Oscar Miranda Escalante
A01630791
17 de septiembre de 2021

Reflexión – Actividad integradora 1

Parte 1:
El algoritmo que utilizamos para encontrar el patrón de los *mcode* en los archivos de transmisión fue el KMP, el resuelve el problema en un tiempo lineal. Este algoritmo primero debe conocer la tabla de prefijos-sufijos que se genera con nuestra función prefix.
Para generar la tabla, la función hace un recorrido por el patrón (en este caso cualquier archivo *mcode*) con 2 índices (i y j). Cuando el índice mayor encuentre una posición cuya letra coincida con la que apunta el índice menor, la tabla comenzará a almacenar enteros que representan el desplazamiento que debería hacer el algoritmo KMP en el patrón buscado cuando no coincide. Esto ayuda a evitar que se realicen comparaciones innecesarias.
Posteriormente, en nuestra implementación de KMP, se puede observar que un ciclo recorre cada elemento en el archivo de transmisión. Antes de ello, declara un índice para el archivo de transmisión y otro para el archivo con código malicioso.

- Si encuentra una coincidencia entre ambos, comienza a incrementar ambos índices
- Si llega al final del patrón que se busca (*mcode*), imprime las posiciones inicial y final.
- Si encuentra elementos que no coinciden, el índice del *mcode* tomará el valor que indique la tabla prefijos-sufijos y solo incrementará el otro índice.

La complejidad es del orden de O(n+m), donde n es la longitud del archivo de transmisión y m la longitud del archivo *mcode*.

Parte2:
Implementamos una solución más intuitiva (fuerza bruta) en la cual se analizan todas las posibilidades de substrings en los archivos de transmisión para encontrar el palíndromo más largo. A grandes rasgos, lo que hace es, por cada caracter en el archivo de transmisión, lleva un apuntador hasta la última posición del archivo, el cual irá disminuyendo de uno en uno. Una vez teniendo ambos apuntadores, el algoritmo incrementa el primero y disminuye el segundo, comprobando si los caracteres coinciden y hasta que se encuentren ambos apuntadores. Si todos los caracteres coinciden, significa que es un palíndromo. De por cada palíndromo encontrado, el algoritmo actualiza una variable con el de mayor longitud.

La complejidad de esta solución es del orden de O(n^2*m), donde n es la longitud del archivo de transmisión y m es la diferencia entre los dos apuntadores que se encuentran en la posición recurrente y al final del archivo en cada iteración.

Parte 3:
El algoritmo utilizado es el Longest Common Substring (LCS) con programación dinámica, el cual tiene una complejidad temporal cuadrática. Es una función que cada caracter de un string contra cada uno del otro para memoizar en una matriz, colocando en el cuadrante donde se encuentra la coincidencia el resultado de sumar 1 al valor del cuadrante superior izquierdo. De esta forma, la tabla va almacenando longitudes de substrings coincidentes y el algoritmo actualiza constantemente una variable con la longitud y posición del substring de mayor tamaño.

La complejidad temporal es del orden de O(n*m), donde n es la longitud de el primer archivo de transmisión y m es la longitud del segundo archivo de transmisión.