Instituto Tecnológico de Estudios Superiores Monterrey

CAMPUS QUERÉTARO

Análisis y diseño de algoritmos avanzados

Ramona Fuéntes Valdéz

Grupo 602

AI1_Actividad Integradora 1

PRESENTAN

Rodrigo Terán Hernández A01704108

Fecha:
01/10/2023

Índice

Problemática

Introducción

En la situación problema se nos presenta un escenario en donde se tienen archivos de transmisión de datos y unos archivos con código malicioso.

Nuestro trabajo es identificar posibles amenazas dentro de los archivos de transmisión utilizando las coincidencias de código que se encuentran en los archivos con código malicioso.

Coincidencias del código

Para identificar cuáles son las coincidencias dañinas vamos a dividir el proceso en 3 etapas:

- Identificar si hay código malicioso en los archivos de transmisión y dónde hallarlos.
- Identificar si hay código espejado.
- Identificar cuál es la secuencia de caracteres más larga que comparten los archivos de transmisión.

Código malicioso

Para identificar si hay código malicioso en los archivos de transmisión debemos de buscar si la secuencia de caracteres se encuentra en el archivo de transmisión, y si el caso es positivo decir en qué lugar se encuentra.

Código espejeado

Si hay una secuencia de caracteres que se puede leer de derecha a izquierda y viceversa de la misma manera es un palíndromo. En esta situación problema este caso también es considerado en la corrupción de archivos. Entonces tenemos que encontrar el palíndromo más grande de cada archivo de transmisión.

Similitud de archivos

También es necesario saber qué tan similares son los archivos de transmisión, por lo tanto debemos de conocer cuál es la secuencia de caracteres más grande entre ambos archivos y saber en dónde está.

Propuesta de solución

En este apartado se presentará la propuesta de solución para cada caso:

Código malicioso

Para encontrar si un patrón existe en un archivo utilicé el algoritmo de Knuth Morris Pratt, ya que es un algoritmo para encontrar patrones en textos de una manera muy eficiente.

Complejidad

Tiene una complejidad de $O(n + m)$

En donde

$$m = La\ longitud\ del\ texto\ a\ buscar$$

$$n = La\ longitud\ del\ patrón\ a\ buscar$$

Selección de este algoritmo

Se escogió este algoritmo ya que es muy eficiente al encontrar un patrón en un texto. En este programa se buscaba que el algoritmo fuera lo más eficiente posible.

Código espejeado

Para encontrar el palíndromo más grande en un texto se decidió utilizar el algoritmo de Manacher.

Complejidad

Tiene una complejidad de $O(n)$

En donde

$$n = La\ longitud\ del\ texto\ a\ buscar\ el\ palíndromo$$

Selección de este algoritmo

Se escogió este algoritmo ya que es muy eficiente al encontrar el palíndromo más grande dentro de un texto.

Similitud de archivos

Para encontrar la subcadena de caracteres más grande en común dentro de dos cadenas de texto se decidió utilizar programación dinámica.

