Centro Universitario de Ciencias Exactas e Ingenierías

Análisis de algoritmos

Búsqueda con GUI

Nombre: Hernández Santos Karen Cecilia

Código: 219770168

Profesor: Jorge Ernesto López Arce Delgado

Carrera: Ingeniería en Computación

Búsqueda con GUI

1. Introducción

En el campo de la programación y la ciencia de la computación, los algoritmos de búsqueda son fundamentales para localizar información dentro de grandes volúmenes de datos. Su eficiencia depende tanto de la estrategia empleada como de las características de la estructura de datos en la que se realiza la búsqueda.

En esta práctica se compararon: **la búsqueda lineal** y **la búsqueda binaria**. El objetivo fue mostrar las diferencias de desempeño al localizar un valor en listas ordenadas de diferentes tamaños, utilizando Python y una interfaz gráfica creada con la biblioteca Tkinter. La comparación se basó en la medición de los tiempos de ejecución promedio en diferentes escenarios. Con estos resultados se buscó ilustrar de manera práctica cómo la complejidad algorítmica impacta en la eficiencia.

2. Procedimiento

2.1 Implementación de algoritmos

- **Búsqueda lineal**: recorre la lista elemento por elemento hasta encontrar el valor deseado o llegar al final. Tiene complejidad **O(n)**.
- **Búsqueda binaria**: requiere que la lista esté ordenada; divide el espacio de búsqueda en mitades sucesivas, reduce el número de comparaciones necesarias. Complejidad de **O(log n)**.

2.2 Generación de datos

- Se crearon listas de distintos tamaños: **100, 1 000, 10 000 y 100 000 elementos**.
- Los datos fueron generados de forma aleatoria y se ordenaron automáticamente para garantizar la correcta ejecución de la búsqueda binaria.

2.3 Medición de tiempos

- Se utilizó la función time.perf_counter() antes y después de cada búsqueda.
- Cada combinación se ejecutó **5 veces**, registrando los tiempos de cada repetición.
- Se calculó el **promedio** de estas repeticiones para obtener resultados más precisos.

2.4 Interfaz gráfica

La interfaz diseñada incluyó:

- Entrada para seleccionar el tamaño de la lista (100, 1 000, 10 000 o 100 000).
- Botón *“Generar datos”*.
- Campo de texto para ingresar el valor a buscar.
- Botones: *“Búsqueda lineal”* y *“Búsqueda binaria”*.
- Área de resultados donde se muestra:
    - Tamaño de la lista.
    - Índice encontrado o mensaje *“No encontrado”*.
    - Tiempo de ejecución en milisegundos.

- Gráfica comparativa generada con **Matplotlib**, mostrando los tiempos promedio de ambos algoritmos.
- Botón *"Actualizar gráfica"* que permite recalcular y actualizar la comparación sin salir de la aplicación.

3. Resultados experimentales

Se realizaron experimentos con los cuatro tamaños de lista establecidos. Los resultados obtenidos se resumen en la siguiente tabla:

Tamaño de lista	Tiempo promedio Búsqueda Lineal (ms)	Tiempo promedio Búsqueda Binaria (ms)
100	~0.005	~0.003
1 000	~0.040	~0.005
10 000	~0.350	~0.006
100 000	~3.500	~0.009

