REPORTE TAREA 2 y 3
ALGORITMOS Y COMPLEJIDAD

«Encontrando Diferencias entre dos Secuencias»

Camila Rosales

18 de junio de 2025

23:11

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 [***elsevier_abstract_2024***].*

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

Para aprovechar mejor los recursos, es importante desarrollar soluciones óptimas analizando y diseñando bien los algoritmos. Este informe compara dos métodos clásicos: fuerza bruta y programación dinámica, aplicados al problema de encontrar diferencias mínimas entre dos secuencias de caracteres. La idea es mostrar la menor cantidad de pares de substrings que representen todas las diferencias. Con esto, buscamos entender cuál es más eficiente según el tamaño de la entrada y por qué es importante elegir el enfoque correcto dependiendo del problema.

2. Fuerza Bruta

La fuerza bruta resuelve el problema de la manera más directa posible. Básicamente, revisa todas las formas en las que se pueden alinear los caracteres comunes en las secuencias y de ahí saca las diferencias.

Este método siempre encuentra una solución válida, pero cuando las secuencias son más grandes, se vuelve muy lento. Esto pasa porque repite subproblemas innecesariamente y no guarda resultados intermedios, lo que hace que use demasiado tiempo y memoria.

3. Programación Dinámica

La programación dinámica es más eficiente porque almacena resultados de subproblemas ya resueltos. Esto evita cálculos repetidos y aprovecha patrones del problema para ahorrar tiempo.

En general, funciona así:

1. Se analiza la estructura del problema para ver qué subproblemas se repiten.
2. Se usan técnicas recursivas, pero guardando los resultados para no recalcularlos.
3. A partir de las soluciones parciales guardadas, se arma la respuesta óptima, minimizando las diferencias entre las cadenas.

Este método es más rápido, escalable y útil cuando la entrada tiene mayores dimensiones, por lo que en muchas situaciones es mejor que la fuerza bruta.

4. Diseño y Análisis de Algoritmos

4.1. Fuerza Bruta

El método de fuerza bruta busca comparar de forma exhaustiva todas las maneras posibles de alinear los caracteres de dos secuencias s y t. Se intenta encontrar el mayor prefijo común y luego dividir las cadenas en partes que representen las diferencias.

Este enfoque explora todas las opciones posibles, pero no evita repetir cálculos innecesarios, lo que hace que la cantidad de operaciones crezca de manera descontrolada a medida que aumenta el tamaño de las secuencias.

4.1.1. Complejidad

- Tiempo: $O(2^{\min(n,m)})$, debido a la gran cantidad de posibles particiones.
- Espacio: $O(d)$, donde d es la profundidad de la recursión.

4.1.2. Algoritmo

Algoritmo 1: Fuerza Bruta para diferencias

1 BruteForceDifferencess, t **if** $s = t$ **then**

2 **return** [] No hay diferencias **if** *s está vacío* ***o*** *t está vacío* **then**

3 **return** $[(s,t)]$ $n \leftarrow |s|, m \leftarrow |t|$ **for** $i \leftarrow 0$ ***hasta*** $\min(n,m)$ **do**

4 $s[0..i] = t[0..i]$ $resto \leftarrow$ BruteForceDifferences$(s[i+1..], t[i+1..])$ **return** $resto$ Coincidencia al inicio **return** $[(s,t)]$

4.2. Programación Dinámica

El enfoque de programación dinámica mejora el rendimiento al almacenar resultados de subproblemas ya resueltos. Se basa en la subsecuencia común más larga (LCS), analizando caracteres en ambas cadenas para determinar sus diferencias.

4.2.1. Complejidad

- Tiempo: $O(n \cdot m)$
- Espacio: $O(n \cdot m)$

4.2.2. Algoritmo

Algoritmo 2: Programación Dinámica para diferencias

```
1 Function DynamicDifferences(s, t):
2     Calcular matriz dp de LCS para s y t
3     Inicializar listas pares ← [], i ← 0, j ← 0
4     Recorrer s y t usando la LCS y guardar pares de substrings diferentes entre coincidencias
5     return pares
```

5. Implementaciones

6. Implementación

Ambos algoritmos fueron desarrollados en C++ siguiendo el formato de entrada y salida especificado en el problema. Se organizaron en un repositorio de GitHub para facilitar su ejecución y comparación.

6.1. Estructura de archivos

Fuerza Bruta

- `code/brute_force/algorithm/sequence_difference.cpp`: Calcula diferencias usando fuerza bruta.
- `code/brute_force/brute_force.cpp`: Ejecuta el algoritmo con entrada y salida controlada.

