Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Mty

Materia

Análisis y diseño de algoritmos avanzados

Nombre del archivo

E1. Actividad Integradora

Cruz Daniel Pérez Jiménez - A01736214

Erwin Porras Guerra - A01734881

Marlon Yahir Martínez Chacón - A01424875

Grupo 601

01 de Octubre del 2023

Los arreglos de sufijos, conocidos en el ámbito de la "stringology" o estudio de las cadenas de texto, representan una herramienta esencial en la manipulación y búsqueda eficiente de cadenas. Estos tienen muchas ventajas sobre otras estructuras de datos similares como los árboles de sufijos, principalmente su complejidad temporal y espacial.

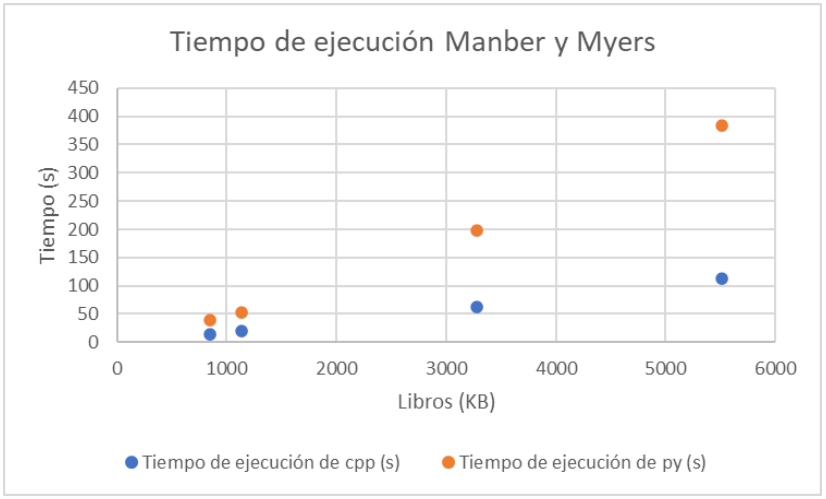
El tiempo de construcción y acceso a los datos asociados a las cadenas de texto es óptimo, alcanzando una complejidad de O(n), esto significa que, a pesar de la creciente complejidad de las cadenas, el rendimiento de los arreglos de sufijos sigue siendo eficiente y lineal, lo que los hace ideales para aplicaciones que involucran grandes conjuntos de datos textuales. También aunque almacenar todos los sufijos de una cadena puede parecer una tarea costosa en términos de memoria, los arreglos de sufijos logran esto de manera eficiente. La cantidad de memoria necesaria para representar un arreglo de sufijos está directamente relacionada con la longitud de la cadena original, lo que implica una complejidad espacial también de O(n). Esta característica hace que los arreglos de sufijos sean una opción viable incluso en entornos con restricciones de memoria.

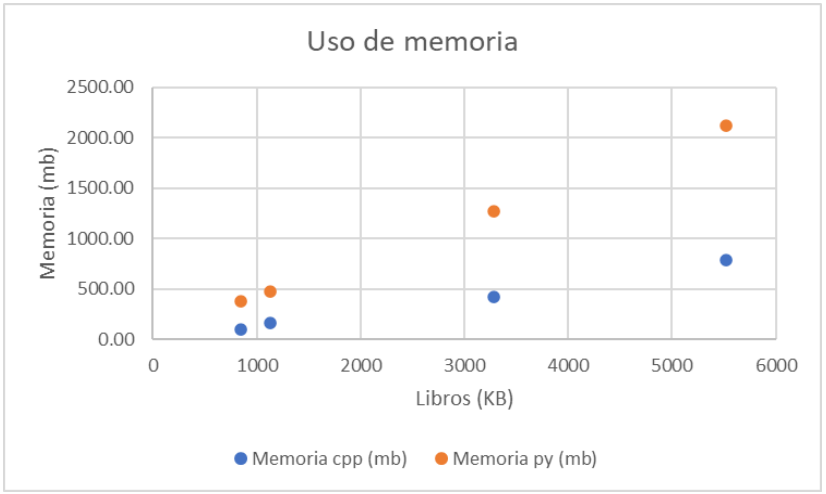
Las ventajas previamente mencionadas convierten a los arreglos de sufijos en una herramienta invaluable para resolver una amplia gama de problemas relacionados con el procesamiento de texto, desde la búsqueda de patrones hasta la recuperación de información en grandes conjuntos de datos textuales. Su eficiencia y versatilidad los posicionan como una elección inteligente en el arsenal de cualquier científico de la computación o ingeniero de software que trabaje con cadenas de texto.

