**UNIVERSIDAD TECNOLOGIA NACIONAL**

**FACULTAD REGIONAL TUCUMAN**

****

**Algoritmo y estructura de datos**

**Trabajo Integrador Grupal - Primer Cuatrimestre**

**TEMA:** *Métodos de ordenamiento* **COMISION:** *1K11*

**ALUMNOS:** *Garcia José Elias*

*Giménez Nahuel Leandro*

*Romero Nataly Ayelen*

**ACTIVIDADES PROPUESTAS**

**1)** ¿Qué es el Análisis Algorítmico?

**2)** Definir Orden de un Algoritmo.

**3)** Analizar los siguientes métodos de ordenamiento:

**a.** Intercambio o burbuja mejorada.

**b.** Inserción o método de la baraja

**c.** Selección o método sencillo

**d.** Rápido o QuickSort

**e.** Por Mezcla o MergeSort

**4)** Diferencia entre los diferentes métodos.

**5)** La defensa será oral y realizada por todos los miembros el grupo. Cualquiera puede empezar y continuar otro compañero seleccionado o no por el docente.

**6)** Pueden utilizar diapositiva, videos, o bien cualquier medio que consideren apropiado para mejorar su presentación.

**7)** La presentación del trabajo se hará a través de la plataforma y la defensa de la misma es en forma oral ante profesor de teoría y en los horarios a convenir con él.

**8)** Se deberá detallar en la última hoja, las fuentes consultadas. (Libros, páginas web, docentes, documentos monográficos, etc.)

**RESOLUCION DE ACTIVIDADES**

1. **El análisis de algoritmos** es una parte importante de la Teoría de complejidad computacional más amplia, que provee estimaciones teóricas para los recursos que necesita cualquier algoritmo que resuelva un problema computacional dado. Estas estimaciones resultan ser bastante útiles en la búsqueda de algoritmos eficientes.

A la hora de realizar un análisis teórico de algoritmos es común calcular su complejidad en un sentido asintótico, es decir, para un tamaño de entrada suficientemente grande. La cota superior asintótica, y las notaciones omegas (cota inferior) y theta (caso promedio) se usan con esa finalidad.

Por ejemplo, la búsqueda binaria decimos que se ejecuta en una cantidad de pasos proporcional a un logaritmo, en O(log(n)), coloquialmente "en tiempo logarítmico". Normalmente las estimaciones asintóticas se utilizan porque diferentes implementaciones del mismo algoritmo no tienen por qué tener la misma eficiencia. No obstante, la eficiencia de dos implementaciones "razonables" cualesquiera de un algoritmo dado están relacionadas por una constante multiplicativa llamada constante oculta.

La medida exacta (no asintótica) de la eficiencia a veces puede ser computada, pero para ello suele hacer falta aceptar supuestos acerca de la implementación concreta del algoritmo, llamada modelo de computación. Un modelo de computación puede definirse en términos de un ordenador abstracto, como la Máquina de Turing, y/o postulando que ciertas operaciones se ejecutan en una unidad de tiempo. Por ejemplo, si al conjunto ordenado al que aplicamos una búsqueda binaria tiene 'n' elementos, y podemos garantizar que una única búsqueda binaria puede realizarse en un tiempo unitario, entonces se requieren como mucho log2 N + 1 unidades de tiempo para devolver una respuesta.

Las medidas exactas de eficiencia son útiles para quienes verdaderamente implementan y usan algoritmos, porque tienen más precisión y así les permite saber cuánto tiempo pueden suponer que tomará la ejecución. Para algunas personas, como los desarrolladores de videojuegos, una constante oculta puede significar la diferencia entre éxito y fracaso.}

Las estimaciones de tiempo dependen de cómo definamos un paso. Para que el análisis tenga sentido, debemos garantizar que el tiempo requerido para realizar un paso se encuentre acotado superiormente por una constante. Hay que mantenerse precavido en este terreno; por ejemplo, algunos análisis cuentan con que la suma de dos números se hace en un paso. Este supuesto puede no estar garantizado en ciertos contextos. Si por ejemplo los números involucrados en la computación pueden ser arbitrariamente grandes, dejamos de poder asumir que la adición requiere un tiempo constante (usando papel y lápiz, compara el tiempo que necesitas para sumar dos enteros de 2 dígitos cada uno y el necesario para hacerlo con dos enteros, pero de 1000 dígitos cada uno).

1. **Orden de los algoritmos**

En computación y matemáticas 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 reordenamiento— 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 y familiar. Por ejemplo, **BubbleSort** fue analizado desde **1956**. Aunque muchos puedan considerarlo un problema resuelto, nuevos y útiles algoritmos de ordenamiento se siguen inventado hasta el día de hoy (por ejemplo, el ordenamiento de biblioteca se publicó por primera vez en el 2004). Los algoritmos de ordenamiento son comunes en las clases introductorias a la computación, donde la abundancia de algoritmos para el problema proporciona una gentil introducción a la variedad de conceptos núcleo de los algoritmos, como notación de O mayúscula, algoritmos divide y vencerás, estructuras de datos, análisis de los casos peor, mejor, y promedio, y límites inferiores.

**Clasificación:**

Los algoritmos de ordenamiento se pueden clasificar en las siguientes maneras:

La más común es clasificar según el lugar donde se realice la ordenación.

**Algoritmo de ordenamiento interno:** son aquellos que son manejados usando la memoria primaria, es decir la memoria de trabajo o memoria RAM.