INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

ANALISIS TEMPORAL

PRACTICA 2

INTEGRANTES

LUIS FERNANDO RESENDIZ CHAVEZ

LUIS RAMIREZ

ERICK QUINTANA MARTÍNEZ

ANALISIS DE ALGORITMOS

EDGARDO ADRIAN FRANCO MARTINEZ

3CM3

Introducción

Objetivo

Realizar el análisis a priori de varios algoritmos de búsqueda conocidos para poder realizar estimaciones de su comportamiento cuando el tamaño de problema crece.

Planteamiento del problema

El equipo debe implementar versiones no recursivas de los siguientes algoritmos de búsqueda en ANSI C para medir los tiempos de búsqueda con distintos parámetros N, que representa la cantidad de elementos que componen el arreglo:

- Búsqueda lineal o secuencial
- Búsqueda en un árbol binario de búsqueda
- Búsqueda binaria o dicotómica
- Búsqueda exponencial
- Búsqueda de Fibonacci

Desarrollo

Todas las pruebas fueron realizadas en una misma computadora que cuenta con un procesador Intel Core i5 @ 1.6 GHz, y 8GB de RAM. Cada programa fue compilado con GCC en el subsistema de Linux para Windows 10 (64 bits), que nos da acceso a una terminal de Linux para escribir los códigos de los algoritmos.

Cada algoritmo debe ser ejecutado varias veces con los distintos parámetros N solicitados. Para facilitar dichas pruebas, se crearon varios scripts Shell que ejecutan el programa varias veces de manera automática.

Para realizar la medición de los tiempos de una manera unificada, todos los algoritmos fueron modificados de las siguientes formas (un ejemplo para la búsqueda lineal se muestra en el anexo):

- Se añadió un arreglo de tamaño 20 que incluye las solicitudes de los números, como lo indica la práctica. Cada programa itera en ese arreglo para enviar sus elementos como argumentos.
- Las búsquedas fueron implementadas con funciones. La medición del tiempo comienza desde el momento que se manda a llamar la función, y termina en el instante en el que el programa regresa a la función principal.
- Todos los resultados de un algoritmo se guardan en varios archivos de texto, uno por cada elemento a buscar. Durante cada iteración, se ejecutan varias operaciones que permiten la apertura y escritura en el archivo; estas operaciones son constantes en cada algoritmo, pero no están incluidas en la medición del tiempo.
- El valor de retorno de cada función de búsqueda es una bandera que representa si se encontró el número solicitado o no. El único caso distinto es durante la implementación de la búsqueda lineal con hilos, en donde se utiliza una variable global que tiene el mismo propósito.
- Todos los resultados se registran de la misma forma:
    - Los nombres de archivo tienen la siguiente estructura: x-testing.txt, donde ‘x’ es el número buscado.
    - Para cada uno de los 10 parámetros de N, hay dos líneas:
        - “N: not? in A”, donde N es el parámetro y la palabra ‘not’ opcional que indica si se encontró o no el número.

- “Tiempo real: M s.”, donde M es el tiempo real de la búsqueda del número.

Definición de algoritmo de búsqueda

El propósito de un algoritmo de búsqueda es checar o extraer información de algún tipo de estructura de datos en la que se encuentra almacenado. De acuerdo al tipo de operación, estos algoritmos se clasifican en dos tipos:

- Búsqueda secuencial: Se navega por todo el arreglo de manera secuencial revisando cada elemento. La búsqueda lineal es un ejemplo de este tipo de búsquedas
- Por intervalos: Algoritmos que buscan los datos en información ordenada aproximando el rango en donde podría estar el elemento para reducir el tiempo de búsqueda. Como ejemplo se presenta la búsqueda binaria.

Búsqueda Lineal

Teoría

En informática, la búsqueda lineal o la búsqueda secuencial es un método para encontrar un valor objetivo dentro de una lista. Esta comprueba secuencialmente cada elemento de la lista para el valor objetivo hasta que es encontrado o hasta que todos los elementos hayan sido comparados.

Es importante mencionar que no se necesita que la lista se encuentre ordenada para ejecutar el algoritmo.

Análisis Teórico

Búsqueda lineal es en tiempo el peor, y marca como máximo n comparaciones, donde n es la longitud de la lista. Si la probabilidad de cada elemento para ser buscado es el mismo, entonces la búsqueda lineal tiene una media de n/2 comparaciones, pero esta media puede ser afectado si las probabilidades de búsqueda para cada elemento varían.

Mejor caso

El mejor caso para la búsqueda lineal es cuando el elemento que estamos buscando se encuentra en la primera posición de la lista, otro mejor caso es cuando la lista está vacía. Ya que solo se tendrán que hacer una comparación si es que se encuentra en la primer posición o 0 comparaciones si la lista está vacía. Por lo tanto, la complejidad temporal para el mejor caso es:

$$f(n) = 3$$

Peor caso

El peor caso para la búsqueda lineal es cuando queremos buscar un número que no exista en la lista o que dicho elemento se encuentre en la última posición, ya que el algoritmo tendrá que comparar con todos los elementos antes de llegar a la última posición o darse cuenta de que el elemento no existe. Por lo tanto, su complejidad temporal para el peor caso es:

$$f(n) = 3n + 4$$

Caso medio

Como vimos en el análisis teórico, si calculamos el promedio de cuantas comparaciones tiene que hacer el algoritmo para saber si un elemento dado existe en una lista, obtenemos n/2 comparaciones, siendo n el número de elementos en la lista. Por lo tanto, la complejidad temporal promedio es:

$$f(n) = \frac{n}{2}$$

Registro de tiempo

Para n = 1000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	1000000	0.0027279854 s	NO
2	10393545	1000000	0.0027120113 s	NO
3	1295390003	1000000	0.0033800602 s	NO
4	1360839354	1000000	0.0027360916 s	NO
5	14700764	1000000	0.0030200481 s	NO
6	1493283650	1000000	0.0014710426 s	SI
7	152503	1000000	0.0027420521 s	NO
8	1843349527	1000000	0.0027401447 s	NO
9	187645041	1000000	0.0027990341 s	NO
10	1980098116	1000000	0.0027267933 s	NO
11	2109248666	1000000	0.0027110577 s	NO
12	2147445644	1000000	0.0027449131 s	NO
13	2147470852	1000000	0.0027379990 s	NO
14	214826	1000000	0.0027089119 s	NO
15	3128036	1000000	0.0030200481 s	NO
16	322486	1000000	0.0030100346 s	NO
17	450057883	1000000	0.0028111935 s	NO
18	5000	1000000	0.0027661324 s	NO
19	61396	1000000	0.0027668476 s	NO
20	6337399	1000000	0.0027160645 s	NO

