Trabajo Integrador

Programación I

Datos Generales

Alumnos:

- Kadar Assad, Germán
- Kadar Assad, Martín

Materia: Programación I

Trabajo Integrador: Análisis Comparativo de Algoritmos de Búsqueda

1. Objetivos del análisis de algoritmos

El propósito de este trabajo práctico fue crear un algoritmo de búsqueda lineal y binaria para poder realizar un análisis de los mismos y compararlos para verificar cual es el más óptimo. La comparación permite fundamentar, desde una perspectiva práctica y teórica, la elección del algoritmo más eficiente para resolver un problema en un contexto determinado.

2. Algoritmos de búsqueda

2.1 Búsqueda lineal

La búsqueda lineal es una técnica directa y sencilla que consiste en examinar cada elemento de una lista, uno por uno, desde el primer hasta el último, comparando cada uno con el valor buscado. El proceso se detiene cuando se encuentra una coincidencia o se ha recorrido toda la estructura de datos sin éxito.

Esta técnica es especialmente útil cuando se trabaja con listas desordenadas o de pequeño tamaño, ya que no requiere ningún preprocesamiento de los datos.

En el trabajo se desarrolló una función que implementa esta lógica de forma secuencial. La función recorre la lista completa y devuelve la posición del valor buscado, o una señal de fallo si el elemento no se encuentra. Es importante destacar que esta función funciona sin importar el orden de los datos.

2.2 Búsqueda binaria

La búsqueda binaria consiste en comparar el valor buscado con el elemento que se encuentra en el centro de la lista, pero requiere que los datos estén ordenados. Si el valor coincide, el proceso finaliza. Si el valor buscado es menor, se repite el procedimiento sobre la mitad izquierda; si es mayor, sobre la mitad derecha.

En el desarrollo práctico, se implementaron dos variantes:

- Versión iterativa: Utiliza un bucle que actualiza los extremos del rango de búsqueda (mínimo y máximo) hasta que se encuentra el valor o se determina que no está presente.
- Versión recursiva: Aplica la misma lógica, pero mediante llamadas recursivas. Cada llamada reduce el rango de búsqueda y se acumula en la pila de ejecución.

Ambas variantes fueron aplicadas sobre listas ordenadas, generadas aleatoriamente y ordenadas mediante funciones propias del lenguaje Python.

3. Análisis de casos

Se consideraron dos tipos de situaciones para analizar la eficiencia de los algoritmos:

- Mejor caso: El valor buscado se encuentra en la primera posición (búsqueda lineal) o en el centro (búsqueda binaria).
- Peor caso: El valor está en la última posición (búsqueda lineal) o no está presente (ambos algoritmos). Para la búsqueda binaria el peor caso sería el primer valor y el último.

4. Metodología y análisis empírico

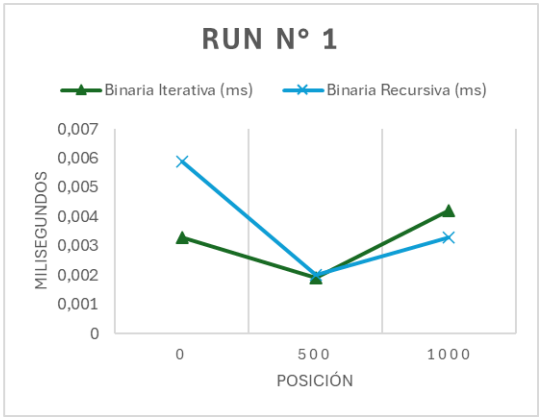
Para validar el análisis teórico, se llevó a cabo una serie de pruebas empíricas:

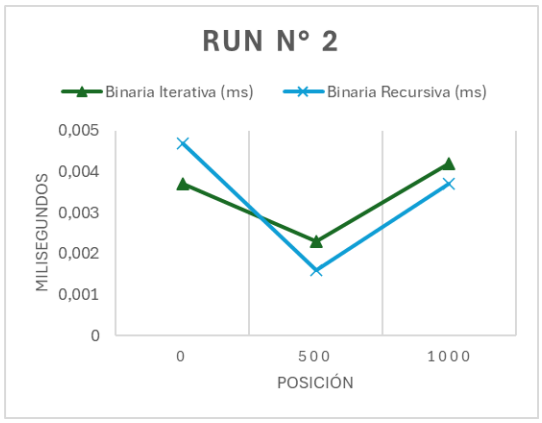
Generación de datos: Se generó una lista de 1.001 números aleatorios, únicos, entre 1 y 1.999. Esta lista fue ordenada para asegurar su compatibilidad con los métodos binarios.

