REPORTE TAREA 2 y 3
ALGORITMOS Y COMPLEJIDAD

«Encontrando Diferencias entre dos Secuencias»

Camila Rosales

19 de junio de 2025

00:00

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.

4.2. 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: $O(n \cdot m)$

5. Implementaciones

Ambos algoritmos fueron desarrollados en C++ siguiendo el formato de entrada y salida especificado en el enunciado.

Ambos enfoques fueron probados con los mismos inputs para garantizar una comparación justa, para poder crear estos es necesario usar el archivo $input_generator.pyqueseencuentraenlacarpetacorrespondientea$

Repositorio: https://github.com/kamiskamis/INF221-2025-1-TAREA-2-3/tree/main

6. Experimentos

Para medir el rendimiento, se registró el tiempo de ejecución en función del tamaño de entrada. Los resultados fueron procesados con Python usando `matplotlib` y `pandas`, generando gráficos que muestran la relación entre tamaño y tiempo de ejecucion.

6.1. Dataset (casos de prueba)

Se generaron distintos tipos de secuencias para evaluar el comportamiento de los algoritmos, incluyendo:

- Secuencias vacías e idénticas.
- Secuencias con caracteres repetidos y diferencias significativas.
- Tamaños de entrada variables (0 a 20 caracteres).

Las pruebas fueron creadas con un script en Python que permite configurar la longitud de las cadenas, el número de diferencias y el tipo de modificación (inserción, eliminación, sustitución). Se realizaron varias ejecuciones para cada caso, promediando los tiempos para reducir la variabilidad.

6.2. Resultados

Los datos analizados provienen de los archivos que se generan en `data/measurements/*.txt` al ejecutar cada enfoque con el input.

4.3. Análisis Comparativo

Los resultados obtenidos reflejan las grandes diferencias en complejidad entre ambos enfoques:

Fuerza Bruta muestra un crecimiento exponencial en tiempo de ejecución, alcanzando el límite de 5 minutos a partir de dimensiones 26–32. Por ejemplo, para entradas de longitud 26, el tiempo supera los 9 segundos, y para 32 se marca como “ERROR” por exceder el tiempo permitido. Esto concuerda con su complejidad exponencial teórica $O(2^{n+m})$.

Programación Dinámica, en comparacion mantiene un tiempo de ejecución muy bajo incluso para cadenas de un mayor tamaño como 32,lo que reafirma su eficiencia $O(n \cdot m)$. La estabilidad de los tiempos indica que este enfoque es más escalable y adecuado para aplicaciones con datos más extensos.

Esta diferencia valida la eleccion de programación dinámica como la solución mas favorable para el contexto en que se analizo, mientras que la fuerza bruta puede ser útil solo para instancias muy pequeñas o como referencia conceptual.