Tiempos de ejecución

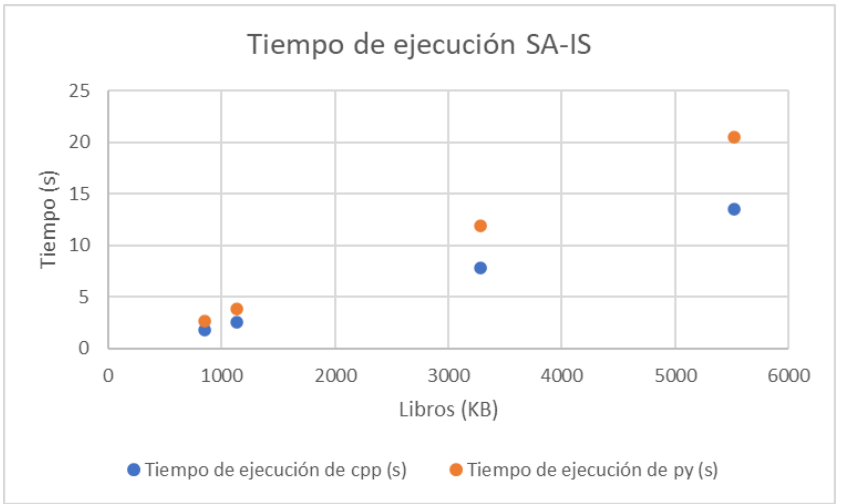
Dracula (849 KB)
Iliada (1,135 KB)
War and Peace (3,281 KB)
The Complete Works of William Shakespeare (5,516 KB)

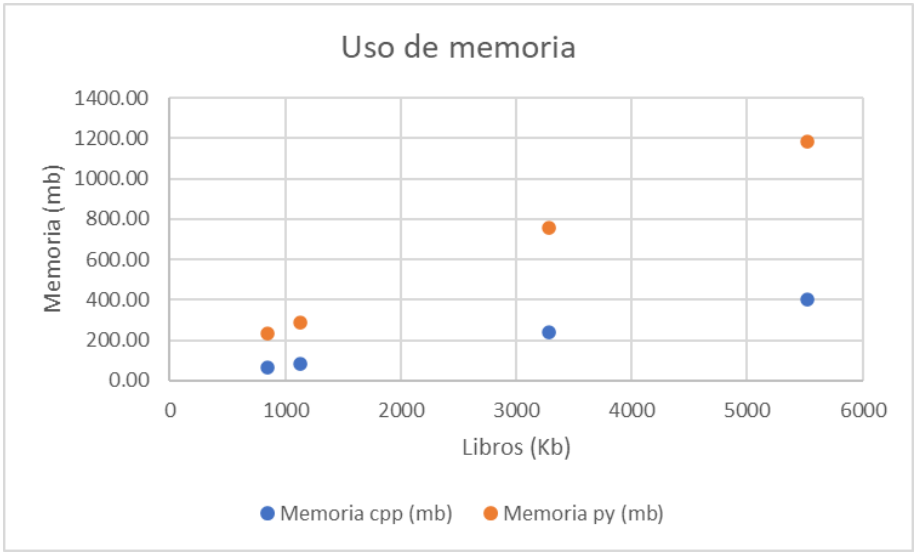
Manber y Myers


Tiempo de ejecución Manber y Myers
Tiempo (s)
450
400
350
300
250
200
150
100
50
0
0
1000
2000
3000
4000
5000
6000
Libros (KB)
Tiempo de ejecución de cpp (s)
Tiempo de ejecución de py (s)


