

# REPORTE TAREA 2-3

## ALGORITMOS Y COMPLEJIDAD

### «Explorando la Distancia entre Cadenas, una Operación a la Vez»

Alejandro Vergara

16 de noviembre de 2024

00:22

#### Resumen

*Un resumen es un breve compendio que sintetiza todas las secciones clave de un trabajo de investigación: la introducción, los objetivos, la infraestructura y métodos, los resultados y la conclusión. Su objetivo es ofrecer una visión general del estudio, destacando la novedad o relevancia del mismo, y en algunos casos, plantear preguntas para futuras investigaciones. El resumen debe cubrir todos los aspectos importantes del estudio para que el lector pueda decidir rápidamente si el artículo es de su interés.*

*En términos simples, el resumen es como el menú de un restaurante que ofrece una descripción general de todos los platos disponibles. Al leerlo, el lector puede hacerse una idea de lo que el trabajo de investigación tiene para ofrecer [1].*

*La extensión del resumen, para esta entrega, debe ser tal que la totalidad del índice siga apareciendo en la primera página. Recuerde que NO puede modificar el tamaño de letra, interlineado, márgenes, etc.*

#### Índice

1. Introducción	2
2. Diseño y Análisis de Algoritmos	3
3. Implementaciones	9
4. Experimentos	10
5. Conclusiones	14
6. Condiciones de entrega	15
A. Apéndice 1	16

# 1. Introducción

La extensión máxima para esta sección es de 2 páginas.

La introducción de este tipo de informes o reportes, tiene como objetivo principal **contextualizar el problema que se va a analizar**, proporcionando al lector la información necesaria para entender la relevancia del mismo.

Es fundamental que en esta sección se presenten los antecedentes del problema, destacando investigaciones previas o principios teóricos que sirvan como base para los análisis posteriores. Además, deben explicarse los objetivos del informe, que pueden incluir la evaluación de un algoritmo, la comparación de métodos o la validación de resultados experimentales.

Aunque la estructura y el enfoque siguen principios de trabajos académicos, se debe recordar que estos informes no son publicaciones científicas formales, sino trabajos de pregrado. Por lo tanto, se busca un enfoque claro y directo, que permita al lector comprender la naturaleza del problema y los objetivos del análisis, sin entrar en detalles excesivos.

Introduction Checklist de *How to Write a Good Scientific Paper* [5], adaptada a nuestro contexto:

- Indique el **campo del trabajo** (Análisis y Diseño de algoritmos en Ciencias de la Computación), por qué este campo es importante y qué se ha hecho ya en este área, con las **citas** adecuadas de la literatura académica o fuentes relevantes.
- Identifique una **brecha** en el conocimiento, un desafío práctico, o plantee una **pregunta** relacionada con la eficiencia, complejidad o aplicabilidad de un algoritmo particular.
- Resuma el propósito del informe e introduzca el análisis o experimento, dejando claro qué se está investigando o comparando, e indique **qué es novedoso** o por qué es significativo en el contexto de un curso de pregrado.
- Evite; repetir el resumen; proporcionar información innecesaria o fuera del alcance de la materia (límitese al análisis de algoritmos o conceptos de complejidad); exagerar la importancia del trabajo (recuerde que se trata de un informe de pregrado); afirmar novedad sin una comparación adecuada con lo enseñado en clase o la bibliografía recomendada.

Recuerde que este es su trabajo, y sólo usted puede expresar con precisión lo que ha aprendido y quiere transmitir. Si lo hace bien, su introducción será más significativa y valiosa que cualquier texto automatizado. ¡Confíe en sus habilidades, y verá que puede hacer un mejor trabajo que cualquier herramienta que automatiza la generación de texto!

## 2. Diseño y Análisis de Algoritmos

Para los dos paradigmas de programación utilizados, se utilizó de estrategia la idea de calcular todos los costos posibles recursivamente, para finalmente elegir el mínimo, es decir, se tendrá una llamada recursiva cuando la función sea sustituir, insertar, eliminar o transponer, donde cada una obtendrá su respectivo costo, eligiendo así el mínimo entre estos.

La diferencia con el problema de mínima distancia entre cadenas es que ahora se podrá ejecutar con distintos costos, y también se le agrega la función de transponer, donde a esta solo se le considero intercambiar con el carácter que tiene adelante solo si los dos caracteres resultantes quedan en la posición que deberían según la cadena 2,

por ejemplo:

palabra: ab

objetivo: ba

en este caso si se podrá ejecutar una transposición,

sin embargo si se tiene algo del tipo:

palabra: acb

objetivo: ba

no se podrá resolver con transposiciones, ya que primero necesitamos eliminar 'c' para que luego sea factible la transposición, sin embargo esto no estará soportado por los algoritmos

Se tomaron otros supuestos, los cuales son, si los dos caracteres a comparar son iguales, no se comparará con ninguna función (sustituir, insertar, eliminar o transponer), ya que como son iguales se considero que el costo será 0.

