Laboratorio 1: Busqueda

29 de Septiembre, 2025

Alumno	Mirko Peñailillo Vasquez
Profesor	José Fuentes Sepúlveda
Ayudante	Luciano Argomedo
N^{ro} Matricula	2021900322

1 Resumen

El objetivo de este laboratorio es implementar, comprender y analizar distintos algoritmos de busqueda en secuencias. Ademas de reforzar el manejo del lenguaje C++ mediante la implementacion y prueba de estos.
Se estudiaron tres algoritmos distintos: Busqueda secuencial, busqueda binaria y busqueda galopante.

2 Análisis Teórico

2.1 Busqueda Secuencial

La busqueda secuencial recorre los elementos de el vector o lista uno a uno hasta encontrar el valor consultado. No requiere que los elementos esten ordenados.
Complejidad temporal: $O(n)$ en peor caso, donde n es el largo del arreglo u $O(p)$, donde p es la posicion del elemento a buscar.

2.2 Busqueda Binaria

La busqueda binaria compara el valor a buscar con el elemento central de la secuencia, dependiendo del resultado, se descarta la mitad izquierda o derecha de los elementos, iterando estos pasos hasta encontrar el elemento consultado. Requiere que los elementos esten ordenados.
Complejidad temporal: $O(log\ n)$, donde n es el largo de la secuencia.

2.3 Busqueda Galopante

La busqueda galopante primero busca el rango o umbral donde podria encontrarse el valor consultado, aumentando el indice en potencias de dos en cada paso $(1, 2, 4, ..., 2^n)$. Una vez hallado este intervalo, se aplcia busqueda binaria en dicho rango.
Complejidad temporal: $O(log\ p)$, donde p es la posicion del elemento consultado.

3 Implementación

- Las implementaciones de los tres algoritmos de busqueda fueron realizadas desde cero en C++.
- La busqueda secuencial y binaria fueron codificadas de manera iterativa.
- La busqueda galopante fue implementada apoyandose en la implementacion de la busqueda binaria. El archivo adjunto "Boletin1.cpp" contiene la implementacion de las tres busquedas.

4 Resultados Experimentales

4.1 Decisiones de inicializacion

- Para la experimentacion realizada respecto a el efecto del tamaño de la secuencia en los tiempos de ejecucion, para cada n (rango 1000 a 100000, con saltos de 1000) se creo un vector<int> ordenado con valores $0, 1, ..., n-1$. El elemento consultado fue elegido de manera aleatoria en cada ejecucion de la busqueda.
- Para la experimentacion realizada respecto a el efecto de la posicion del elemento en una secuencia de tamaño fijo, se fijo un N_FIXED de tamaño 100000 y se consulto por el elemento en la posicion "pos", barriendo pos en el rango 1000 a 100000, con saltos de 1000.
- Para cada experimento realizado se utilizo $RUNS = 32$ en el programa "uhr.cpp" disponible en el repositorio del curso [1].

4.2 Uso de datasets externos

No se utilizaron datasets externos para la experimentacion.

4.3 Caracteristicas de la maquina donde se realizaron los experimentos

- Procesador: AMD Ryzen 5 3600 (3.6 GHz)
- Memoria RAM: 16GB DDR4 (3600 MHz)

4.4 Descripcion de los resultados mediante tablas o graficos

- En cada figura podemos observar los graficos correspondientes a los resultados obtenidos con cada algoritmo de busqueda utilizando un tamaño de secuencia variable (figuras 1, 2 y 3) y un tamaño de secuencia fijo, variando la posicion del elemento a buscar (figuras 4,5 y 6).
- Notar que en cada figura aun se puede observar un ligero ruido por efecto de cache/OS, pero se pueden observar claramente las tendencias de las curvas.
- Graficos
  En la figura 1, correspondiente a el grafico del tiempo promedio por busqueda de la busqueda binaria con un tamaño de secuencia variable, podemos observar una curva con crecimiento muy lento, consistente con $O(log\ n)$.

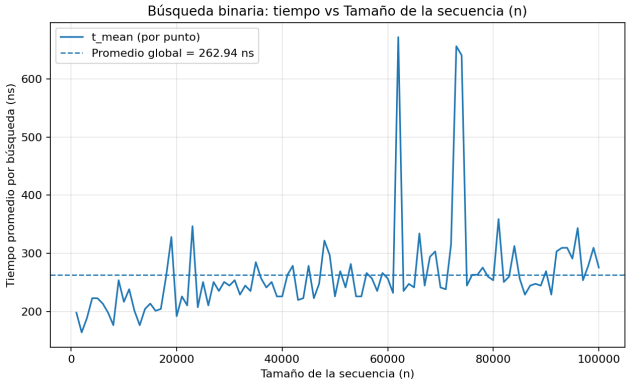

Figure 1: Busqueda binaria con tamaño de secuencia variable

En la figura 2, podemos observar un comportamiento similar al de la figura 1, con un notable "sobre costo" debido a la fase de galopado.

