Estructuras de datos y algoritmos avanzados

Laboratorio 2

Rodrigo Alexander Richards Valenzuela

Matrícula: 2016404813

El problema dado para este laboratorio es implementar un Suffix Array y búsquedas de patrones sobre el, el tamaño del Suffix Array estará dado por el largo del string que estará dado por n y la búsqueda dependerá del tamaño del patrón que estará dado por m. Para realizar los experimentos, se retornan las ocurrencias del patrón en vez de las posiciones en las que se encuentra, debido a que la búsqueda por fuerza bruta no permite retornar las posiciones en un tiempo constante.

Complejidades

1. Construcción de Suffix Array

   La construcción consiste en comparar sufijos de un string e ir ordenándolos, esta comparación toma tiempo $O(n)$ ya que el sufijo más largo sería el mismo string y se realizarán comparaciones de otros sufijos con este. Para ordenar se utiliza un algoritmo de sort en tiempo $O(nlog(n))$, el cual se sobrecarga con las comparaciones de los sufijos para ordenarlos y así obtener el arreglo, de esta forma la complejidad de la construcción del Suffix Array es de $O(n^2log(n))$.

2. Búsqueda binaria en Suffix Array

   Para buscar las ocurrencias de un patrón, se realiza una primera búsqueda binaria hasta encontrar un sufijo que sea mayor o igual lexicográficamente al patrón buscado y así obtener un límite inferior para la o las veces que se puede encontrar el patrón dentro de los sufijos, esta búsqueda al ser binaria, toma tiempo $O(nlog(n))$ ya que se deben realizar a lo más n comparaciones entre el patrón y los sufijos. Para encontrar el límite superior se realiza una nueva búsqueda binaria, hasta encontrar un sufijo que sea mayor al patrón buscado, la complejidad de esto también está dada por $O(nlog(n))$, de esta forma, la complejidad de este algoritmo de búsqueda es $O(nlog(n))$.

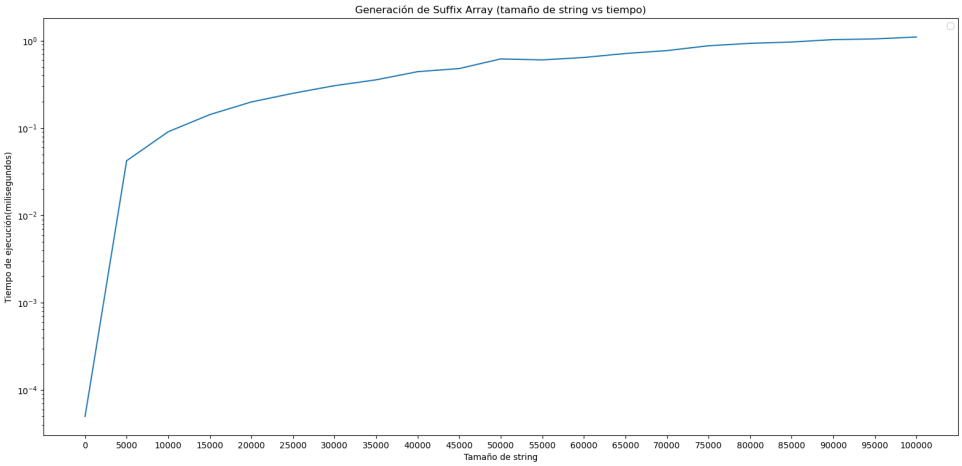
3. Búsqueda por fuerza bruta

   Para encontrar las ocurrencias por fuerza bruta se realizan comparaciones de cada sufijo del string dado con el patrón, de esta forma se realizarán n comparaciones por el tamaño del string y m comparaciones entre caracteres debido al tamaño del patrón, para ver si es que el patrón se encuentra dentro del sufijo en cada iteración. De esta manera, la complejidad de este algoritmo es $O(nm)$.