Promedio de tiempo real: **0.002752423 s**

Para n = 2000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	2000000	0.0052800179 s	NO
2	10393545	2000000	0.0054290295 s	NO
3	1295390003	2000000	0.0054008961 s	NO
4	1360839354	2000000	0.0054349899 s	NO
5	14700764	2000000	0.0065989494 s	NO
6	1493283650	2000000	0.0015010834 s	SI
7	152503	2000000	0.0054550171 s	NO
8	1843349527	2000000	0.0054118633 s	NO
9	187645041	2000000	0.0054490566 s	NO
10	1980098116	2000000	0.0054240227 s	NO
11	2109248666	2000000	0.0054781437 s	NO
12	2147445644	2000000	0.0054371357 s	NO
13	2147470852	2000000	0.0053429604 s	NO
14	214826	2000000	0.0054559708 s	NO
15	3128036	2000000	0.0037231445 s	SI
16	322486	2000000	0.0054090023 s	NO
17	450057883	2000000	0.0054719448 s	NO
18	5000	2000000	0.0053508282 s	NO
19	61396	2000000	0.0054388046 s	NO
20	6337399	2000000	0.0054500103 s	NO

Promedio de tiempo real: **0.005197144 s**

Para n = 3000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	3000000	0.0081670284 s	NO
2	10393545	3000000	0.0081720352 s	NO
3	1295390003	3000000	0.0084331036 s	NO
4	1360839354	3000000	0.0081448555 s	NO
5	14700764	3000000	0.0081169605 s	NO
6	1493283650	3000000	0.0014619827 s	SI
7	152503	3000000	0.0090100765 s	NO
8	1843349527	3000000	0.0066080093 s	SI
9	187645041	3000000	0.0081741810 s	NO
10	1980098116	3000000	0.0081770420 s	NO
11	2109248666	3000000	0.0081539154 s	NO
12	2147445644	3000000	0.0109829903 s	NO
13	2147470852	3000000	0.0081961155 s	NO
14	214826	3000000	0.0081651211 s	NO
15	3128036	3000000	0.0037231445 s	SI
16	322486	3000000	0.0081250668 s	NO
17	450057883	3000000	0.0082371235 s	NO
18	5000	3000000	0.0081579685 s	NO
19	61396	3000000	0.0081741810 s	NO
20	6337399	3000000	0.0081079006 s	NO

Promedio de tiempo real: **0.00772444 s**

Para n = 4000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	4000000	0.0108771324 s	NO
2	10393545	4000000	0.0108609200 s	NO
3	1295390003	4000000	0.0108819008 s	NO
4	1360839354	4000000	0.0108587742 s	NO
5	14700764	4000000	0.0139510632 s	NO
6	1493283650	4000000	0.0014629364 s	SI
7	152503	4000000	0.0108730793 s	NO
8	1843349527	4000000	0.0065760612 s	SI
9	187645041	4000000	0.0120141506 s	NO
10	1980098116	4000000	0.0108618736 s	NO
11	2109248666	4000000	0.0111000538 s	NO
12	2147445644	4000000	0.0108819008 s	NO
13	2147470852	4000000	0.0109579563 s	NO
14	214826	4000000	0.0108768940 s	NO
15	3128036	4000000	0.0039260387 s	SI
16	322486	4000000	0.0111169815 s	NO
17	450057883	4000000	0.0109000206 s	NO
18	5000	4000000	0.0109169483 s	NO

19	61396	4000000	0.0108819008 s	NO
20	6337399	4000000	0.0111808777 s	NO

Promedio de tiempo real: **0.010097873 s**

Para n = 5000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	5000000	0.0129120350 s	NO
2	10393545	5000000	0.0127830505 s	NO
3	1295390003	5000000	0.0125179291 s	NO
4	1360839354	5000000	0.0132009983 s	NO
5	14700764	5000000	0.0134611130 s	NO
6	1493283650	5000000	0.0013868809 s	SI
7	152503	5000000	0.0129220486 s	NO
8	1843349527	5000000	0.0064399242 s	SI
9	187645041	5000000	0.0129818916 s	NO
10	1980098116	5000000	0.0130889416 s	NO
11	2109248666	5000000	0.0133619308 s	NO
12	2147445644	5000000	0.0125300884 s	NO
13	2147470852	5000000	0.0132119656 s	NO
14	214826	5000000	0.0129339695 s	NO
15	3128036	5000000	0.0035939217 s	SI
16	322486	5000000	0.0160489082 s	NO
17	450057883	5000000	0.0136799812 s	NO
18	5000	5000000	0.0127439499 s	NO
19	61396	5000000	0.0130281448 s	NO
20	6337399	5000000	0.0131440163 s	NO

Promedio de tiempo real: **0.011798584 s**

Para n = 6000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	6000000	0.0154318810 s	NO
2	10393545	6000000	0.0158200264 s	NO
3	1295390003	6000000	0.0162498951 s	NO
4	1360839354	6000000	0.0155642033 s	NO
5	14700764	6000000	0.0163018703 s	NO
6	1493283650	6000000	0.0013859272 s	SI
7	152503	6000000	0.0152971745 s	NO
8	1843349527	6000000	0.0063729286 s	SI
9	187645041	6000000	0.0157818794 s	NO
10	1980098116	6000000	0.0154170990 s	NO
11	2109248666	6000000	0.0155529976 s	NO
12	2147445644	6000000	0.0158050060 s	NO
13	2147470852	6000000	0.0155010223 s	NO
14	214826	6000000	0.0158431530 s	NO
15	3128036	6000000	0.0037488937 s	SI
16	322486	6000000	0.0165319443 s	NO

17	450057883	6000000	0.0158259869 s	NO
18	5000	6000000	0.0153200626 s	NO
19	61396	6000000	0.0160799026 s	NO
20	6337399	6000000	0.0163390636 s	NO

Promedio de tiempo real: **0.014008546 s**

Para n = 7000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	7000000	0.0171041489 s	NO
2	10393545	7000000	0.0166928768 s	SI
3	1295390003	7000000	0.0188028812 s	NO
4	1360839354	7000000	0.0186600685 s	NO
5	14700764	7000000	0.0168199539 s	NO
6	1493283650	7000000	0.0013959408 s	SI
7	152503	7000000	0.0210759640 s	NO
8	1843349527	7000000	0.0061659813 s	SI
9	187645041	7000000	0.0177059174 s	NO
10	1980098116	7000000	0.0172760487 s	NO
11	2109248666	7000000	0.0162899494 s	SI
12	2147445644	7000000	0.0169298649 s	NO
13	2147470852	7000000	0.0169730186 s	NO
14	214826	7000000	0.0181009769 s	NO
15	3128036	7000000	0.0033349991 s	SI
16	322486	7000000	0.0175158978 s	NO
17	450057883	7000000	0.0181720257 s	NO
18	5000	7000000	0.0182120800 s	NO
19	61396	7000000	0.0169918537 s	NO
20	6337399	7000000	0.0160050392 s	SI

Promedio de tiempo real: **0.015511274 s**

Para n = 8000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	8000000	0.0218999386 s	NO
2	10393545	8000000	0.0181601048 s	SI
3	1295390003	8000000	0.0192790031 s	NO
4	1360839354	8000000	0.0207998753 s	NO
5	14700764	8000000	0.0205688477 s	NO
6	1493283650	8000000	0.0013389587 s	SI
7	152503	8000000	0.0210759640 s	NO
8	1843349527	8000000	0.0090529919 s	SI
9	187645041	8000000	0.0192408562 s	NO
10	1980098116	8000000	0.0193359852 s	NO
11	2109248666	8000000	0.0169630051 s	SI
12	2147445644	8000000	0.0194039345 s	NO
13	2147470852	8000000	0.0216081142 s	NO
14	214826	8000000	0.0196249485 s	NO