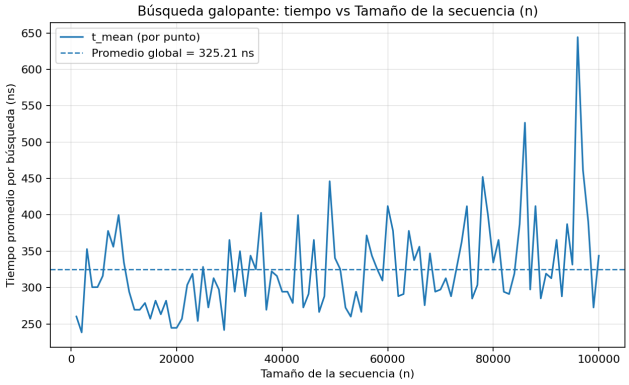

Figure 2: Busqueda galopante con tamaño de secuencia variable

En la figura 3 podemos observar la tendencia de la busqueda lineal hacia crecer proporcionalmente a el tamaño de la secuencia.

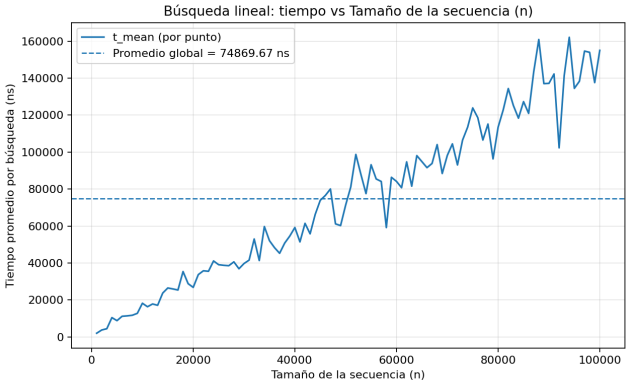

Figure 3: Busqueda lineal con tamaño de secuencia variable

En la figura 4 podemos observar que, si bien existe ruido, el tiempo es practicamente independiente de la posicion consultada, lo que coincide con la naturaleza de $O(log\ n)$

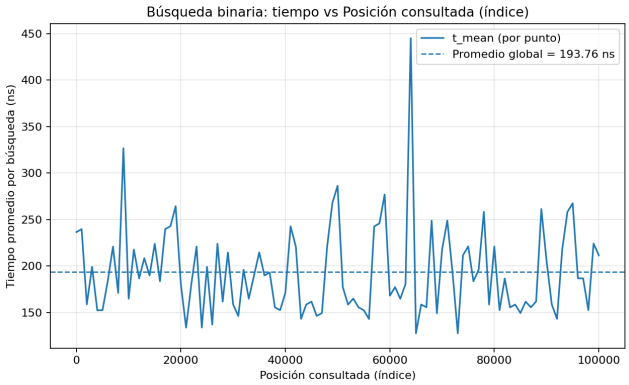

Figure 4: Busqueda binaria con posicion de clave variable

En la figura 5 podemos observar que el tiempo crece suavemente a medida que aumenta la posicion de la clave consultada, lo que es consistente con la complejidad temporal $O(log\ p)$ de la busqueda galopante.

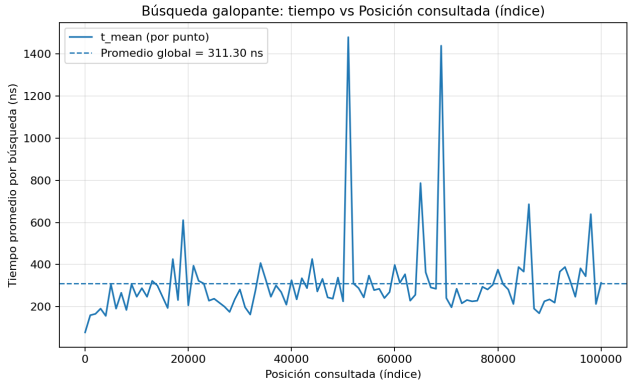

Figure 5: Busqueda galopante con posicion de clave variable

En la figura 6 podemos observar que el tiempo aumenta casi linealmente con la posicion consultada, lo que es consistente con la complejidad temporal de $O(p)$.

