Boletín 1:
Problema inicial

Nombre del estudiante: Dazhi Enrique Feng Zong
Número de matrícula: 2019435710

Introducción

En este primer boletín, se tiene como objetivos mejorar la programación, compilación y ejecución de programas, e implementar los algoritmos de búsqueda en secuencias vistos en clases. En particular, se implementarán: la búsqueda **secuencial, binaria** y **galopante**.

La búsqueda secuencial examina elementos uno por uno para encontrar el objetivo. Es la única opción para secuencias no ordenadas, pero es más lenta que otros enfoques.

La búsqueda binaria es altamente eficiente, pero requiere que la secuencia esté ordenada. Divide repetidamente la secuencia a la mitad, descartando la mitad que no contiene el elemento buscado.

La búsqueda galopante es una variante optimizada de la búsqueda binaria. A diferencia de esta última, que divide el rango de búsqueda a la mitad en cada paso, la búsqueda galopante aumenta gradualmente el tamaño del intervalo de búsqueda mediante saltos exponenciales. Después de encontrar un intervalo que contiene el elemento deseado, se realiza una búsqueda binaria dentro de ese intervalo.

Desarrollo:

Análisis Teórico:

Búsqueda Secuencial

Recorre cada elemento del arreglo en orden secuencial hasta encontrar el elemento deseado. Entonces:

- **Caso promedio:** En este caso, si el elemento deseado se encuentra en la posición p, se requerirá recorrer p elementos, lo que resulta en una complejidad $\mathcal{O}(p)$.
- **Mejor caso:** Cuando el elemento buscado está en la primera posición ($p = 0$), el algoritmo realiza una única comparación y se detiene, con una complejidad constante de $\mathcal{O}(1)$.
- **Peor caso:** Si el elemento buscado se encuentra al final del arreglo o no está presente, se deben comparar todos los n elementos del arreglo en el peor caso, resultando en una complejidad de $\mathcal{O}(n)$.

Búsqueda Binaria

Divide repetidamente el arreglo en dos mitades y compara el elemento buscado con el elemento en el centro de el arreglo. Entonces:

- **Caso promedio:** No hay un caso promedio que dependa de la posición p; su complejidad se mantiene logarítmica a menos que se alcance el mejor caso.
- **Mejor caso:** Cuando el elemento buscado es el elemento central de el arreglo ($p = \frac{n}{2}$), el algoritmo realiza una única comparación y se detiene, con una complejidad constante de $\mathcal{O}(1)$.
- **Peor caso:** Si el elemento buscado no está presente en el arreglo o se encuentra en un extremo, el algoritmo descarta la mitad de los elementos restantes en cada paso. Como resultado, el número de pasos es $\log_2 n$, lo que se traduce en una complejidad de $\mathcal{O}(\log n)$.

Búsqueda Galopante (Exponencial)

Duplica repetidamente el tamaño del intervalo de búsqueda hasta que encuentra un intervalo que contiene el elemento buscado. Luego, realiza una búsqueda binaria en ese intervalo. Entonces:

- **Caso promedio:** En cada paso, el algoritmo duplica el tamaño del intervalo de búsqueda. Si el elemento deseado está en la posición p, se necesitarán aproximadamente $\log_2 p$ pasos para encontrar un intervalo que lo contenga, y luego se realiza una búsqueda binaria en ese intervalo. En conjunto, la complejidad es $\mathcal{O}(\log p)$ para encontrar el elemento.
- **Mejor caso:** Al igual que en la búsqueda secuencial, si el elemento buscado está en la primera posición ($p = 0$), el algoritmo realiza una comparación y se detiene, con una complejidad constante de $\mathcal{O}(1)$.
- **Peor caso:** Similar a la búsqueda binaria, en el peor caso el algoritmo realiza $\log_2 n$ pasos, donde n es el tamaño de el arreglo, lo que se traduce en una complejidad de $\mathcal{O}(\log n)$.