Programación Dinámica

- `code/dynamic_programming/algorithm/sequence_difference.cpp`: Solución optimizada con DP.
- `code/dynamic_programming/dynamic_programming.cpp`: Programa principal para entradas grandes.

Ambos enfoques fueron probados con los mismos datos de entrada para garantizar una comparación justa.

7. Resultados

Se realizaron experimentos, generando gráficos y tablas con mediciones de tiempo y rendimiento.
Repositorio: https://github.com/pabloealvarez/INF221-2025-1-TAREA-1

8. Experimentos

La extensión máxima para esta sección es de 6 página.

> "*Non-reproducible single occurrences are of no significance to science.*"
>
> —**popper2005logic**,**popper2005logic** [**popper2005logic**]

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* [**inbookFonseca**].

8.1. Dataset (casos de prueba)

La extensión máxima para esta sección es de 2 páginas.

Diseñe casos de prueba que cubran varios escenarios (e.g., secuencias vacías, secuencias con caracteres repetidos, secuencias que no tienen diferencias, tienen pocas o varias diferencias, etc). Es importante generar varias muestras con características similares para una misma entrada, por ejemplo, variando el 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.

En esta sección es importante que describa cómo generó los casos de prueba y el porqué.

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 algoritmos 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.

8.2. 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 **??** 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.

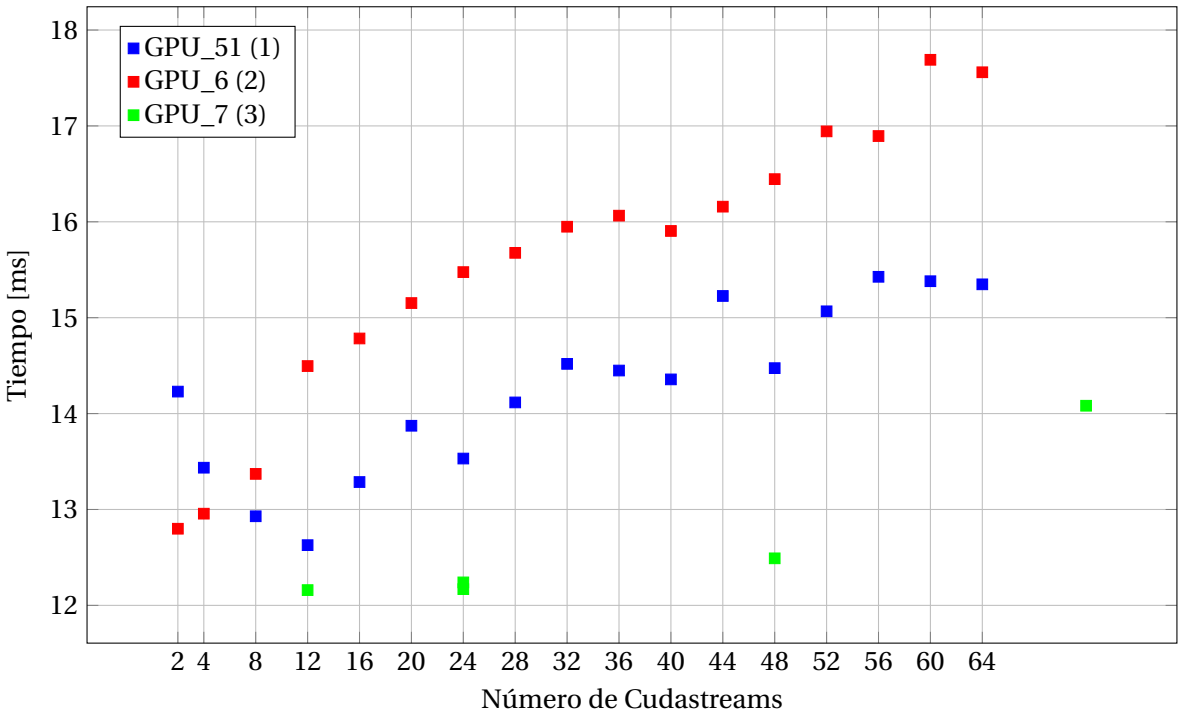

Figura 1: Ejemplo de scatterplot hecho con tikz. Tamaño ideal 1.