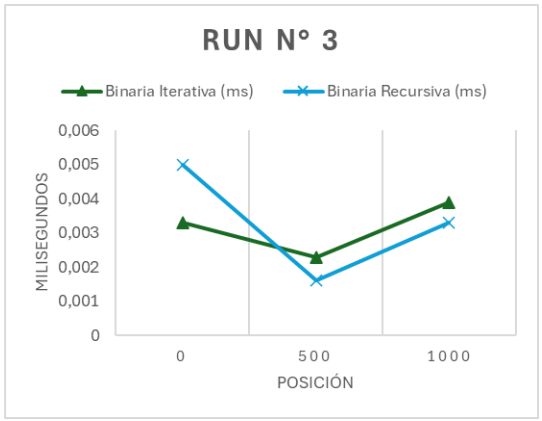
Selección de valores de prueba: Se seleccionaron tres posiciones representativas dentro de la lista (0, 500 y 1.000), que cubren desde el inicio hasta el final de la estructura de datos.

Instrumentación: La medición se realizó con herramientas estándar del lenguaje Python que permiten calcular el tiempo de ejecución con precisión. Se garantizó que todas las búsquedas operaran sobre los mismos datos para asegurar la equidad de la comparación.

Ejecución y medición: Para cada valor seleccionado se aplicaron las tres variantes de búsqueda (lineal, binaria iterativa y binaria recursiva). Cada prueba fue repetida tres veces y se promediaron los tiempos de ejecución para reducir el efecto del entorno de ejecución. Los resultados fueron los siguientes:







Este enfoque permitió evaluar con precisión el comportamiento de cada algoritmo y detectar diferencias relevantes en su rendimiento, especialmente cuando el tamaño de la lista o la posición del valor buscado varía.

5. Conclusión y aprendizajes

El análisis y las pruebas realizadas confirmaron que la búsqueda binaria es significativamente más eficiente que la búsqueda lineal en listas ordenadas, incluso para volúmenes de datos moderados como el usado en este trabajo.

La búsqueda lineal demostró tener un rendimiento aceptable únicamente cuando el elemento se encuentra al comienzo de la lista. Su desempeño cae abruptamente cuando el valor está cerca del final o no está presente, lo que la convierte en una opción poco eficiente para listas extensas.

Por su parte, la búsqueda binaria mantuvo tiempos de ejecución bajos y consistentes en los tres casos evaluados. En las tres ocasiones tomadas para el análisis, la función recursiva fue más eficiente que la iterativa, con una diferencia que oscila entre el 12% y 21% más rápida. Sin embargo, podemos observar que al buscar valores al principio de la lista, la versión iterativa es más eficiente (entre 21% y 44% más rápida) que la recursiva.

Este trabajo permitió aplicar conocimientos teóricos sobre algoritmos, validar predicciones mediante mediciones prácticas, y entender el impacto de decisiones de implementación como el uso de recursividad. En conjunto, se logró una visión más profunda y concreta sobre cómo y cuándo utilizar cada tipo de búsqueda.

Aprendizajes clave:

- Las pruebas empíricas permiten validar la teoría y descubrir detalles prácticos que pueden no ser evidentes en el análisis matemático.
- Herramientas de medición como timeit resultan útiles para obtener mediciones fiables del rendimiento de un algoritmo.

Este trabajo integrador permitió poner en práctica conceptos de análisis de algoritmos, mejorar habilidades en Python y fortalecer el enfoque crítico para seleccionar soluciones eficientes en programación.

Link al video de YouTube: https://youtu.be/EtfTSW-oekQ

Referencias

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms. MIT Press.
- Skiena, S. S. (2017). The Algorithm Design Manual. Springer.
- Sedgewick, R. (2011). Algorithms in Python. Addison-Wesley.
- Python Software Foundation. https://docs.python.org/
- Wikipedia contributors. Búsqueda lineal. https://es.wikipedia.org/wiki/B%C3%BAsqueda_lineal
- Wikipedia contributors. Búsqueda binaria. https://es.wikipedia.org/wiki/B%C3%BAsqueda_binaria