Lab 1: Evaluación Experimental de Búsquedas

Daniel Andrés Moreno Cartagena

10 de septiembre de 2021

1. Consideraciones

Los códigos de búsquedas se realizaron mediante una implementación iterativa, con diferentes tamaños de inputs, los cuales se muestran en la Sección 2. Asimismo, se realizaron 100.000 replicas para cada tamaño considerando el peor caso y 100.000 replicas por posición de elementos buscados. Además se consideraron que los vectores estuvieran ordenados para no incurrir en un costo de $O(n\ log\ n)$ con los algoritmos de búsqueda binaria y doblada que necesitan que el vector este ordenado.

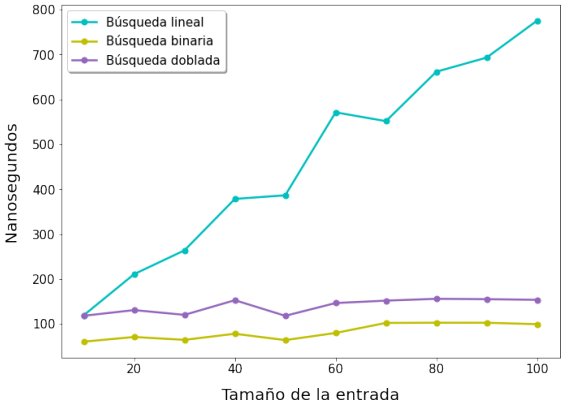
2. Ejercicios

1. Los algoritmos se implementaron en el lenguaje de programación C++.
2. Razonamiento de las complejidades en $Big-Oh$. La explicación más detallada de cada punto se puede leer en el ejercicio 4.
   - **Búsqueda lineal:** El algoritmo tiene una complejidad en notación O grande de $O(n)$, lo que en general provoca que el tiempo de búsqueda para n pequeños sea muy exhaustivo, como se puede observar en la Figura (1a), lo que a la vez complica la comparación entre estos tres algoritmos para n de tamaños medianos-grandes.
   - **Búsqueda binaria:** El algoritmo tiene una complejidad en notación O grande de $O(log\ n)$, lo que reduce significativamente el tiempo de buscar un elemento, con respecto al algoritmo de búsqueda lineal, para tamaños de n pequeños (como se puede observar en la Figura 1a), y por consiguiente, para n de tamaños medianos-grandes.
   - **Búsqueda doblada:** El algoritmo tiene una complejidad en notación O grande de $O(log\ n)$, lo que reduce significativamente el tiempo de buscar un elemento, con respecto al algoritmo de búsqueda lineal, para tamaños de n pequeños (como se puede observar en la Figura 1a), y por consiguiente, para n de tamaños medianos-grandes. Sin embargo, a pesar de que, en teoría, la búsqueda doblada tenga la misma complejidad que el algoritmo de búsqueda binaria, los tiempos que demoran en la practica, en responder a la consulta, difieren entre si.
3. Intervalo de evaluación de los algoritmos.
   - **Tamaño (n):** Se evaluaron 10 tamaños distintos, en dos escenarios, desde n pequeños hasta n lo suficientemente grandes para observar diferentes comportamientos de los algoritmos. Los intervalos de evaluación de los tamaños para n pequeños fueron de: 10, 20, 30, 40, 50, 60, 70, 80, 90 y 100, en donde se utilizaron los tres algoritmos en la comparación. Por otro lado, para intervalos de n medianos-grandes, se utilizaron los dos algoritmos $O(log\ n)$, ya que el tiempo computacional del algoritmo de búsqueda lineal era significativamente mayor. Los puntos evaluados fueron: 1.000.000, 10.000.000, 20.000.000, 30.000.000, 40.000.000, 50.000.000, 60.000.000, 70.000.000, 80.000.000, 90.000.000.

