UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA CAMPUS ZACATECAS
INSTITUTO POLITÉCNICO NACIONAL

PRÁCTICA 1: ANÁLISIS DE CASOS

16 de septiembre de 2023

Análisis y diseño de algoritmos
Juan F. Alba Avalos 3CM1
Unidad I

0.1. Objetivo de la práctica

El objetivo principal de esta práctica es que los estudiantes adquieran una comprensión sólida y profunda de los conceptos clave relacionados con la complejidad computacional de los algoritmos. Los tres casos principales a analizar, mejor caso, peor caso y caso promedio, permiten una evaluación completa del rendimiento de un algoritmo bajo diferentes circunstancias.

0.1.1. Objetivos particulares

- Comprender la Variabilidad de la Ejecución: Los algoritmos pueden comportarse de manera diferente según las características de los datos de entrada. Al analizar el mejor, peor y caso promedio, los estudiantes aprenderán a apreciar cómo un algoritmo puede variar en su rendimiento.
- Aplicar Conceptos de Complejidad Computacional: La práctica ayudará a los estudiantes a aplicar conceptos fundamentales de la teoría de la complejidad computacional, como la notación Ogrande (Big O), para describir y comparar el rendimiento de los algoritmos
- Desarrollar Habilidades de Análisis Crítico: A través del análisis de diferentes casos, los estudiantes mejorarán sus habilidades de análisis crítico, aprendiendo a identificar situaciones en las que un algoritmo puede ser más eficiente o ineficiente.
- Preparación para Desarrollo de Algoritmos Eficientes: Al comprender los casos de mejor, peor y caso promedio, los estudiantes estarán mejor preparados para diseñar y seleccionar algoritmos eficientes en situaciones reales, lo que es fundamental en la resolución de problemas computacionales.
- Promover la Resolución de Problemas: A través de la práctica, los estudiantes aprenderán a abordar y resolver problemas computacionales de manera más efectiva, seleccionando o adaptando algoritmos en función de las restricciones de tiempo y recursos.

0.2. Desarrollo de la práctica

0.2.1. Implementación de los algoritmos

0.2.1.1. Calcular el mejor, el peor y el caso promedio

Algoritmo de burbuja:

Mejor Caso:

El mejor caso ocurre cuando la lista ya está ordenada. En este caso, el algoritmo solo realizará una pasada por la lista para verificar que no haya intercambios necesarios. El número de comparaciones será igual a n-1, donde n es el tamaño de la lista.

Ejemplo: 1, 2, 3, 4, 5

Peor Caso:

El peor caso ocurre cuando la lista está ordenada en orden inverso, y el algoritmo necesita realizar el máximo número de intercambios posible en cada paso. Esto resulta en un rendimiento cuadrático, con n*(n-1)/2 comparaciones y n*(n-1)/2 intercambios.

Ejemplo: 5, 4, 3, 2, 1

Caso Promedio:

El caso promedio depende de factores como la distribución de datos en la lista y la estrategia de ordenamiento. En promedio, se espera que el algoritmo realice aproximadamente n*(n-1)/4 comparaciones e intercambios si los datos son aleatorios.

Ejemplo: 5, 3, 2, 1, 4

0.2.2. Análisis de casos

Algoritmo de burbuja optimizada:

Los casos que ocurren para el algoritmo de burbuja optimizada es similar al de burbuja. Sin embargo, la versión optimizada puede tener un mejor rendimiento en listas parcialmente ordenadas, lo que puede hacer que el caso promedio sea ligeramente mejor en tales situaciones.

Mejor caso burbuja optimizada:

- 1, 2, 3, 4, 5
- 10, 20, 30, 40, 50, 60
- 100, 200, 300, 400, 500, 600, 700

Peor caso burbuja optimizada:

- 5, 4, 3, 2, 1
- 50, 49, 48, 47, 46, 45
- 500, 499, 498, 497, 496, 495, 494

Caso promedio burbuja optimizada (Aleatorio):

- 3, 1, 5, 2, 4
- 20, 10, 30, 60, 40, 50
- 400, 300, 100, 500, 700, 600, 200

0.2.3. Análisis de casos

Para justificar a cuál clase de complejidad pertenece cada caso de cada algoritmo, debemos analizar cuántas operaciones o comparaciones realiza el algoritmo en función del tamaño de la entrada (n)

Algoritmo de Burbuja:

- Mejor Caso (Ordenada): O(n)

En el mejor caso, el algoritmo realiza un número lineal de comparaciones, ya que solo necesita una pasada por la lista para verificar que no haya intercambios necesarios.

- Peor Caso (Inversamente Ordenada): $O(n^2)$

En el peor caso, el algoritmo realiza un número cuadrático de comparaciones e intercambios, ya que necesita realizar muchos intercambios en cada paso.

- Caso Promedio (Aleatorio): $O(n^2)$

En el caso promedio, el algoritmo también realiza un número cuadrático de comparaciones e intercambios si los datos son aleatorios.

Algoritmo de Burbuja Optimizada:

- Mejor Caso (Ordenada): O(n)

Al igual que en el caso de burbuja regular, en el mejor caso, el algoritmo realiza un número lineal de comparaciones.

- Peor Caso (Inversamente Ordenada): $O(n^2)$

Aunque la versión optimizada reduce el número de comparaciones en comparación con el caso regular, todavía es cuadrático en el peor caso.

- Caso Promedio (Aleatorio): $O(n^2)$

Al igual que en el caso de burbuja regular, en el caso promedio, el algoritmo realiza un número cuadrático de comparaciones e intercambios si los datos son aleatorios.

0.2.4. Comparación de resultados

- El mejor caso para ambos algoritmos es O(n) porque realizan un número lineal de comparaciones en ese escenario
- El peor caso para ambos algoritmos es O(n al 2) porque realizan un número cuadrático de comparaciones en el peor de los casos
- El caso promedio para ambos algoritmos es O(n al 2) porque, en promedio, también realizan un número cuadrático de comparaciones e intercambios si los datos son aleatorio

Estas complejidades pueden variar ligeramente según la implementación que tengan además de los detalles del algoritmo, pero usualmente estas son las clasificaciones típicas para los dos algoritmos.

0.3. Conclusión

Durante la práctica se pudieron analizar dos tipos de ordenamiento para una lista, el método de burbuja y de burbuja optimizada, métodos que en un principio son bastante similares, sin embargo, llegan a tener sus diferencias y caracteristicas que los hacen una mejor o peor opcion según sea el caso.
Se realizó una comparativa donde se pudo clasificar cuál era su complejidad Big O según el caso que se pudiera llegar a tener (mejor caso, peor caso y un caso aleatorio) para poder determinar sus ventajas y desventajas en una aplicacion sencilla como un ordenamiento de una lista determinada por el usuario.