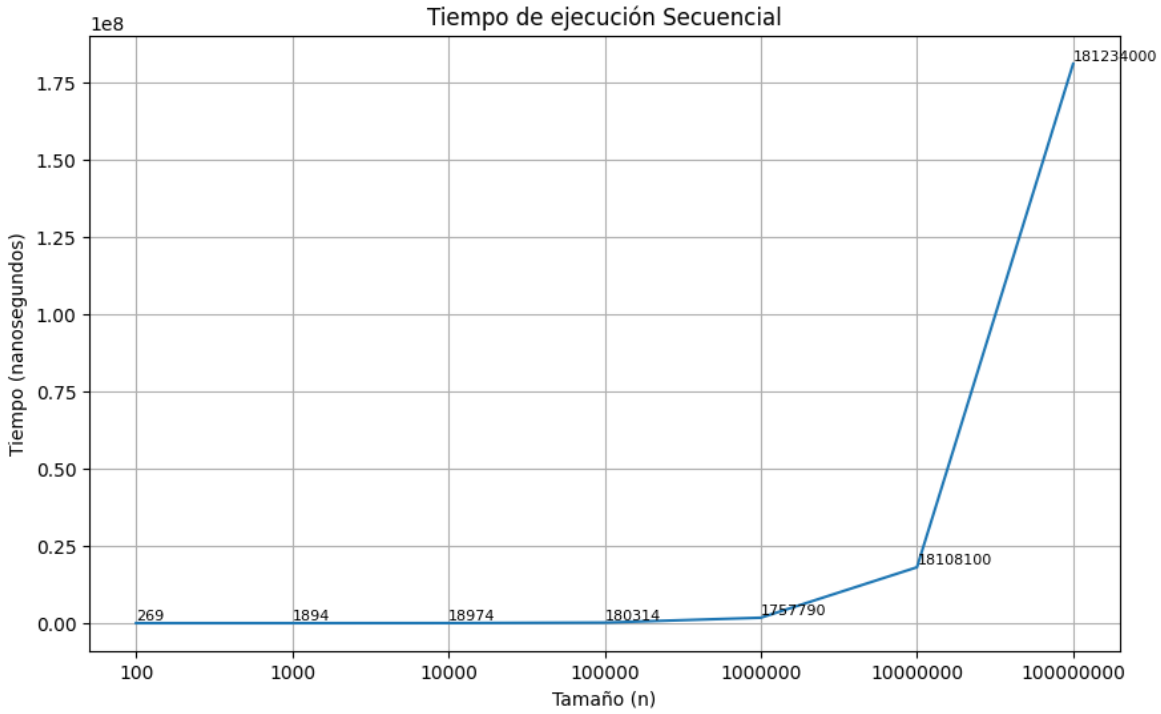
Análisis Experimental:

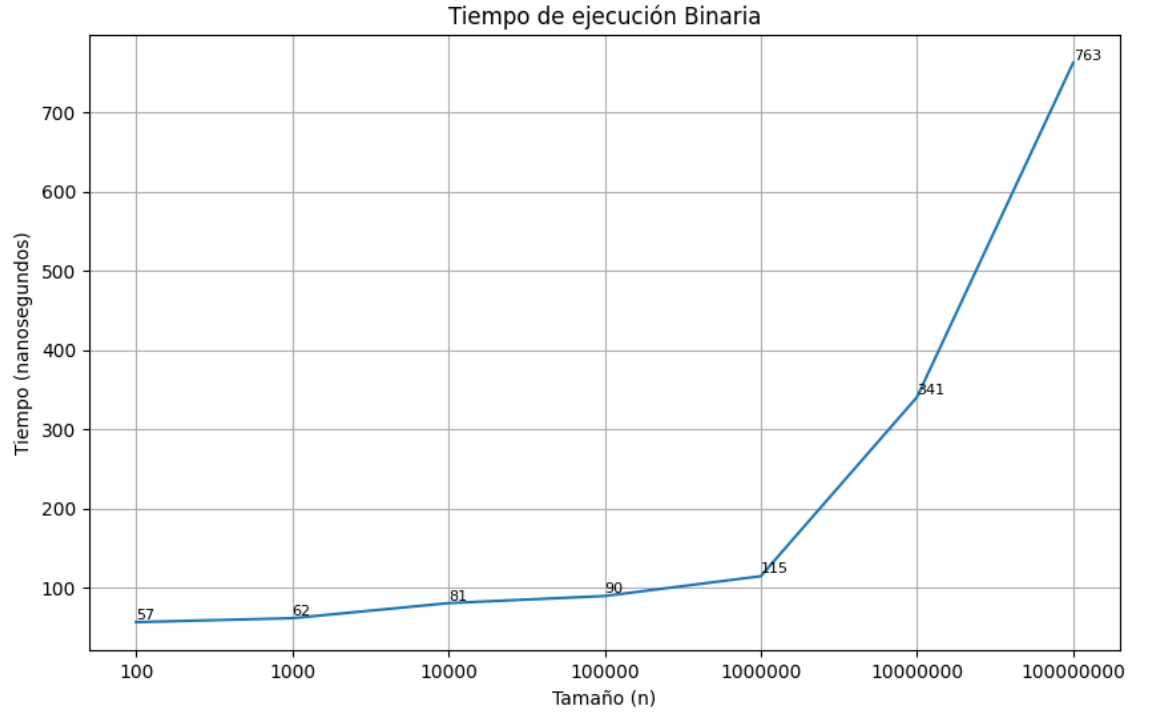
Se llevaron a cabo dos experimentos diferentes para analizar el rendimiento de la búsqueda de elementos en una secuencia. En el primer experimento, se investigó el efecto del tamaño de la secuencia en los tiempos de ejecución, manteniendo constante el elemento a buscar. El tamaño de la secuencia varió, comenzando con 100 elementos y multiplicándose por 10 en cada iteración hasta llegar a 100,000,000. El elemento que se buscó en cada caso fue uno que no estaba presente en el arreglo.

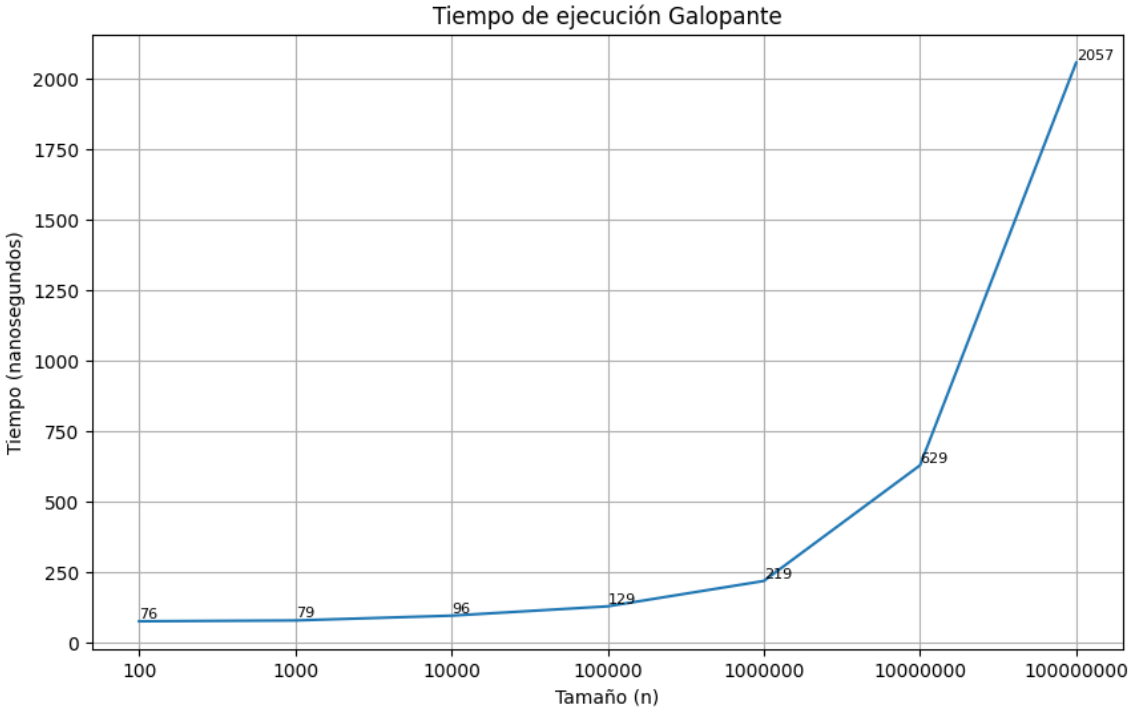
En el segundo experimento, se mantuvo el tamaño del arreglo constante en 100,000,000 elementos, pero se varió el número a buscar. Se realizaron búsquedas del elemento en la posición 0, luego en la posición 10,000,000, y así sucesivamente, hasta llegar a la posición 100,000,000.