La entrada consiste en dos cadenas de caracteres, la primera será la que buscaremos transformar a la segunda, estas dos las etiquetaremos, la primera será simplemente **palabra**, y la segunda será **objetivo**, estas dos serán guardadas como variables globales para las dos implementaciones, también los costos de cada función estarán designados de forma matricial/vectorial que especificara cada costo específico dependiendo de cual/es caracteres estén implicados.

Por último tenemos que la palabra en realidad no se vera modificada, puesto que no es necesario para calcular el costo mínimo, solo es necesario saber que si se quiere eliminar un carácter el tamaño de la palabra se reducirá 1, si queremos insertar un carácter, se asumirá que se insertará el correcto, por ende revisaremos el siguiente carácter de objetivo con el actual de palabra, es decir se reducirá 1 a objetivo, si se quiere sustituir, hay que reducir palabra y objetivo en 1 puesto que ya cumplimos con ese carácter, y por último si transponemos (se cumplen las condiciones) se reducirán los dos en 2.

## 2.1. Fuerza Bruta

*“Indeed, brute force is a perfectly good technique in many cases; the real question is, can we use brute force in such a way that we avoid the worst-case behavior?”*

— Knuth, 1998 [4]

Para el enfoque de fuerza bruta tenemos dos variables globales (strings), una será la palabra y la otra el objetivo. Esta función recibe dos parámetros, **i**, tamaño de la palabra a resolver - 1, y **j**, que es el tamaño del objetivo a resolver - 1, es importante recalcar que la palabra se va subdividiendo, generando así todos los subproblemas posibles y resolviéndolos con fuerza bruta.

---

### Algoritmo 1: Distancia Mínima de Edición - Fuerza Bruta

---

```

1 String palabra, objetivo
2 Procedure DISTANCIAEDICIONFB(i, j)
3   if palabra vacia (i < 0) and objetivo vacio (j < 0) then
4     return 0
5   if palabra vacia (i < 0) then
6     return (costo insertar objetivoj) + DISTANCIAEDICIONFB(i, j-1)
7   if objetivo vacio (j < 0) then
8     return (costo eliminar palabrai) + DISTANCIAEDICIONFB(i-1, j)
9   if palabra[i] == objetivo[j] then
10    return DISTANCIAEDICIONFB(i-1, j-1)
11  eliminar ← DISTANCIAEDICIONFB(i-1, j) + (costo eliminar palabrai)
12  insertar ← DISTANCIAEDICIONFB(i, j-1) + (costo insertar objetivoj)
13  sustituir ← DISTANCIAEDICIONFB(i-1, j-1) + (costo sustituir palabrai por objetivoj)
14  transponer ← ∞
15  if i y j son > 0 and palabrai == objetivoj-1 and palabrai-1 == objetivoj then
16    transponer ← DISTANCIAEDICIONFB(i-2, j-2) + (costo transponer palabrai por palabrai-1)
17  return (minimo entre eliminar, insertar, sustituir, transponer)

```

---

Analizando este algoritmo tenemos que en el peor caso se generan 4 llamadas recursivas, las cuales son:

- 1 Eliminar: DistanciaEdicionFB(*i*-1, *j*)
- 2 Insertar: DistanciaEdicionFB(*i*, *j*-1)
- 3 Sustituir: DistanciaEdicionFB(*i*-1, *j*-1)
- 4 transponer: DistanciaEdicionFB(*i*-2, *j*-2)

por ende podemos generar una relación de recurrencia que se aproxima al problema, esta es:

$$T(i, j) = 4 * T(i - 1, j - 1)$$

además tenemos que  $T(-1, -1) = 1$ , esto ya que en este caso solo retornará 0, dejando que el problema trivial de dos palabras vacías  $\in O(1)$ , ahora si tenemos una palabra de tamaño  $n$ , y un objetivo de tamaño

m, tendremos que  $T(n,m)$  ira aumentando en potencia de 4, es decir:

$$T(n, m) = 4 * T(n - 1, m - 1)$$

$$T(n, m) = 4 * 4 * T(n - 2, m - 2)$$

$$T(n, m) = 4 * 4 * \dots * T(-1, -1)$$

Hay que tener en cuenta de que la relacion de recurrencia no se cumple siempre, por ejemplo si  $i < 0$ , tenemos que ya no habran 4 llamadas recursivas sino 1, por ende como estamos considerando el peor caso tendremos una cota superior para la complejidad temporal, que será:

$$T(n, m) \in O(4^{\max(n,m)})$$

Para la complejidad espacial tenemos que, ignorando la utilizacion de matrices/vectores de costos, el algoritmo utilizara un espacio extra  $O(1)$

