REPORTE TAREA 2 y 3
ALGORITMOS Y COMPLEJIDAD

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

Luis Zegarra Stuardo

14 de noviembre de 2024

04:20

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

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* [7], 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 (limítese 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!

—

Hoy en día, en la era de la tecnología, la **calidad y eficiencia de los programas computacionales** están creciendo enormemente. Con el avance de la tecnología, el aumento de la cantidad de datos y la complejidad de los procesos, es fundamental contar con métodos para **consultar, organizar y manejar** toda

esta información de manera eficiente. En este contexto, el campo de **Análisis y Diseño de Algoritmos en Ciencias de la Computación** cobra especial relevancia, ya que permite abordar la **resolución de problemas complejos y la creación de programas optimizados**, que pueden ejecutarse en tiempos razonables y con un consumo de memoria reducido.

Un problema recurrente en este campo es el cálculo de la **distancia mínima de edición**, ampliamente utilizado en aplicaciones como el **procesamiento de lenguaje natural, la recuperación de información, la biología computacional y la inteligencia artificial**. En estos casos, la necesidad de comparar y procesar cadenas de texto de manera precisa y eficiente es clave para obtener resultados de calidad. El cálculo de la distancia mínima de edición, además de ser una tarea común, es esencial en estos ámbitos, ya que permite medir la similitud entre dos secuencias mediante la transformación de una en otra.

En este documento se **presentará la implementación de dos algoritmos para calcular la distancia mínima de edición entre dos cadenas**, aplicando dos enfoques: **fuerza bruta** y **programación dinámica**. Estos algoritmos permitirán comparar la eficiencia de los enfoques mencionados. Ambos algoritmos incorporan operaciones de **inserción, eliminación, sustitución y transposición** con costos variables, lo cual aumenta la complejidad y utilidad del cálculo.

El algoritmo de **fuerza bruta** explora todas las posibles transformaciones de manera exhaustiva, y aunque es un enfoque simple, su tiempo de ejecución crece exponencialmente conforme aumenta el tamaño de las cadenas. Esto se traduce en una solución viable solo para casos de pequeña escala. Por otro lado, el algoritmo de **programación dinámica** optimiza la búsqueda de soluciones almacenando los resultados de subproblemas ya resueltos, lo cual reduce la cantidad de operaciones necesarias para obtener el resultado final. Sin embargo, esta optimización viene con un mayor consumo de memoria, dado que es necesario almacenar los subproblemas en una estructura de datos.

Este estudio se realiza para **verificar la eficiencia de ambos métodos**, por lo que se analizarán el **tiempo de ejecución** y el **uso de memoria** en diferentes pares de palabras con diversas características. Se dejará evidencia de las implementaciones realizadas, demostrando de manera gráfica los pros y contras de ambos algoritmos. De esta manera, se busca ayudar al lector a comprender las características de cada algoritmo en función del input que recibe, así como la cantidad de operaciones necesarias para lograr obtener una solución.

2. Diseño y Análisis de Algoritmos

La extensión máxima para esta sección es de 5 páginas.

Diseñar un algoritmo por cada técnica de diseño de algoritmos mencionada en la sección de objetivos. Cada algoritmo debe resolver el problema de distancia mínima de edición extendida, dadas dos cadenas `S1` y `S2`, utilizando las operaciones y costos especificados.

- Describir la solución diseñada.
- Incluir pseudocódigo (ver ejemplo **??**)
- Proporciones un ejemplo paso a paso de la ejecución de sus algoritmos que ilustren cómo sus algoritmos manejan diferentes escenarios, particularmente donde las transposiciones o los costos variables afectan el resultado. Haga referencias a los programas expresados en psudocódigo (además puede hacer diagramas).
- Analizar la Complejidad temporal y espacial de los algoritmos diseñados en términos de las longitudes de las cadenas de entrada $S1$ y $S2$
- Discute cómo la inclusión de transposiciones y costos variables impacta la complejidad.

Los pseudocódigos los he diseñado utilizando el paquete *Algorithm2e documentation* [3] para la presentación de algoritmos. Se recomienda consultar *Algorithm2e on CTAN* [4] y *Writing Algorithms in LaTeX* [8].