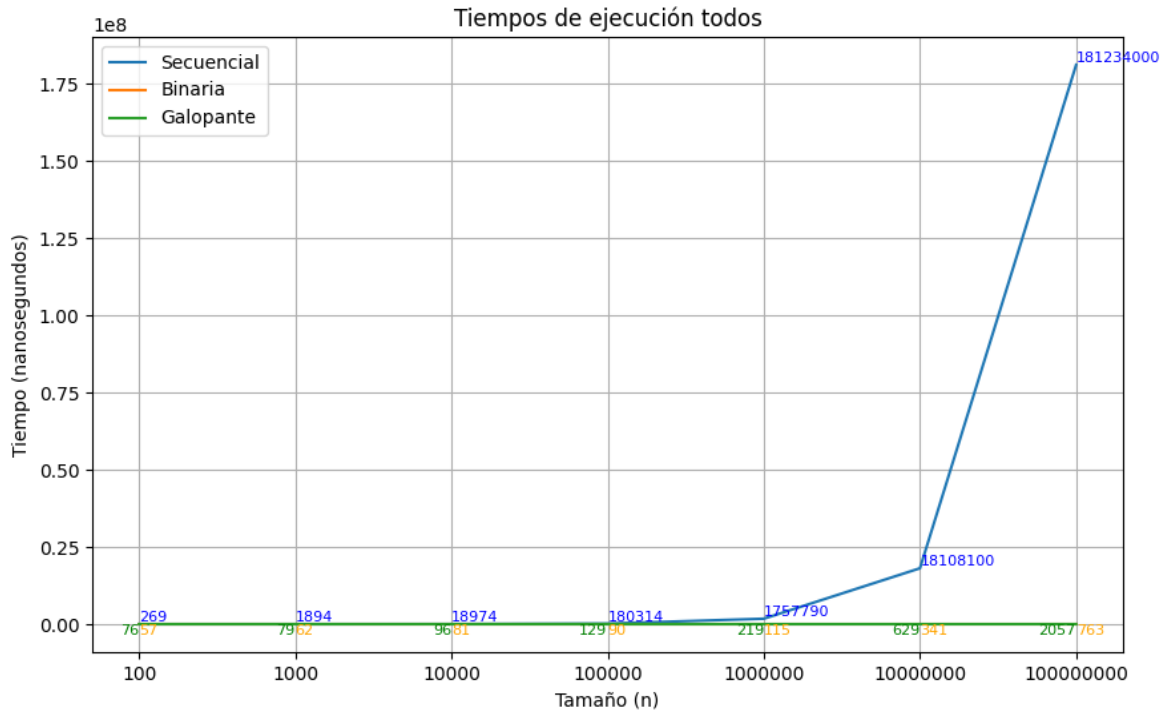
Para ambos experimentos, se repitió la ejecución 150 veces. Las primeras 50 repeticiones se consideraron "cold runs" y no se incluyeron en el análisis final. Las 100 repeticiones restantes fueron consideradas en el análisis.