15	3128036	8000000	0.0035181046 s	SI
16	322486	8000000	0.0240509510 s	NO
17	450057883	8000000	0.0191419125 s	NO
18	5000	8000000	0.0203309059 s	NO
19	61396	8000000	0.0192589760 s	NO
20	6337399	8000000	0.0155239105 s	SI

Promedio de tiempo real: **0.017508864 s**

Para n = 9000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	9000000	0.0216441154 s	NO
2	10393545	9000000	0.0171999931 s	SI
3	1295390003	9000000	0.0200321674 s	SI
4	1360839354	9000000	0.0217289925 s	NO
5	14700764	9000000	0.0234591961 s	NO
6	1493283650	9000000	0.0013279915 s	SI
7	152503	9000000	0.0239031315 s	NO
8	1843349527	9000000	0.0058119297 s	SI
9	187645041	9000000	0.0226519108 s	NO
10	1980098116	9000000	0.0261850357 s	NO
11	2109248666	9000000	0.0159409046 s	SI
12	2147445644	9000000	0.0216081142 s	NO
13	2147470852	9000000	0.0190019608 s	SI
14	214826	9000000	0.0222561359 s	NO
15	3128036	9000000	0.0035350323 s	SI
16	322486	9000000	0.0250680447 s	SI
17	450057883	9000000	0.0217130184 s	NO
18	5000	9000000	0.0228121281 s	NO
19	61396	9000000	0.0228409767 s	NO
20	6337399	9000000	0.0177350044 s	SI

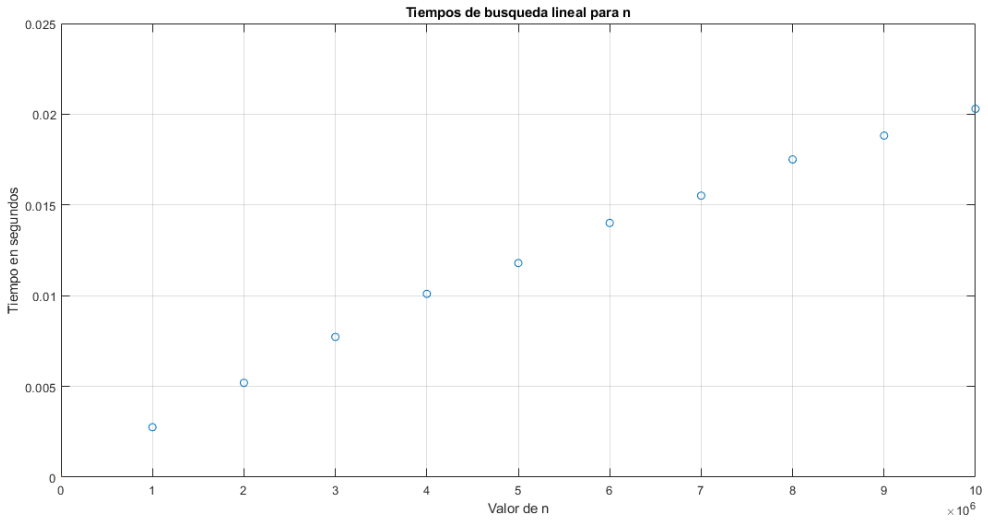
Promedio de tiempo real: **0.018822789 s**

Para n = 10000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	10000000	0.0242350101 s	NO
2	10393545	10000000	0.0167357922 s	SI
3	1295390003	10000000	0.0259799957 s	SI
4	1360839354	10000000	0.0246920586 s	NO
5	14700764	10000000	0.0246090889 s	NO
6	1493283650	10000000	0.0012919903 s	SI
7	152503	10000000	0.0249240398 s	NO
8	1843349527	10000000	0.0058548450 s	SI
9	187645041	10000000	0.0288279057 s	NO
10	1980098116	10000000	0.0275411606 s	NO
11	2109248666	10000000	0.0160119534 s	SI
12	2147445644	10000000	0.0242629051 s	NO

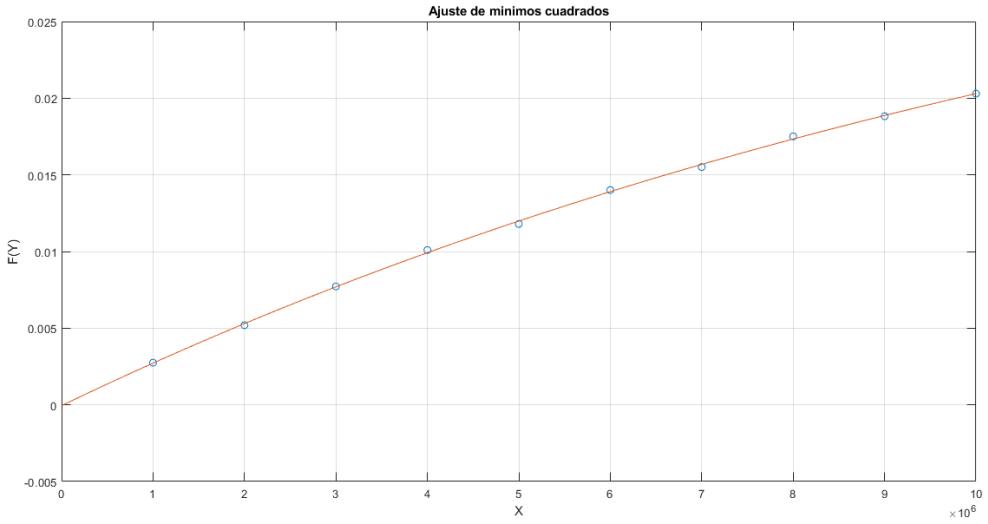
13	2147470852	10000000	0.0191059113 s	SI
14	214826	10000000	0.0241851807 s	NO
15	3128036	10000000	0.0039007664 s	SI
16	322486	10000000	0.0208680630 s	SI
17	450057883	10000000	0.0241689682 s	NO
18	5000	10000000	0.0253510475 s	NO
19	61396	10000000	0.0257239342 s	NO
20	6337399	10000000	0.0176701546 s	SI

Promedio de tiempo real: **0.020297039 s**



Aproximación del comportamiento temporal

$$f(n) = n^3 + n^2 + n - 0.5135$$



Tiempos para el peor caso

n	*Tiempo estimado*
50000000	$2.2x10^{-6}$ s
100000000	$2.4x10^{-6}$ s
500000000	$2.7x10^{-6}$ s
1000000000	$3.5x10^{-5}$ s
5000000000	$2.74x10^{-6}$ s

Idea para mejorar el algoritmo

La mejora al trabajar con hilos es que cada hilo puede realizar una tarea distinta, pero todos al mismo tiempo, sin embargo, la optimización que proponemos es crear únicamente dos hilos, el principal y uno extra, y que cada uno analice una mitad del arreglo al mismo tiempo.

