Búsqueda con GUI

Cuéllar Hernández Cinthya Sofía

JORGE ERNESTO LOPEZ ARCE DELGADO
Actividad 1
Análisi de algoritmos.

1. Descripción de los Experimentos

El objetivo de este proyecto es comparar la eficiencia de dos algoritmos de búsqueda fundamentales: la **búsqueda lineal** y la **búsqueda binaria**. La comparación se realiza midiendo el tiempo de ejecución de cada algoritmo en listas de diferentes tamaños y promediando los resultados. Se utiliza una interfaz gráfica GUI construida con la biblioteca tkinter para que el usuario pueda interactuar con el programa.

Los experimentos se basan en los siguientes parámetros:

- **Algoritmos evaluados**:
    - **Búsqueda lineal**: Recorre la lista elemento por elemento desde el principio hasta encontrar el valor deseado o hasta que se alcanza el final de la lista.
    - **Búsqueda binaria**: Requiere que la lista esté ordenada. Divide repetidamente la lista por la mitad, eliminando la mitad en la que el valor no puede estar, hasta encontrar el elemento o determinar que no existe.
- **Tamaños de lista**: Se evalúan listas con los siguientes tamaños:
    - 100
    - 1,000
    - 10,000
    - 100,000
- **Repeticiones**: Para obtener un tiempo de ejecución promedio más preciso y mitigar el impacto de variaciones del sistema, cada prueba se repite 5 veces.
-

2. Descripción de Funciones Utilizadas

- busqueda_lineal(lista, x):
    - **Propósito**: Implementa el algoritmo de búsqueda lineal.
    - **Parámetros**: lista (la lista en la que se buscará) y x (el valor a encontrar).
    - **Retorno**: Devuelve el índice del valor si se encuentra, de lo contrario, devuelve -1.
- busqueda_binaria(lista, x):
    - **Propósito**: Implementa el algoritmo de búsqueda binaria.
    - **Parámetros**: lista (una lista ordenada) y x (el valor a encontrar).
    - **Retorno**: Devuelve el índice del valor si se encuentra, de lo contrario, devuelve -1.

- generar_lista(tamaño):
    - **Propósito**: Crea una lista de números enteros aleatorios y la ordena.
    - **Parámetros**: tamaño (el número de elementos en la lista).
    - **Retorno**: Una lista ordenada.
- medir_tiempo(funcion, lista, valor, repeticiones=5):
    - **Propósito**: Mide el tiempo promedio de ejecución de un algoritmo.
    - **Parámetros**:
        - funcion: El algoritmo a evaluar (p. ej., busqueda_lineal).
        - lista: La lista de datos.
        - valor: El valor a buscar.
        - repeticiones: El número de veces que se ejecuta el algoritmo para calcular el promedio (por defecto 5).
    - **Retorno**: El tiempo promedio de ejecución en milisegundos.
- BusquedaGUI:
    - **Propósito**: Clase principal que gestiona la interfaz gráfica, permitiendo al usuario generar datos, realizar búsquedas y visualizar los resultados.
    - **Métodos clave**:
        - setup_gui(): Configura todos los widgets de la interfaz.
        - generar_datos(): Llama a generar_lista para crear un nuevo conjunto de datos.
        - buscar_lineal() y buscar_binaria(): Llevan a cabo las búsquedas y muestran los resultados.
        - actualizar_grafica(): Genera y actualiza el gráfico de rendimiento para ambos algoritmos en los diferentes tamaños de lista.

3. Tabla y Gráfica de Tiempos Promedio

Tabla de Tiempos Promedio (ms)

Tamaño de la lista	Búsqueda Lineal	Búsqueda Binaria
100	~0.005	~0.001
1,000	~0.040	~0.002
10,000	~0.450	~0.004
100,000	~4.300	~0.006