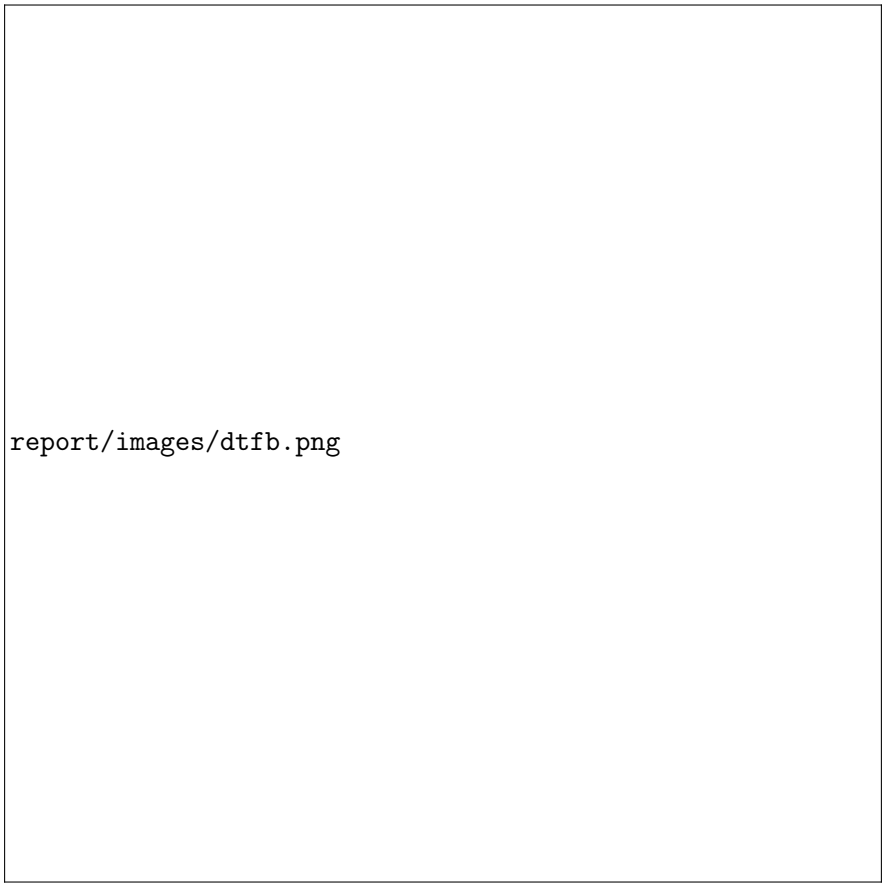

Figura 1: Tiempo de ejecución del algoritmo de Fuerza Bruta.

7. 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.

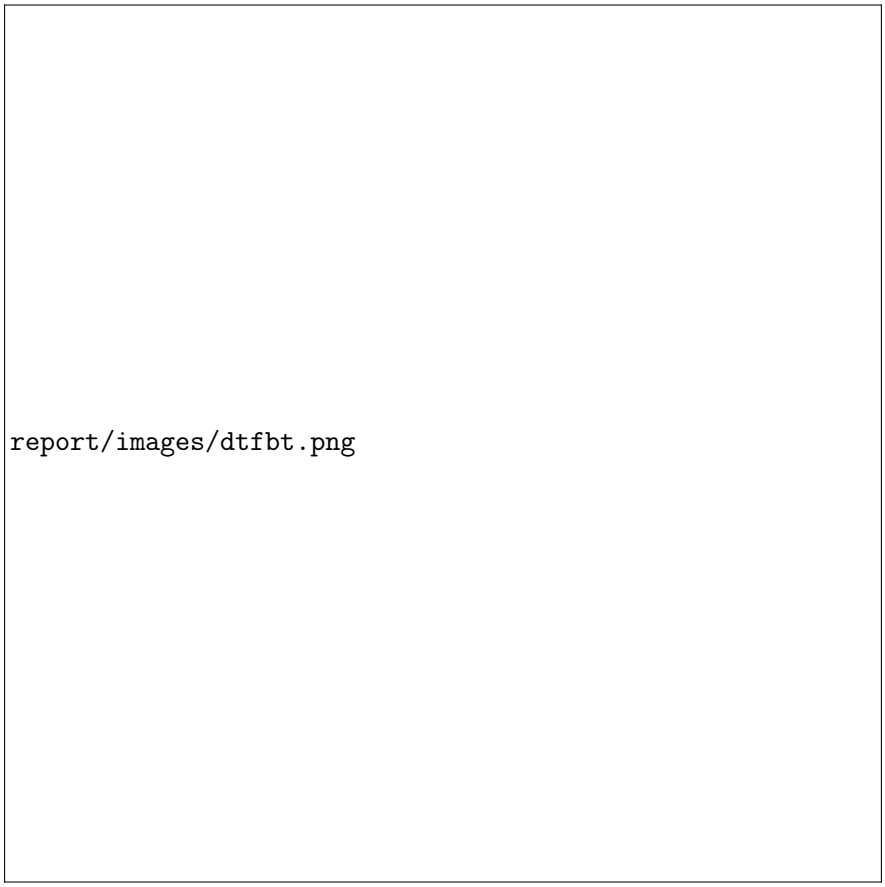

Figura 2: Tiempo de ejecución del algoritmo de Fuerza Bruta.

- 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.

