UNIVERSIDAD TECNOLÓGICA METROPOLITANA del Estado de Chile
UTEM

Luis H. Herrera
I-2024

Análisis y Diseño de Algoritmos

Trabajo II

Particionamiento de palíndromos

0.0.1 Introducción

Dada una cadena str, una partición de la cadena es una partición palíndorme si cada subcadena de la partición es un palíndromo. La tarea es encontrar el número mínimo de cortes necesarios para la partición palíndrome de la cadena dada.

Ejemplo 1:

Cadena: **ababbbabbababa**

Particionamiento de palíndromos: **a — babbbab — b — aba**

Ejemplo 2:

Cadena: **geek**

Particionamiento de palíndromos: **g — ee — k**

Explicación: Necesitamos hacer un mínimo de 2 cortes, es decir, ”g — ee — k”.

0.0.2 Algoritmos

A- Para optimizar los subproblemas superpuestos en el enfoque recursivo para el particionamiento de palíndromos en $O(n^3)$:

En esta solución, podemos usar dos arreglos 2D, C[][] y P[][], para almacenar el resultado computado.

- C$[i][j]$ almacena el número mínimo de cortes necesarios para la subcadena cadena$[i..j]$,
- mientras que P$[i][j]$ almacena si la subcadena cadena$[i..j]$ es un palíndromo o no.
- Comienza con subcadenas más pequeñas y gradualmente se construye hasta la cadena completa.

B- Enfoque de programación dinámica para el particionamiento de palíndromos en $O(n^2)$:

El problema se puede resolver encontrando el sufijo que comienza en j y termina en el índice i, $(1 \leq j \leq i \leq n-1)$, que son palíndromos. Por lo tanto, podemos hacer un corte aquí que requiere 1 + el mínimo de cortes del subtexto restante $[0, j-1]$. Para todos estos sufijos palíndromos que comienzan en j y terminan en i, seguimos minimizando en minCutDp$[i]$. Del mismo modo, necesitamos calcular resultados para todos estos i. $(1 \leq i \leq n-1)$ y finalmente, minCutDp$[n-1]$ será el número mínimo de cortes necesarios para el particionamiento palíndromo de la cadena dada.

0.0.3 Objetivos

Se le pide:

- Implementar los algoritmos anteriormemte descritos.
- Medir el tiempo de ejecución de cada algoritmo para diferentes tamaños de cadenas formadas de forma aleatoria y guardadas en un archivo de texto.
- Analizar la complejidad computacional de cada algoritmo de forma teórica.
- Comparar el rendimiento de los diferentes algoritmos a través de gráficos para las mismas entradas de datos.

0.0.4 Metodología

- Estudio de la bibliografía (libros, videos, etc.).
- Programación: Implementación del algoritmo.
- Análisis de complejidad: Analizar la complejidad computacional de cada algoritmo utilizando recurrencias y los métodos vistos en clase.
- Comparación de resultados: Comparar el rendimiento de los diferentes algoritmos en base a las mediciones de tiempo y la complejidad computacional.
- Redactar un informe final que incluya la introducción, metodología, resultados, conclusiones y referencias en formato latex.
- El alumno deberá enviar un documento en pdf hecho en latex, también el archivo .tex donde escribió su informe final, además de los 2 algoritmos juntos en un solo programa el programa deberá de un archivo generado por usted mismo con palabras, computarlas con los dos procedimientos y generar un gráfico.