Ejemplo de ejecucion:  
considerando los siguientes costos:

- $\text{costo\_sub}(a, b) = 3$  si  $a \neq b$  y 0 si  $a = b$
- $\text{costo\_ins}(b) = 1$  para cualquier caracter  $b$
- $\text{costo\_del}(a) = 1$  para cualquier caracter  $a$
- $\text{costo\_trans}(a, b) = 1$  para cualquier par de caracteres adyacentes  $a, b$

y los strings:

- palabra = 'zcoal'
- objetivo = 'hola'

Con el proposito de resumir, solo se considerara el camino mas optimo para conseguir que la palabra se transforme en objetivo, comenzaremos llamando a la funcion con  $i = 4$  y  $j = 3$ .

- DistanciaEdicionFB(4, 3)

Se ejecutaran las llamadas recursivas de eliminar, insertar y sustituir, pero al momento de entrar en el condicional para transponer nos damos cuenta de que si se respeta, logrando transponer la palabra y llamandose recursivamente con  $i = 2$  y  $j = 1$ .

- DistanciaEdicionFB(2, 1)

Se comparan los dos caracteres, y notamos que son los mismos, por lo tanto solo se revisaran los siguientes caracteres  $i = 1$  y  $j = 0$ .

- DistanciaEdicionFB(1, 0)

En este momento en palabra tendremos c y en objetivo h, por lo cual se llama recursivamente a cada una de las funciones, pero no a transponer porque no cumple con las condiciones, aca se generan varios caminos pero se optara por que lo mejor sera eliminar, esto es porque ahora estamos buscando que 'zc' se transforme en 'h', con lo cual se podria simplemente sustituir c por h pero esto tiene un costo de 3, sale mas barato eliminar caracteres y luego insertarlos, lo cual tendria un costo de 2, por ende la llamada recursiva sera con  $i = 0$  y  $j = 0$ .

- DistanciaEdicionFB(0, 0)

siguiendo lo expuesto anteriormente tambien eliminaremos este caracter, por ende la llamada recursiva sera con  $i = -1$  y  $j = 0$ .

- DistanciaEdicionFB(-1, 0)

Ahora i pasa a ser negativo, esto indica que el subproblema se limita a solo agregar los caracteres que faltan (if de la linea 5 pseudocodigo), como solo nos queda esta opcion la llamada recursiva sera con  $i = -1$  y  $j = -1$ .

- DistanciaEdicionFB(-1, -1)

Como los dos indices son negativos retorna 0, indicando que ha terminado.

Recopilando los costos efectuados tenemos un total de 4, que corresponden a:

terminar(0) + insertar(1) + eliminar(1) + eliminar(1) + transponer(1)

## 2.2. Programación Dinámica

*Dynamic programming is not about filling in tables. It's about smart recursion!*

Erickson, 2019 [2]

Para este enfoque se utilizo una tabla que indica los costos de cada subproblema, haciendo que el programa realice menos llamadas recursivas, logrando un mejor rendimiento.

En esta implementacion a palabra y a objetivo se les agrega un caracter al inicio, el cual es '-', esto es para simplificar el acceso a la matriz de subproblemas, tambien tendremos la matriz que estara implementada con un vector de 2 dimensiones, esta estara inicializada con valores -1 en todos los puntos menos en el (0,0), en este punto valdra 0, que es el caso base, la solucion al problema trivial de palabra=" y objetivo = ".

---

**Algoritmo 2:** Distancia Minima de Edición - Programación Dinamica
 

---

```

1 String palabra, objetivo
2 Matriz tabla
3 Procedure DISTANCIAEDICIONPD(i, j)
4   if tablaij  $\neq$  -1 then
5     return tablaij
6   if palabra vacia (i == 0) then
7     tablaij  $\leftarrow$  (costo insertar objetivoj) + DISTANCIAEDICIONPD(i, j-1)
8     return tablaij
9   if objetivo vacio (j == 0) then
10    tablaij  $\leftarrow$  (costo eliminar palabrai) + DISTANCIAEDICIONPD(i-1, j)
11    return tablaij
12  if palabra[i] == objetivo[j] then
13    tablaij  $\leftarrow$  DISTANCIAEDICIONPD(i-1, j-1) return tablaij
14  eliminar  $\leftarrow$  DISTANCIAEDICIONPD(i-1, j) + (costo eliminar palabrai)
15  insertar  $\leftarrow$  DISTANCIAEDICIONPD(i, j-1) + (costo insertar objetivoj)
16  sustituir  $\leftarrow$  DISTANCIAEDICIONPD(i-1, j-1) + (costo sustituir palabrai por objetivoj)
17  transponer  $\leftarrow$   $\infty$ 
18  if i y j son >0 and palabrai == objetivoj-1 and palabrai-1 == objetivoj then
19    transponer  $\leftarrow$  DISTANCIAEDICIONPD(i-2, j-2) + (costo transponer palabrai por palabrai-1)
20  tablaij  $\leftarrow$  (minimo entre eliminar, insertar, sustituir, transponer)
21  return tablaij
22 Procedure SET()
23   ...
24   palabra  $\leftarrow$  " " + leertxt
25   objetivo  $\leftarrow$  " " + leertxt
26   setear tabla con valores -1
27   tabla[0][0]  $\leftarrow$  0
28   ...