Todo lo correspondiente a esta sección es, digamos, en "lapiz y papel", en el sentido de que no necesita de implementaciones ni resultados experimentales.

Recuerde que lo importante es diseñar algoritmos que cumplan con los paradigmas especificados.

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.

Algoritmo 1: Funciones de costos para las operaciones de edición

```
1 Function COSTO_INSERT(b)
2     return costosInsert[b - 'a']
3 Function COSTO_DELETE(a)
4     return costosDelete[a - 'a']
5 Function COSTO_SUB(a, b)
6     return costosReplace[a - 'a'][b - 'a']
7 Function COSTO_TRANSPOSE(a, b)
8     return costosTranspose[a - 'a'][b - 'a']
```

no cambies nada mas

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 [6]

Algoritmo 2: Algoritmo de distancia de edición con fuerza bruta.

```
Procedure DISTANCIAEDICIONFUERZABRUTA(S1, S2, i, j, operaciones)
    if i es igual a la longitud de S1 then
        costo ← 0
        for k desde j hasta la longitud de S2 - 1 do
            Escribir "inserción" + S2[k] en operaciones
            costo ← costo + costo_insert(S2[k])
        return costo
    else if j es igual a la longitud de S2 then
        costo ← 0
        for k desde i hasta la longitud de S1 - 1 do
            Escribir "eliminación" + S1[k] en operaciones
            costo ← costo + costo_delete(S1[k])
        return costo
    costoSustitucion ← costo_sub(S1[i], S2[j]) + DISTANCIAEDICIONFUERZABRUTA(S1, S2, i + 1, j + 1, operaciones)
    costoInsercion ← costo_insert(S2[j]) + DISTANCIAEDICIONFUERZABRUTA(S1, S2, i, j + 1, operaciones)
    costoEliminacion ← costo_delete(S1[i]) + DISTANCIAEDICIONFUERZABRUTA(S1, S2, i + 1, j, operaciones)
    costoTransposicion ← INT_MAX
    if i + 1 < longitud de S1 y j + 1 < longitud de S2 y S1[i] = S2[j + 1] y S1[i + 1] = S2[j] then
        Escribir "transposición" + S1[i] + S1[i + 1] en operaciones
        costoTransposicion ← costo_transpose(S1[i], S1[i + 1]) + DISTANCIAEDICIONFUERZABRUTA(S1, S2, i + 2, j + 2,
          operaciones)
    if costoSustitucion es el mínimo de los costos then
        Escribir "sustitución" + S1[i] + "->" + S2[j] en operaciones
    else if costoInsercion es el mínimo de los costos then
        Escribir "inserción" + S2[j] en operaciones
    else if costoEliminacion es el mínimo de los costos then
        Escribir "eliminación" + S1[i] en operaciones
    return el mínimo entre {costoSustitucion, costoInsercion, costoEliminacion, costoTransposicion}
```

2.1.1. Ejemplo Paso a Paso

Para demostrar el funcionamiento de ambos algoritmos (fuerza bruta y programación dinámica), aplicaremos un análisis paso a paso con las cadenas $S1$ = "abcd" y $S2$ = "abc", así como con $S1$ = "abcd" y $S2$ = "abdc". Los costos específicos de cada operación de edición son:

- **Costo de Inserción**: 1
- **Costo de Eliminación**: 1

- **Costo de Sustitución**: 2
- **Costo de Transposición**: 1

Este algoritmo examina todas las posibles transformaciones de $S1$ a $S2$, evaluando cada combinación de inserción, eliminación, sustitución y transposición para encontrar el costo mínimo.

2.1.2. Ejemplo 1: $S1$ = "**abcd**" y $S2$ = "**abc**"

1. Comparamos los primeros caracteres:"a" de $S1$ y "a" de $S2$.
   - Son iguales, avanzamos al siguiente carácter en ambas cadenas sin costo adicional.
2. Comparamos los siguientes caracteres: "b" de $S1$ y "b" de $S2$.
   - Son iguales, avanzamos al siguiente carácter sin costo adicional.
3. Comparamos "c" de $S1$ y "c" de $S2$.
   - Son iguales, avanzamos al siguiente carácter sin costo adicional.
4. Resta un carácter adicional en $S1$: "d". Realizamos una eliminación con un costo de 1.