TAMAÑO DE N	TIEMPO REAL (SIN HILOS)	TIEMPO REAL (CON HILOS)	MEJORA
1000000	0.002752423 s		
2000000	0.005197144 s		
3000000	0.007724440 s		
4000000	0.010097873 s		
5000000	0.011798584 s		
6000000	0.014008546 s		
7000000	0.015511274 s		
8000000	0.017508864 s		
9000000	0.018822789 s		
10000000	0.020297039 s		

Registro de tiempo

Búsqueda en árbol binario sin recursión

Teoría

Un árbol binario de búsqueda (Binary Search Tree) es una variante del árbol binario en donde cada nodo almacena un entero y dos subárboles, el subárbol derecho contiene valores más grandes que el del nodo actual mientras que el izquierdo guarda valores más pequeños. Así como el nombre indica, el recorrido en el árbol se realiza con una búsqueda binaria.

Todas las pruebas con el árbol binario de búsqueda fueron realizadas sobre árboles no balanceados, i.e., la diferencia de alturas entre subárboles puede ser mayor a uno.

Análisis Teórico

Cada vez que se visita un nodo del árbol, se realizan tres comparaciones para verificar cuál es el siguiente paso. Si el elemento en el nodo es:

- Mayor al número buscado, la búsqueda continúa en el subárbol derecho
- Menor al número buscado, la búsqueda continúa en el subárbol izquierdo
- Igual al número buscado, el algoritmo se detiene indicando que el elemento se encuentra en el árbol.

Estas tres comparaciones serán las instrucciones elementales para el análisis.

Mejor caso

Si el número buscado está almacenado en el primer nodo, es decir, se encuentra en la raíz del árbol, se realizan las tres comparaciones mencionadas arriba, y finalmente se termina la búsqueda.

$$f(h_l, h_r) = 2$$

Peor caso

Cuando el arreglo de números a buscar está ordenado de manera decreciente o no decreciente, el procedimiento de búsqueda en el árbol se convierte en un proceso secuencial, y el algoritmo se comporta de manera muy similar a la búsqueda lineal.

Expresado en función de la altura, dado que cada número en el árbol es un nivel distinto,

$$f(h_l, h_r) = 2h_l + h_r$$

Si el arreglo cumple la condición mencionada arriba, dado que el árbol sólo crece hacia un lado, uno de los términos de la expresión siempre es cero. Esta anotación nos permite analizar para ambos casos propuestos.

$$f(h_l, h_r) = \begin{cases} h_r, & A\ es\ no\ decreciente \\ 2h_l, & A\ es\ no\ creciente \end{cases}$$

Entonces, observamos que el peor caso (en esta implementación) se da cuando el arreglo A es no creciente.

Caso medio

Cada vez que el elemento del nodo no es el número deseado, el algoritmo baja un nivel en el árbol, y puede continuar por el subárbol izquierdo o el derecho. También existe la probabilidad de que el número deseado se encuentre en el siguiente nodo. Entonces se puede argumentar que el caso medio está en función de la altura del árbol, y concretamente, la altura de los subárboles derecho e izquierdo.

$$f(h_l, h_r) = \frac{1}{3}(h_l + h_r + 2)$$

Registro de tiempo

Para n = 1000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	1000000	0.0000028610 s	NO
2	10393545	1000000	0.0000081062 s	NO
3	1295390003	1000000	0.0000050068 s	NO
4	1360839354	1000000	0.0000050068 s	NO
5	14700764	1000000	0.0000038147 s	NO
6	1493283650	1000000	0.0000050068 s	SÍ
7	152503	1000000	0.0000040531 s	NO
8	1843349527	1000000	0.0000059605 s	NO
9	187645041	1000000	0.0000047684 s	NO
10	1980098116	1000000	0.0000050068 s	NO
11	2109248666	1000000	0.0000050068 s	NO

12	2147445644	1000000	0.0000040531 s	NO
13	2147470852	1000000	0.0000030994 s	NO
14	214826	1000000	0.0000050068 s	NO
15	3128036	1000000	0.0000059605 s	NO
16	322486	1000000	0.0000059605 s	NO
17	450057883	1000000	0.0000050068 s	NO
18	5000	1000000	0.0000040531 s	NO
19	61396	1000000	0.0000050068 s	NO
20	6337399	1000000	0.0000040531 s	NO

Promedio de tiempo real: **0.0000048399 s**

Para n = 2000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	2000000	0.0000040531 s	NO
2	10393545	2000000	0.0000140667 s	NO
3	1295390003	2000000	0.0000050068 s	NO
4	1360839354	2000000	0.0000059605 s	NO
5	14700764	2000000	0.0000038147 s	NO
6	1493283650	2000000	0.0000050068 s	SÍ
7	152503	2000000	0.0000040531 s	NO
8	1843349527	2000000	0.0000050068 s	NO
9	187645041	2000000	0.0000040531 s	NO
10	1980098116	2000000	0.0000050068 s	NO
11	2109248666	2000000	0.0000061989 s	NO
12	2147445644	2000000	0.0000040531 s	NO
13	2147470852	2000000	0.0000038147 s	NO
14	214826	2000000	0.0000059605 s	NO
15	3128036	2000000	0.0000059605 s	SÍ
16	322486	2000000	0.0000071526 s	NO
17	450057883	2000000	0.0000059605 s	NO
18	5000	2000000	0.0000040531 s	NO
19	61396	2000000	0.0000059605 s	NO
20	6337399	2000000	0.0000059605 s	NO

Promedio de tiempo real: **0.0000055552 s**

Para n = 3000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	3000000	0.0000050068 s	NO
2	10393545	3000000	0.0000059605 s	NO
3	1295390003	3000000	0.0000059605 s	NO
4	1360839354	3000000	0.0000081062 s	NO
5	14700764	3000000	0.0000038147 s	NO
6	1493283650	3000000	0.0000050068 s	SÍ

7	152503	3000000	0.0000050068 s	NO
8	1843349527	3000000	0.0000050068 s	SÍ
9	187645041	3000000	0.0000059605 s	NO
10	1980098116	3000000	0.0000050068 s	NO
11	2109248666	3000000	0.0000059605 s	NO
12	2147445644	3000000	0.0000090599 s	NO
13	2147470852	3000000	0.0000050068 s	NO
14	214826	3000000	0.0000050068 s	NO
15	3128036	3000000	0.0000050068 s	SÍ
16	322486	3000000	0.0000061989 s	NO
17	450057883	3000000	0.0000050068 s	NO
18	5000	3000000	0.0000059605 s	NO
19	61396	3000000	0.0000050068 s	NO
20	6337399	3000000	0.0000059605 s	NO

