TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

Programación I

Algoritmos de Búsqueda y Ordenamiento en Python

Alumnos:

- Rodriguez Claudio - rodriguezop8@gmail.com
- Rinaldi Patricio Tomas - patricio.rinaldi@hotmail.com

Comisión: 20

Índice

1. Introducción

Los algoritmos de búsqueda y ordenamiento son pilares fundamentales en el desarrollo de software, proporcionando soluciones eficientes para la gestión y organización de información. La elección de este tema se fundamenta en su importancia práctica dentro de la programación, ya que estos algoritmos son ampliamente utilizados en bases de datos y sistemas de archivos, aplicaciones web y motores de búsqueda, entre otros.

La importancia de estos algoritmos está en su capacidad para optimizar el acceso y manipulación de datos, mejorando significativamente el tiempo de ejecución de programas que manejan grandes volúmenes de información.

2. Marco Teórico

Algoritmos de Ordenamiento

Los algoritmos de ordenamiento permiten reorganizar una colección de elementos de acuerdo a un criterio específico, como orden alfabético o numérico. Los algoritmos implementados en este trabajo son:

Bubble Sort (Ordenamiento Burbuja): Funciona comparando pares de elementos y intercambiándolos si están en el orden incorrecto. Este proceso se repite hasta que no se requieren más intercambios. Su complejidad temporal es $O(n^2)$ en el peor caso.

Selection Sort (Ordenamiento por Selección): Funciona encontrando el elemento más pequeño del arreglo y colocándolo en la primera posición, luego encuentra el segundo más pequeño y lo coloca en la segunda posición, y así sucesivamente. Su complejidad temporal es $O(n^2)$ pero realiza menos intercambios que Bubble Sort.

Insertion Sort (Ordenamiento por Inserción): Construye la lista ordenada elemento por elemento, insertando cada nuevo elemento en la posición correcta dentro de la porción ya ordenada. Es eficiente para listas pequeñas o parcialmente ordenadas, con complejidad $O(n)$ en el mejor caso.

Algoritmos de Búsqueda

Los algoritmos de búsqueda tienen como objetivo encontrar un elemento específico dentro de una colección de datos:

Búsqueda Lineal: Recorre secuencialmente todos los elementos de la lista hasta encontrar el elemento deseado o llegar al final. No requiere que los datos estén previamente ordenados. Su complejidad temporal es $O(n)$.

Búsqueda Binaria: Es un algoritmo eficiente que funciona solo en listas ordenadas. Divide repetidamente el espacio de búsqueda por la mitad, comparando el elemento central con el valor buscado. Su complejidad temporal es $O(\log n)$.

3. Caso Práctico

Se desarrolló un programa en Python que permite al usuario gestionar una lista de personas aplicando diferentes algoritmos de búsqueda y ordenamiento.

Características Principales:

1. **Carga de Datos:** El usuario ingresa los datos manualmente o genera daros aleatorios para pruebas
2. **Algoritmos de Ordenamiento:**
    - Bubble Sort
    - Selection Sort
    - Insertion Sort
3. **Algoritmos de Búsqueda:**
    - Búsqueda lineal
    - Búsqueda binaria

4. Metodología Utilizada

El desarrollo del trabajo se realizó siguiendo una metodología estructurada en las siguientes etapas:

Investigación Previa:

- Consulta de documentación oficial de Python
- Revisión del material brindado en la plataforma
- Análisis de implementaciones de referencia

Diseño y Planificación:

- Definición de la arquitectura del programa
- Diseño de la interfaz de usuario
- Planificación de las funcionalidades requeridas

Implementación:

- Desarrollo incremental de funcionalidades
- Implementación de cada algoritmo con validación individual

Pruebas y Validación:

- Validación con diferentes conjuntos de datos
- Verificación de correctitud de resultados

Herramientas Utilizadas:

- **IDE:** Visual Studio Code
- **Lenguaje:** Python 3
- **Control de Versiones:** Git
- **Librerias:** Faker

5. Resultados Obtenidos

Funcionalidad Exitosa:

- Todos los algoritmos implementados funcionan correctamente
- El programa maneja adecuadamente diferentes tamaños de datos
- La interfaz de usuario es intuitiva y funcional

Análisis de Rendimiento:

Los resultados confirman la teoría sobre complejidad algorítmica:

- Bubble Sort muestra los mayores tiempos de ejecución
- Selection Sort muestra tiempos mejores que Bubble Sort
- Insertion Sort es eficiente para listas pequeñas o parcialmente ordenadas
- La búsqueda binaria supera ampliamente a la búsqueda lineal en listas grandes

6. Conclusiones

El desarrollo de este trabajo proporcionó una comprensión profunda de los algoritmos fundamentales de búsqueda y ordenamiento. La experiencia práctica de implementación, prueba y optimización de algoritmos es muy importante en nuestra formación como programadores.

7. Bibliografía

- Documentación oficial de Python. Disponible en: https://docs.python.org/
- Material del curso - Programación I, Tecnicatura Universitaria en Programación.
- Sedgewick, R., Wayne, K. (2011). *Algorithms*. Addison-Wesley Professional.

8. Anexos

Link al Video explicativo: https://youtu.be/LKov69HIsik