ENUNCIADO TAREA 2
ALGORITMOS Y COMPLEJIDAD

«[TODO: Añadir titulo]»

Fecha límite de entrega: **21 de abril de 2025**

Equipo ALGORITMOS Y COMPLEJIDAD 2025-2

6 de octubre de 2025
00:33
Versión 0.0

Índice

1. Objetivos

El objetivo de esta tarea 2es comparar diferentes técnicas para resolver un problema específico, evaluando tanto el **tiempo de ejecución** como el **uso de memoria**.

En particular, los estudiantes deberán implementar tres enfoques distintos: un **algoritmo de fuerza bruta**, dos **algoritmos greedy** (subóptimos o no necesariamente correctos) y un **algoritmo de programación dinámica**. La idea es analizar y comparar el rendimiento de cada técnica, y además, en el caso de los algoritmos greedy, evaluar **qué tan correctas o cercanas a la óptima son las soluciones obtenidas**.

1.1. Problema

En **CppCorp**, una empresa de desarrollo de software fundada en 2001, se busca optimizar la productividad de los desarrolladores. A lo largo de los años, la empresa ha crecido y cuenta con un equipo diverso de programadores con distintos lenguajes favoritos y niveles de experiencia. En el año 2025, Carlos fue nombrado jefe de la empresa y se propuso mejorar la eficiencia y productividad de todos los equipos de desarrollo.

Los empleados están organizados en fila según el orden de ingreso a la empresa, es decir, el empleado i fue el i-ésimo en entrar. Cada desarrollador i tiene un **lenguaje de programación favorito** C_i y si programa en su lenguaje favorito su productividad es A_i, mientras que si programa en otro lenguaje su productividad es B_i.

Carlos desea formar equipos que maximicen la **productividad total de la empresa**, definida como la suma de las productividades de todos los equipos. Cada desarrollador **pertenece exactamente a un equipo**, y no hay empleados fuera de los equipos. Para ello, los equipos se forman como **segmentos contiguos no vacíos** de la fila de empleados.

La productividad individual de un empleado i dado que programa con el lenguaje L se define como

$$p_i(L) = \begin{cases} A_i & \text{si } C_i = L \\ B_i & \text{si } C_i \neq L \end{cases}$$

La productividad de un equipo se obtiene sumando las productividades individuales de todos los miembros del equipo usando el lenguaje que más miembros prefieren, y en caso de empate se toma la opción que genere la menor productividad posible. Formalmente, para un segmento $[l, r]$, la productividad del equipo se define como:

$$\text{productividad\_equipo}(l, r) = \min_{L \in \arg\max_\ell |\{i \in [l,r] : C_i = \ell\}|} \sum_{i=l}^{r} p_i(L)$$

Aquí, $\arg\max_x f(x)$ representa **el conjunto de valores** x **que maximizan** $f(x)$. En nuestra fórmula, $|\{i \in [l, r] : C_i = \ell\}|$ representa la cantidad de miembros del segmento cuyo lenguaje favorito es ℓ, y por lo tanto

$$\arg\max_\ell |\{i \in [l, r] : C_i = \ell\}|$$

es el **conjunto de lenguajes más frecuentes** en el segmento.

La **productividad total de la empresa** se obtiene sumando las productividades de todos los segmentos que forman los equipos:

$$\text{ProdTotal} = \sum_{\text{segmentos } [l,r]} \text{productividad\_equipo}(l, r)$$

Cada segmento $[l, r]$ contribuye con su propia productividad según las reglas descritas, y el objetivo es encontrar la **partición óptima** de la fila que maximice ProdTotal, asegurando que todos los desarrolladores estén incluidos en algún equipo.

2. Especificaciones

En esta sección se describen los pasos a seguir para completar la tarea, la cual consiste en desarrollar código en C++ «Implementaciones» y en realizar un informe «Informe». Se espera que cada uno de los pasos se realice de manera ordenada y siguiendo las instrucciones dadas.

> Abra este documento en algún lector de pdf que permita hipervínculos, ya que en este documento el texto en color azul suele indicar un hipervínculo.