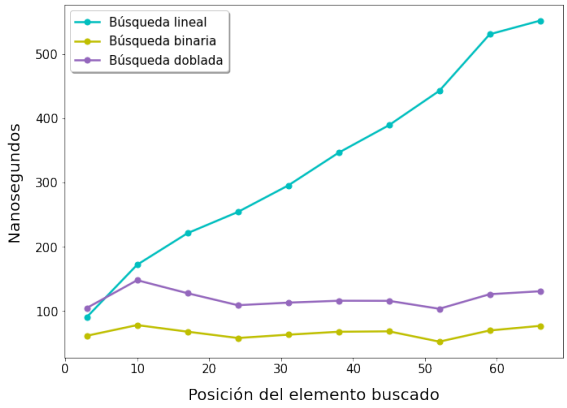
- **Posición (p):** Se evaluaron 10 posiciones distintas, en dos escenarios, con un tamaño del vector fijo de 100 y 90.000.000. En ambos escenarios se varió el elemento buscado en la misma proporción a lo largo del vector. La posición de los elementos en el primer escenario fue de: 3, 10, 17, 24, 31, 38, 45, 52, 59 y 66. En cambio, la posición de los elementos en el segundo escenario fue de: 8700000, 17400000, 26100000, 34800000, 43500000, 52200000, 60900000, 69600000, 78300000 y 87000000

4. Resultados y conclusiones.

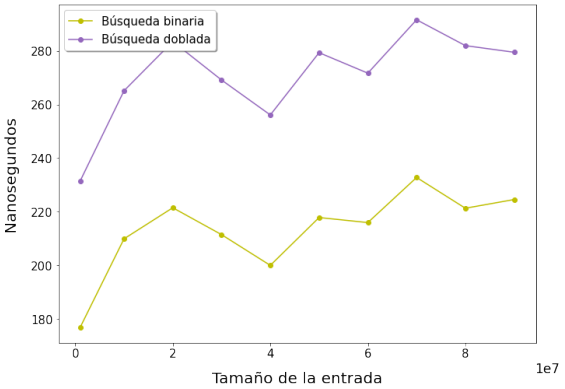
   Como se puede observar en la Figura (1a), el algoritmo de búsqueda lineal demostró tener un rendimiento excesivamente menor con respecto a las búsquedas binarias y dobladas utilizando el peor caso. Asimismo, se muestra, que a pesar de que el algoritmo de búsqueda binaria y búsqueda doblada, en teoría tengan el mismo tiempo de complejidad, en la practica el algoritmo de búsqueda doblada para tamaños de n pequeños (Figura 1a) y n grandes (Figura 1c) termina siendo, en general, mas costoso con respecto al tiempo de respuesta de la búsqueda. Esto debido a la constante implícita que tiene el algoritmo de búsqueda doblada ($2 * log\ n$, en el peor caso), lo cual no se observa en el análisis asintótico.



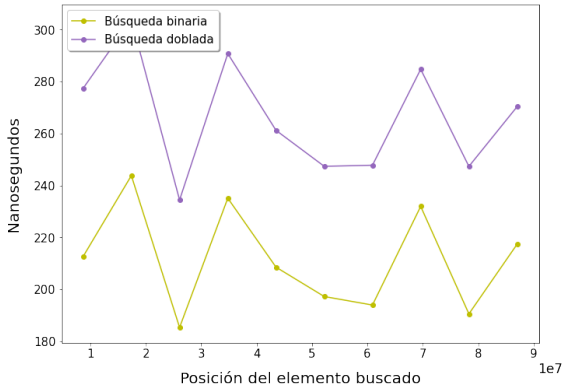
(a) Tamaños de entrada pequeños



(b) Variación de posición (Tamaño fijo: 100)



(c) Tamaños de entrada mediana-grande



(d) Variación de posición (Tamaño fijo: 90.000.000)

Figura 1: Evaluación experimental de algoritmos de búsquedas clásicos

En lo que concierne a la experimentación con un tamaño fijo y cambiando de posición el elemento buscado, se demostró que el costo computacional de los tres algoritmos de búsquedas para un escenario con tamaño fijo de 100 (Figura 1b) mantenían las mismas conclusiones que las mencionadas anteriormente, es decir, el algoritmo de búsqueda lineal tuvo un peor rendimiento en

buscar diferentes elementos dentro del vector, con tamaño fijo de 100, y el algoritmo de búsqueda binaria mantuvo el primer lugar en cuanto al menor tiempo computacional en la respuesta a la consulta. Asimismo, para el escenario con un tamaño fijo de 90.000.000 (Figura 1d), el algoritmo de búsqueda binaria mantuvo un mejor rendimiento con respecto al algoritmo de búsqueda doblada para todos los tamaños de n.

