

*4) METODO DE ORDENAMIENTO POR**QUICK SORT*

***DESCRIPCIÓN***

*Este método de ordenamiento, es un algoritmo basado en la técnica “divide y vencerás” que permite ordenar n elementos en un tiempo proporcional.*

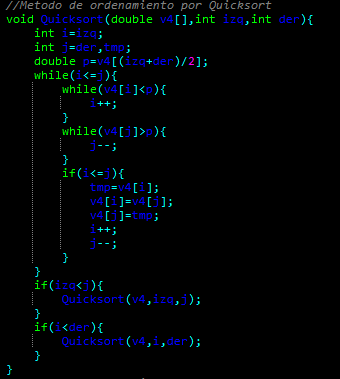
***1)***  *Recorre la lista simultáneamente con i y j: por la izquierda con i (desde el primer elemento) y por la derecha con j (desde el último elemento).*

***2)*** *Cuando lista [i] sea mayor que el pivote y lista [j] sea menor, los intercambias.*

***3)*** *Repite esto hasta que se crucen los índices.*

***4)*** *El punto donde se cruzan los índices, es la posición adecuada para colocar el pivote.*

**Método de ordenamiento** **“Por Quicksort”**



*Pregunta si el número que está en la posición 0 del vector es mayor a la variable p que tiene asignado la mitad de posiciones del vector supondremos que j disminuye una sola vez y por lo tanto será*  ***j=n-1***

*Preguntó si ¿El Numero que está en la posición 0 del vector es menor a la variable* ***p?*** *que tiene asignado la mitad de posiciones del vector supondremos que aumenta un asola ves y por lo tanto será*  ***i=1***

*El while recorre siempre y cuando el iterador sea menor o igual a j que j toma el valor de n*

*Divido el total del vector en dos y lo asigno a “p”*

*//LLAMADA DE FUNCION*

*Así se llama la función en el programa principal*

***Quicksort ( v4, 0, n );***

***void*** *Quicksort (****double***  *v4[] ,* ***int*** *izq,*  ***int*** *der);*

*Así se asigna los valores como se muestra*

*Si el if se cumple entones se pasa nuevamente como parámetros el valor de j que indica el rango de posiciones máxima que se va analizar haciendo una llamada recursiva*

*El elemento que se encuentra en la posición 1 del vector se guarda en la variable tmp*

*El elemento que se encuentra en la posición* ***n-1*** *del vector se guarda en la posición 1 de vector*

*Si el if se cumple entones se pasa nuevamente como parámetros el valor de i que indica la posición mínima en el que va a analisar nuevamente haciendo una llamada recursiva*

*El elemento que se encuentra en la variable tem se asigna en la posición* ***n-1*** *del vector*

*5) METODO DE ORDENAMIENTO POR**MEZCLA*

***DESCRIPCIÓN***

Este método aplica

la técnica divide-y-vencerás, dividiendo la secuencia de datos en dos

subsecuencias hasta que las subsecuencias tengan un único elemento, luego

se ordenan mezclando dos subsecuencias ordenadas en una secuencia

ordenada, en forma sucesiva hasta obtener una secuencia única ya ordenada.

Fue desarrollado en 1945 por John Von Neumann.

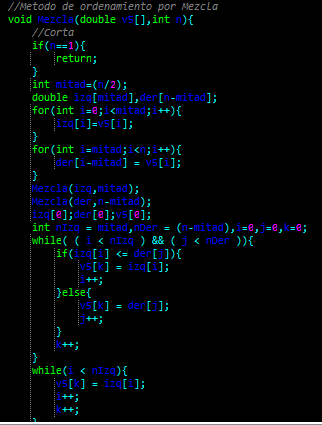
Este algoritmo también es llamado de Intercalación o combinación, debido que

combina (intercala) dos estructuras previamente ordenadas

*Mediante este algoritmo divide y vencerás se parte a partir del problema en varios subproblemas de tamaño menor, resolver cada uno de esos subproblemas por separado aplicando la misma técnica y finalmente juntar estas soluciones parciales en una solución completa del problema mayor.*

*Como siempre sucede con las soluciones recursivas, debemos encontrar un caso base en el cual no se aplica la llamada recursiva. Si la lista es pequeña ya está ordenada y no hay nada que hacer.*

*Se crea un vector llamado* ***izq****. con el máximo de elementos definido por la mitad de posiciones del vector total. Luego se crea un vector llamado* ***der*** *que se define mediante la resta del total de elementos del vector cargado inicialmente esto se da para asegurar que ningún elemento quede afuera, por ejemplo si tenemos un vector de 7 elementos, al dividirlo en dos tendríamos que la mitad valdría 3 por que solo se guarda la parte entera y entonces el vector* ***izq[3]*** *y el vector* ***der[7-3****] quedando* ***der[4]***



*Se pregunta si el número en la posición 0 del vector* ***izq*** *es menor o igual al número en la posición 0 del vector* ***der***

*El while recorre siempre cuando* ***i*** *sea menor a* ***nIzq= mitad*** *y j sean menor a* ***nDer=n -mitad***

*Son llamadas recursivas*

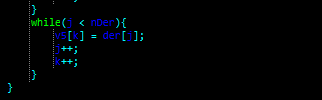
*Se extrae del vector total solo los elementos que están desde la mitad hasta el final del vector y se guarda en un vector*

*llamado der realizamos (i menor la mitad) para asegurarnos que los elementos se van extrayendo se ubiquen desde la posición 0 del vector der*

*El for recorre desde la mitad hasta el total del vector*

*Se extrae del vector total de los elementos que se encuentran desde la posición 0 hasta la mitad y se guardan en un* vector llamado **izq** definido anteriormente

*El for recorre desde cero hasta la mitad de posiciones totales del vector*



*Si el if se cumple entonces se asigna en un vector llamado* ***v5****, el número que está en la posición 0 del vector* ***izq*** *luego el iterador* ***i*** *aumenta* i

*Si el while se cumple entonces se asigna al vector llamado v5 en la posición 1 el valor del vector* ***Izq*** *en la posición 1 Luego de eso el iterador*  ***i*** *y* ***k*** *aumentan*

*Si el if no se cumple entonces se le asigna al vector llamado* ***v5*** *el número que está en posición 0 del vector der y el iterador* ***j*** *aumenta*

Bibliografía

*El while recorrerá siempre y cuando el iterador i sea menor* ***nIzq*** *que este caso vale lo que vale la variable mitad*

*El while recorrerá siempre y cuando el iterador i sea menor* ***nDer***  *que este caso vale lo que vale la variable n-mitad*

*Si el while se cumple entonces se asigna al vector llamado v5 en la posición 1 el valor del vector* ***Izq*** *en la posición 1 Luego de eso el iterador*  ***i*** *y* ***k*** *aumentan*

Vazquez Gomez Juan Bernardo, *Análisis*