```

---

El algoritmo con el enfoque de programación dinámica, específicamente utilizando el enfoque top down, es muy similar a fuerza bruta, de hecho de igual forma se calculan todos los caminos posibles, pero en este con este enfoque no se volverá a recalcular, sino que los resultados parciales (resultados del subproblema) se irán guardando, ahorrando así llamadas recursivas y mejorando el rendimiento.

Este algoritmo divide el problema en varios subproblemas, dependiendo de la última función que se ejecute a la palabra, la manera más fácil de verlo es que se fija el último carácter y se decide hacer una operación, en donde si se agrega, o se sustituye un carácter, siempre se considerará que es el que beneficia al problema, por ende tendremos una fracción del problema resuelto, faltaría el resto, ese sería el primer subproblema.

Para la complejidad temporal tenemos que lo que buscaremos realizar es llenar la matriz con los subproblemas, esta tendrá un tamaño de  $n * m$  donde  $n$  es el tamaño de palabra y  $m$  es el tamaño de objetivo, en conclusión su complejidad será:

$$T(n, m) \in O(n * m)$$

Por otro lado para la complejidad espacial tendremos que de espacio extra (ignorando espacio utilizado

para los costos) tendremos la matriz de subproblemas, la cual consta de  $n$  filas y  $m$  columnas, por ende el espacio estara definido por:

$$E(n, m) \in O(n * m)$$

Ejemplo de ejecucion: considerando los siguientes costos:

- $\text{costo\_sub}(a, b) = 3$  si  $a \neq b$  y  $0$  si  $a = b$
- $\text{costo\_ins}(b) = 1$  para cualquier caracter  $b$
- $\text{costo\_del}(a) = 1$  para cualquier caracter  $a$
- $\text{costo\_trans}(a, b) = 1$  para cualquier par de caracteres adyacentes  $a, b$

y los strings:

- palabra = 'abc'
- objetivo = 'bad'

como el tamaño de palabra y de objetivo es 3, sin embargo le agregaremos el '-' antes, que como ya fue explicado anteriormente es para acceder de manera mas sencilla a la matriz, sabiendo esto los dos strings quedaran de largo 4, por ende comenzaremos llamando a la funcion con  $i = 3$  y  $j = 3$ .

Para este ejemplo el algoritmo se comportara igual que el de fuerza bruta e ira llenando la matriz de a poco, en este problema es visible que la menor distancia es eliminar 'c', agregar 'd', transponer 'ab', con un costo total de 3, el algoritmo generara la siguiente matriz:

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 1 & 2 & 1 & 2 \\ 2 & 1 & 1 & 2 \\ 3 & 2 & 2 & 3 \end{array} \quad (1)$$

de la cual podemos apreciar que el algoritmo efectivamente revisa todos los  $i$  y los  $j$  posibles, y que el resultado será 3 ( $\text{DistanciaEdicionPD}(2,3) + 1$ ), como ya sabemos el resultado, podemos seguir el camino de regreso, como la primera operación fue 'eliminar' tenemos que ir a (2, 3), que guarda el valor 2, esto nos indica que  $\text{DistanciaEdicionPD}(2,3) = 2$ , luego la siguiente operacion fue 'agregar' en consecuencia visitaremos la matriz en (2,2), que es 1, entonces  $\text{DistanciaEdicionPD}(2,2) = 1$ , por ultimo la operacion que viene sera transponer, la cual tiene un costo de 1, que viene de  $\text{DistanciaEdicionPD}(0,0) + 1$ , pero como  $i=0$  y  $j=0$ , el resultado es 0



### 3. Implementaciones

Ahora se especificara mas acerca de las implementaciones de estos programas, estas se encuentran en el siguiente [repositorio](#), en este se encuentra la información de todo lo relacionado con el informe.

La implementación tiene 4 archivos de gran importancia, estos son:

- `costos.cpp`:

En este subprograma se tendran las implementaciones de los costos de las funciones.