Promedio de tiempo real: **0.0000056505 s**

Para n = 4000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	4000000	0.0000040531 s	NO
2	10393545	4000000	0.0000050068 s	NO
3	1295390003	4000000	0.0000050068 s	NO
4	1360839354	4000000	0.0000050068 s	NO
5	14700764	4000000	0.0000038147 s	NO
6	1493283650	4000000	0.0000061989 s	SÍ
7	152503	4000000	0.0000050068 s	NO
8	1843349527	4000000	0.0000040531 s	SÍ
9	187645041	4000000	0.0000050068 s	NO
10	1980098116	4000000	0.0000040531 s	NO
11	2109248666	4000000	0.0000050068 s	NO
12	2147445644	4000000	0.0000040531 s	NO
13	2147470852	4000000	0.0000040531 s	NO
14	214826	4000000	0.0000040531 s	NO
15	3128036	4000000	0.0000050068 s	SÍ
16	322486	4000000	0.0000059605 s	NO
17	450057883	4000000	0.0000050068 s	NO
18	5000	4000000	0.0000040531 s	NO
19	61396	4000000	0.0000050068 s	NO
20	6337399	4000000	0.0000040531 s	NO

Promedio de tiempo real: **0.0000046730 s**

Para n = 5000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	5000000	0.0000081062 s	NO

2	10393545	5000000	0.0000090599 s	NO
3	1295390003	5000000	0.0000109673 s	NO
4	1360839354	5000000	0.0000121593 s	NO
5	14700764	5000000	0.0000081062 s	NO
6	1493283650	5000000	0.0000090599 s	YES
7	152503	5000000	0.0000088215 s	NO
8	1843349527	5000000	0.0000100136 s	YES
9	187645041	5000000	0.0000088215 s	NO
10	1980098116	5000000	0.0000090599 s	NO
11	2109248666	5000000	0.0000097752 s	NO
12	2147445644	5000000	0.0000081062 s	NO
13	2147470852	5000000	0.0000078678 s	NO
14	214826	5000000	0.0000100136 s	NO
15	3128036	5000000	0.0000109673 s	YES
16	322486	5000000	0.0000109673 s	YES
17	450057883	5000000	0.0000100136 s	NO
18	5000	5000000	0.0000140667 s	NO
19	61396	5000000	0.0000100136 s	NO
20	6337399	5000000	0.0000100136 s	NO

Promedio de tiempo real: **0.0000097990 s**

Para n = 6000000

# Test	***Número por buscar***	***Tamaño de n***	***Tiempo real***	***Encontrado***
1	0	6000000	0.0000078678 s	NO
2	10393545	6000000	0.0000259876 s	NO
3	1295390003	6000000	0.0000100136 s	NO
4	1360839354	6000000	0.0000119209 s	NO
5	14700764	6000000	0.0000078678 s	NO
6	1493283650	6000000	0.0000071526 s	NO
7	152503	6000000	0.0000078678 s	NO
8	1843349527	6000000	0.0000259876 s	SÍ
9	187645041	6000000	0.0000090599 s	NO
10	1980098116	6000000	0.0000078678 s	NO
11	2109248666	6000000	0.0000100136 s	NO
12	2147445644	6000000	0.0000069141 s	NO
13	2147470852	6000000	0.0000081062 s	NO
14	214826	6000000	0.0000200272 s	NO
15	3128036	6000000	0.0000100136 s	SÍ
16	322486	6000000	0.0000109673 s	NO
17	450057883	6000000	0.0000090599 s	NO
18	5000	6000000	0.0000071526 s	NO
19	61396	6000000	0.0000071526 s	NO
20	6337399	6000000	0.0000119209 s	NO

Promedio de tiempo real: **0.0000111341 s**

Para n = 7000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	7000000	0.0000090599 s	NO
2	10393545	7000000	0.0000090599 s	SÍ
3	1295390003	7000000	0.0000100136 s	NO
4	1360839354	7000000	0.0000109673 s	NO
5	14700764	7000000	0.0000090599 s	NO
6	1493283650	7000000	0.0000078678 s	SÍ
7	152503	7000000	0.0000088215 s	NO
8	1843349527	7000000	0.0000081062 s	SÍ
9	187645041	7000000	0.0000119209 s	NO
10	1980098116	7000000	0.0000081062 s	NO
11	2109248666	7000000	0.0000088215 s	SÍ
12	2147445644	7000000	0.0000069141 s	NO
13	2147470852	7000000	0.0000078678 s	NO
14	214826	7000000	0.0000100136 s	NO
15	3128036	7000000	0.0000090599 s	SÍ
16	322486	7000000	0.0000200272 s	NO
17	450057883	7000000	0.0000119209 s	NO
18	5000	7000000	0.0000090599 s	NO
19	61396	7000000	0.0000090599 s	NO
20	6337399	7000000	0.0000150204 s	SÍ

Promedio de tiempo real: **0.0000100374 s**

Para n = 8000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	8000000	0.0000090599 s	NO
2	10393545	8000000	0.0000088215 s	SÍ
3	1295390003	8000000	0.0000100136 s	NO
4	1360839354	8000000	0.0000097752 s	NO
5	14700764	8000000	0.0000090599 s	NO
6	1493283650	8000000	0.0000090599 s	SÍ
7	152503	8000000	0.0000081062 s	NO
8	1843349527	8000000	0.0000090599 s	SÍ
9	187645041	8000000	0.0000090599 s	NO
10	1980098116	8000000	0.0000078678 s	NO
11	2109248666	8000000	0.0000100136 s	SÍ
12	2147445644	8000000	0.0000081062 s	NO
13	2147470852	8000000	0.0000090599 s	SÍ
14	214826	8000000	0.0000100136 s	NO
15	3128036	8000000	0.0000090599 s	SÍ
16	322486	8000000	0.0000109673 s	NO
17	450057883	8000000	0.0000100136 s	NO
18	5000	8000000	0.0000078678 s	NO

19	61396	8000000	0.0000090599 s	NO
20	6337399	8000000	0.0000090599 s	SÍ

Promedio de tiempo real: **0.0000091553 s**

Para n = 9000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	9000000	0.0000069141 s	NO
2	10393545	9000000	0.0000100136 s	SÍ
3	1295390003	9000000	0.0000100136 s	SÍ
4	1360839354	9000000	0.0000100136 s	NO
5	14700764	9000000	0.0000109673 s	NO
6	1493283650	9000000	0.0000081062 s	SÍ
7	152503	9000000	0.0000078678 s	NO
8	1843349527	9000000	0.0000088215 s	SÍ
9	187645041	9000000	0.0000097752 s	NO
10	1980098116	9000000	0.0000078678 s	NO
11	2109248666	9000000	0.0000090599 s	SÍ
12	2147445644	9000000	0.0000078678 s	NO
13	2147470852	9000000	0.0000140667 s	SÍ
14	214826	9000000	0.0000088215 s	NO
15	3128036	9000000	0.0000090599 s	SÍ
16	322486	9000000	0.0000119209 s	SÍ
17	450057883	9000000	0.0000100136 s	NO
18	5000	9000000	0.0000078678 s	NO
19	61396	9000000	0.0000100136 s	NO
20	6337399	9000000	0.0000100136 s	SÍ