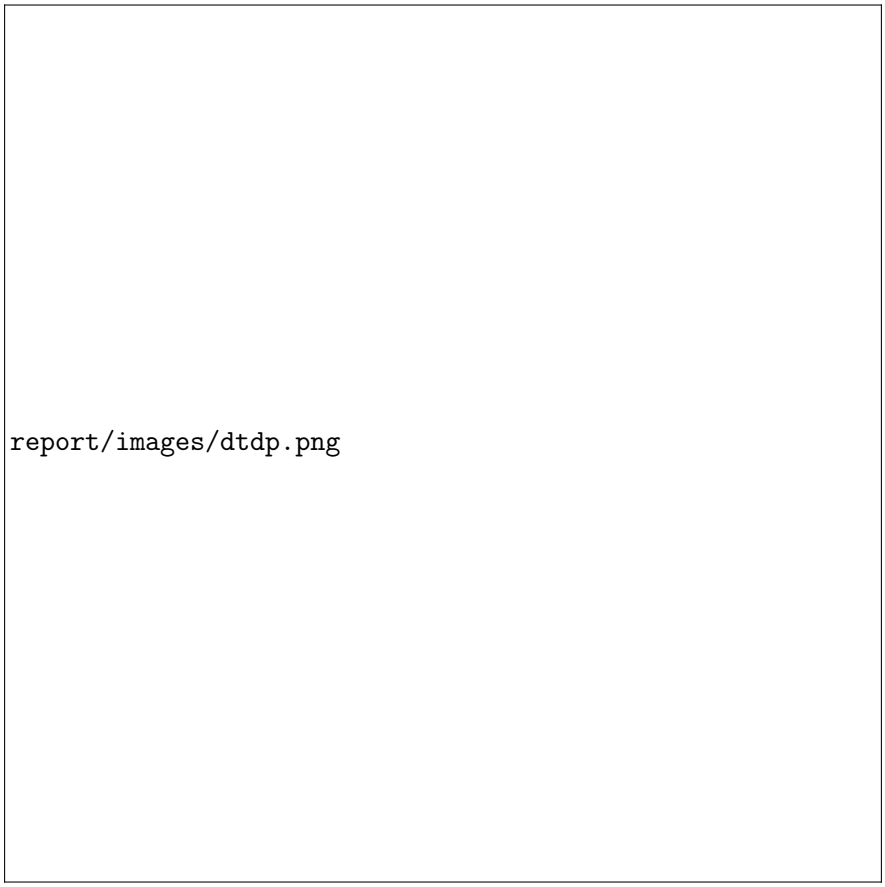

Figura 3: Tiempo de ejecución del algoritmo de Programación Dinámica.

8. 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).

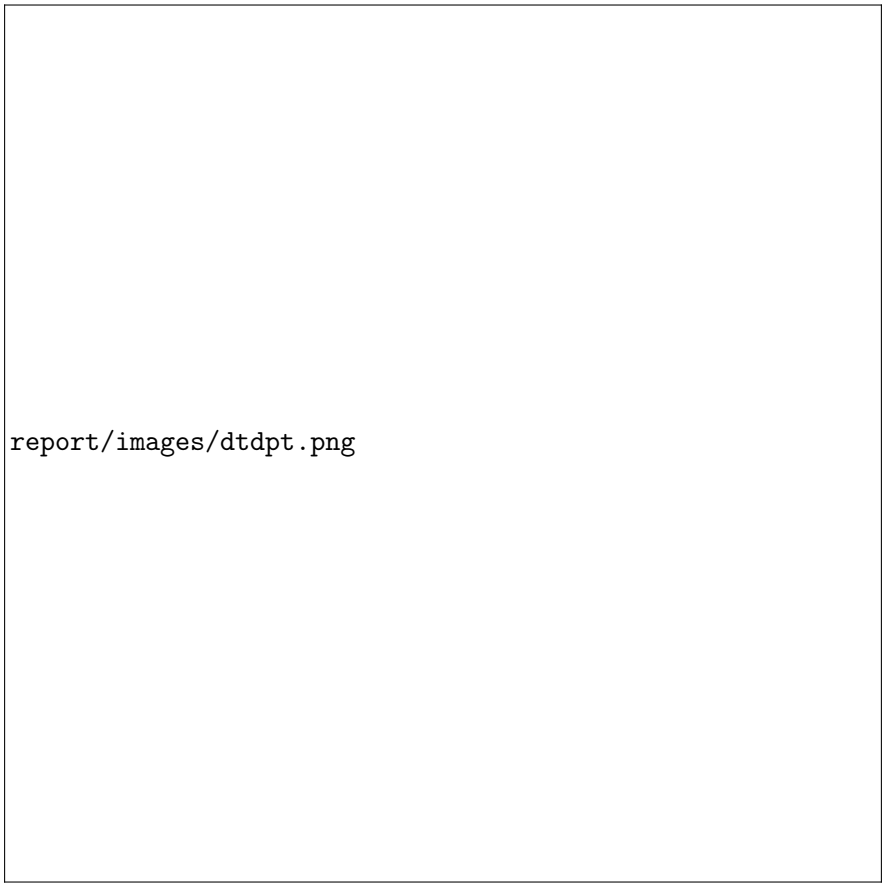

Figura 4: Tiempo de ejecución del algoritmo de Programación Dinámica.

- 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