Análisis Experimental 1: Entrada variable y número a buscar fijo

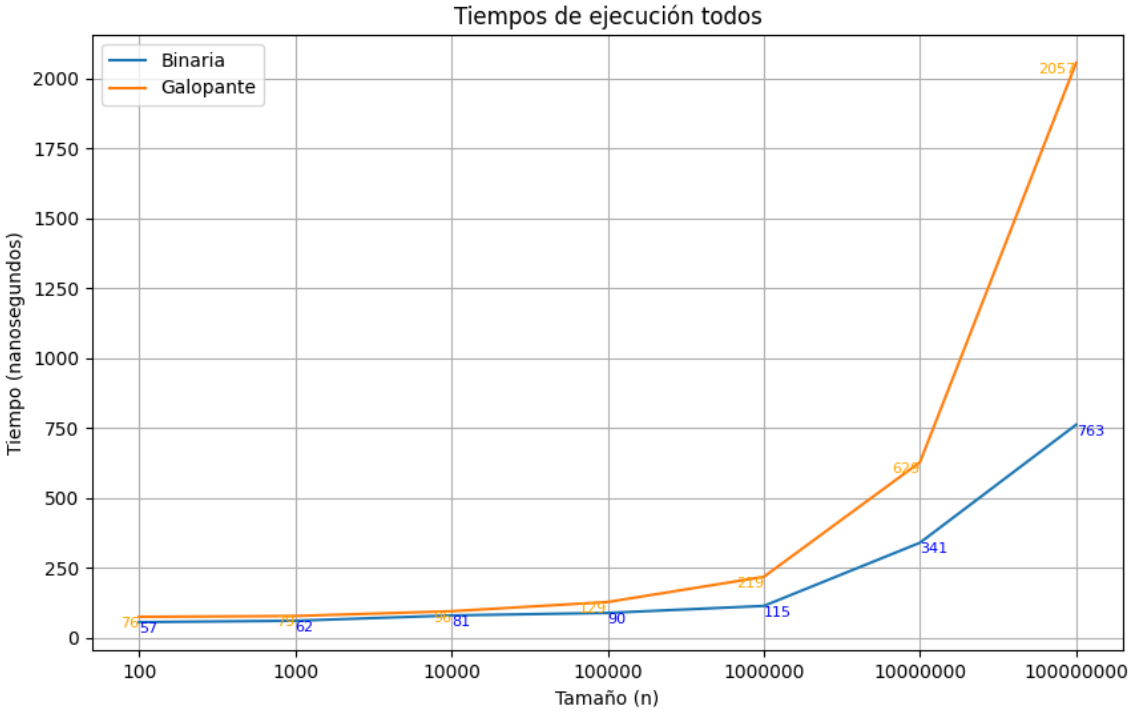





Tiempo de ejecución Galopante
Tiempo (nanosegundos)
Tamaño (n)
0
250
500
750
1000
1250
1500
1750
2000
100
1000
10000
100000
1000000
10000000
100000000
76
79
96
129
219
629
2057


Tiempos de ejecución todos
1e8
Secuencial
Binaria
Galopante
Tiempo (nanosegundos)
Tamaño (n)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
100
1000
10000
100000
1000000
10000000
100000000
181234000
18108100
1757790
180314
18974
1894
269
7657
7962
9681
12990
21915
62941
205763

