# Tarea 1 – INF221

**Algoritmo y complejidad**

**Evaluación Experimental de Algoritmos de Ordenamiento y Multiplicación de Matrices**

**Introducción:**

Introducción: Breve reseña de todo el trabajo realizado. Descripción a alto nivel de los problemas y algoritmos (implementados y provistos por bibliotecas), herramientas, fuentes de datos y conclusiones preliminares.

**1.1. Objetivo**

El objetivo del presente informe es evaluar y experimentar con distintos algoritmos de ordenamiento (ordenar una lista de forma ascendente) y multiplicación de matrices. Luego, en base a la experimentación realizada de cada algoritmo, estudiar su complejidad big O, que nos indica la cota superior, una aproximación de como aumenta el tiempo de ejecución según el tamaño de la entrada ‘n’ que recibirá cada función de los algoritmos a analizar. Además, teniendo la complejidad O(n) en consideración, se realiza un análisis con tablas y gráficos para facilitar la visualización de cómo afecta la complejidad O(n) a los tiempos de ejecución del algoritmo, y que a pesar de que un algoritmo tenga una complejidad un poco menor asintóticamente, esto no implica que los tiempos serán si o si menor a otro algoritmo con una complejidad un poco mayor, ya que hay algoritmos que representan efectividad con una entrada n bastante grande.

**1.2. Resumen del Trabajo**

El trabajo, como se mencionó anteriormente, estudiará algoritmos de ordenamiento y multiplicación de matrices. Este trabajo incluye los siguientes puntos:

* Implementación de algoritmos de ordenamiento (Selection Sort, Merge Sort, Quick Sort y sort de la biblioteca estándar de C++), y multiplicación de matrices (Algoritmo iterativo cúbico tradicional, Algoritmo iterativo cubico optimizado, Algoritmo de Strassen). Estos algoritmos están implementados en C++ para su estudio ya que este lenguaje de programación permite tener mayor control en la gestión de memoria en comparación con otros lenguajes como Python que son lenguajes interpretes y en una sola función puede realizar cosas que no se necesitan y perder tiempo significativo para el estudio de algoritmos de ordenamiento y multiplicación de matrices.
* Para medir el rendimiento de los algoritmos en principio se optó por una función de kcachegrind, pero debido a su complejo entendimiento, los algoritmos están medidos con una función “chronos” de las librerías de C++, este nos permite una mayor precisión en tiempo (ms), y se registraran los distintos tiempos con diferentes datasets para realizar el respectivo análisis y ver como se comporta con ayuda de un gráfico de dispersión de Excel.
* Para la creación de datos se uso un código creado en Python para una mayor facilidad y flexibilidad a la hora de generar los datasets, ya que, caso contrario, haber usado C++ para crear los datasets no hubiera sido lo más optimo. Ejemplo de dataset creado con Python:

**1.3. Herramientas Utilizadas**

Mencionar las herramientas de programación, profiling y generación de datasets que fueron utilizadas. Ejemplos:

* **Lenguaje de programación**: C++
* **Herramientas de profiling**: Valgrind, gprof
* **Generación de datasets**: [mencionas la herramienta utilizada]

**1.4. Conclusiones Preliminares**

Dar una idea general de las conclusiones que se espera obtener, como qué algoritmos se espera que tengan un mejor rendimiento en diferentes condiciones.

**Descripción de algoritmos:**

Entrega de todos los códigos como adjuntos al documento o enlaces a dónde encontrarlos en el mismo. El código debe estar documentado y seguir algún estándar de codificación adoptado por el estudiante. Para cada algoritmo, especificar una descripción general y consignar el costo del mejor y peor caso. Reseñar las funciones de bibliotecas estándar utilizadas  ¿Qué algoritmos funcionan por debajo y cuáles son sus costos?