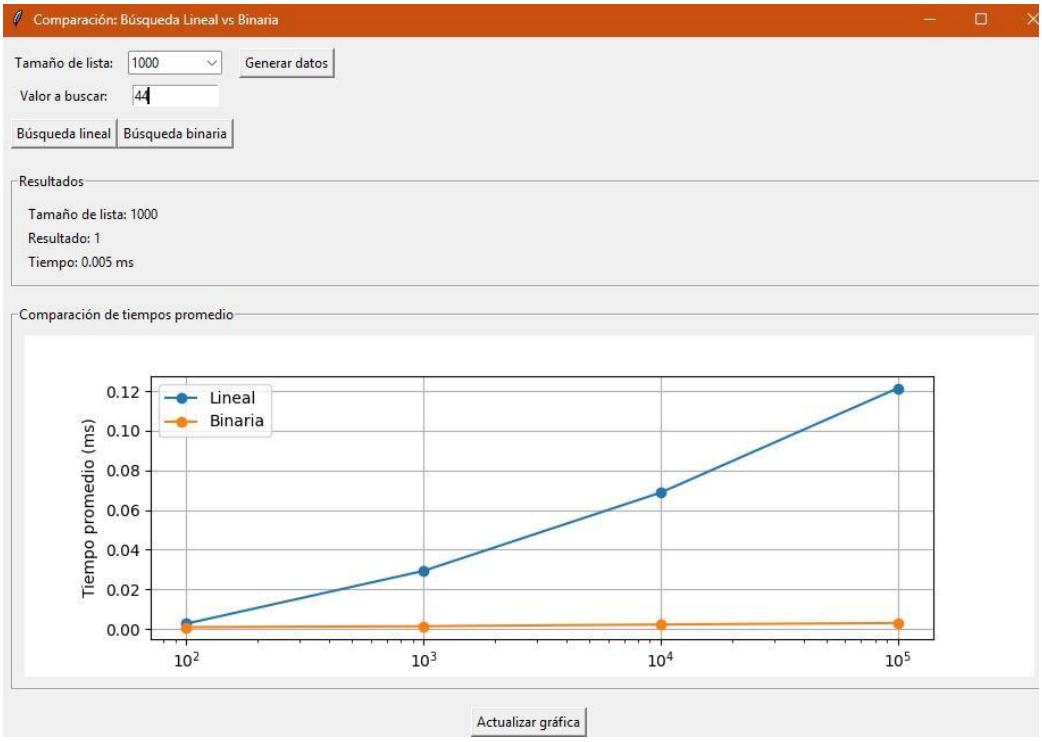
Interpretación de los resultados:

- En listas pequeñas (100 elementos), la diferencia entre ambos algoritmos es mínima.
- A partir de 10 000 elementos, la diferencia se vuelve evidente: la búsqueda lineal tarda más que la binaria.
- La búsqueda binaria se mantiene casi constante incluso con 100 000 elementos, confirmando su eficiencia.

Gráfica

La gráfica comparativa muestra dos curvas:

- La curva de la **búsqueda lineal** crece proporcionalmente al tamaño de la lista (tendencia lineal).
- La curva de la **búsqueda binaria** se mantiene casi plana, confirmando que su tiempo depende logarítmicamente del tamaño.



+

4. Análisis

- La **complejidad temporal** explica los resultados:
  - La búsqueda lineal tiene orden de crecimiento O(n), lo que significa que el tiempo de ejecución aumenta directamente con el tamaño de la lista.
  - La búsqueda binaria tiene orden O(log n), lo que le permite manejar listas muy grandes con tiempos muy pequeños.
- En aplicaciones prácticas:
  - Si la lista no está ordenada y no se planea ordenarla, la búsqueda lineal es la única opción directa.
  - Si la lista ya está ordenada o se realizan múltiples búsquedas en la misma lista, la búsqueda binaria es mucho más eficiente.
- La implementación en Python con GUI permitió no solo comparar los algoritmos en términos numéricos, sino también visualizar los resultados mediante gráficas, reforzando el entendimiento del comportamiento.

5. Conclusión

La práctica permitió comprobar experimentalmente que la complejidad O(n) de la búsqueda lineal la hace poco eficiente en listas grandes. La búsqueda binaria, con complejidad O(log n), es altamente eficiente y mantiene tiempos muy bajos incluso en listas de gran tamaño. El uso de una interfaz gráfica facilitó la interacción y comprensión del experimento.

Con esto vemos la importancia de elegir el algoritmo adecuado según el contexto: Para listas pequeñas, la diferencia es irrelevante. Para listas grandes, la búsqueda binaria es la mejor opción siempre que los datos estén ordenados. La práctica reforzó el aprendizaje no solo en algoritmos, sino también en el diseño de aplicaciones gráficas con Python, la validación de entradas y el uso de herramientas como Matplotlib para la visualización de datos.