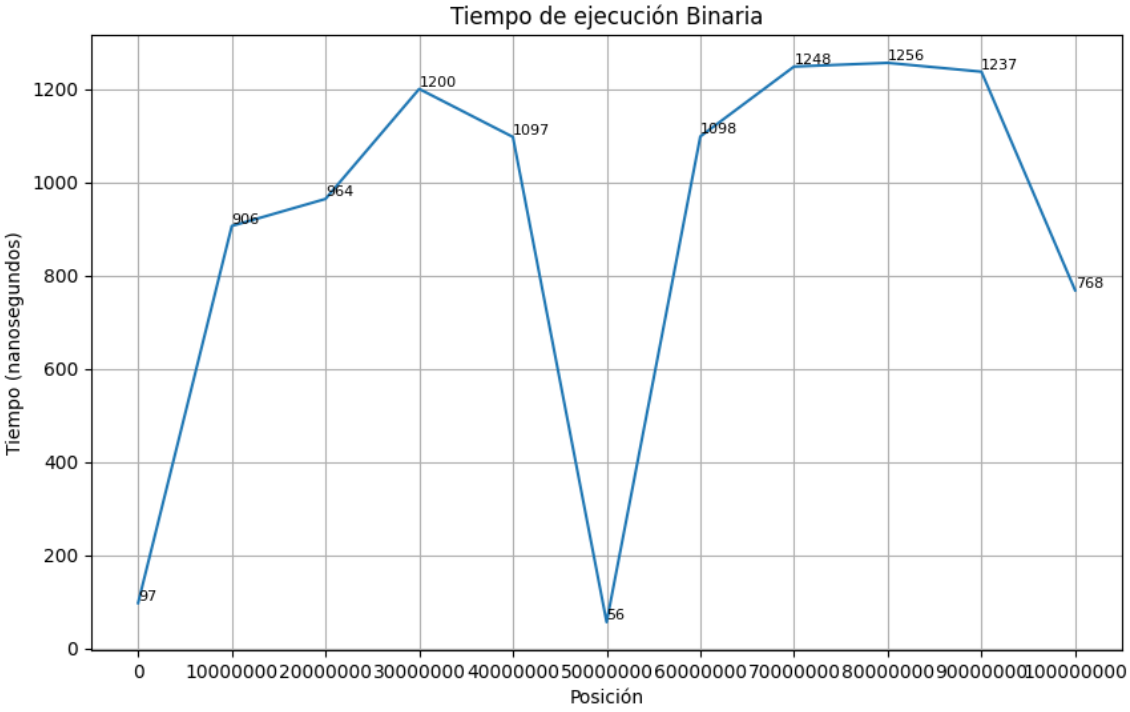
Se realizó un acercamiento a las dos búsquedas logarítmicas para una mejor visualización:

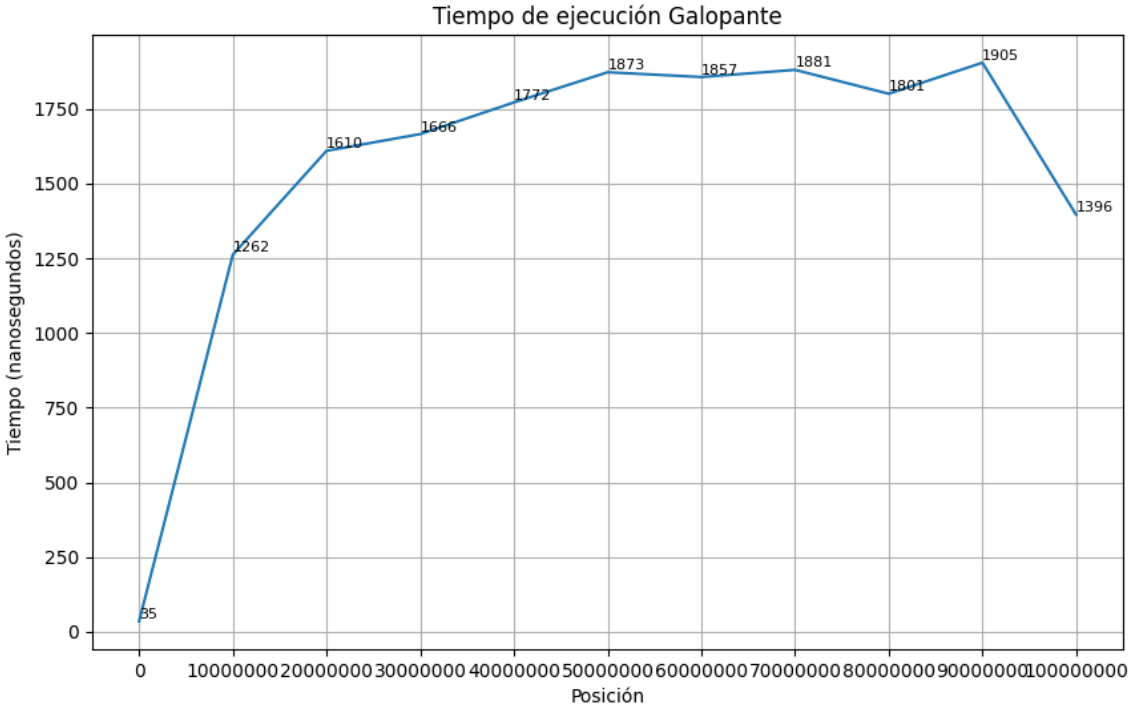


n	TiempoSec	VarSec	TiempoBin	VarBin	TiempoExp	VarExp
100	269	2739	57	2651	76	1824
1000	1894	114764	62	2556	79	2259
10000	18974	1.19147e+07	81	3339	96	2184
100000	180314	3.29594e+08	90	1700	129	7259
1000000	1.75779e+06	5.40432e+09	115	1275	219	19939
10000000	1.81081e+07	1.75176e+11	341	106419	629	27459
100000000	1.81234e+08	1.71148e+13	763	51531	2057	1.89405e+07

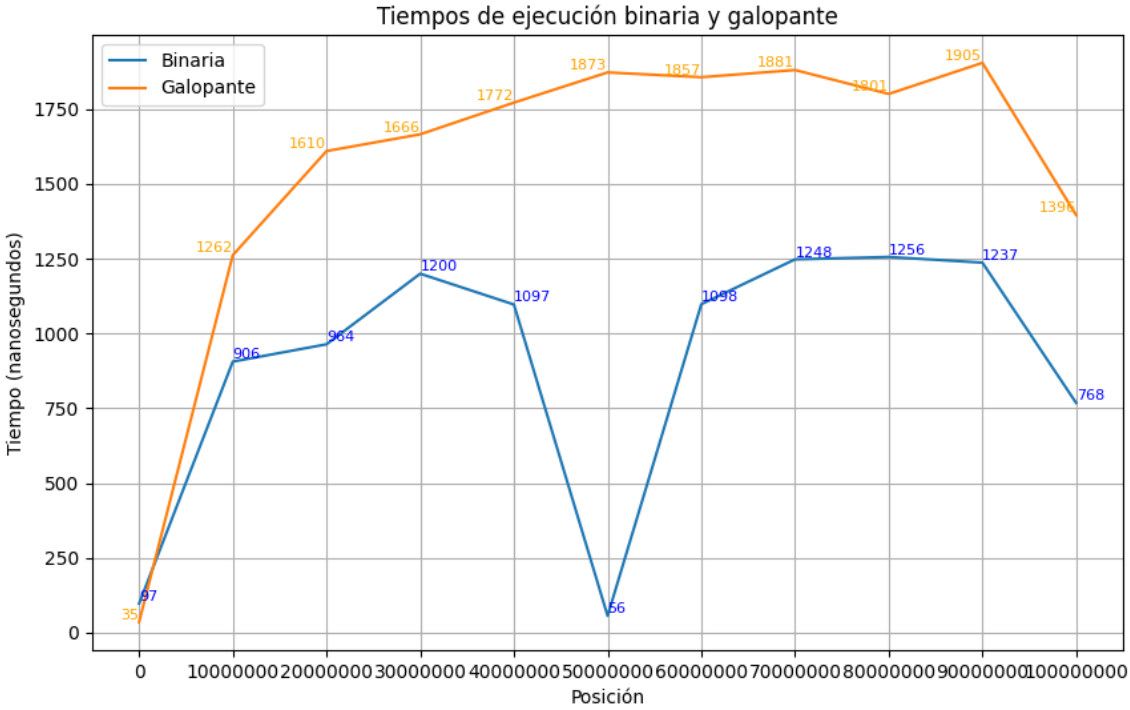
Cuadro 1: Resultados de los tiempos de ejecución en nanosegundos para entrada variable y número a buscar constante. Var indica la varianza.

