Reflexión Individual

Parte 1

Objetivo: Búsqueda de códigos maliciosos en los archivos de transmisión

Algoritmo: Utiliza el algoritmo Z

Complejidad: O(N)

Explicación de Algoritmo: Este algoritmo se utilizó para saber si los contenidos maliciosos se encontraban en en los archivos transmissión dados. La forma en la que funciona es la siguiente: Dado un string y un patrón, en este caso la transmisión y el código malicioso, respectivamente. Se crea un arreglo de enteros de tamaño string.size() que representa la cantidad de caracteres que coinciden con el patrón dado. De este modo, si el tamaño de caracteres del mayor prefijo possible del patrón es encontrado en el string, entonces significa que sí está.

Parte 2

Objetivo: Búsqueda del palíndromo más largo

Algoritmo: Manacher

Complejidad: O(N)

Explicación de Algoritmo: Este algoritmo se utilizó para encontrar el políndromo más grande dentro de una cadena de texto. En este caso, las cadena de texto fueron los archivos de transmissión. Y se buscó el el políndromo más grande porque se asumió que el código malicioso es un políndromo. Se utilizó el algoritmo de manacher para lograr esta función porque este aprovecha la simetría de los políndromos para no repetir iteraciones previamente realizadas. Esto se puede hacer gracias a que hay palindromos dentro de palindromos. En efecto, la complejidad del algoritmo es lineal. La forma en la que funciona es la siguiente: Se crea una lista del tamaño de la longitud de la cadena de texto para guardar las longitudes de los polindromos. Después, el algoritmo itera caracter por caracter, tomonado al caracter como el centro del políndromo y de esta manera, asumiendo que es el aso estandar, donde existe simetría y los lados son iguales y no ha llegado a los límites del arreglo, el algoritmo calcula la longitud del polindromo y lo guarda en el arreglo. Por el otro lado, el algoritmo también considera casos especiales como:

- Caso en donde el subpalíndromo a la izquiera sale del palíndromo mayor y no puede ser igualado por el de la derecha.
- Caso en el que los palíndromos de ambos lados tienen las mismas condiciones, pero el de la derecha todavía tiene espacio para seguir creciendo.
-

Finalmente, conforme almacenamos longitudes en el arreglo de longitudes, se checa si estamos trabajando con la longitud más grande, y si es el caso, reemplzada el valor de la variable que previamente guardaba a la longitud más grande. Con el objetivo de que cuando se termine la iteración, sea facil desplegar la posición de inicio y final del políndromo más grande.

Parte 3

Objetivo: Búsqueda de la subcadena común más larga

Complejidad:O(N*M)

Algoritmo: Programación Dinámica

Explicación de Algoritmo: Se utilizó este algoritmo para encontrar la subcadena de texto más larga dentro de 2 diferentes cadenas de texto. En este caso, estas dos cadenas de texto fueron los archivos de transmisión. Para lograr esto, primero se crea una tabla de tamaño de la longitud de la cadena de texto 1 por la la longitud de la cadena de texto 2, que guarda las longitudes de los sufijos comunes. De este modo, se iteran ambas cadenas utilizando 2 ciclos fors. Esto nos permite guardan la longitud de los sufijos comunes en los indices adecuados. Además, esta iteración también nos permite guardar la longitud de la subcadena más grande y su ubicación. En efecto, teniendo el indice inicial de la subcadena y su longitud (longitud de la subcadena común más larga), al terminar la iteración desplegar los resultados es muy simple.