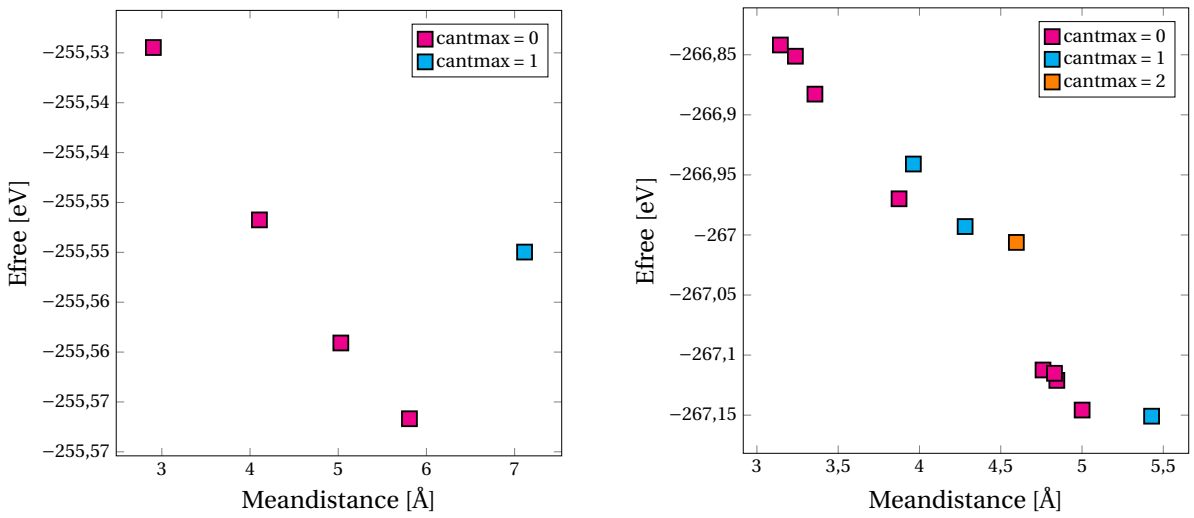

Figura 2: Ejemplo de scatterplot hecho con tikz. Tamaño ideal 2.

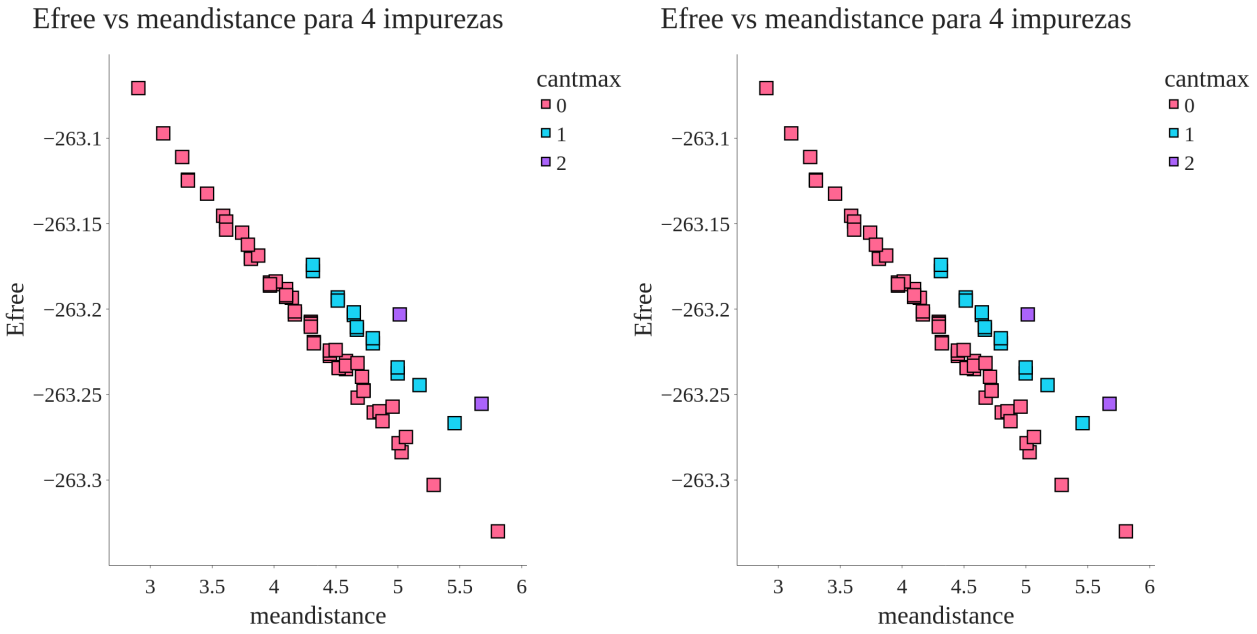

Figura 3: Ejemplo de scatterplot hecho con matplotlib.

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.

9. 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.

10. 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 6 de junio de 2025.

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.