**2. Descripción de los Algoritmos a Ser Comparados**

**2.1. Algoritmos de Ordenamiento**

* **Selection Sort**: Descripción breve del algoritmo, su complejidad en el mejor y peor caso, y las características principales (e.g., algoritmo in-place).
* **MergeSort**: Descripción del algoritmo, su enfoque divide y vencerás, y su complejidad O(nlog⁡n)O(n \log n)O(nlogn).
* **QuickSort**: Descripción del algoritmo, selección de pivote, su complejidad promedio y peor caso.

**2.2. Algoritmos de Multiplicación de Matrices**

* **Algoritmo Iterativo Cúbico Tradicional**: Explicación del enfoque básico y su complejidad O(n3)O(n^3)O(n3).
* **Algoritmo Iterativo Cúbico Optimizado**: Descripción de la optimización mediante transposición para mejorar la localidad de los datos.
* **Algoritmo de Strassen**: Descripción del algoritmo, enfoque recursivo, y su complejidad O(nlog⁡27)O(n^{\log\_2 7})O(nlog2​7).

**2.3. Bibliotecas Estándar Utilizadas**

* Mencionar si se usaron funciones o algoritmos de bibliotecas estándar, como std::sort en C++, y describir brevemente su funcionamiento y complejidad.

**Descripción de datasets:**

En esta sección se pide listar los conjuntos de datos con los que se ha trabajado, así como caracterizarlos (tamaño del dataset, valores límites u otras características consideradas). De igual forma, se pide especificar el tipo de archivos utilizado para la entrada y salida, así como el formato de los mismos.

**3. Descripción de los Datasets**

**3.1. Características de los Datasets**

* **Tamaño**: Descripción de los tamaños de los datasets utilizados (e.g., número de elementos en las listas, dimensiones de las matrices).
* **Distribución de Datos**: Descripción de cómo se distribuyen los datos (e.g., completamente desordenados, semi-ordenados).
* **Matrices**: Diferentes tipos de matrices utilizadas (e.g., matrices cuadradas, matrices de distintas dimensiones).

**3.2. Formato de los Archivos**

* Descripción del formato utilizado para almacenar los datasets (e.g., archivos de texto, binarios) y el formato de entrada y salida para los algoritmos.

**Resultados experimentales:**

En esta sección se deberán mostrar los resultados comparativos de los distintos algoritmos en forma de tablas, gráficos y comentar, dentro del texto, observaciones que consideren que vale la pena resaltar en las tablas/gráficos.

**4. Resultados Experimentales**

**4.1. Tablas y Gráficos**

* Presentación de tablas comparativas que muestren el rendimiento de cada algoritmo en función del tamaño del dataset.
* Gráficos que resalten las diferencias en tiempos de ejecución entre los algoritmos.

**4.2. Observaciones**

* Comentarios sobre las tendencias observadas en los resultados.
* Discusión sobre las situaciones en las que ciertos algoritmos se desempeñan mejor o peor.

**Conclusion:**

En esta sección se deberán consignar observaciones/comentarios/sentencias que evalúen el trabajo realizado, así como los resultados obtenidos. En especial, se aconseja abundar alrededor del poder predictivo del análisis asintótico del peor/mejor caso y sobre la importancia de aspectos relacionados con la implementación (i.e. algoritmos inplace vs no-inplace, preservación de la localidad de los datos, otros) en los resultados experimentales.

**5. Conclusiones**

**5.1. Evaluación del Trabajo Realizado**

* Reflexión sobre la efectividad de los algoritmos probados y cómo se comparan los resultados con las expectativas teóricas.

**5.2. Importancia del Análisis Asintótico**

* Discusión sobre la relevancia del análisis de complejidad asintótica para predecir el comportamiento de los algoritmos en la práctica.

**5.3. Factores de Implementación**

* Consideraciones sobre factores como la localidad de los datos, la diferencia entre algoritmos in-place y no in-place, y cómo estos afectan el rendimiento en la práctica.