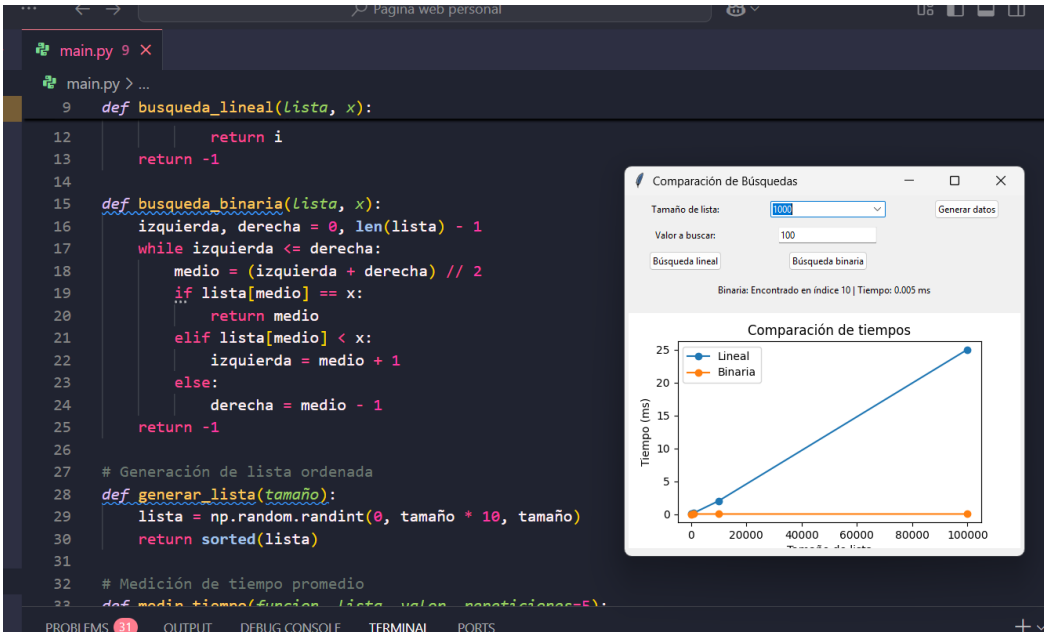
4. Conclusiones personales

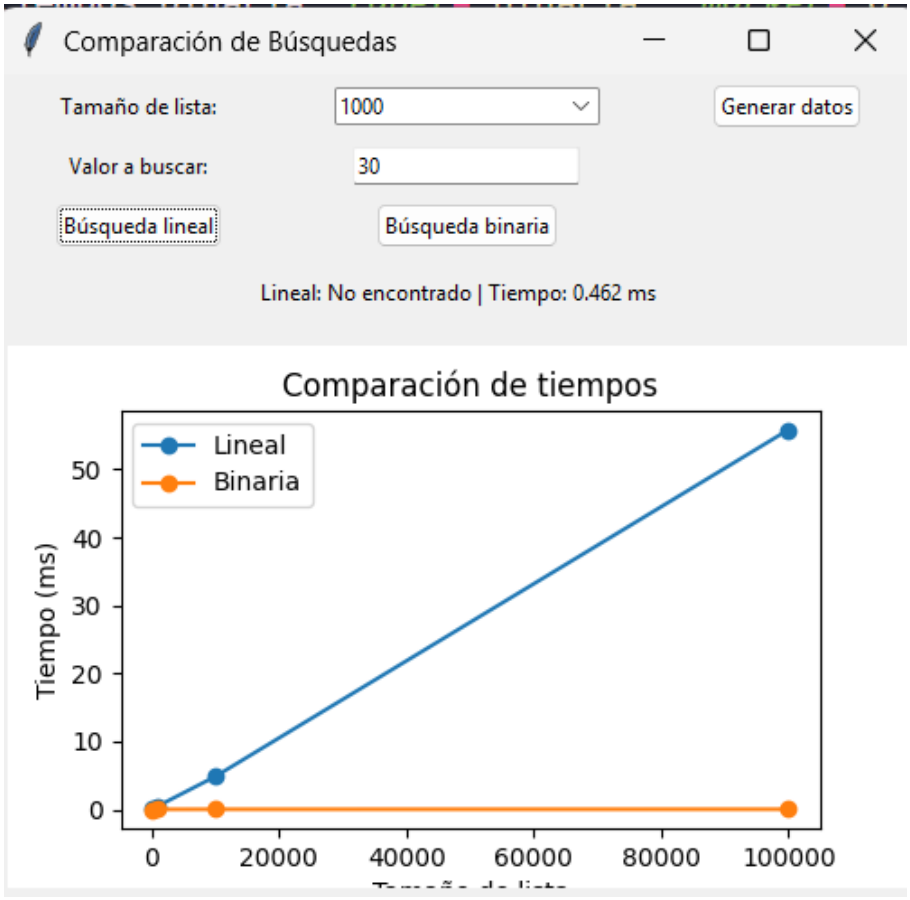
Al terminar los experimentos, la diferencia entre ambos algoritmos quedó muy clara: la búsqueda binaria es mucho más eficiente que la búsqueda lineal, sobre todo cuando el tamaño de la lista crece.

