Tecnológico de Monterrey

E1. Actividad Integradora 1

Análisis y diseño de algoritmos avanzados (Gpo 603)

Diego Enrique Vargas Ramírez | A01635782

Francisco Javier Romo Juárez | A01643189

Iñaki González Morales | A01612126

Adair Virgilio Figueroa Medina | A00572826

Ernesto Puga Araujo | A00572845

09/ 22/ 2024

Algoritmos

1. fileRead()

- Complejidad: O(n), donde n es el número total de líneas en todos los archivos.
- Justificación: La función lee cada línea de cada archivo una vez.

Como se mencionó anteriormente, esta función sirve para leer cada linea de cada archivo(tanto los ‘transmission’ como los ‘mcode’. Después de leer una línea, esta se ingresa a un vector que es diferente para cada archivo. Finalmente se cierran los archivos.

2. printFiles()

- Complejidad: O(n), donde n es el número total de líneas en todas las transmisiones.
- Justificación: Itera sobre cada línea de las transmisiones una vez.

Esta función sirve para imprimir todos los archivos de entrada.

3. preKMP()

- Complejidad: O(m), donde m es la longitud del código (pattern).
- Justificación: Utiliza un bucle que recorre el código una vez.

La función preKMP() usa el algoritmo Knuth-Morris-Pratt (KMP) para calcular el vector Longest Prefix which is also Suffix (LPS). Primero se recorre el string que se recibe como argumento y se va construyendo el vector LPS. Si encuentra un carácter igual entre la posición actual y un prefijo anterior, incrementa el valor de LPS. De no ser así, usa el valor anterior de LPS para continuar la búsqueda.

4. KMP()

- Complejidad: O(n * m), donde n es la longitud total de todas las líneas de transmisión y m es la longitud del código.
- Justificación: Para cada línea de la transmisión (n), realiza el algoritmo KMP que tiene una complejidad de O(m).

Esta función implementa el algoritmo KMP que nos permite encontrar todas las ocurrencias de un patrón dado en cada línea del vector ‘transmission’.

5. preManacher()

- Complejidad: O(m), donde m es la longitud del código.
- Justificación: Recorre el código una vez, añadiendo caracteres.

Esta función pre-procesa cada línea de un código que se ingresa como argumento para después ser usada en la función ‘manacher()’. Lo que hace es insertar un símbolo ’# entre cada carácter, empezando ’ desde la posición 0 de un string hasta la posición final de este.

6. manacher()

- Complejidad: O(n), donde n es la longitud total de todas las líneas de transmisión.
- Justificación: Para cada carácter en la transmisión, realiza operaciones constantes.

Se utiliza la función preManacher() para poder pre-procesar cada línea. Después de esto se encuentra el palíndromo más largo en cada línea.

7. LCS()

- Complejidad: O(n * m), donde n es la longitud de la primera transmisión y m es la longitud de la segunda transmisión.
- Justificación: Utiliza una matriz de programación dinámica de tamaño n x m y la llena completamente.

El algoritmo de Longest Common Substring construye una tabla para almacenar las longitudes de las subcadenas comunes en las posiciones correspondientes de los strings. Utilizando esta tabla, el programa nos brinda el substring común entre ambos archivos de transmisión más largo.

COMPLEJIDAD GENERAL

En general, la complejidad del programa está dominado por las funciones más costosas para el procesamiento, es decir KMP() y LCS(). Por eso la complejidad general es O(n * m), donde n es la longitud total de los flujos y m es la longitud del código más largo o la longitud del flujo más largo,dependiendo lo que sea mayor.