(1) En caso de cualquier duda, contactarse directamente con **Pablo Álvarez**. Para esto pueden hacerlo por mensaje directo en discord (`pabloealvarez`) o correo electrónico (`pablo.alvarezs@sansano.usm.cl`). En caso de cualquier de modificaciones, todas se informarán tanto por aula como por discord.

(2) Todo lo necesario para realizar la tarea se encuentra en la **rama master** del repositorio de github:

`https://github.com/pabloealvarez/INF221-2025-1-TAREA-1/tree/master/code`

(3) En el repositorio del punto (2) pueden existir otras ramas, pero master siempre será donde se encuentra la información oficial.

2.1. Implementaciones

(1) Implementar cada uno los algoritmos en C++:

- Para el problema de ordenamiento de un arreglo unidimensional de números enteros, se deben implementar en C++, en los archivos correspondientes, los algoritmos de ordenamiento SELECTION SORT, MERGE SORT, QUICK SORT y el algoritmo SORT de la librería estándar de C++.
  - code/sorting/algorithms/selectionsort.cpp
  - code/sorting/algorithms/mergesort.cpp
  - code/sorting/algorithms/quicksort.cpp
  - code/sorting/algorithms/sort.cpp

  El algoritmo SORT ya se encuentra implementado en el repositorio con el fin de que sea incluido en sus mediciones del punto (2) de la subsección «Implementaciones».
- Para el problema de multiplicación de matrices cuadradas de números enteros, se deben implementar en C++, en los archivos correspondientes, los algoritmos de multiplicación NAIVE y STRASSEN.
  - code/matrix_multiplication/algorithms/naive.cpp
  - code/matrix_multiplication/algorithms/strassen.cpp

(2) Implementar el programa que realizará las mediciones de tiempo y memoria en C++ (programas principales) y su respectivo `makefile`, que ejecutará los algoritmos y generará los archivos de salida en cada uno de los directorios measurements sorting y measurements matrix multiplication con los resultados de cada uno de los algoritmos.

- code/sorting/sorting.cpp
- code/matrix_multiplication/matrix_multiplication.cpp

(3) Implementar el programa que generará los gráficos en PYTHON y que se encargará de leer los archivos generados por los programas principales guardados en measurements sorting y measurements matrix multiplication, para luego graficar los resultados obtenidos y guardarlos en formato PNG en plots sorting y plots matrix multiplication.

- code/sorting/algorithms/scripts/plot_generator.py
- code/matrix_multiplication/scripts/plot_generator.py

(4) Documentar Cada uno de los pasos anteriores

- Completar el archivo `README.md` del directorio `code`
- Documentar en cada uno de sus programas, al inicio de cada archivo, fuentes de información, referencias y/o bibliografía utilizada para la implementación de cada uno de los algoritmos.

2.2. Informe

Luego de realizar las implementaciones y experimentos, se debe generar un informe en LaTeX que contenga los resultados obtenidos y una discusión sobre ellos. En el siguiente repositorio podrá encontrar el Template que **deben utilizar**, en esta entrega:

https://github.com/pabloealvarez/INF221-2025-1-TAREA-1/tree/master/report

- No se debe modificar la estructura del informe.
- Las indicaciones se encuentran en el archivo `README.md` del repositorio y en la plantilla de LaTeX.

3. Condiciones de entrega

(1) La tarea se realizará individualmente (esto es grupos de una persona), sin excepciones.

(2) La entrega debe realizarse vía `http://aula.usm.cl`, entregando la url del repositorio privado de GitHub donde se encuentra el código fuente de su tarea. Debe clonar el siguiente repositorio y crear uno nuevo privado en su cuenta de GitHub con el mismo nombre que el repositorio original (`INF221-2025-1-TAREA-1`).

`https://github.com/pabloealvarez/INF221-2025-1-TAREA-1/tree/master/code`

- El repositorio de github **DEBE SER PRIVADO**, ya que de lo contrario cualquier persona podrá acceder a su código y cometer plagio, siendo usted responsable de ello.
- **DEBERÁ DAR ACCESO A LOS AYUDANTES DE SU RESPECTIVO CAMPUS**, antes de la fecha de entrega. Para ello, se informará antes de la fecha de entrega el nombre de usuario de los ayudantes de su respectivo campus y deberá agregar a los ayudantes como colaboradores del repositorio que contiene su entrega.