Costo mínimo para transformar $S1$ = "**abcd**" **en** $S2$ = "**abc**" **es 1.**

2.1.3. Ejemplo 2: $S1$ = "**abcd**" y $S2$ = "**abdc**"

1. Comparamos "a" de $S1$ y "a" de $S2$.
   - Son iguales, avanzamos al siguiente carácter sin costo.
2. Comparamos "b" de $S1$ y "b" de $S2$.
   - Son iguales, avanzamos al siguiente carácter sin costo.
3. Comparamos "c" de $S1$ y "d" de $S2$.
   - Son diferentes, con cuatro opciones:
     a) Eliminar "c" en $S1$, logrando $S1$ = "abd" (costo acumulado: 1).
     b) Insertar "d" en $S1$, logrando $S1$ = "abcd" (costo acumulado: 1).
     c) Sustituir "c" por "d" (costo acumulado: 2).
     d) Transponer "c" y "d" en $S1$, logrando $S1$ = "abdc" (costo acumulado: 1).

El costo mínimo para transformar $S1$ = "**abcd**" **en** $S2$ = "**abdc**" **es 1.**

2.1.4. Análisis de Complejidad

La complejidad temporal del algoritmo de fuerza bruta es exponencial, $O(4^{\text{máx}(m,n)})$, donde m y n son las longitudes de $S1$ y $S2$, respectivamente. La falta de almacenamiento de subproblemas resueltos obliga a recalcular cada transformación posible, resultando en una alta demanda de tiempo de ejecución para cadenas largas. La complejidad espacial es $O(\text{máx}(m,n))$ debido a la profundidad máxima de la pila de recursión.

La inclusión de transposiciones y costos variables aumenta la cantidad de posibles operaciones a evaluar en cada paso, incrementando la carga computacional en comparación con la versión estándar del algoritmo de distancia de edición.

2.2. Programación Dinámica

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

Erickson, 2019 [2]

1) Describa la solución recursiva.
2) Escriba la relación de recurrencia, incluyendo condiciones y casos base.
3) Identifique subproblemas.
4) Defina estructura de datos a utilizar y especifique el orden de calculo que realiza su programa que utiliza programación dinámica.

2.2.1. Descripción de la Solución Recursiva

En programación dinámica, el problema se descompone en subproblemas, almacenando cada resultado para evitar cálculos repetitivos. Esto permite calcular la distancia de edición entre $S1$ y $S2$ de manera más eficiente.

2.2.2. Relación de Recurrencia

La relación de recurrencia es la siguiente:

$$\text{DP}[i][j] = \text{mín} \begin{cases} \text{DP}[i-1][j] + \text{costo\_eliminación} \\ \text{DP}[i][j-1] + \text{costo\_inserción} \\ \text{DP}[i-1][j-1] + \text{costo\_sustitución} & \text{si } S1[i] \neq S2[j] \\ \text{DP}[i-2][j-2] + \text{costo\_transposición} & \text{si hay transposición} \end{cases}$$

Los casos base son:

$$\text{DP}[0][j] = j \times \text{costo\_inserción} \quad \text{y} \quad \text{DP}[i][0] = i \times \text{costo\_eliminación}$$

2.2.3. Identificación de Subproblemas

Cada subproblema DP[i][j] representa la distancia mínima de edición para transformar los primeros i caracteres de $S1$ en los primeros j caracteres de $S2$.