En conclusión, la búsqueda lineal debería utilizarse para n pequeños, en donde se tenga la certeza de no incurrir en el peor caso y no tener que recorrer mas de la mitad del vector, ya que como se demostró, si el tamaño del vector es al menos de 20 elementos y se incurre en el peor caso, entonces es preferible utilizar las búsquedas binarias y dobladas (Figura 1a, segundo tamaño evaluado). Asimismo, las búsquedas binarias deberían utilizarse en todos los casos en donde no se conozcan posibles patrones en los elementos almacenados en el vector, ya que siempre obtuvo un mejor rendimiento computacional que los otros dos algoritmos evaluados. Por ultimo, con respecto a la búsqueda doblada, no se pueden obtener mayores conclusiones de los experimentos realizados, ya que en general su costo computacional se mantuvo entre la búsqueda lineal y binaria

3. Evidencia de ejecución

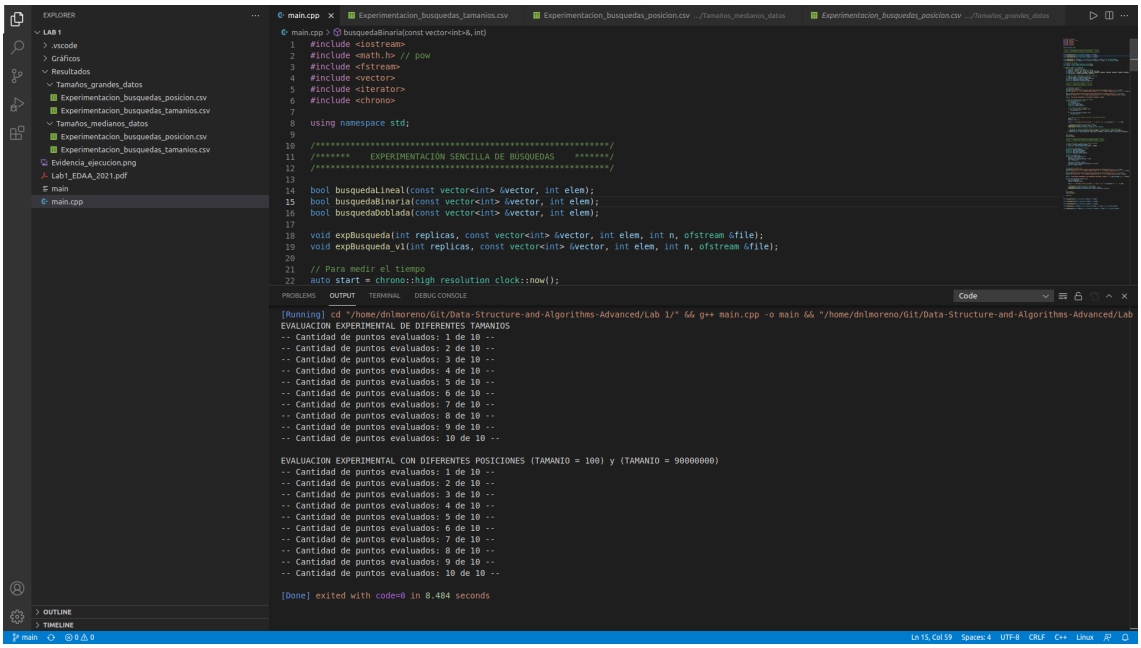


Figura 2: Evidencia de ejecución del código