Análisis Experimental 2: Entrada constante y número a buscar variable





Dado que la búsqueda secuencial es mejor solo al inicio ($p = 0$), nos enfocamos en comparar la búsqueda binaria con la búsqueda galopante:



n	TiempoSec	VarSec	TiempoBin	VarBin	TiempoExp	VarExp
0	25	1875	97	491	35	2275
10000000	1.82881e+07	3.67426e+11	906	276564	1262	90756
20000000	3.6527e+07	4.55734e+11	964	91904	1610	638300
30000000	5.48768e+07	2.74836e+12	1200	375800	1666	281844
40000000	7.28659e+07	1.27536e+12	1097	72891	1772	220616
50000000	9.14206e+07	4.14412e+12	56	2464	1873	111171
60000000	1.09365e+08	2.18262e+12	1098	81996	1857	116251
70000000	1.27843e+08	6.02648e+12	1248	124896	1881	247539
80000000	1.4536e+08	2.35084e+12	1256	96064	1801	99099
90000000	1.63694e+08	3.15525e+12	1237	84131	1905	165075
100000000	1.81946e+08	4.68056e+12	768	47176	1396	61984

Cuadro 2: Resultados de los tiempos de ejecución en nanosegundos para entrada constante y número a buscar variable. Var indica la varianza.

Discusión y Conclusión:

En el primer experimento, se investigó cómo el tamaño de la secuencia afecta los tiempos de ejecución, manteniendo constante el elemento a buscar. Los resultados muestran un aumento predecible en los tiempos de ejecución a medida que aumenta el tamaño de la secuencia, lo cual es una observación intuitiva. En el caso de la búsqueda secuencial, observamos un aumento lineal, donde el tiempo se incrementa proporcionalmente al tamaño de entrada, lo que coincide con el análisis teórico (el tiempo se incrementa aproximadamente en un factor de 10 cuando aumentamos el tamaño de la entrada en un factor de 10). Por otro lado, tanto la búsqueda binaria como la búsqueda galopante también aumentan en tiempo con el tamaño de entrada, pero no de manera lineal. En lugar de eso, tiene un crecimiento logarítmico con diferencias de tiempo muy pequeñas, como predice el análisis teórico.

En el segundo experimento, se mantuvo constante el tamaño de la secuencia pero se varió la posición del elemento a buscar en diferentes posiciones. Se observó que la posición del elemento a buscar también influye en los tiempos de ejecución, siendo más cortos cuando el elemento se encuentra al principio de la secuencia y más largos cuando se encuentra al final. Este patrón se evidenció claramente en la búsqueda secuencial. En cuanto a la búsqueda binaria y galopante, notamos una diferencia más significativa al buscar el primer elemento ($p = 0$) en comparación con los demás. Sin embargo, la búsqueda entre los otros elementos muestra tiempos muy similares, excepto en el mejor caso de búsqueda binaria, que es buscar en la mitad ($p = n/2$).

En base al análisis realizado y los resultados obtenidos, se puede concluir que la búsqueda secuencial es la opción menos eficiente, pero es la única viable si el arreglo no está ordenado. Además, es la más sencilla de implementar. En cuanto a la búsqueda binaria y galopante, teóricamente, la búsqueda galopante debería ser más rápida que la binaria en los extremos. Sin embargo, en la práctica, no se observó una diferencia significativa, ya que la búsqueda binaria resultó superior en ambos análisis. Es importante mencionar que las diferencias de tiempo son minúsculas, medidas en nanosegundos, y apenas perceptibles. Además, cabe destacar que los tiempos son muy volátiles, con una alta variabilidad.