Algoritmo 3: Algoritmo de distancia de edición con programación dinámica.

```
1  Procedure DISTANCIAEDICIONPROGDINAMICA(S1, S2, operaciones)
2      m ← longitud de S1
3      n ← longitud de S2
4      Crear matriz dp de dimensiones (m + 1) × (n + 1) inicializada en 0
5      for i desde 1 hasta m do
6          dp[i][0] ← dp[i - 1][0] + costo_delete(S1[i - 1])
7          Escribir "eliminación" + S1[i - 1] en operaciones
8      for j desde 1 hasta n do
9          dp[0][j] ← dp[0][j - 1] + costo_insert(S2[j - 1])
10         Escribir "inserción" + S2[j - 1] en operaciones
11     for i desde 1 hasta m do
12         for j desde 1 hasta n do
13             costoSustitucion ← dp[i - 1][j - 1] + costo_sub(S1[i - 1], S2[j - 1])
14             costoInsercion ← dp[i][j - 1] + costo_insert(S2[j - 1])
15             costoEliminacion ← dp[i - 1][j] + costo_delete(S1[i - 1])
16             dp[i][j] ← mínimo entre {costoSustitucion, costoInsercion, costoEliminacion}
17             if dp[i][j] es costoSustitucion then
18                 Escribir "sustitución" + S1[i - 1] + "->" + S2[j - 1] en operaciones
19             else if dp[i][j] es costoInsercion then
20                 Escribir "inserción" + S2[j - 1] en operaciones
21             else if dp[i][j] es costoEliminacion then
22                 Escribir "eliminación" + S1[i - 1] en operaciones
23             if i > 1 y j > 1 y S1[i - 1] = S2[j - 2] y S1[i - 2] = S2[j - 1] then
24                 costoTransposicion ← dp[i - 2][j - 2] + costo_transpose(S1[i - 2], S1[i - 1])
25                 if dp[i][j] > costoTransposicion then
26                     dp[i][j] ← costoTransposicion
27                     Escribir "transposición" + S1[i - 2] + S1[i - 1] en operaciones
28     return dp[m][n]
```

2.2.4. Ejemplo Paso a Paso

Para este algoritmo de programación dinámica, utilizamos una matriz DP donde cada celda $DP[i][j]$ representa el costo mínimo para transformar los primeros i caracteres de $S1$ en los primeros j caracteres de $S2$.Los costos específicos de cada operación de edición son:

- **Costo de Inserción**: 1
- **Costo de Eliminación**: 1
- **Costo de Sustitución**: 2

- **Costo de Transposición**: 1

2.2.5. Ejemplo 1: $S1$ = "**abcd**" y $S2$ = "**abc**"

Inicializamos la matriz DP de tamaño 5×4 (porque $S1$ tiene 4 caracteres y $S2$ tiene 3), con las condiciones base:

- $DP[i][0] = i \times$ costo de eliminación
- $DP[0][j] = j \times$ costo de inserción

	""	"a"	"ab"	"abc"
""	0	1	2	3
"a"	1	0	1	2
"ab"	2	1	0	1
"abc"	3	2	1	0
"abcd"	4	3	2	1

El costo mínimo para transformar $S1$ = "abcd" en $S2$ = "abc" es el valor en $DP[4][3] = 1$.

2.2.6. Ejemplo 2: $S1$ = "**abcd**" y $S2$ = "**abdc**"

Para este caso, la matriz tendrá en cuenta la posibilidad de transposiciones.

	""	"a"	"ab"	"abc"	
""	0	1	2	3	4
"a"	1	0	1	2	3
"ab"	2	1	0	1	2
"abc"	3	2	1	2	1
"abcd"	4	3	2	1	1

La celda $DP[4][4] = 1$ indica que el costo mínimo es 1, logrado mediante la transposición entre "c" y "d" en $S1$.

2.2.7. Análisis de Complejidad

La complejidad temporal de la versión dinámica es $O(m \times n)$, ya que se calcula cada celda de la matriz DP una sola vez. La complejidad espacial también es $O(m \times n)$ debido a la matriz utilizada para almacenar los subproblemas. La inclusión de transposiciones y costos variables agrega un costo adicional en el cálculo de cada celda de DP, pero no cambia la complejidad asintótica.

3. Implementaciones

La extensión máxima para esta sección es de 1 página.

Aquí deben explicar la estructura de sus programas haciendo referencias a los archivos y funciones de su entrega. No adjunte código en esta sección.