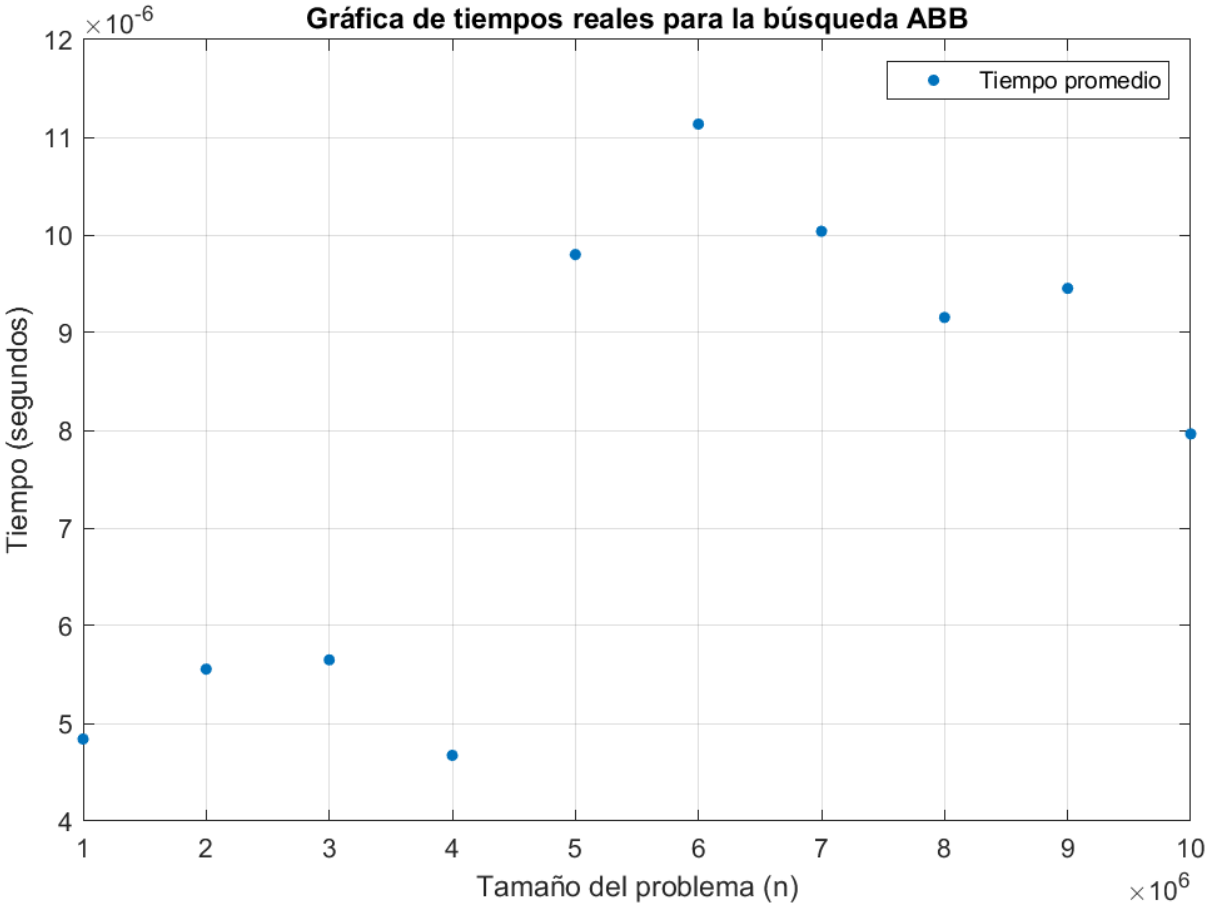
Promedio de tiempo real: **0.0000094533 s**

Para n = 10000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	10000000	0.0000069141 s	NO
2	10393545	10000000	0.0000059605 s	SÍ
3	1295390003	10000000	0.0000090599 s	SÍ
4	1360839354	10000000	0.0000100136 s	NO
5	14700764	10000000	0.0000059605 s	NO
6	1493283650	10000000	0.0000069141 s	SÍ
7	152503	10000000	0.0000071526 s	NO
8	1843349527	10000000	0.0000109673 s	SÍ
9	187645041	10000000	0.0000100136 s	NO
10	1980098116	10000000	0.0000078678 s	NO
11	2109248666	10000000	0.0000078678 s	SÍ
12	2147445644	10000000	0.0000069141 s	NO

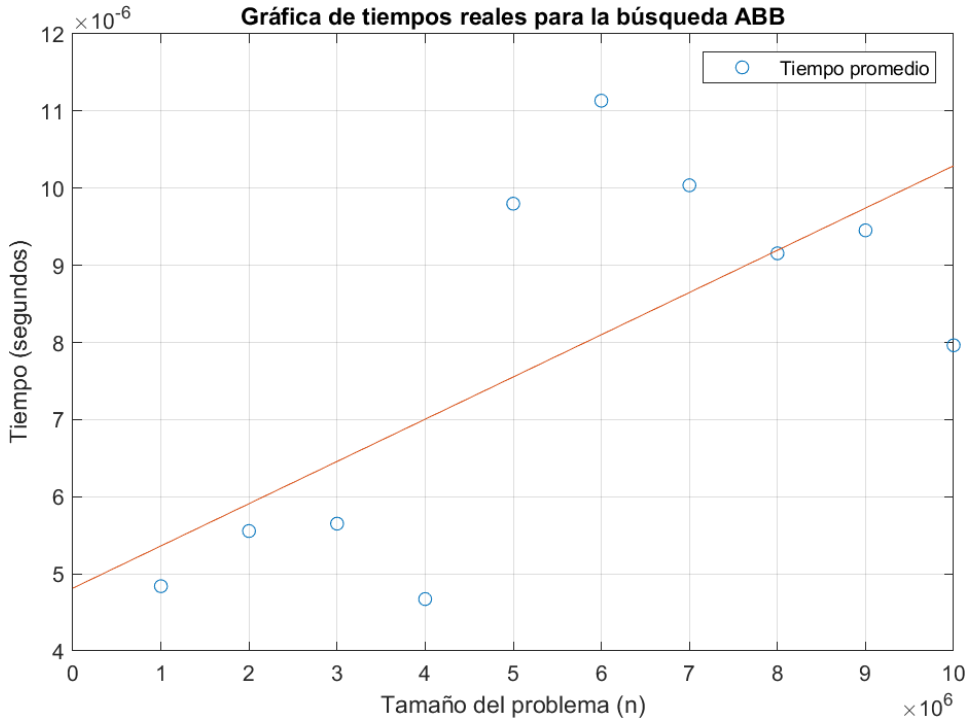
13	2147470852	10000000	0.0000059605 s	SÍ
14	214826	10000000	0.0000069141 s	NO
15	3128036	10000000	0.0000069141 s	SÍ
16	322486	10000000	0.0000119209 s	SÍ
17	450057883	10000000	0.0000090599 s	NO
18	5000	10000000	0.0000069141 s	NO
19	61396	10000000	0.0000100136 s	NO
20	6337399	10000000	0.0000059605 s	SÍ

Promedio de tiempo real: **0.0000079632 s**



Aproximación del comportamiento temporal

$$P(x) = 5.46 \times 10^{-13}\, n + 4.81 \times 10^{-6}$$



Si bien podemos observar en la imagen anterior que los puntos de tiempo promedio están dispersos alrededor de la recta, es importante notar que la diferencia entre los valores de tiempo es realmente despreciable (la diferencia entre el valor más alto y el más pequeño es de tan sólo $6.5x10^{-6}$ segundos, es decir, en los microsegundos).

