Práctica 01

Hiram Caleb Gutiérrez Olague

September 15, 2023

Análisis y diseño de algoritmos

1 Introducción

La complejidad computacional de un algoritmo es un concepto fundamental en informática que afecta la eficiencia y el rendimiento de los programas informáticos. Esta práctica está diseñado para proporcionar una comprensión sólida y profunda de estos conceptos clave, centrándose en los tres casos principales de análisis de algoritmos: mejor caso, peor caso y caso promedio. A través de esta práctica, se explorará cómo los algoritmos se comportan de manera diferente según las características de los datos de entrada y cómo aplicar conceptos de complejidad computacional (como la notación "Big O") para describir y comparar el rendimiento de los algoritmos. . Además, desarrollarán habilidades de análisis crítico identificando situaciones en las que los algoritmos pueden ser más o menos eficientes, algo fundamental para la resolución de problemas computacionales.

2 Objetivos

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.

2.1 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.

3 Desarrollo

3.1 Análisis de casos

3.1.1 Casos del ordenamiento de burbuja normal

El peor caso para el ordenamiento de burbuja normal tiene lugar cuando la lista esté en el orden opuesto, por ejemplo:

1. $[5, 4, 3, 2, 1]$
2. $[10, 8, 6, 4, 2, 0]$
3. $[11, 9, 7, 5, 3, 1]$

El mejor caso para el ordenamiento de burbuja normal se cumple cuando la lista esté en el orden correcto desde un inicio, por ejemplo:

1. $[1, 2, 3, 4, 5]$
2. $[0, 2, 4, 6, 8, 10]$
3. $[1, 3, 5, 7, 9, 11]$

Por otro lado, los casos promedio se llevan a cabo cuando el orden de la lista es aleatorio, por ejemplo:

1. $[1, 3, 2, 5, 4]$
2. $[0, 8, 10, 6, 2, 4]$
3. $[11, 3, 5, 1, 9, 7]$

3.1.2 Casos del ordenamiento de burbuja optimizado

Al igual que el caso del ordenamiento burbuja normal, el peor caso que se puede tener con el ordenamiento de burbuja optimizado es cuando la lista está en el orden opuesto:

1. $[16, 12, 8, 4, 0]$
2. $[25, 20, 15, 10, 5]$
3. $[18, 15, 12, 9, 6, 3]$

El mejor caso para el ordenamiento de burbuja optimizado se cumple cuando la lista esté en el orden correcto desde un inicio, por ejemplo:

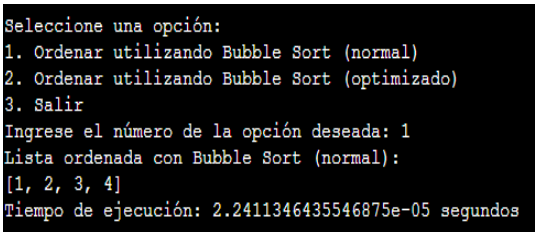
1. $[0, 4, 8, 12, 16]$
2. $[5, 10, 15, 20, 25]$
3. $[3, 6, 9, 12, 15, 18]$

Por otro lado, los casos promedio se llevan a cabo cuando el orden de la lista es aleatorio, por ejemplo:

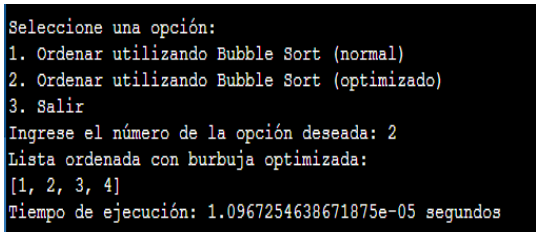
1. $[0, 12, 8, 16, 4]$
2. $[25, 10, 5, 15, 20]$
3. $[15, 9, 12, 3, 6, 18]$

3.1.3 Comparación de los ordenamientos en cuanto a tiempo

$$\text{Lista} = [1, 4, 2, 3]$$



(a) Burbuja normal



(b) Burbuja optimizada

Figure 1: Tiempo de ejecucion de los ordenamientos

Como podemos notar, el ordenamiento burbuja optimizado dentro del caso promedio tardó poco más de la mitad del tiempo menos que el ordenamiento burbuja normal.

3.2 Notación Big O de ambos ordenamientos

La notación Big O de ambos tipos de ordenamiento es $O(n^2)$ ya que el tiempo de ejecución varia en función al cuadrado de la entrada. En el caso del ordenamiento de burbuja normal (OBN) se comparan los elementos adyacentes y se cambian de lugar, si es necesario, hasta que la lista esté ordenada. A diferencia del OBN, el ordenamiento de burbuja optimizado, se comparan todos los elementos de la lista y se comienzan a ordenar del menor al mayor.

4 Conclusiones

La práctica de analizar la complejidad computacional de los algoritmos es esencial para construir una base sólida en informática. A lo largo de la experiencia, exploramos los conceptos de mejor de los casos, peor de los casos y caso promedio, lo que les permitió comprender cómo cambia el rendimiento del algoritmo según la entrada. Además, aprendió a aplicar la notación "Big O" y otros conceptos de complejidad para describir y comparar algoritmos de manera efectiva. Esta práctica también mejora las habilidades analíticas críticas al identificar situaciones en las que los algoritmos pueden ser más o menos efectivos.

5 Referencias

González-Díaz, J. (2021, June 30). El problema del camino más corto: algoritmos y aplicaciones. https://minerva.usc.es/xmlui/handle/10347/28914
Gude Santos, A. El problema del camino más corto: algoritmos y aplicaciones.