TP: Gestión de Catálogo de Biblioteca en Python

Alumnos:

- Facundo Gastón Vázquez – facugaston98@gmail.com
- Víctor Tolaba - viktor.tsp14@gmail.com

Materia: Programación I

Profesor: Nicolás Quirós

Fecha de Entrega: 9 de junio de 2025

Índice

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

---

1. Introducción

Este informe presenta el desarrollo y análisis de algoritmos de ordenamiento y búsqueda aplicados a la simulación de un catálogo de biblioteca. Se generan listados de ISBN, se ordenan con diferentes métodos y se mide la eficiencia de cada operación.

2. Marco Teórico

Ordenamiento

- Bubble Sort: compara elementos adyacentes y los intercambia si están fuera de orden. O(n²).
- Quick Sort: elige un pivote, divide la lista y ordena recursivamente. O(n log n) promedio.

Búsqueda

- Lineal: recorre secuencialmente hasta encontrar el elemento. O(n).
- Binaria: requiere lista ordenada; divide el espacio de búsqueda en mitades. O(log n).

Optimización en Python

- Python emplea Timsort para *list.sort()*, O(n log n).
- Los diccionarios permiten búsquedas en tiempo promedio O(1).

3. Caso Práctico

Para utilizar el programa en tu sistema y aprovechar el menú interactivo, sigue estos pasos:

1. Abrir la terminal o consola de comandos.
2. Ubicar la carpeta del proyecto donde están los archivos *main.py*, *ordenamiento.py* y *busqueda.py*.
3. Ejecutar el programa con el siguiente comando:

   ```
   python main.py
   ```

4. Interactuar con el menú que aparecerá en pantalla:
   - Opción 1: Genera un catálogo de 1 000 ISBN y muestra los tiempos de todas las operaciones.
   - Opción 2: Genera un catálogo nuevo y muestra el tiempo de Bubble Sort.

- Opción 3: Genera un catálogo nuevo y muestra el tiempo de Quick Sort.
- Opción 4: Permite ingresar un ISBN y elegir búsqueda lineal o binaria; muestra la posición o indica si no se encuentra.
- Opción 5: Sale del programa.

5. Visualizar los resultados directamente en la consola:

   ISBN de prueba: 9781234567890
   Bubble Sort: 0.250123 s
   Quick Sort: 0.015456 s
   Lineal: 0.000523 s
   Binaria: 0.000021 s

4. Metodología Utilizada Metodología Utilizada

1. Separación en módulos: *ordenamiento.py*, *busqueda.py* y *main.py*.
2. Generación de datos: 1 000 y 10 000 ISBN aleatorios.
3. Medición de tiempos: uso de *timeit.Timer*, 5 repeticiones.
4. Menú interactivo en consola para seleccionar operación.
5. Registro de resultados en tablas comparativas.

5. Resultados Obtenidos

Operación	Algoritmo	n=1 000 (s)	n=10 000 (s)
Bubble Sort	$O(n^2)$	0.2	20.5
Quick Sort	O(n log n)	0.01	0.15
Búsqueda Lin.	O(n)	0.0005	0.005
Búsqueda Bin.	O(log n)	0.00002	0.0001

6. Conclusiones

- Bubble Sort resulta aceptable en catálogos pequeños (< 1 000 ítems).
- Quick Sort es más eficiente y escalable en volúmenes grandes.
- La búsqueda binaria reduce significativamente el tiempo en listas ordenadas.

7. Bibliografía

- García, A., & Morales, L. *Programación en Python: Fundamentos y aplicaciones*. Editorial Alfa, 2020. (Consulta utilizada para la estructura modular del programa y manejo de funciones en Python.)
- Rodríguez, M. *Algoritmos y estructuras de datos en Python*. Editorial Marcombo, 2019. (Soporte teórico para la implementación y análisis de Bubble Sort y Quick Sort.)
- Fernández, P. *Introducción a las bibliotecas y sistemas de catalogación*. Editorial Síntesis, 2018. (Referencia para los conceptos de catálogo de biblioteca e identificación con ISBN.)
- Python Software Foundation. *Documentación oficial de Python 3 en español*. https://docs.python.org/es/3/library/timeit.html (Guía oficial en español del módulo *timeit* para medición de rendimiento.)

8. Anexos

- Video explicativo dentro del README
- Tablas comparativas de tiempos para n=1 000 y n=10 000.
- Repositorio: https://github.com/facuvazquez1/IntregadorProgramacion1