- `entrada.txt`:

Aqui se ingresara en la primera linea la palabra, y en la segunda el objetivo para resolver el problema.

- `FuerzaBruta/distanciaDeEdicion.cpp`

En este archivo se guarda la implementacion con fuerza bruta del problema, esta posee dos funciones, las cuales son `distanciaEdicion(i,j)`, y `set()`, la primera resuelve el problema con fuerza bruta revisando todos los casos posibles, la segunda lee los strings del archivo `entrada.txt`.

- `prg_dinamica/distanciaDeEdicion.cpp`

En este archivo se guarda la implementacion con programacion dinamica del problema, esta posee dos funciones, las cuales son `distanciaEdicion(i,j)`, y `set()`, la primera resuelve el problema con programacion dinamica revisando todos los casos posibles pero sin recalcularlos, y la segunda lee los strings del archivo `entrada.txt` y a estas se les agrega un '-' al inicio, esto es solo para facilitar el acceso a la matriz, no perjudica a la entrada.

También se necesitan 4 archivos txt para los costos, estos tendran un tamaño donde  $n = 26$  (alfabeto inglés, solo con letras minusculas) estos son; `cost_insert.txt`, `cost_delete.txt`, `cost_replace.txt`, `cost_transpose.txt`, estos estaran guardados en la carpeta `costos`, y cada uno guardara  $n$  (insert, delete) o  $n * n$  (replace, transpose) numeros enteros, de igual manera en el repositorio hay un generador de costos aleatorio.

Para correr los programas, solo hay que modificar `entrada.txt`, ingresando la palabra y el objetivo deseados, luego compilar con:

```
g++ -o out fuerzabruta/distanciadeedicion.cpp -Wall
```

si se quiere por fuerza bruta, o

```
g++ -o out prg_dinamica/distanciadeedicion.cpp -Wall
```

si se quiere por programacion dinamica

## 4. Experimentos

La extensión máxima para esta sección es de 6 página.

*“Non-reproducible single occurrences are of no significance to science.”*

—Popper, 2005 [6]

En la sección de Experimentos, es fundamental detallar la infraestructura utilizada para asegurar la reproducibilidad de los resultados, un principio clave en cualquier experimento científico. Esto implica especificar tanto el hardware (por ejemplo, procesador Intel Core i7-9700K, 3.6 GHz, 16 GB RAM DDR4, almacenamiento SSD NVMe) como el entorno software (sistema operativo Ubuntu 20.04 LTS, compilador g++ 9.3.0, y cualquier librería relevante). Además, se debe incluir una descripción clara de las condiciones de entrada, los parámetros utilizados y los resultados obtenidos, tales como tiempos de ejecución y consumo de memoria, que permitan a otros replicar los experimentos en entornos similares. *La replicabilidad es un aspecto crítico para validar los resultados en la investigación científica computacional* [3].

Para los experimentos realizados, se utilizaron costos fijos, es decir para todos los casos se utilizaron las mismas matrices y vectores, y se utilizó un hardware con las siguientes especificaciones:

- Procesador Intel Core i3-12100F
- RAM 16 GB DDR4
- Almacenamiento ssd SATA
- SO windows 10 pro, pero los algoritmos fueron ejecutados en la máquina virtual de ubuntu 18.04.6
- Librerías utilizadas: iostream, string, fstream, random, vector, algorithm, climits

### 4.1. Dataset (casos de prueba)

La extensión máxima para esta sección es de 2 páginas.

Es importante generar varias muestras con características similares para una misma entrada, por ejemplo, variando tamaño del input dentro de lo que les permita la infraestructura utilizada en este informe, con el fin de capturar una mayor diversidad de casos y obtener un análisis más completo del rendimiento de los algoritmos.

Aunque la implementación de los algoritmos debe ser realizada en C++, se recomienda aprovechar otros lenguajes como Python para automatizar la generación de casos de prueba, ya que es más amigable para crear gráficos y realizar análisis de los resultados. Python, con sus bibliotecas como

matplotlib o pandas, facilita la visualización de los datos obtenidos de las ejecuciones de los distintos algoritmo bajo diferentes escenarios.

Debido a la naturaleza de las pruebas en un entorno computacional, los tiempos de ejecución pueden variar significativamente dependiendo de factores externos, como la carga del sistema en el momento de la ejecución. Por lo tanto, para obtener una medida más representativa, siempre es recomendable ejecutar múltiples pruebas con las mismas características de entrada y calcular el promedio de los resultados.