Para el peor caso, considerando a N como 10.000.000, tenemos que la constante multiplicativa de cada operación condicional (las instrucciones elementales) es de $7.9632x10^{-13}$

Regresando al análisis, con el modelo matemático, podemos obtener los valores de tiempo cuando N incrementa:

n	***Tiempo estimado***
50000000	$3x10^{-5}$ s
100000000	$5x10^{-5}$ s
500000000	$2.7x10^{-4}$ s
1000000000	$5.5x10^{-4}$ s
5000000000	$2.74x10^{-3}$ s

Concluimos que la búsqueda ABB es un algoritmo muy eficiente incluso con valores muy grandes. Por esa razón, la implementación de hilos no es necesaria: no mejora el rendimiento del algoritmo. Esta situación es prevalente en el resto de los algoritmos de búsqueda en intervalos.

Búsqueda Binaria

Teoría

En ciencias de la computación y matemáticas, la búsqueda binaria, también conocida como búsqueda de intervalo medio o búsqueda logarítmica, es un algoritmo de búsqueda que encuentra la posición de un valor en un array ordenado. Compara el valor con el elemento en el medio del

array, si no son iguales, la mitad en la cual el valor no puede estar es eliminada y la búsqueda continúa en la mitad restante hasta que el valor se encuentre o evaluemos el último elemento y nos demos cuenta de que no existe en el array.

Análisis Teórico

Mejor caso

El mejor caso para este algoritmo es constante, ya que puede existir el caso en que el elemento que estamos buscando se encuentre en medio del arreglo y por lo tanto se realice una sola comparación y el arreglo termine.

$$O(1)$$

Peor caso

El peor caso para este algoritmo es cuando el elemento no se encuentre, o se encuentre a la primer o última posición en el arreglo. Ya que tendrá que hacer log(n) (logaritmo base 2) comparaciones hasta llegar a los bordes del arreglo o darse cuenta de que el elemento que buscamos no se encuentre.

$$O(\log(n))$$

Caso medio

Para obtener el caso promedio del algoritmo es necesario obtener la probabilidad de cada elemento en el arreglo.

Registro de tiempo

Para n = 1000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	1000000	0.0000030994 s	SI
2	10393545	1000000	0.0000038147 s	SI
3	1295390003	1000000	0.0000028610 s	NO
4	1360839354	1000000	0.0000030994 s	NO
5	14700764	1000000	0.0000038147 s	NO
6	1493283650	1000000	0.0000030994 s	NO
7	152503	1000000	0.0000030994 s	NO
8	1843349527	1000000	0.0000028610 s	NO
9	187645041	1000000	0.0000040531 s	NO
10	1980098116	1000000	0.0000028610 s	NO
11	2109248666	1000000	0.0000030994 s	NO
12	2147445644	1000000	0.0000038147 s	NO
13	2147470852	1000000	0.0000030994 s	NO
14	214826	1000000	0.0000030994 s	NO
15	3128036	1000000	0.0000040531 s	SI
16	322486	1000000	0.0000040531 s	SI
17	450057883	1000000	0.0000030994 s	NO
18	5000	1000000	0.0000028610 s	NO
19	61396	1000000	0.0000040531 s	NO
20	6337399	1000000	0.0000040531 s	SI

Promedio de tiempo real: **3.39744E-06 s**

Para n = 2000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	2000000	0.0000040531 s	SI
2	10393545	2000000	0.0000030994 s	SI
3	1295390003	2000000	0.0000030994 s	NO
4	1360839354	2000000	0.0000071526 s	NO
5	14700764	2000000	0.0000038147 s	NO
6	1493283650	2000000	0.0000040531 s	NO
7	152503	2000000	0.0000030994 s	NO
8	1843349527	2000000	0.0000030994 s	NO
9	187645041	2000000	0.0000040531 s	NO
10	1980098116	2000000	0.0000028610 s	NO
11	2109248666	2000000	0.0000028610 s	NO
12	2147445644	2000000	0.0000040531 s	NO
13	2147470852	2000000	0.0000028610 s	NO
14	214826	2000000	0.0000028610 s	NO
15	3128036	2000000	0.0000090599 s	SI
16	322486	2000000	0.0000040531 s	SI
17	450057883	2000000	0.0000038147 s	NO
18	5000	2000000	0.0000040531 s	NO
19	61396	2000000	0.0000030994 s	NO
20	6337399	2000000	0.0000040531 s	SI

Promedio de tiempo real: **3.95773E-06 s**

Para n = 3000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	3000000	0.0000040531 s	SI
2	10393545	3000000	0.0000040531 s	SI
3	1295390003	3000000	0.0000040531 s	NO
4	1360839354	3000000	0.0000038147 s	NO
5	14700764	3000000	0.0000050068 s	NO
6	1493283650	3000000	0.0000040531 s	NO
7	152503	3000000	0.0000038147 s	NO
8	1843349527	3000000	0.0000040531 s	NO
9	187645041	3000000	0.0000050068 s	NO
10	1980098116	3000000	0.0000028610 s	NO
11	2109248666	3000000	0.0000040531 s	NO
12	2147445644	3000000	0.0000038147 s	NO
13	2147470852	3000000	0.0000040531 s	NO
14	214826	3000000	0.0000050068 s	NO
15	3128036	3000000	0.0000050068 s	SI
16	322486	3000000	0.0000059605 s	SI
17	450057883	3000000	0.0000038147 s	NO
18	5000	3000000	0.0000040531 s	NO

19	61396	3000000	0.0000040531 s	NO
20	6337399	3000000	0.0000050068 s	SI

Promedio de tiempo real: **4.27961E-06 s**

Para n = 4000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	4000000	0.0000028610 s	SI
2	10393545	4000000	0.0000038147 s	SI
3	1295390003	4000000	0.0000030994 s	NO
4	1360839354	4000000	0.0000028610 s	NO
5	14700764	4000000	0.0000040531 s	NO
6	1493283650	4000000	0.0000030994 s	NO
7	152503	4000000	0.0000028610 s	NO
8	1843349527	4000000	0.0000028610 s	NO
9	187645041	4000000	0.0000028610 s	NO
10	1980098116	4000000	0.0000038147 s	NO
11	2109248666	4000000	0.0000030994 s	NO
12	2147445644	4000000	0.0000040531 s	NO
13	2147470852	4000000	0.0000030994 s	NO
14	214826	4000000	0.0000030994 s	NO
15	3128036	4000000	0.0000040531 s	SI
16	322486	4000000	0.0000050068 s	SI
17	450057883	4000000	0.0000040531 s	NO
18	5000	4000000	0.0000030994 s	NO
19	61396	4000000	0.0000030994 s	NO
20	6337399	4000000	0.0000030994 s	SI

