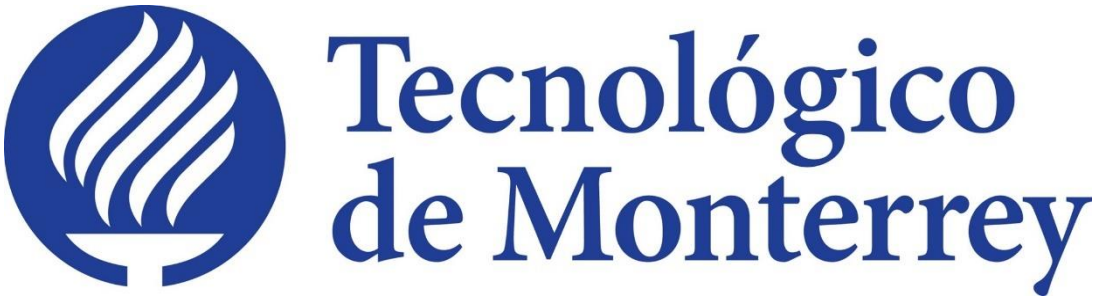

Reflexión Actividad Integradora 1

Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Guadalajara

Diego Sú Gómez – A01620476

Santiago Ortiz Pérez - A01620647

Estefanía Pérez Yeo - A01639270

Análisis y diseño de algoritmos avanzados

Fecha de entrega: martes 4 de octubre de 2022

Semestre agosto – diciembre 2022

Reflexión Actividad Integradora 1

Para la primer actividad integradora del curso, lo que se tuvo que hacer fue un programa que pudiera simular la búsqueda de código malicioso dentro de archivos de transmisión de datos. Para esto, se pidió realizar un código que recibiera 5 archivos de texto, los cuales eran 2 que simulaban archivos de transmisión, y 3 que simulaban código malicioso. Estos tres archivos ("mcode1/2/3.txt") contenían información que se debía de procesar para poder detectar este código dentro de los archivos de transmisión. La actividad pedía procesar estos archivos de tres diferentes formas.

La primera de ellas tenía que analizar si los archivos de transmisión contenían la información de los archivos con código malicioso. Esto significaba que se tenía que hacer una función que pudiera revisar si un archivo de transmisión tenía el contenido de un archivo con texto malicioso y, regresar "True" o "False" dependiendo del escenario. Si el archivo de transmisión contenía el código malicioso, también se tenía que desplegar la posición del archivo de transmisión donde comenzaba el código malicioso. Esta función se podría resumir en el siguiente diagrama:

- **Función 1 Actividad Integradora (analizarArchivos)**
  - analizarArchivos (Transmisión 1 -> MCode1, MCode2, MCode3)
  - analizarArchivos (Transmisión 2 -> MCode1, MCode2, MCode3)
  - Entradas:
    - Archivo de transmisión
    - Archivo de código malicioso
  - Salidas:
    - False/True
      - True: Posición Inicial del MCode en el archivo de transmisión

La segunda función consistía en buscar código espejeado, es decir, palíndromos en los archivos de transmisión. Esto quiere decir que la función debía ser capaz de buscar dentro de los archivos de transmisión el palíndromo más largo. Además, se especificó que se podía asumir que siempre iba a encontrarse este código espejeado, lo que permitió modelar la función de esta manera.

- **Función 2 Actividad Integradora (palindromoMasLargo)**
  - maxPalindromo(Transmisión 1)
  - maxPalindromo(Transmisión 2)
  - Entradas;
    - Archivo de transmisión
  - Salidas:
    - Posición inicial del palíndromo más largo
    - Posición final del palíndromo más largo

Finalmente, la tercer y última función requería analizar ambos archivos de transmisión entre sí, para poder encontrar el sub-string más largo en común de ambos archivos. La función se explica por sí misma, pero se puede entender más a fondo con el diagrama de a continuación.

- **Función 3 Actividad Integradora (substringMasLargo)**
  - substringMasLargo(Transmisión 1, Transmisión 2)
  - Entradas:
    - Archivo de transmisión 1
    - Archivo de transmisión 2
  - Salidas:
    - Posición inicial del palíndromo más largo en el archivo de transmisión 1
    - Posición final del palíndromo más largo en el archivo de transmisión 1

Después de haber explicado lo que cada una de las funciones pedía, lo siguiente que se tuvo que hacer fue implementar ya cada una de ellas en un programa, para lograr hacer lo que pedía la evidencia. A continuación, se procederá a explicar detalladamente la implementación de cada una de las funciones, junto con su complejidad temporal.

