Tecnológico de Monterrey

InstitutoTecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Análisis y diseño de algoritmos avanzados (Gpo 604)

E1. Actividad Integradora 1

Profesor:
Felipe Castillo Rendón

Alumno:
Mauricio Lozano Zárate A00833216
Aleksandra Stupiec A00835071
Enrique Macías López A01641402

Fecha de entrega:
03/10/2024

Reflexión sobre los algoritmos aplicados y su complejidad

En este proyecto, desarrollamos un programa en C++ que analiza archivos de transmisión y busca secuencias maliciosas, identifica palíndromos, y encuentra la subsecuencia común más larga entre dos archivos de transmisión. Aquí explicamos cómo funcionan los algoritmos que usamos y qué tan eficientes son.

Parte 1: Búsqueda de secuencias maliciosas en las transmisiones

Para buscar las secuencias maliciosas, usamos la función find() de C++. Esta función busca si una cadena de texto (la secuencia maliciosa) aparece dentro de otra cadena (el archivo de transmisión).

- **Algoritmo**: Búsqueda de subcadenas usando find().
- **Complejidad**: La función find() recorre el archivo hasta encontrar la secuencia o llegar al final, lo que toma tiempo proporcional al tamaño del archivo, es decir, O(n), donde n es la longitud del archivo.

Parte 2: Búsqueda de palíndromos en las transmisiones

Para encontrar el palíndromo más largo en las transmisiones, comparamos todas las posibles subcadenas del archivo para ver si son palíndromos. Un palíndromo es una secuencia de caracteres que se lee igual de adelante hacia atrás.

- **Algoritmo**: Comparamos todas las subcadenas del archivo para ver si son palíndromos.
- **Complejidad**: Este proceso es más lento, con una complejidad de $O(n^2)$, porque para cada subcadena tenemos que revisar si es un palíndromo, donde n es la longitud del archivo.

Parte 3: Búsqueda del substring más largo común entre dos transmisiones

Para encontrar la subsecuencia común más larga entre dos transmisiones, usamos un método llamado programación dinámica. Esto significa que vamos comparando poco a poco ambas transmisiones y guardamos los resultados de comparaciones anteriores para no repetir trabajo.

- **Algoritmo**: Programación dinámica que compara las dos transmisiones letra por letra.
- **Complejidad**: La complejidad es O(m * n), donde m y n son las longitudes de las transmisiones. Esto es más eficiente que comparar todas las subcadenas una por una.

Complejidad general

1. **Parte 1**: Búsqueda de secuencias maliciosas: O(n).
2. **Parte 2**: Búsqueda de palíndromos: $O(n^2)$.
3. **Parte 3**: Substring común más largo: O(m * n).