(3) Si se utiliza algún código, idea, o contenido extraído de otra fuente, este **debe** ser citado en el lugar exacto donde se utilice.

(4) Asegúrese que todas sus entregas tengan sus datos completos: número de la tarea, ramo, semestre, nombre y rol. Puede incluirlas como comentarios en sus fuentes LaTeX (en TeX comentarios son desde % hasta el final de la línea) o en posibles programas. Anótese como autor de los textos.

(5) Si usa material adicional al discutido en clases, detállelo. Agregue información suficiente para ubicar ese material (en caso de no tratarse de discusiones con compañeros de curso u otras personas).

(6) No modifique `preamble.tex`, `report.tex`, `rules.tex`, estructura de directorios, nombres de archivos, configuración del documento, etc. Sólo agregue texto, imágenes, tablas, código, etc. En el códigos fuente de su informe/reporte, no agregue paquetes, ni archivos .tex.

(7) La fecha límite de entrega es el día **21 de abril de 2025** .

NO SE ACEPTARÁN TAREAS FUERA DE PLAZO.

(8) Nos reservamos el derecho de llamar a interrogación sobre algunas de las tareas entregadas. En tal caso, la nota de la tarea será la obtenida en la interrogación.

NO PRESENTARSE A UN LLAMADO A INTERROGACIÓN SIN JUSTIFICACIÓN PREVIA SIGNIFICA AUTOMÁTICAMENTE NOTA 0.

A. Casos de prueba

A.0.1. Ordenamiento de un arreglo unidimensional de números enteros

Entrada:

- Leer un arreglo unidimensional A desde el archivo `{n}_{t}_{d}_{m}.txt`.
  - n hace referencia a la cantidad de elementos (o largo del arreglo) y pertenece al conjunto $\mathcal{N} = \{10^1, 10^3, 10^5, 10^7\}$.
  - t hace referencia al tipo de matriz, y pertenece al conjunto $\mathcal{T} = \{\text{ascendente}, \text{descendente}, \text{aleatorio}\}$.
  - d hace referencia al conjunto dominio de cada elemento del arreglo. $d = \{D1, D7\}$, donde $D7$ implica que el dominio es $\{0,1,2,3,4,5,6,7,8,9\}$ y $D7$ que el dominio es $\{0,1,2,3,...,10^7\}$.
  - m hace referencia a la muestra aleatoria (o caso de prueba) y pertenece al conjunto $\mathcal{M} = \{a, b, c\}$.

Salida:

- Escribir la matriz resultante $M_1 \times M_2$ en el archivo `{n}_{t}_{d}_{m}_out.txt`.

Los programas que generan estos arreglos se encuentran en

code/sorting/scripts/array_generator.py

A.0.2. Multiplicación de matrices cuadradas de números eneteros

Entrada:

- Leer dos matrices cuadradas de entrada M_1 y M_2 desde los archivos `{n}_{t}_{d}_{m}_1.txt` y `{n}_{t}_{d}_{m}_2.txt`, respectivamente.
  - n hace referencia a la dimensión de la matriz (n filas y n columnas) y pertenece al conjunto $\mathcal{N} = \{2^4, 2^6, 2^8, 2^{10}\}$.
  - t hace referencia al tipo de matriz, y pertenece al conjunto $\mathcal{T} = \{\text{dispersa}, \text{diagonal}, \text{densa}\}$.
  - d hace referencia al dominio de cada coeficiente de la matriz $d = \{D0, D10\}$, donde $D0$ implica que el dominio es $\{0,1\}$ y $D10$ que el dominio es $\{0,1,2,3,4,5,6,7,8,9\}$.
  - m hace referencia a la muestra aleatoria (o caso de prueba) y pertenece al conjunto $\mathcal{M} = \{a, b, c\}$.

Salida:

- Escribir la matriz resultante $M_1 \times M_2$ en el archivo `{n}_{t}_{d}_{m}_out.txt`.

Los programas que generan estas matrices se encuentran en

code/matrix_multiplication/scripts/matrix_generator.py