Plantilla para Trabajos Integradores – Programación I

Datos Generales

- **Título del trabajo**: Algoritmos de Búsqueda y Ordenamiento Aplicados: Un Gestor de Notas de Alumnos en Python.
- **Alumnos**: Cristian Gabriel Aguirre – cga1985@hotmail.com

  Gino Paolo Canevaro - renfra2002@gmail.com - dacowboy92@gmail.com
- **Materia**: Programación I
- **Profesor/a**: Cinthia Rigoni
- **Fecha de Entrega**: (09/06/2025)

Índice

1. Introducción
2. Marco Teórico
3. Caso Práctico
4. Metodología Utilizada
5. Resultados Obtenidos
6. Conclusiones
7. Bibliografía
8. Anexos

---

1. Introducción

En el ámbito de la programación, la eficiencia y la optimización son pilares fundamentales para el desarrollo de software. Dentro de este contexto, los algoritmos de búsqueda y ordenamiento juegan un papel fundamental, ya que son la base para el manejo y procesamiento eficaz de

grandes volúmenes de datos. Este trabajo integrador se enfoca en la investigación y aplicación práctica de estos algoritmos utilizando el lenguaje de programación Python.

Se eligió este tema debido a su importancia en cualquier aplicación que involucre la gestión de información. Comprender cómo funcionan internamente los algoritmos de búsqueda y ordenamiento, y cómo su elección impacta directamente en el rendimiento de un sistema, es una habilidad indispensable para cualquier futuro programador. La capacidad de encontrar datos rápidamente y de presentarlos de manera organizada no solo mejora la experiencia del usuario, sino que también optimiza el uso de recursos.

El presente trabajo se propone alcanzar los siguientes objetivos:

- Investigar y comprender los principios teóricos de los algoritmos de búsqueda lineal y binaria, así como los de ordenamiento de burbuja y por selección.
- Implementar estos algoritmos en Python dentro de un caso práctico concreto: un sistema de gestión de notas de alumnos.
- Realizar un análisis comparativo de la eficiencia de los algoritmos implementados, utilizando diferentes volúmenes de datos para demostrar su impacto en el rendimiento.
- Documentar el proceso de desarrollo y las conclusiones obtenidas a partir de la experiencia práctica.

A través de este proyecto, se busca aplicar los conocimientos adquiridos en la materia de Programación, y mediante los algoritmos aprendidos utilizarlos para la optimización de los softwares.

---

2. Marco Teórico

Este apartado proporciona la base conceptual sobre los algoritmos de búsqueda y ordenamiento y el análisis de su eficiencia.

Algoritmos de Búsqueda

Procedimientos para localizar elementos dentro de una estructura de datos.

Búsqueda Lineal

- Concepto: Recorre la lista elemento por elemento desde el inicio hasta encontrar la coincidencia o llegar al final.
- Característica: No requiere que la lista esté ordenada, es muy eficiente para buscar en listas no tan grandes y es de fácil codificación.

Búsqueda Binaria

- Concepto: Divide repetidamente por la mitad la porción de la lista donde podría estar el elemento.
- Característica: Requiere que la lista esté previamente ordenada, es más eficiente para grandes listas, es más complicada su codificación.

Algoritmos de Ordenamiento

Procedimientos para ordenar elementos de una lista

Ordenamiento de Burbuja

- Concepto: Compara y, si es necesario, intercambia elementos a, "burbujeando" los valores hacia su posición correcta.
- Eficiencia: $O(n^2)$ - cuadrática, muy ineficiente en listas grandes.

Ordenamiento por Selección

- Concepto: Encuentra repetidamente el elemento mínimo (o máximo) de la parte no ordenada y lo coloca al principio de la parte ordenada.
- Eficiencia: $O(n^2)$ - cuadrática, ineficiente en listas grandes.

Análisis de Algoritmos: Eficiencia y Notación Big O

El análisis de algoritmos mide los recursos que necesita un algoritmo. La notación Big O (O) describe cómo el tiempo de ejecución de un algoritmo se escala con el tamaño de la entrada (n).

- O(1) - Constante: Tiempo fijo, independientemente de 'n'.
- O(log n) - Logarítmica: Tiempo crece muy lentamente con 'n' (ej. Búsqueda Binaria).
- O(n) - Lineal: Tiempo crece proporcionalmente a 'n' (ej. Búsqueda Lineal).

- O(n^2) - Cuadrática: Tiempo crece exponencialmente con 'n', volviéndose muy lento para 'n' grandes (ej. Ordenamiento Burbuja, Selección).

La optimización busca reducir el tiempo para mejorar el rendimiento del software, especialmente con grandes volúmenes de datos.

---

3. Caso Práctico

Para aplicar en la práctica este tema, bastó con hacer un script de una lista global en la cual se almacenan alumnos, es decir sus nombres y sus respectivas ID's de cada alumno de forma automática, ya que todo esto funciona como una base de datos, y los algoritmos de búsqueda y ordenamiento se asesoran de mantener ordenados en la lista a cada alumno con su respectivo ID. Pero hay que aclarar que se usó una función de python llamada "Faker" que hace que los nombres sean ficticios y las notas aleatorias, en este contexto eso es algo ideal para hacer ese tipo de pruebas de códigos, aunque con unos pocos cambios el script se podría aplicar en una situación real; en una escuela real con alumnos reales.

---

4. Metodología Utilizada en proceso

Se comenzó con una investigación teórica sobre algoritmos de búsqueda y ordenamiento.

Se diseñó el programa en Python utilizando la librería Faker para generar datos.

Se implementaron y probaron los algoritmos por separado, utilizando Visual Studio Code como entorno de desarrollo.

El trabajo se realizó en grupo, dividiendo tareas entre el diseño, la codificación y las pruebas finales.

---

5. Resultados Obtenidos en proceso

Se obtuvieron los resultados esperados aunque no fue tan fácil, ya que se presentaron distintos errores en el script que no permitían que se ejecute correctamente, hubo que reescribir algunas líneas de código, probando varias veces ejecutarlas. O por ejemplo en un momento el script se ejecutaba hasta cierto punto y se detenía o bugueaba, lo cual costó un poco arreglarlo pero fue posible solucionarlo, para previamente hacer pequeños cambios y optimizar el programa en general. Se pudo observar que mediante los diferentes métodos para ordenar los datos, hay unos que son más eficientes o rápidos que otros.

---

6. Conclusiones

Se concluyó que los algoritmos de búsqueda y ordenamiento son pilares fundamentales en la computación moderna. Su correcta selección (según el tamaño de los datos, el nivel de orden previo y los recursos disponibles) determina la eficiencia de un sistema. Mientras que la búsqueda binaria y el ordenamiento rápido destacan por su velocidad, otros como la búsqueda lineal o el burbujeo son útiles en contextos específicos. Dominar estos algoritmos permite desarrollar soluciones optimizadas, escalables y adaptables a las demandas tecnológicas actuales. Estos algoritmos permiten organizar datos de forma estructurada y recuperarlos rápidamente, optimizando el rendimiento de aplicaciones y sistemas.

---

7. Bibliografía

https://medium.com/@mise/algoritmos-de-b%C3%BAsqueda-y-ordenamiento-7116bcea03d0

https://www.youtube.com/watch?v=xiUKhMeLTR4

https://runestone.academy/ns/books/published/pythoned/SortSearch/toctree.html

https://www.datacamp.com/es/tutorial/binary-search-python

---

8. Anexos

PDF Informe técnico (este mismo archivo)

Powerpoint con capturas de pantalla y comentarios

Video explicativo en YouTube

Script Python

-Todo se encuentra en el repositorio de GitHub-