Complejidad

Tiene una complejidad de $O(m * n)$

En donde

$$m = El\ primer\ texto$$

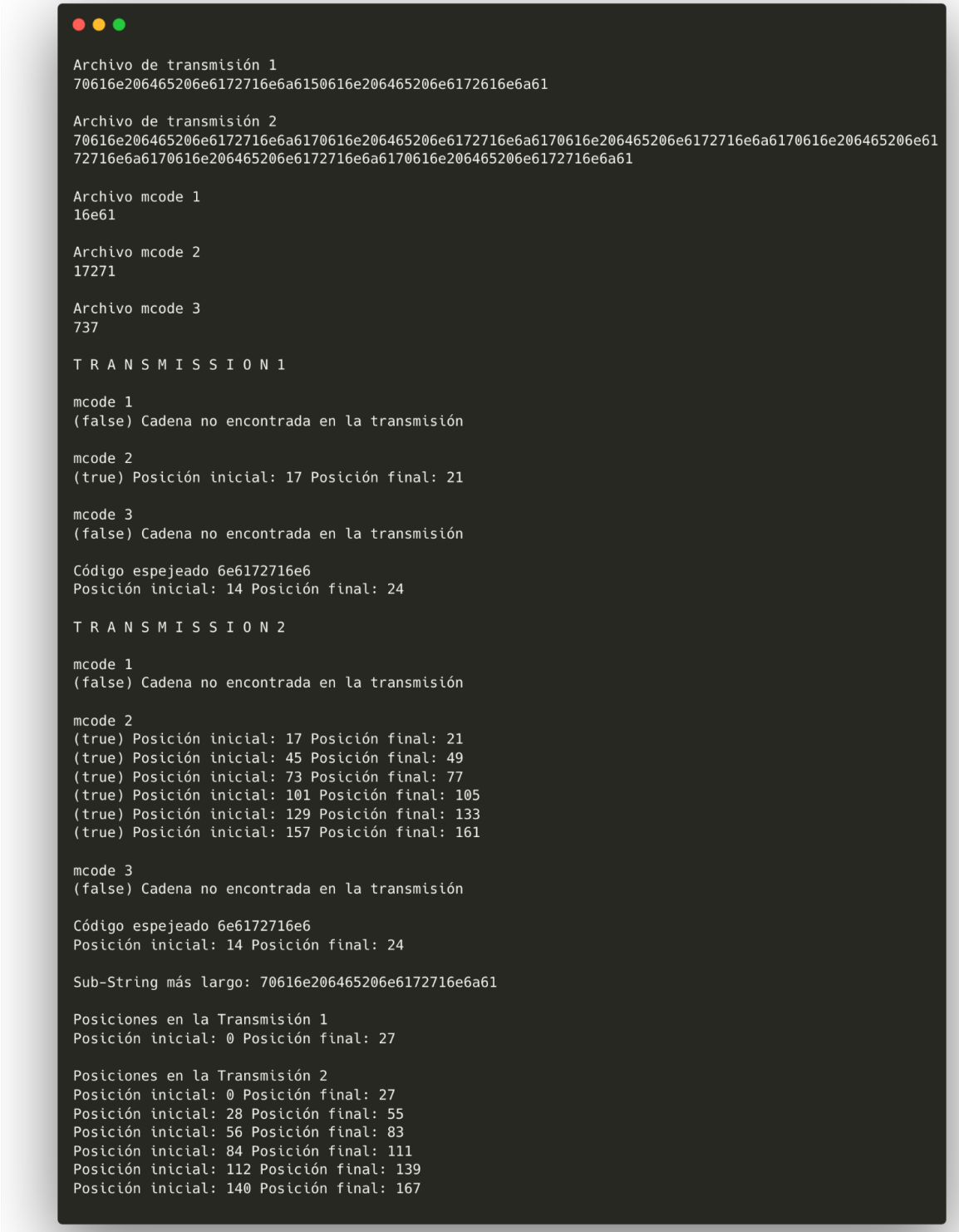
$$n = El\ segundo\ texto$$

Selección de este algoritmo

Se escogió la programación dinámica ya que en este caso es muy fácil implementarlo. Con ayuda de la programación dinámica vamos construyendo las soluciones desde el caso más sencillo hasta llegar al caso original. También la complejidad algorítmica es muy eficiente.

Ejemplo

Este es un ejemplo de la salida de el programa implementado:



```
Archivo de transmisión 1
70616e206465206e6172716e6a6150616e206465206e6172616e6a61

Archivo de transmisión 2
70616e206465206e6172716e6a6170616e206465206e6172716e6a6170616e206465206e6172716e6a6170616e206465206e61
72716e6a6170616e206465206e6172716e6a6170616e206465206e6172716e6a61

Archivo mcode 1
16e61

Archivo mcode 2
17271

Archivo mcode 3
737

T R A N S M I S S I O N 1

mcode 1
(false) Cadena no encontrada en la transmisión

mcode 2
(true) Posición inicial: 17 Posición final: 21

mcode 3
(false) Cadena no encontrada en la transmisión

Código espejeado 6e6172716e6
Posición inicial: 14 Posición final: 24

T R A N S M I S S I O N 2

mcode 1
(false) Cadena no encontrada en la transmisión

mcode 2
(true) Posición inicial: 17 Posición final: 21
(true) Posición inicial: 45 Posición final: 49
(true) Posición inicial: 73 Posición final: 77
(true) Posición inicial: 101 Posición final: 105
(true) Posición inicial: 129 Posición final: 133
(true) Posición inicial: 157 Posición final: 161

mcode 3
(false) Cadena no encontrada en la transmisión

Código espejeado 6e6172716e6
Posición inicial: 14 Posición final: 24

Sub-String más largo: 70616e206465206e6172716e6a61

Posiciones en la Transmisión 1
Posición inicial: 0 Posición final: 27

Posiciones en la Transmisión 2
Posición inicial: 0 Posición final: 27
Posición inicial: 28 Posición final: 55
Posición inicial: 56 Posición final: 83
Posición inicial: 84 Posición final: 111
Posición inicial: 112 Posición final: 139
Posición inicial: 140 Posición final: 167
```

Conclusiones

El uso de algoritmos eficientes en problemas que requieran el manejo de strings es de suma importancia. A menudo damos por hecho que el manejo de strings es muy sencillo ya que nosotros como humanos vemos el texto como algo de toda la vida… sencillo de buscar información. Pero cuando realmente nos ponemos a analizar la complejidad que conlleva encontrar patrones, similitudes y formas muy específicas de cadenas de texto el asunto se empieza a dificultar.

En este situación problema predominó el uso de los strings. Al final de la resolución de esta actividad destaco la importancia de encontrar patrones, ya que en un entorno real de software esta situación puede ser crucial. Es posible que un archivo haya sido contaminado y encontrar un código malicioso es muy importante.

Referencias

Longest Common Substring | DP-29. Geeks for Geeks. Recuperado de: https://www.geeksforgeeks.org/longest-common-substring-dp-29/

Manacher's Algorithm. Scaler. Recuperado de: https://www.scaler.com/topics/data-structures/manachers-algorithm/

KMP Algorithm for Pattern Searching. Geeks for Geeks. Recuperado de: https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/