## 4.2. Resultados

**La extensión máxima para esta sección es de 4 páginas.**

En esta sección, los resultados obtenidos, como las gráficas o tablas, deben estar respaldados por los datos generados durante la ejecución de sus programas. Es fundamental que, junto con el informe, se adjunten los archivos que contienen dichos datos para permitir su verificación. Además, se debe permitir y especificar como obtener esos archivos desde una ejecución en otro computador (otra infraestructura para hacer los experimentos).

**No es necesario automatizar la generación de las gráficas**, pero sí es imprescindible que se pueda confirmar que las visualizaciones presentadas son producto de los datos generados por sus algoritmos, aunque la trazabilidad de los datos hasta las visualizaciones es esencial para garantizar que su validez: describa cómo se generaron los datos, cómo se procesaron y cómo se visualizaron de manera que pueda ser replicado por quien lea su informe.

Agregue gráficas que muestren los resultados de sus experimentos. La cantidad de páginas es limitada, por lo tanto escoja las gráficas más representativas y que muestren de manera clara los resultados obtenidos. Esta elección es parte de lo que se evaluará en la sección de presentación de resultados. Referencie las figuras en el texto, describa lo que se observa en ellas y por qué son relevantes.

En la [fig. 1](#) se muestra un scatterplot hecho con [TikZ](#) con el tamaño ideal cuando se incluyen dos figuras. Queda a criterio de usted el decidir qué figuras incluir.

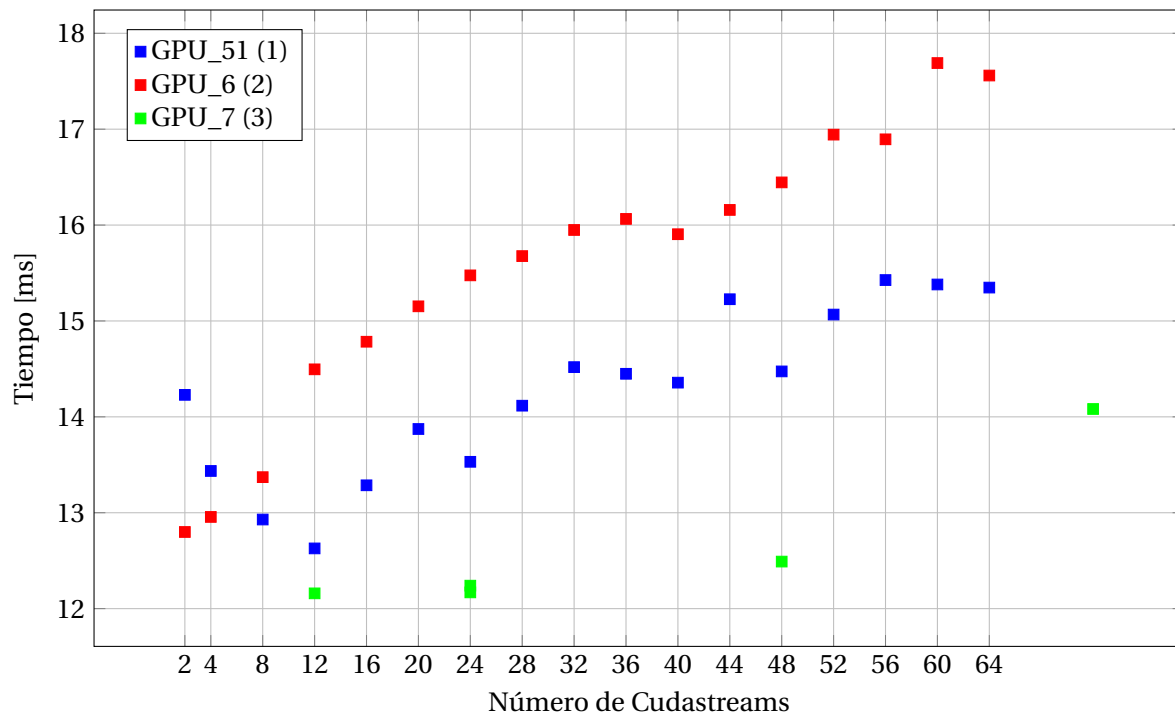


Figura 1: Ejemplo de scatterplot hecho con tikz. Tamaño ideal 1.