Uso de memoria
Memoria (mb)
2500.00
2000.00
1500.00
1000.00
500.00
0.00
0
1000
2000
3000
4000
5000
6000
Libros (KB)
Memoria cpp (mb)
Memoria py (mb)

SA-IS


Tiempo de ejecución SA-IS
Tiempo (s)
25
20
15
10
5
0
0
1000
2000
3000
4000
5000
6000
Libros (KB)
Tiempo de ejecución de cpp (s)
Tiempo de ejecución de py (s)



Complejidad temporal y espacial de los algoritmos

Manber-Myers:

El algoritmo de Manber-Myers alcanza una complejidad temporal de O(n log^2 n), el cual después de ser optimizado puede llegar a una complejidad de O(n log n), donde n es la longitud de la cadena de texto de entrada. Este algoritmo emplea una técnica de divide y vencerás para construir el arreglo de sufijos, en la fase de división de subcadenas se hace una complejidad temporal de O(n), luego para ordenarlas se hace una complejidad de O(nlogn), resultando en una complejidad final de O(nlogn).

En cuanto a la complejidad espacial, el algoritmo de Manber-Myers llega a una complejidad lineal O(n), esto significa que la cantidad de memoria para almacenar el arreglo de sufijos es proporcional a la longitud de la cadena original.

SA-IS:

El algoritmo de SA-IS es un algoritmo muy avanzado y optimizado que tiene una complejidad temporal de O(n). La clave de esta eficiencia radica en varias técnicas como la inducción y clasificación, a través de “buckets”, una categorización de cada sufijo ya sea de L o S, y un ordenamiento radix, el cual tiene una complejidad de tiempo lineal O(n). La combinación de todos estas técnicas permiten que este algoritmo alcance una complejidad lineal, convirtiéndose en uno de los algoritmos más eficientes para la construcción de arreglos de sufijos.

Para el almacenamiento del arreglo, este es igual que el algoritmo de Manber-Myers, teniendo una complejidad lineal de O(n), proporcional a la longitud de la cadena original.

Conclusiones

Cruz Daniel Pérez Jiménez:

Durante esta actividad me resultó muy interesante el proceso de traducción del código de un lenguaje “sencillo” a uno un poco más “complejo”, ya que el entendimiento del algoritmo fue fundamental para que a pesar de la diferencia de sintaxis se hiciese conceptualmente lo mismo, además, con esta actividad pude notar el gran impacto que se puede llegar a lograr con una implementación que priorice el uso de memoria (al igual que el compilador del lenguaje), ya que tanto el tiempo de ejecución como el uso de memoria tienen un gran impacto de acuerdo al lenguaje utilizado.

Erwin Porras Guerra:

Las problemáticas que presenta el área de “stringology” no solo son muy interesantes, sino también presentan una necesidad muy grande para cumplir con funciones que frecuentemente damos por hecho, sin embargo como computólogos es importante saber cómo resolver estos problemas con algoritmos eficientes. Los algoritmos que analizamos en esta entrega son algunos de los más utilizados y fue muy interesante entender cómo funcionan, para facilitar el procesamiento de texto avanzado con niveles muy altos de eficiencia.

Marlon Yahir Martínez Chacón

Los arreglo de sufijos presenta una solución interesante para los problemas que nos ofrece el área de la stringología, los suffix array pueden llegar a volverse más complejos conforme más estructuras de datos se le van incorporando al algoritmo para crearlo, pero gracias a esto podemos obtener un algoritmo más eficiente que se demuestra en el tiempo de ejecución y en el uso de la memoria donde entre los tiempos suelen ser mucho menores a comparación del uso del algoritmo como fue ideado por manber y myers en sus inicios al igual que la memoria donde se puede observar que en el SA-IS es mucho menor que en el manber. Por lo que los algoritmos van evolucionando cada vez más haciendo que sea más eficientes y esto también se ve en su complejidad tanto temporal como espacial donde en el caso de SA-IS es menor que el que se nos da en manber.

Link al repositorio: https://github.com/ErwinPo/AlgoritmosE1