## Trabajo practico integrador

# PROGRAMACION I


**TECNICATURA UNIVERSITARIA EN PROGRAMACION - UTN**

**MONTENEGRO SERGIO**
**MORALES BRUNO**

# Algoritmos de Búsqueda y Ordenamiento

**Los algoritmos de búsqueda y ordenamiento son fundamentales en informática y se utilizan para organizar y encontrar datos de manera eficiente, lo que es esencial para optimizar el rendimiento de las aplicaciones.**
**Debemos comprender como se implementan y en que situaciones pueden aplicarse para realizar un desarrollo de software efectivo.**
**En este trabajo mencionaremos los aspectos más relevantes de estos algoritmos**

# ORDENAMIENTO

**El ordenamiento es un proceso fundamental en la programación que consiste en organizar un conjunto de elementos (letras, números, objetos) siguiendo un criterio especifico, como de menor a mayor o de mayor a menor. Esta tarea puede parecer simple, pero es esencial para muchas aplicaciones, desde la búsqueda de información hasta la visualización de datos.**

**Los algoritmos de ordenación se pueden clasificar en función de los siguientes parámetros:**

1. **La cantidad de intercambios o inversiones requeridas**
2. **El número de comparaciones**
3. **Ya sea que usen recursividad o no**
4. **Ya sean estables o inestables**
5. **La cantidad de espacio adicional requerido**

# ORDENAMIENTO

**Existen diferentes algoritmos de ordenamiento, cada uno con sus ventajas y desventajas en términos de velocidad y uso de memoria. Algunos de los más conocidos son:**

- **Bubble Sort (ordenamiento burbuja):** Compara elementos adyacentes e intercambia si están desordenados. Es fácil de entender, pero poco eficiente.
- **Selection Sort (ordenamiento por selección):** Encuentra el menor elemento en cada iteración y lo coloca en la posición correcta.
- **Insertion Sort (ordenamiento por inserción):** Inserta cada elemento en su lugar dentro una lista ordenada. Es útil para conjuntos pequeños.
- **Merge Sort (ordenamiento por mezcla):** Divide y vencerás: separa la lista en partes más pequeñas, las ordena y las une de nuevo.
- **Quick Sort(ordenamiento rápido):** Elige un elemento "pivote" y reordena la lista en dos partes. Es uno de los más rápidos en la práctica.
- **Heap Sort(ordenamiento por montículo):** Usa una estructura de datos llamada heap para ordenar elementos eficientemente.

# BUSQUEDA

**Es el proceso de encontrar un elemento específico dentro de una estructura de datos, como una lista, un arreglo o una base de datos. Es fundamental para optimizar el rendimiento de programas y algoritmos.**

**Existen varios tipos de algoritmos de búsqueda, entre ellos:**

- **Búsqueda lineal: Recorre la estructura de datos elemento por elemento hasta encontrar el objetivo.**
- **Búsqueda binaria: Divide la lista ordenada en mitades y descarta la mitad donde no está el elemento buscado.**
- **Búsqueda por interpolación: Similar a la binaria, pero usa una estimación matemática para decidir por dónde empezar la búsqueda. Ajusta su posición de búsqueda en función del valor clave buscado.**
- **Búsqueda por saltos: También conocida como búsqueda por bloques, está diseñada específicamente para matrices ordenadas. A diferencia de la búsqueda lineal, que examina cada elemento secuencialmente, la búsqueda por saltos salta en incrementos de tamaño fijo, omitiendo múltiples elementos para realizar búsquedas más rápidas.**

# CASO PRACTICO

**Para este trabajo vamos a utilizar el ordenamiento rápido y una búsqueda binaria.**

**ORDENAMIENTO RÁPIDO (quick sort)**

**Peor caso: es de $O(n^2)$ cuando el pivote seleccionado es el menor o mayor de cada partición (lista ordenada).**

**Mejor caso: O(log n) cuando el pivote divide la lista en dos partes aproximadamente iguales.**

**Promedio: O(log n)**

**BÚSQUEDA BINARIA.**

**Peor caso: O(log n) cuando el elemento está en uno de los extremos o cuando directamente no exista. La búsqueda se dividirá tantas veces hasta que solo quede un elemento a analizar.**

**Mejor caso: O(1), cuando el elemento se encuentra en la mitad de la lista. En la primera comparación lo encuentra.**

**Caso promedio: O(log n), el elemento se encuentra después de unas cuantas divisiones de la lista, manteniendo un rendimiento cercano a log n**

# CONCLUSIONES

**Los algoritmos de búsqueda y ordenamiento son pilares esenciales de la informática. Entender bien sus ventajas y desventajas permite optimizar los procesos de búsqueda y organización de datos. En este trabajo destacamos la búsqueda binaria, pero para poder utilizarla realizamos un ordenamiento rápido (quick sort). Ambos fueron muy efectivos.**