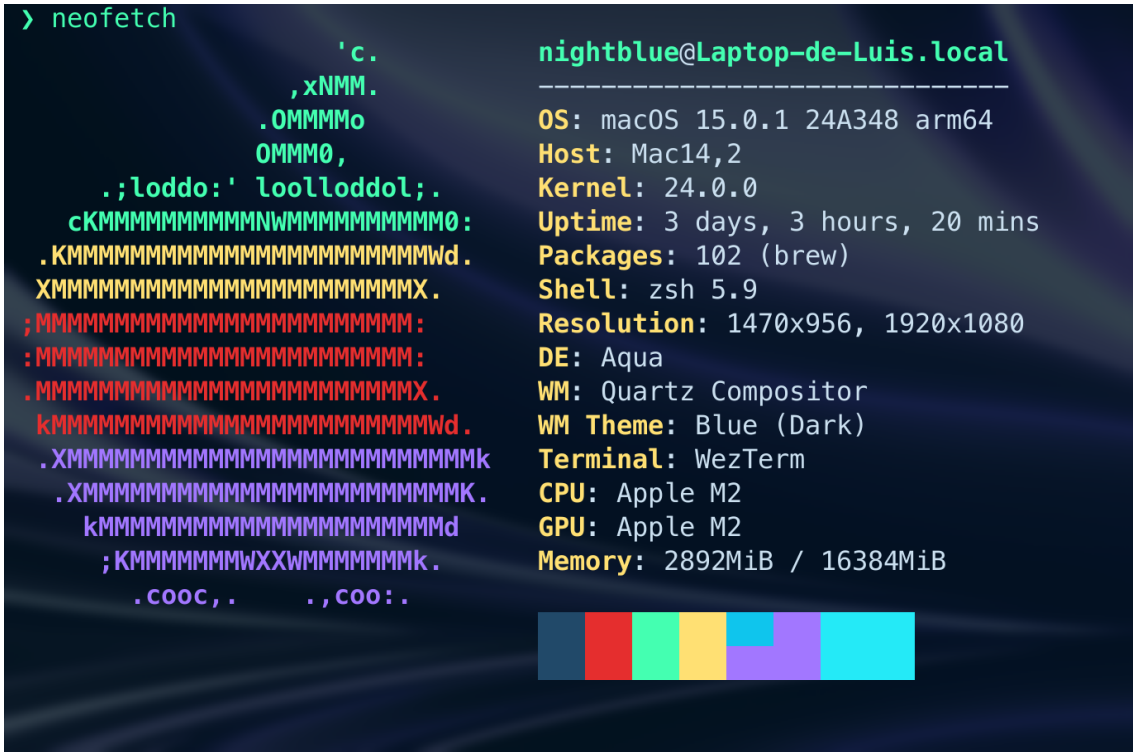
Si se utiliza algún código, idea, o contenido extraído de otra fuente, este **debe** ser citado en el lugar exacto donde se utilice, en lugar de mencionarlo al final del informe.

4. Experimentos

La extensión máxima para esta sección es de 6 página.

> *"Non-reproducible single occurrences are of no significance to science."*
>
> —Popper,2005 [9]

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* [5].



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 ests 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 especficiar como obtener esos archivos desde una ejecución en otro computador (otra infraestructura para hacer lso 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 evaluara 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.