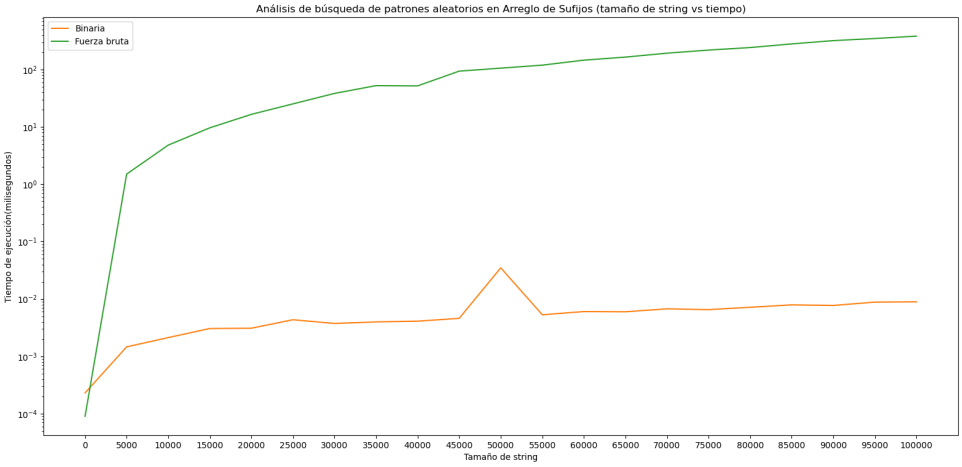
Gráficos

Para la realización de los siguientes gráficos se tomaron algunas entradas de las secuencias de ADN de las colecciones de texto de Pizza&Chili Corpus, estas entradas fueron reducidas para poder trabajar con menos cantidad de datos y tamaños de strings más pequeños.

Generación de Suffix Array

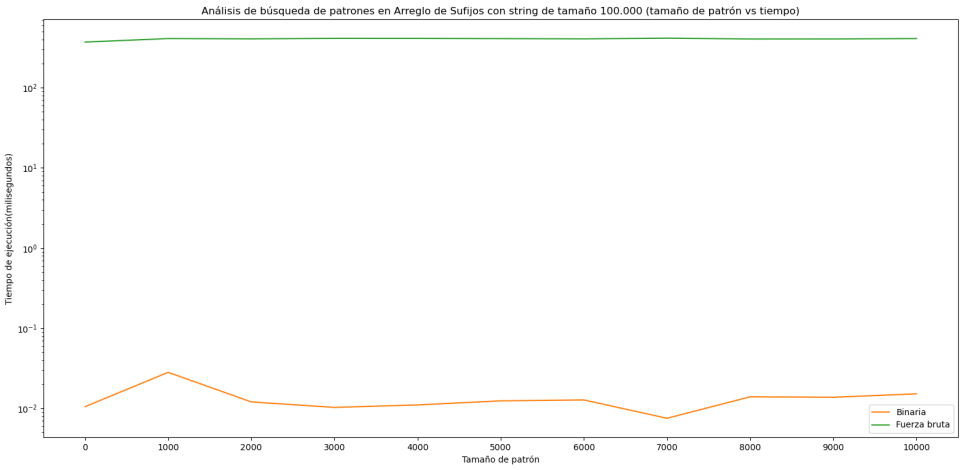


Tamaño de string creciente y patrones aleatorios



Para el gráfico anterior, se tomaron patrones aleatorios dentro del string completo y se buscaron ocurrencias de ese patrón dentro del string. Realizando búsqueda por fuerza bruta se observa que el desempeño es entre un orden de 10^0 y 10^2 milisegundos, aumentando aún más con tamaños de string mayores, en cambio con búsqueda binaria, el desempeño no varía tan drásticamente al aumentar el tamaño del string, estando en ordenes cercanos a 10^{-3} milisegundos, esto ya que la búsqueda por fuerza bruta recorre todo el string y realiza m comparaciones, mientras que la búsqueda binaria va dividiendo el tamaño de la búsqueda por cada iteración.

Tamaño de string fijo (100000) y patrones de distinto tamaño



Del gráfico, se observa que para un tamaño fijo de string y un tamaño de patrón de distinto tamaño, la búsqueda por fuerza bruta se mantiene casi constante, mientras que la búsqueda binaria posee algunos cambios, esto puede deberse a que mientras más grande el tamaño de patrón, se deben realizar más comparaciones en la búsqueda binaria y también a que esta búsqueda puede encontrar (o no) en pocas iteraciones el patrón buscado (tamaño de patrón 7000 en el gráfico), en cambio la búsqueda por fuerza bruta siempre buscará sobre el string completo y realizando comparaciones del patrón con cada sufijo dentro del string.