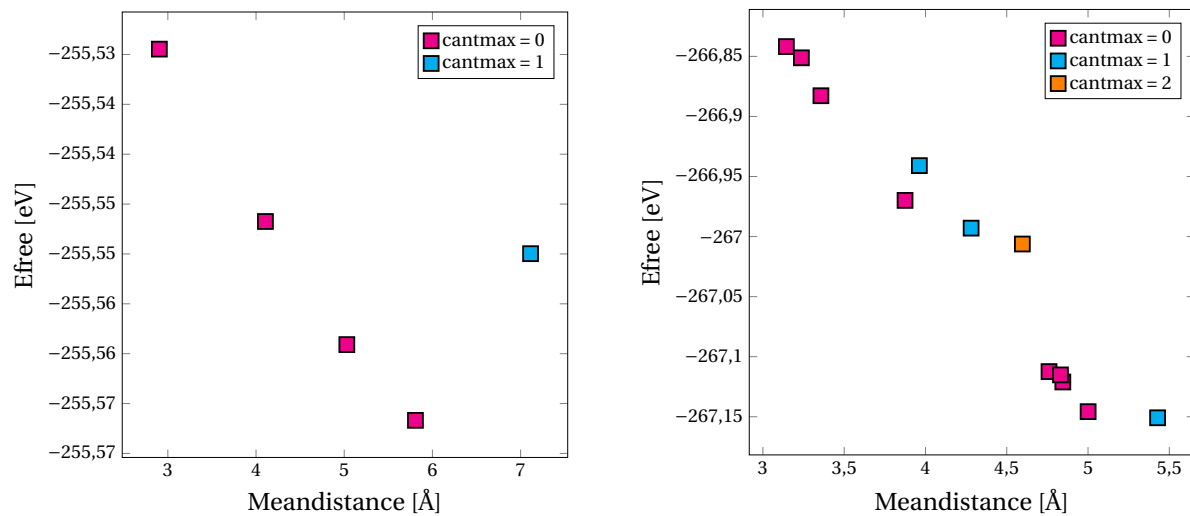


Figura 2: Ejemplo de scatterplot hecho con tikz. Tamaño ideal 2.

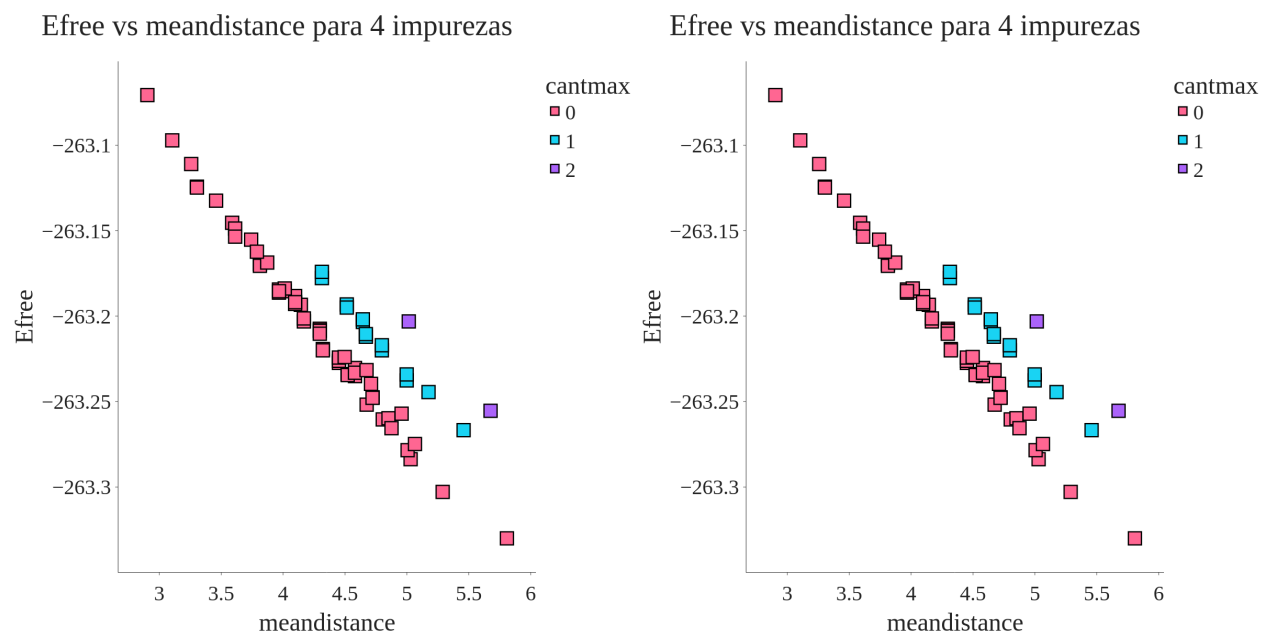


Figura 3: Ejemplo de scatterplot hecho con matplotlib.

Recuerde que es imprescindible que se pueda replicar la generación de las gráficas, por lo que usted debe incluir cómo generó esos datos y cómo podría generarlos la persona que revisa su entrega y ejecuta sus programas. Por ejemplo, si genera un scatterpolot con Tikz, usted debe explicar cómo obtener la tupla de valores que se usaron para generar la gráfica.

