

Trabajo Práctico Anual

de Matemática Discreta

Sección 2: Relaciones

**Alternativa elegida: Alternativa para programadores**

Ejercicio nº2: Implementar un método de ordenamiento algorítmico con código propio, donde dado un conjunto de valores o vector, devuelva el conjunto ordenado, indicando cantidad de pasos que debieron realizarse para hacerlo. Incluir en el documento una breve introducción a la complejidad algorítmica y su relación con el algoritmo usado.

**Introducción a la complejidad algorítmica:**

La complejidad algorítmica representa la cantidad de recursos (temporales) que necesita un algoritmo para resolver un problema y por tanto permite determinar la eficiencia de dicho algoritmo. Los criterios que se van a emplear para evaluar la complejidad algorítmica no proporcionan medidas absolutas sino medidas relativas al tamaño del problema.

La medida del tiempo tiene que ser independiente:

* de la máquina
* del lenguaje de programación
* del compilador
* de cualquier otro elemento hardware o software que influya en el análisis.

Para conseguir esta independencia una posible medida abstracta puede consistir en determinar cuantos pasos se efectúan al ejecutarse el algoritmo. El tiempo empleado por el algoritmo se mide en pasos. El coste depende del tamaño de los datos.

A la hora de evaluar el coste se debe de tener en consideración tres posibles casos:

* El coste esperado o promedio
* El coste mejor
* El coste peor

Si el tamaño de los datos es grande lo que importa es el comportamiento asintótico de la eficiencia.

Coste esperado o promedio: tiempo promedio que tarda el algoritmo en resolver el problema.

Coste mejor: Tiempo ideal, es decir, menor tiempo posible que tardaría el algoritmo en resolver el problema.

Coste peor: Tiempo mayor que podría tardar el algoritmo en resolver el problema, es decir, el peor caso de problema a resolver y el cual demandará el mayor tiempo de resolución por parte del algoritmo.

Con esto explicado, proponemos relacionar la información previamente explicada con el algoritmo que tuvimos que desarrollar para resolver el problema planteado al inicio del trabajo.

**Resolución del problema:**

El problema nos pide que dado un conjunto de datos introducidos por el usuario, el programa desarrollado identifique cada dato, lo compare y finalmente deje ordenado cada dato introducido. En este caso, elegimos trabajar con valores numéricos enteros (positivos y negativos) y elegimos un ordenamiento sencillo de menor a mayor.

Para la resolución del problema, hemos utilizado un algoritmo capaz de ordenar los valores de manera de que dado un conjunto de datos introducidos, en formato de un arreglo de datos o “array” de datos, cada valor es leído y comparado con el siguiente valor del conjunto y, en caso de que ese valor sea mayor que el siguiente, se intercambiarán de posición en el conjunto. Del mismo modo se hará con los valores restantes del conjunto y una vez que finalice el recorrido en el dicho conjunto, se volverá a recorrer tantas veces de vuelta como valores haya en el conjunto, de manera de que luego de finalizar todos los recorridos, el conjunto quedará completamente ordenado. A este procedimiento se lo conoce como burbujeo de datos y se utiliza para varios tipos de estructuras de datos. Aunque el algoritmo utilizado varía un poco en los distintos casos de estructuras utilizadas, el formato es siempre similar.

Dicho esto nos proponemos a relacionar lo explicado de la complejidad algorítmica con el algoritmo desarrollado. La complejidad algorítmica, como bien se ha explicado, determina el coste o tiempo que demanda el algoritmo en resolver el problema. Ese tiempo queda determinado por variables. En este caso, la complejidad algorítmica queda determinada por 2 variables principales muy importantes.

M: La cantidad de valores introducidos.

N: La cantidad de permutaciones o intercambios que es necesario realizar para poder ordenar los datos.

La cantidad de valores (M) determina la cantidad de recorridos que es necesario realizar en el ordenamiento. La cantidad de permutaciones (N) determina la cantidad de intercambios que debe realizar el algoritmo en un mismo recorrido.

De esta manera queda determinada la cantidad de pasos que deba realizar el algoritmo para poder ordenar el conjunto en función de la cantidad de valores y la cantidad de intercambios dentro de cada recorrido:

***Pasos a realizar = T (M (T(N)).***

Ahora es necesario aclarar el coste mejor y el coste peor en el algoritmo desarrollado.

El coste peor quedará determinado cuando el conjunto sea lo más grande posible y a la vez este exactamente en el orden inverso al que debería estar. De esta manera, las variables N y M serán las mayores y por lo tanto dejarán lugar al mayor coste de trabajo.

El coste mejor quedará determinado cuando el conjunto sea lo más pequeño posible, y a la vez que el conjunto introducido este en el orden correcto, de manera de que el algoritmo no deba realizar permutaciones dentro de cada recorrido. Eso permitirá que el número de pasos solo se limite a la cantidad de recorridos a realizar.

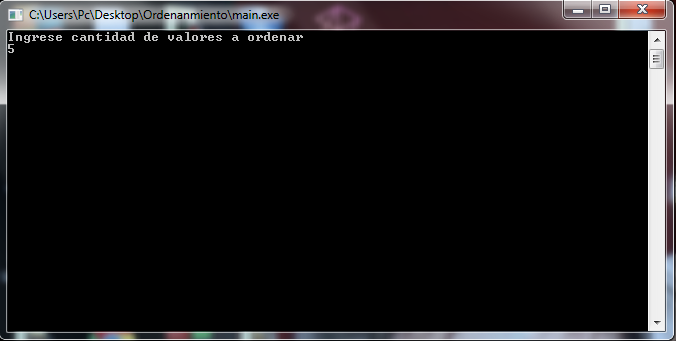
Por ejemplo, para el caso del conjunto de 5 elementos 1-2-3-4-5, el coste peor quedará determinado cuando el orden de ingreso de los datos sea en el orden inverso al que debería estar, es decir, de mayor a menor (5 4 3 2 1). En cambio, si el orden ingresado es el orden buscado, es decir, de menor a mayor (1 2 3 4 5) el costo será el mejor, ya que el orden ingresado no exige permutaciones para lograr el orden del conjunto. La cantidad de pasos entonces en el caso mejor será 5, ya que solo tiene que hacer los recorridos y no implica ninguna permutación de valores. Si en cambio, el orden es el inverso al deseado, la cantidad de pasos será 5 + cantidad de permutaciones necesarias. El coste promedio se calcula considerando todos los resultados posibles.

**Manual de Uso:**

El programa al ejecutarse solicita al usuario que ingrese la cantidad de valores que desea ordenar. Posteriormente a esto, se solicitara que ingrese los valores a ordenar. Luego de ingresar el último dato, el programa ordenara automáticamente todos los datos ingresados de menor a mayor. Finalmente, mostrará por pantalla el conjunto ordenado, y seguido a esto la cantidad de pasos que fueron necesarios para ello. Los pasos necesarios queda determinado por la fórmula:

*Cantidad de pasos = cantidad de valores ingresados + cantidad de permutaciones que fueron realizadas*

1. **Se ingresa la cantidad de valores a ordenar y se presiona la tecla *enter*:**



1. **Se ingresan los valores a ordenar, se ingresa el valor (positivo o negativo) y se presiona la tecla *enter*:**



1. **Se muestra por pantalla los datos ordenados y la cantidad de pasos necesarios.**