La primera función implementada fue una que no se pedía en la actividad, pero fue de mucha utilidad para simplificar el código. Esta función tiene de nombre "leerArchivo", y toma como parámetro un string con el nombre del archivo, y regresa otro string con el contenido de este archivo. Esta función tiene una **complejidad temporal** de **O(n)**, es decir que es constante. La forma en la que funciona es que el string que recibe lo asigna a una variable FILE, lo que permite abrir este archivo, que está ubicado en la misma locación que el programa, para después leer los caracteres y enviarlos al string que se retorna como valor final, y luego cerrar el archivo. Esto permitió simplificar mucho código para no tener que abrir cada uno de los archivos.

La siguiente función implementada fue la de analizar los archivos para comprobar si había código malicioso de los archivos MC (MCode1, Mcode2, Mcode3) en los de transmisión. Esta función fue llamada analizarArchivos, y era una función vacía que tomaba como argumentos los strings del archivo de transmisión y del código malicioso, junto con los strings de los nombres de los archivos, para dar formato a la salida. Esta función tiene una **complejidad temporal** de **O(n)**, puesto que únicamente tiene un ciclo "for" que recorre el string del archivo de transmisión. El principio de funcionamiento de esta función es que va a recorrer el string del archivo de transmisión caracter por caracter, y va a ir formando sub-strings del tamaño de la longitud del string del archivo malicioso. Este sub-string luego va a ser comparado con el string del archivo malicioso y, si hay una coincidencia idéntica, se va a mostrar "True" y la posición inicial del sub-string, que sería el índice actual del ciclo 'for'. Si llega a completar el ciclo y no encontró coincidencia, se mostrará el "False" y se termina la función.

La tercera función que se implementó fue la de encontrar el palíndromo más largo dentro de los archivos de transmisión. La función se llama maxPalindromo y tiene una **complejidad temporal** de **O(n^2)**. Esta función lo que hace es que va a tomar el string del archivo de transmisión, para evaluar los palíndromos dentro de ella y después encontrar el más largo y mostrarlo. Para lograr esto, se crea una matriz de dos dimensiones, en donde se va a almacenar un valor booleano si es palíndromo o no. Se considera que cualquier string de longitud 1 es palíndromo, y después se valúan los strings de dos caracteres, donde tienen que ser exactamente iguales ambos caracteres para que sea válido el palíndromo. Después de eso, se evalúan los strings de más de 2 caracteres, y esto se hace por medio de un ciclo, donde se buscan los palíndromos, y si se llega a encontrar uno, se evalúa la longitud máxima del palíndromo encontrado. Después de terminar todos estos ciclos, se muestra el palíndromo más largo encontrado.

Finalmente, la cuarta función que se implementó fue la de encontrar el sub-string más largo en común entre los archivos de transmisión. Esta tiene de nombre substringMasLargo, igualmente es una función vacía que pide de argumentos los dos strings de los archivos de transmisión. Esta función tiene una **complejidad temporal** de **O(n^2)**, y la razón se explicará a continuación. La función va a recorrer cada uno de los strings de los archivos de transmisión, y evaluará cada carácter de cada string. Si encuentra una coincidencia entre caracteres, guardará su índice en un vector, uno para cada archivo, y luego en un arreglo de dos dimensiones, almacenará la longitud más larga del sub-string. Después de haber hecho eso, va a evaluar si la longitud máxima actual es mayor a la almacenada. Si es el caso, se va a actualizar y se va a guardar la posición del arreglo. Si no es el caso, se va a modificar el valor del arreglo a 0, para finalmente después de que concluya el ciclo, obtener la longitud mayor del sub-string. Con eso hecho, se obtiene el índice inicial del archivo uno con el primer elemento del vector, y luego se obtiene el índice final con el inicial + la longitud máxima -1 para encontrar ambas posiciones.

Estas funciones permitieron resolver la situación problema presentada para esta actividad integradora, además de que ayudaron a comprender mejor los temas vistos en clase debido a que contribuyeron a la implementación de estas funciones y a poder completar las tareas realizadas. Esta actividad además, ayudó a desarrollar mis habilidades computacionales y las de trabajo en equipo, ya que fue algo muy importante para poder completar la actividad. El estar comunicados entre los miembros del equipo fue clave para que todos estuviéramos en la misma página y pudiéramos realizar adecuadamente la actividad. Por otro lado, también fue muy útil la cooperación porque ayudó a detectar errores, corregirlos, y a eficientizar el código y su funcionamiento. En general, esta actividad creo que fue muy fructífera por lo aprendido y las habilidades que me ayudó a realizar.