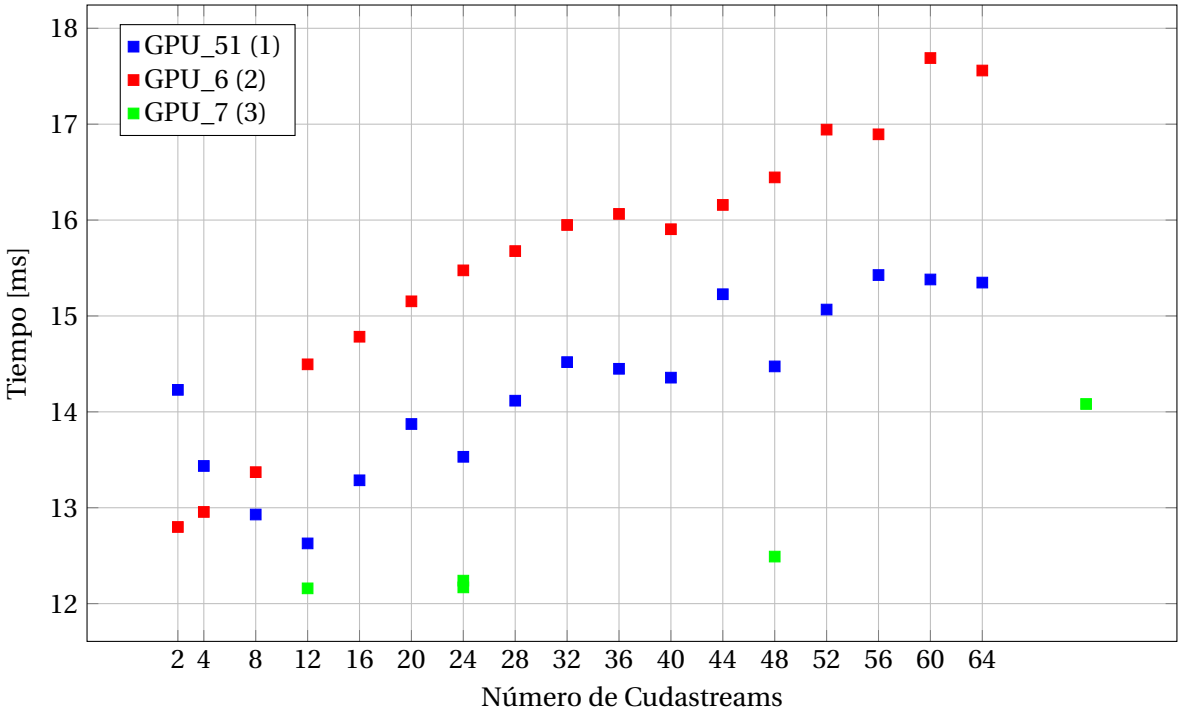


Figura 1: Ejemplo de scatterplot hecho con tikz. Tamaño ideal 1.

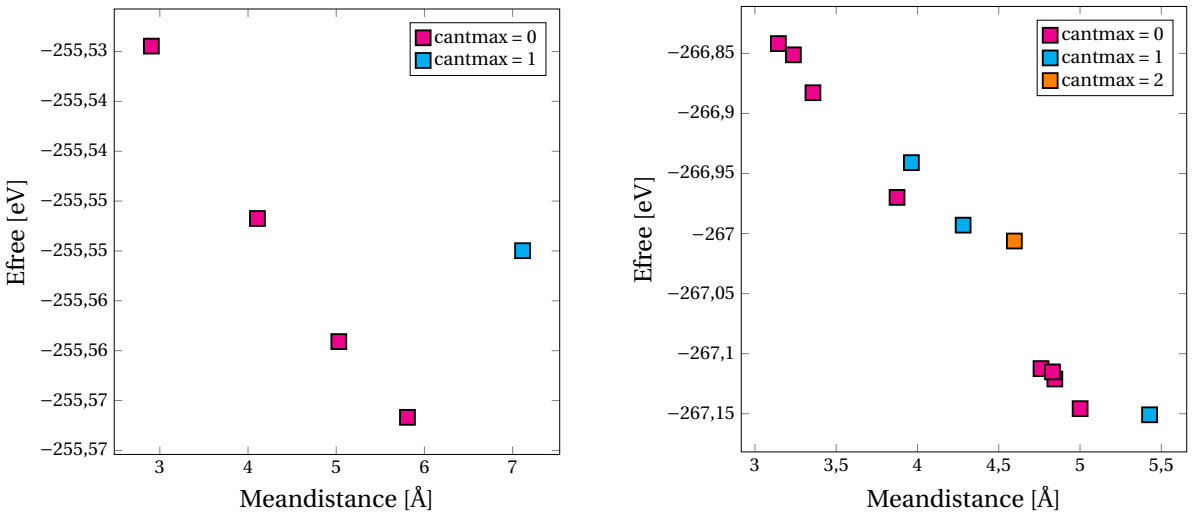


Figura 2: Ejemplo de scatterplot hecho con tikz. Tamaño ideal 2.

