UTN

TECNICATURA UNIVERSITARIA
EN PROGRAMACION
A DISTANCIA

Trabajo Integrador - Propuesta de Investigación

Cátedra de Programación I

Investigación aplicada en Python

Algoritmos de Búsqueda y Ordenamiento

Gustavo H. Tiseira • Daiana A. Trejo

Profesora: Martina Wallace

Tutor: Walter Pintos

Fecha de Entrega: 09/06/25

Índice

1. Introducción

Los supermercados manejan miles de productos diariamente, necesitando sistemas eficientes para organizar y encontrar información rápidamente. Los algoritmos de búsqueda y ordenamiento son fundamentales para estas operaciones.

¿Por qué elegimos este tema?

Elegimos desarrollar un sistema de inventario para supermercados porque es un ejemplo concreto y familiar donde los algoritmos de búsqueda y ordenamiento se aplican constantemente en situaciones reales.

Importancia en la programación

En un supermercado, es crucial poder ordenar productos por diferentes criterios (precio, stock, fecha de vencimiento) y buscar productos específicos de manera rápida para atender consultas de clientes y gestionar el inventario eficientemente.

Objetivos del trabajo

- Implementar algoritmos básicos de ordenamiento (Bubble Sort, Selection Sort)
- Desarrollar algoritmos de búsqueda (lineal y binaria)
- Crear un sistema práctico de gestión de inventario usando programación estructurada
- Comparar la eficiencia de diferentes algoritmos
- Aplicar conceptos de funciones y modularización en Python

Uso de Inteligencia Artificial

Para el desarrollo de este proyecto, se implementó inicialmente toda la lógica de los algoritmos y funcionalidades por cuenta propia utilizando programación estructurada. Una vez completada la implementación funcional, se utilizó inteligencia artificial generativa para mejorar aspectos estéticos y de presentación del código, incluyendo el embellecimiento de la interfaz de usuario, la mejora de comentarios y documentación, y la optimización visual de los mensajes mostrados al usuario. La lógica fundamental de los algoritmos de búsqueda y ordenamiento se mantuvo intacta.

2. Marco Teórico

2.1 Algoritmos de Ordenamiento

Bubble Sort (Ordenamiento Burbuja)

Es el algoritmo más simple de entender. Compara elementos adyacentes y los intercambia si están en orden incorrecto.

Funcionamiento:

1. Compara cada par de elementos adyacentes
2. Si están desordenados, los intercambia
3. Repite el proceso hasta que no haya más intercambios

Complejidad: O(n²) - Cuadrática

Selection Sort (Ordenamiento por Selección)

Busca el elemento más pequeño y lo coloca en la primera posición, luego el segundo más pequeño en la segunda posición, y así sucesivamente.

Funcionamiento:

1. Encuentra el elemento mínimo
2. Lo intercambia con el primer elemento
3. Repite para el resto de la lista

Complejidad: O(n²) - Cuadrática

2.2 Algoritmos de Búsqueda

Búsqueda Lineal

Examina cada elemento de la lista uno por uno hasta encontrar el elemento buscado.

Funcionamiento:

1. Comienza desde el primer elemento
2. Compara cada elemento con el valor buscado
3. Si lo encuentra, retorna la posición
4. Si llega al final sin encontrarlo, retorna -1

Complejidad: O(n) - Lineal

Búsqueda Binaria

Solo funciona en listas ordenadas. Divide la lista por la mitad repetidamente hasta encontrar el elemento.

Funcionamiento:

1. Compara el elemento central con el valor buscado
2. Si son iguales, lo encontró
3. Si el valor es menor, busca en la mitad izquierda
4. Si es mayor, busca en la mitad derecha
5. Repite hasta encontrar o agotar opciones

Complejidad: O(log n) - Logarítmica

3. Caso Práctico

3.1 Descripción del Problema

Desarrollamos un Sistema de Inventario para Supermercado que permite:

- Registrar productos con código, nombre, precio y stock
- Ordenar productos por diferentes criterios
- Buscar productos por código o nombre
- Mostrar productos con stock bajo
- Comparar velocidad de diferentes algoritmos

3.2 Implementación

El sistema está implementado usando programación estructurada en Python, organizado en módulos funcionales:

Archivos del sistema:

- **productos.py:** Contiene funciones para crear productos, generar datos de ejemplo y mostrar información de productos
- **ordenamiento.py:** Implementa las funciones de ordenamiento Bubble Sort y Selection Sort
- **busqueda.py:** Contiene las funciones de búsqueda lineal y binaria
- **main.py:** Programa principal con interfaz de usuario mediante menú interactivo

Estructura de datos utilizada:

- Los productos se representan como diccionarios con las claves: 'codigo', 'nombre', 'precio', 'stock'
- La lista de productos es una lista de diccionarios
- No se utilizan clases, manteniendo un enfoque de programación estructurada

Diseño modular: Cada archivo tiene una responsabilidad específica, permitiendo mantener el código organizado y fácil de mantener. Las funciones están bien documentadas y son reutilizables.

3.3 Decisiones de Diseño

¿Por qué estas decisiones?