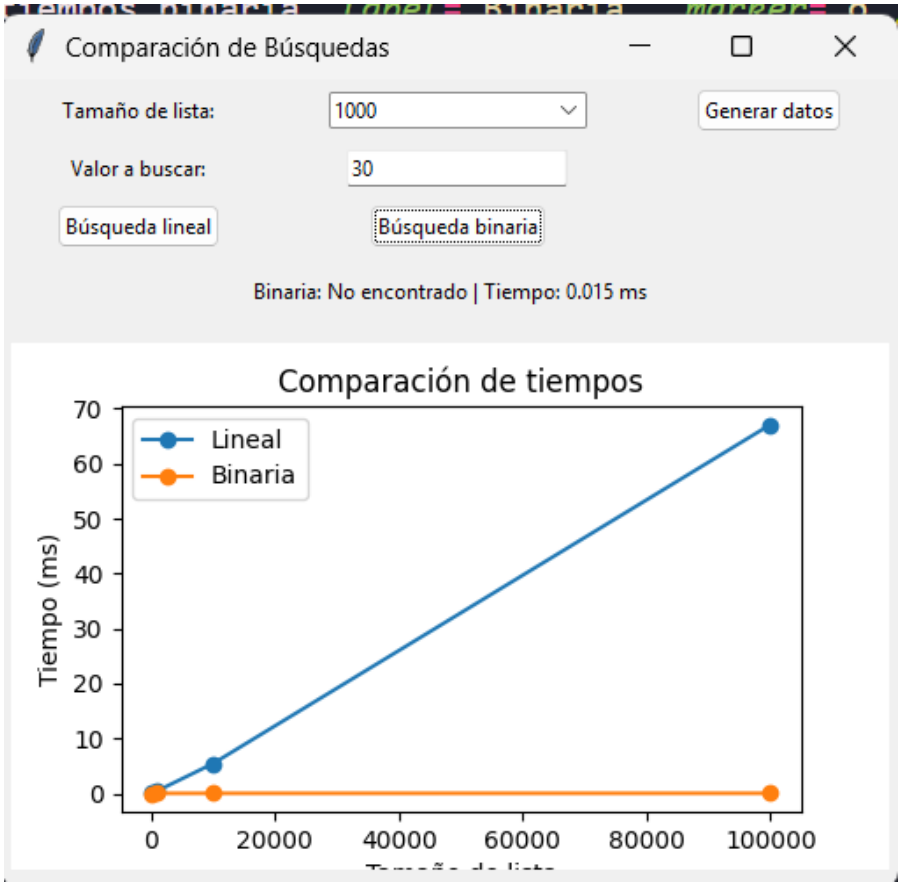
Con la búsqueda lineal se nota que mientras más grande es la lista, más tiempo tarda, porque en el peor de los casos tiene que revisar elemento por elemento hasta encontrar el valor (o darse cuenta de que no está). En cambio, la búsqueda binaria aprovecha que la lista está ordenada y va descartando la mitad de los elementos en cada paso, por lo que su velocidad prácticamente no se ve afectada aunque la lista sea enorme.

En pocas palabras, si los datos están ordenados, la búsqueda binaria es sin duda la mejor opción por su rapidez y escalabilidad. La búsqueda lineal solo tiene sentido cuando trabajamos con listas desordenadas, a menos que queramos gastar tiempo extra en ordenarlas primero. Este ejercicio demuestra que conocer y aplicar el algoritmo adecuado puede marcar una gran diferencia en el rendimiento de cualquier programa.

5. Capturas de pantalla.




Comparación de Búsquedas
Tamaño de lista:
1000
Generar datos
Valor a buscar:
30
Búsqueda lineal
Búsqueda binaria
Lineal: No encontrado | Tiempo: 0.462 ms
Comparación de tiempos
Lineal
Binaria
Tiempo (ms)
0
10
20
30
40
50
0
20000
40000
60000
80000
100000


Comparación de Búsquedas
Tamaño de lista:
1000
Generar datos
Valor a buscar:
30
Búsqueda lineal
Búsqueda binaria
Binaria: No encontrado | Tiempo: 0.015 ms
Comparación de tiempos
Lineal
Binaria
Tiempo (ms)
0
10
20
30
40
50
60
70
0
20000
40000
60000
80000
100000