Estructuras de datos y algoritmos avanzados

Laboratorio 3

Francisco Flores

1. Introducción

En el presente informe se estudia y analiza *string matching* mediante los algoritmos de *pattern matching* basado en fuerza bruta y arreglo de sufijos (*suffix array*). Se mide el los tiempos de ejecución de cada uno y se comparan

2. Estructuras de datos y algoritmos

Un arreglo de sufijos es un arreglo que almacena los índices de todos los sufijos de un string. Para crearlo primero se deben determinar todos los sufijos del string, luego ordenarlos alfabéticamente y cada uno de ellos tiene asociado su índice correspondiente al string original. Esto requiere tiempo adicional en crear el arreglo de sufijos, sin embargo permite realizar búsquedas de un patrón en el texto en tiempo O(nlogn).

Por otro lado la búsqueda por fuerza bruta debe comprar cada carácter del patrón con cada uno de los caracteres del string, lo que toma tiempo O(nm) con n: largo del patron y m: largo del string. Este algoritmo no requiere de estructuras adicionales para su ejecución.

3. Implementación

Para implementar el arreglo de sufijos se ha creado una clase en Python llamada `SuffixArray` la cual recibe como parámetro al constructor el texto (string) que se utilizará para crear el arreglo de sufijos. Lo primero que hace es agregar el caracter $. El método `suffix` crea el arreglo de sufijos iterando sobre el texto aumentando en 1 el índice del inicio, para luego ordenarlo con la función `sorted` de Python de forma alfabética. Esto ocupa un espacio de O(n).

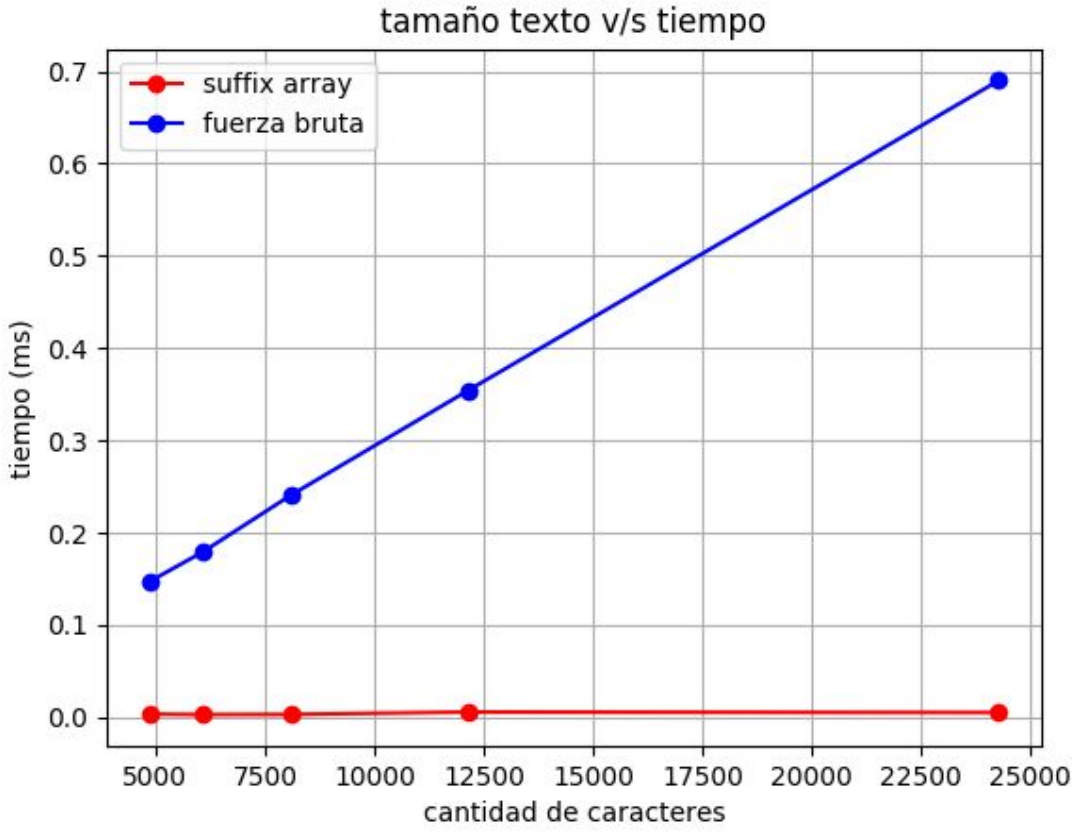
El método `search` recibe el string que se quiere buscar en el texto y hace búsqueda binaria dos veces para encontrar el mínimo y máximo índice que hace match con el patrón. Compara >, < o != el patrón con cada uno de los sufijos (cuyos índices están el el arreglo). Retorna una lista con todos los índices entre los intervalos calculados. Toma tiempo O(mlogn) para un patrón de largo m.

Fuerza bruta itera sobre todo el texto y busca el patrón por cada caracter en el texto. Tiene complejidad O(nm) con largo del patron m y largo del texto n.

4. Experimentos y resultados

Se probaron ambos métodos con distintos tamaños (cantidad de caracteres) para el texto de entrada y se buscó el mismo patrón “ACGTT” en cada uno guardando los tiempos de ejecución del método buscar (`serach & findBrute`). El dataset seleccionado corresponde a un subconjunto de un dataset de adn (dna.50MB from here) del que se extrajeron los primeros 24277 caracteres.

Los resultados arrojados se muestran en el siguiente gráfico



Se puede apreciar claramente que en este experimento por lejos el que obtuvo mejores resultados fue SuffixArray, ya que solo realiza búsqueda binaria sobre el arreglo de sufijos, mientras que el algoritmo de fuerza bruta realiza una búsqueda exhaustiva comparando el patrón por cada carácter del texto. Sin embargo el tiempo que toma SuffixArray para crear el arreglo es bastante considerable para dataset de gran tamaño, llegando a tomar más tiempo que fuerza bruta (para textos pequeños).

cantidad de caracteres	**SuffixArray.search (ms)**	**findBrute (ms)**
4855	0.00360	0.14703
6069	0.00303	0.17896
8092	0.00324	0.24073
12138	0.00582	0.35405
24276	0.00525	0.69005

Tabla con los valores graficados.

Para realizar el mismo experimento, leer el archivo README.md adjunto.