- **Programación estructurada:** Utilizamos funciones en lugar de clases para mantener la simplicidad y facilitar la comprensión
- **Diccionarios para productos:** Estructura simple que permite acceso directo a los atributos del producto
- **Algoritmos básicos:** Usamos Bubble Sort y Selection Sort porque son fáciles de entender, implementar y explicar
- **Menú interactivo:** Permite probar todas las funcionalidades fácilmente
- **Medición de tiempo:** Cada algoritmo mide su tiempo de ejecución para comparar eficiencia
- **Datos de ejemplo:** Incluimos productos típicos de supermercado para hacer pruebas realistas
- **Separación en módulos:** Facilita el mantenimiento y la comprensión del código

4. Metodología Utilizada

4.1 Etapas del Desarrollo

1. **Investigación teórica:** Estudiamos los algoritmos de búsqueda y ordenamiento básicos
2. **Diseño del sistema:** Definimos las funciones y estructura de datos necesarias
3. **Implementación:** Programamos cada algoritmo paso a paso usando programación estructurada

4. **Mejoras estéticas con IA:** Utilizamos inteligencia artificial para embellecer la presentación
5. **Pruebas:** Testeamos con diferentes datos y casos
6. **Documentación:** Elaboramos este informe

4.2 Herramientas Utilizadas

- **Python 3:** Lenguaje de programación principal
- **Módulo time:** Para medir la eficiencia de los algoritmos
- **Editor de código:** Visual Studio Code
- **Git:** Para control de versiones del proyecto
- **Inteligencia Artificial Generativa:** Para mejoras estéticas y embellecimiento de la interfaz

4.3 División del Trabajo

- **Tiseira Gustavo:** Implementación de algoritmos de ordenamiento y productos.
- **Trejo Daiana:** Implementación de algoritmos de búsqueda, archivo principal y readme.

4.4 Proceso de Desarrollo con IA

El desarrollo siguió un enfoque híbrido donde:

Desarrollo autónomo: Toda la lógica algorítmica, estructura de funciones y funcionalidades principales fueron desarrolladas completamente por nosotros usando programación estructurada.

Asistencia de IA para mejoras estéticas: Una vez completado el sistema funcional, se empleó IA generativa específicamente para:

- Mejorar la presentación visual de los menús y mensajes
- Añadir elementos visuales para hacer la interfaz más atractiva
- Optimizar la documentación y comentarios del código
- Estructurar mejor la organización del código en múltiples archivos

Esta metodología nos permitió mantener el aprendizaje completo de los algoritmos y la programación estructurada mientras aprovechamos las herramientas modernas para mejorar la experiencia del usuario.

5. Resultados Obtenidos

5.1 Funcionamiento de los Algoritmos

Algoritmos de Ordenamiento:

- **Bubble Sort:** Funcionó correctamente ordenando productos por precio, mostrando cada comparación e intercambio
- **Selection Sort:** Ordenó eficientemente los productos por stock, minimizando el número de intercambios
- Ambos algoritmos mostraron tiempos de ejecución similares con pocos productos

Algoritmos de Búsqueda:

- **Búsqueda Lineal:** Encontró productos correctamente, mostrando cada paso de la búsqueda
- **Búsqueda Binaria:** Fue más rápida, pero requiere que los datos estén ordenados primero

5.2 Comparación de Eficiencia

Con 15 productos de prueba:

- **Bubble Sort:** ~0.000015 segundos
- **Selection Sort:** ~0.000012 segundos
- **Búsqueda Lineal:** ~0.000008 segundos
- **Búsqueda Binaria:** ~0.000003 segundos

5.3 Casos de Prueba Realizados

- Ordenamiento con diferentes cantidades de productos
- Búsqueda de productos existentes y no existentes
- Búsqueda por nombre parcial
- Identificación de productos con stock bajo
- Comparación de tiempos de ejecución entre algoritmos

6. Conclusiones

6.1 Aprendizajes Principales

Este trabajo nos permitió comprender cómo los algoritmos teóricos se aplican en problemas reales usando programación estructurada. Aprendimos que:

- Bubble Sort y Selection Sort son fáciles de implementar con funciones, pero no muy eficientes
- La búsqueda binaria es mucho más rápida que la lineal, pero necesita datos ordenados
- Elegir el algoritmo correcto depende del contexto y los datos disponibles
- La programación estructurada con funciones es efectiva para proyectos de mediana complejidad
- La modularización del código facilita el mantenimiento y la comprensión

Adicionalmente, este proyecto nos enseñó sobre el uso responsable de herramientas de IA como complemento al desarrollo de software, manteniendo siempre el control sobre la lógica fundamental y utilizando la IA únicamente para mejoras de presentación y estética.

6.2 Aplicación Práctica

Los algoritmos implementados son útiles para:

- Sistemas de punto de venta en supermercados
- Gestión de inventarios en comercios
- Búsqueda rápida de productos por código de barras
- Organización de productos por diferentes criterios

6.3 Dificultades Encontradas

- **Medición precisa del tiempo:** Los algoritmos son tan rápidos que necesitamos usar microsegundos
- **Manejo de datos:** Asegurar que las comparaciones en diccionarios funcionen correctamente
- **Interfaz de usuario:** Crear un menú claro y fácil de usar con funciones
- **Organización del código:** Dividir correctamente las responsabilidades entre archivos

6.4 Posibles Mejoras

- Agregar más algoritmos de ordenamiento (como Quick Sort)
- Implementar guardado de datos en archivos

- Añadir más campos a los productos (categoría, proveedor)
- Crear una interfaz gráfica más amigable
- Implementar manejo de excepciones más robusto

7. Bibliografía

- Apuntes de Cátedra - Programación I. (2025). Universidad Tecnológica Nacional.
- Documentación oficial de Python. (2024). Tutorial de Python. Disponible en: https://docs.python.org/es/3/tutorial/

8. Anexos

Anexo A: Código Fuente Completo

Repositorio del proyecto: https://github.com/brujoh88/TPIntegradorProgramacion1

Archivos principales:

- main.py - Programa principal e interfaz de usuario
- productos.py - Gestión de productos y datos de ejemplo
- ordenamiento.py - Algoritmos Bubble Sort y Selection Sort
- busqueda.py - Algoritmos de búsqueda lineal y binaria

Anexo B: Video Explicativo

Enlace al vídeo: https://youtu.be/f-FYkDhBkvM