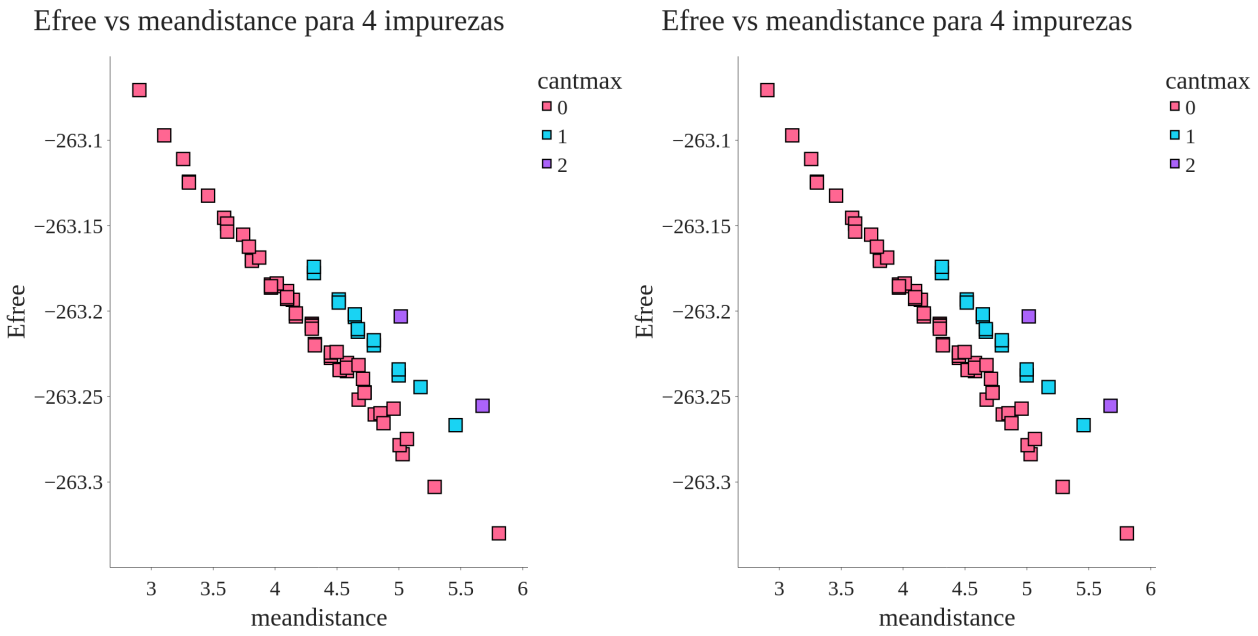


Figura 3: Ejemplo de scatterplot hecho con matplotlib.

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 y 3-rol.tar.gz` (`rol` con dígito verificador y sin guión).

  Dicho tarball debe contener las fuentes en LaTeX 2ε (al menos `tarea-2 y 3.tex`) de la parte escrita de su entrega, además de un archivo `tarea-2 y 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 LaTeX (en 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 códigos funte 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] Christophe Fiorio. *Algorithm2e documentation*. http://ctan.math.illinois.edu/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf. 2023.

[4] Christophe Fiorio. *Algorithm2e on CTAN*. https://ctan.org/pkg/algorithm2e. 2023.

[5] 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.

[6] 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.

[7] Chris Mack y Lola Muxamedova. *How to Write a Good Scientific Paper*. Nov. de 2019.

[8] Overleaf. *Writing Algorithms in LaTeX*. https://www.overleaf.com/learn/latex/Algorithms. 2023.

[9] K. Popper. *The Logic of Scientific Discovery*. Routledge Classics. Taylor & Francis, 2005. ISBN: 9781134470020. URL: https://books.google.cl/books?id=LWSBAgAAQBAJ.