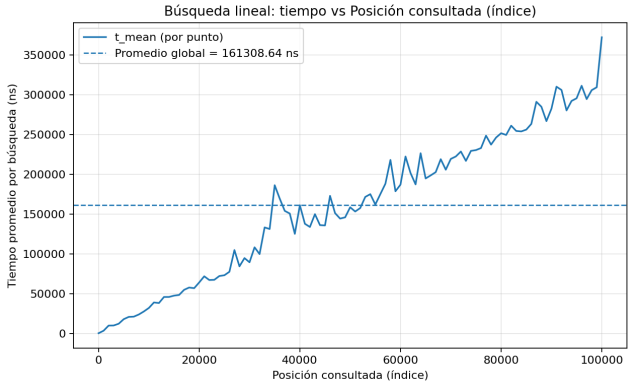

Figure 6: Busqueda lineal con posicion de clave variable

5 Conclusiones

5.1 Validacion de complejidades temporales teoricas

En los resultados presentados se puede observar que, si bien existe ruido, los algoritmos presentan una tendencia a las complejidades temporales que fueron discutidas en clases y presentadas en la seccion Análisis Teórico de este informe.

- La busqueda lineal o secuencial muestra, en promedio, un crecimiento proporcional a el tamaño de la secuencia ($O(n)$) al buscar un elemento en una posicion aleatoria. Mientras que al tener un tamaño fijo de arreglo, muestra un crecimiento proporcional a la posicion del elemento a buscar ($O(p)$).

- En el caso de la busqueda binaria podemos observar que presenta un leve crecimiento en la curva al aumentar el tamaño de la secuencia, mientras que si el tamaño de la secuencia se mantiene constante y solo varia la posicion del elemento a buscar, la curva no presenta una tendencia notable. Este comportamiento es consistente con la complejidad temporal de $O(log\ n)$.

- La busqueda galopante presenta un comportamiento similar a la busqueda binaria, con un crecimiento en la curva mas notable al aumentar el indice del elemento a buscar, ademas podemos notar un tiempo de ejecucion generalmente mayor a la busqueda binaria debido a la primera parte del algoritmo de busqueda galopante, que busca el rango donde se encuentra el indice de el elemento que buscamos, notemos que el costo total de la busqueda galopante es $(2 \cdot log(\ p))$ debido a las dos busquedas que hace el algoritmo sobre la secuencia, aunque en el analisis de complejidad temporal escribimos $(O(log\ p))$.

5.2 Promedio de tiempo con tamaño de secuencia variable

- Busqueda lineal: $74869,67\ ns$
- Busqueda binaria: $262,94\ ns$
- Busqueda galopante: $325,21\ ns$

Podemos observar de el total del tiempo en toda la experimentacion de cada algoritmo de busqueda cuando consideramos un tamaño de secuencia variable, que la busqueda lineal fue aproximadamente 285 veces mas lenta que la busqueda binaria y 230 veces mas lenta que la busqueda galopante. Ademas, la busqueda galopante fue $1,23$ veces mas lenta que la busqueda binaria, por el costo adicional de acotar el rango de busqueda.

5.3 Que algoritmo usar y cuando

Como ha sido estudiado durante el curso y presentado en este informe, los tres algoritmos de busqueda presentan ventajas y desventajas que es importante conocer al momento de decidir que algoritmo utilizar para un problema en particular.

- Si bien durante este informe hemos observado que la busqueda lineal presenta un peor desempeño temporal en comparacion con la busqueda binaria y galopante, es el unico algoritmo de busqueda entre los tres que puede ser utilizado en secuencias desordenadas.
- La busqueda binaria presenta los mejores resultados experimentales, sin embargo, es importante notar que el algoritmo requiere conocer el tamaño de la secuencia y que esta este ordenada. por ello que en problemas en los que se conozca el tamaño de una secuencia ordenada sobre la que se realizaran consultas se recomienda el uso de busqueda binaria
- El desempeño de la busqueda galopante es similar a el de la busqueda binaria, con un pequeño "costo" temporal adicional en la experimentacion, sin embargo, el algoritmo no necesita conocer el tamaño de la secuencia completa. Debido a esto, en problemas en los que se desconozca el tamaño de una secuencia ordenada sobre la que se realizaran consultas se recomienda el uso de busqueda galopante.
- Finalmente algo importante de recalcar es que, si bien la busqueda lineal es la unica que permite buscar elementos en una secuencia desordenada, si se sabe que se realizaran muchas consultas sobre la secuencia puede resultar conveniente primero ordenar la secuencia una vez y luego realizar las consultas con una busqueda binaria, amortizando el costo inicial.

6 Referencias

References

[1] jfuentess. "Edaa." Repositorio en GitHub, Accessed: Sep. 28, 2025. [Online]. Available: `https://github.com/jfuentess/edaa`.