INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Escuela de Ingeniería

Análisis y diseño de algoritmos avanzados (Gpo 2)

Reflexión individual de la "Actividad Integradora 1"

Presenta:

Ana Cristina Munguía Romero, Matrícula A01740019

Guadalajara, Jal., 17 de septiembre de 2021

Reflexión individual de la "Actividad Integradora 1"

La actividad integradora constó de 3 grandes partes: determinar si existía código malicioso (*mcode1.txt*, *mcode2.txt* o *mcode3.txt*) dentro de los archivos de transmisión (*transmission1.txt* y *transmission2.txt*), encontrar el palíndromo más largo dentro de cada archivo de transmisión y encontrar la subcadena común más larga (*longest common substring*) entre ambos archivos de transmisión.

Antes de poder analizar los archivos de texto para resolver cada parte planteada, sería necesario crear un método para leer los caracteres de un archivo de texto que nos permitiera almacenarlos en una cadena (*string*) para facilitar su manipulación más adelante. Este método tiene una complejidad temporal de *O(n)*, puesto que, cada vez que se corre, realiza *n* iteraciones (donde *n* es el número total de caracteres que se deben leer).

Teniendo resuelta la lectura de archivos, empezamos con la implementación de los métodos para resolver las partes de la actividad. Para la primera parte (búsqueda de una subcadena dentro de otra cadena; en este caso, el código malicioso dentro de los archivos de transmisión), decidimos usar el algoritmo *Knuth-Morris-Pratt*, pues no era demasiado complicado de implementar y tiene una complejidad temporal lineal de *O(n+m)* en el peor de los casos, donde *n* es la cantidad de caracteres contenidos en la cadena y *m* es el número de caracteres contenidos en la subcadena que se está buscando. Este algoritmo necesita de un método que sirva como "preprocesamiento" para generar un arreglo que contenga la cantidad de saltos que se deben dar hacia adelante en la cadena en caso de que algún carácter deje de coincidir con la subcadena buscada.

Para la segunda parte (encontrar el palíndromo más largo en cada archivo de transmisión), optamos por hacer uso del algoritmo de *Manacher*, pues es una solución que aprovecha las características propiedades de los palíndromos para eficientizar el proceso de búsqueda, logrando una complejidad temporal de *O(n)*, donde *n* es el número de caracteres contenidos en la cadena dentro de la cual se busca el palíndromo. Conforme avanzamos, nos dimos cuenta de que el código no funcionaría para cadenas de tamaño par, por lo que decidimos insertar caracteres de gato (#) antes y después de cada caracter para asegurar un tamaño impar; esta edición de las cadenas no afecta la complejidad final de la función para encontrar palíndromos, pues su complejidad de *O(n)* se suma a la complejidad lineal del algoritmo de *Manacher*, dando una complejidad final de *O(2n)* que sigue siendo, para fines prácticos, una complejidad lineal de *O(n)*.

La tercera y última parte de la actividad requería encontrar la subcadena común más larga entre ambos archivos de transmisión. Para lograrlo, hicimos uso de programación dinámica (basándonos fuertemente en el ejemplo de subsecuencias visto en clase, pero alterado para buscar subcadenas); para esto utilizamos una matriz donde almacenaríamos la longitud de la subcadena más larga hasta el momento (colocando un 0 cuando la coincidencia terminara) e identificamos la diagonal más extensa en donde los caracteres coincidían. La complejidad temporal de esta función terminó siendo de $O(nm)$, donde n es la cantidad de caracteres contenidos en el primer archivo de transmisión (*transmission1.txt*) y m es la cantidad de caracteres contenidos en el segundo archivo de transmisión (*transmission2.txt*).