## 5. Conclusiones

**La extensión máxima para esta sección es de 1 página.**

La conclusión de su informe debe enfocarse en el resultado más importante de su trabajo. No se trata de repetir los puntos ya mencionados en el cuerpo del informe, sino de interpretar sus hallazgos desde un nivel más abstracto. En lugar de describir nuevamente lo que hizo, muestre cómo sus resultados responden a la necesidad planteada en la introducción.

- No vuelva a describir lo que ya explicó en el desarrollo del informe. En cambio, interprete sus resultados a un nivel superior, mostrando su relevancia y significado.
- Aunque no debe repetir la introducción, la conclusión debe mostrar hasta qué punto logró abordar el problema o necesidad planteada en el inicio. Reflexione sobre el éxito de su análisis o experimento en relación con los objetivos propuestos.
- No es necesario restablecer todo lo que hizo (ya lo ha explicado en las secciones anteriores). En su lugar, centre la conclusión en lo que significan sus resultados y cómo contribuyen al entendimiento del problema o tema abordado.
- No deben centrarse en sí mismos o en lo que hicieron durante el trabajo (por ejemplo, evitando frases como "primero hicimos esto, luego esto otro...").
- Lo más importante es que no se incluyan conclusiones que no se deriven directamente de los resultados obtenidos. Cada afirmación en la conclusión debe estar respaldada por el análisis o los datos presentados. Se debe evitar extraer conclusiones generales o excesivamente amplias que no puedan justificarse con los experimentos realizados.

## 6. Condiciones de entrega

- La tarea se realizará **individualmente** (esto es grupos de una persona), sin excepciones.
- La entrega debe realizarse vía <http://aula.usm.cl> en un **tarball** en el área designada al efecto, en el formato `tarea-2-3-rol.tar.gz` (rol con dígito verificador y sin guión).  
Dicho **tarball** debe contener las fuentes en  $\text{\LaTeX}$  (al menos `tarea-2-3.tex`) de la parte escrita de su entrega, además de un archivo `tarea-2-3.pdf`, correspondiente a la compilación de esas fuentes.
- Si se utiliza algún código, idea, o contenido extraído de otra fuente, este **debe** ser citado en el lugar exacto donde se utilice, en lugar de mencionarlo al final del informe.
- 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  $\text{\LaTeX}$  (en  $\text{\TeX}$  comentarios son desde % hasta el final de la línea) o en posibles programas. Anótese como autor de los textos.
- 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).
- No modifique `preamble.tex`, `tarea_main.tex`, `condiciones.tex`, estructura de directorios, nombres de archivos, configuración del documento, etc. Sólo agregue texto, imágenes, tablas, código, etc. En el código fuente de su informe, no agregue paquetes, ni archivos `.tex` (a excepción de que agregue archivos en `/tikz`, donde puede agregar archivos `.tex` con las fuentes de gráficos en `TikZ`).
- La fecha límite de entrega es el día **10 de noviembre de 2024**.

### **NO SE ACEPTARÁN TAREAS FUERA DE PLAZO.**

- 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. Apéndice 1

Aquí puede agregar tablas, figuras u otro material que no se incluyó en el cuerpo principal del documento, ya que no constituyen elementos centrales de la tarea. Si desea agregar material adicional que apoye o complemente el análisis realizado, puede hacerlo en esta sección.

Esta sección es solo para material adicional. El contenido aquí no será evaluado directamente, pero puede ser útil si incluye material que será referenciado en el cuerpo del documento. Por lo tanto, asegúrese de que cualquier elemento incluido esté correctamente referenciado y justificado en el informe principal.

## Referencias

- [1] Elsevier. *Differentiating between an introduction and abstract in a research paper*. Accessed: 2024-10-02. 2024. URL: <https://scientific-publishing.webshop.elsevier.com/manuscript-preparation/differentiating-between-and-introduction-research-paper/>.
- [2] Jeff Erickson. *Algorithms*. Jun. de 2019. ISBN: 978-1-792-64483-2.
- [3] Jorge Fonseca y Kazem Taghva. «The State of Reproducible Research in Computer Science». En: ene. de 2020, págs. 519-524. ISBN: 978-3-030-43019-1. DOI: [10.1007/978-3-030-43020-7\\_68](https://doi.org/10.1007/978-3-030-43020-7_68).
- [4] Donald E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. USA: Addison Wesley Longman Publishing Co., Inc., 1998. ISBN: 0201896850.
- [5] Chris Mack y Lola Muxamedova. *How to Write a Good Scientific Paper*. Nov. de 2019.
- [6] K. Popper. *The Logic of Scientific Discovery*. Routledge Classics. Taylor & Francis, 2005. ISBN: 9781134470020. URL: <https://books.google.cl/books?id=LWSBAGAAQBAJ>.