Promedio de tiempo real: **3.39744E-06 s**

Para n = 5000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	5000000	0.0000040531 s	SI
2	10393545	5000000	0.0000040531 s	SI
3	1295390003	5000000	0.0000040531 s	NO
4	1360839354	5000000	0.0000030994 s	NO
5	14700764	5000000	0.0000040531 s	NO
6	1493283650	5000000	0.0000030994 s	NO
7	152503	5000000	0.0000030994 s	NO
8	1843349527	5000000	0.0000030994 s	NO
9	187645041	5000000	0.0000040531 s	NO
10	1980098116	5000000	0.0000030994 s	NO
11	2109248666	5000000	0.0000030994 s	NO
12	2147445644	5000000	0.0000059605 s	NO
13	2147470852	5000000	0.0000030994 s	NO
14	214826	5000000	0.0000028610 s	NO
15	3128036	5000000	0.0000030994 s	SI
16	322486	5000000	0.0000059605 s	SI

17	450057883	5000000	0.0000038147 s	NO
18	5000	5000000	0.0000028610 s	NO
19	61396	5000000	0.0000038147 s	NO
20	6337399	5000000	0.0000040531 s	SI

Promedio de tiempo real: **3.71931E-06 s**

Para n = 6000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	6000000	0.0000030994 s	SI
2	10393545	6000000	0.0000028610 s	SI
3	1295390003	6000000	0.0000030994 s	NO
4	1360839354	6000000	0.0000030994 s	NO
5	14700764	6000000	0.0000040531 s	NO
6	1493283650	6000000	0.0000028610 s	NO
7	152503	6000000	0.0000038147 s	NO
8	1843349527	6000000	0.0000030994 s	NO
9	187645041	6000000	0.0000119209 s	NO
10	1980098116	6000000	0.0000040531 s	NO
11	2109248666	6000000	0.0000028610 s	NO
12	2147445644	6000000	0.0000081062 s	NO
13	2147470852	6000000	0.0000030994 s	NO
14	214826	6000000	0.0000030994 s	NO
15	3128036	6000000	0.0000069141 s	SI
16	322486	6000000	0.0000059605 s	SI
17	450057883	6000000	0.0000050068 s	NO
18	5000	6000000	0.0000030994 s	NO
19	61396	6000000	0.0000038147 s	NO
20	6337399	6000000	0.0000059605 s	SI

Promedio de tiempo real: **4.49417E-06 s**

Para n = 7000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	7000000	0.0000030994 s	SI
2	10393545	7000000	0.0000030994 s	SI
3	1295390003	7000000	0.0000038147 s	SI
4	1360839354	7000000	0.0000040531 s	NO
5	14700764	7000000	0.0000040531 s	NO
6	1493283650	7000000	0.0000030994 s	SI
7	152503	7000000	0.0000040531 s	NO
8	1843349527	7000000	0.0000028610 s	NO
9	187645041	7000000	0.0000040531 s	NO
10	1980098116	7000000	0.0000028610 s	NO
11	2109248666	7000000	0.0000030994 s	NO
12	2147445644	7000000	0.0000030994 s	NO
13	2147470852	7000000	0.0000028610 s	NO
14	214826	7000000	0.0000030994 s	NO

15	3128036	7000000	0.0000040531 s	SI
16	322486	7000000	0.0000050068 s	SI
17	450057883	7000000	0.0000040531 s	NO
18	5000	7000000	0.0000030994 s	NO
19	61396	7000000	0.0000040531 s	NO
20	6337399	7000000	0.0000040531 s	SI

Promedio de tiempo real: **3.57626E-06 s**

Para n = 8000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	8000000	0.0000028610 s	SI
2	10393545	8000000	0.0000040531 s	SI
3	1295390003	8000000	0.0000050068 s	SI
4	1360839354	8000000	0.0000040531 s	NO
5	14700764	8000000	0.0000040531 s	NO
6	1493283650	8000000	0.0000040531 s	SI
7	152503	8000000	0.0000028610 s	NO
8	1843349527	8000000	0.0000038147 s	NO
9	187645041	8000000	0.0000040531 s	NO
10	1980098116	8000000	0.0000030994 s	NO
11	2109248666	8000000	0.0000028610 s	NO
12	2147445644	8000000	0.0000059605 s	NO
13	2147470852	8000000	0.0000030994 s	NO
14	214826	8000000	0.0000078678 s	NO
15	3128036	8000000	0.0000040531 s	SI
16	322486	8000000	0.0000061989 s	SI
17	450057883	8000000	0.0000050068 s	NO
18	5000	8000000	0.0000069141 s	NO
19	61396	8000000	0.0000030994 s	NO
20	6337399	8000000	0.0000040531 s	SI

Promedio de tiempo real: **4.35113E-06 s**

Para n = 9000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	9000000	0.0000030994 s	SI
2	10393545	9000000	0.0000028610 s	SI
3	1295390003	9000000	0.0000038147 s	SI
4	1360839354	9000000	0.0000038147 s	NO
5	14700764	9000000	0.0000030994 s	NO
6	1493283650	9000000	0.0000050068 s	SI
7	152503	9000000	0.0000030994 s	NO
8	1843349527	9000000	0.0000040531 s	SI
9	187645041	9000000	0.0000030994 s	NO
10	1980098116	9000000	0.0000028610 s	NO
11	2109248666	9000000	0.0000030994 s	NO
12	2147445644	9000000	0.0000040531 s	NO

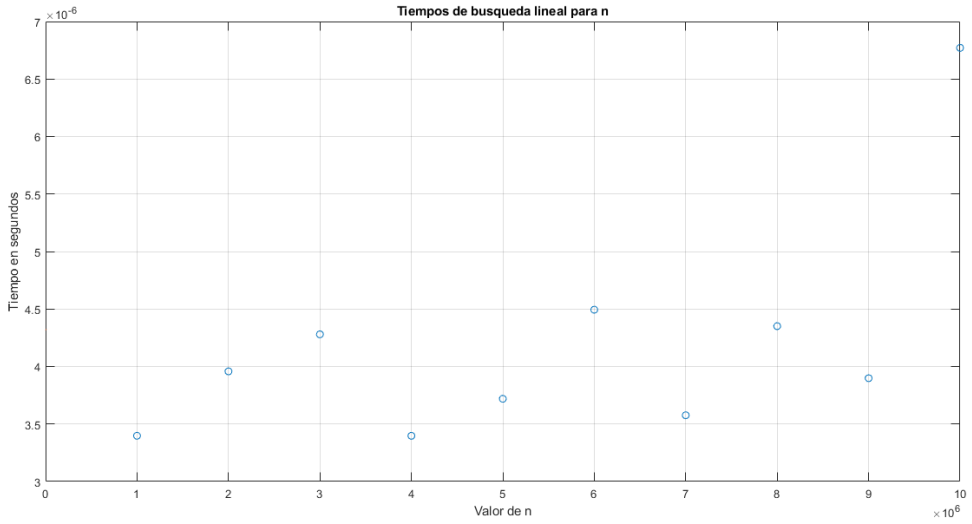
13	2147470852	9000000	0.0000030994 s	NO
14	214826	9000000	0.0000028610 s	NO
15	3128036	9000000	0.0000050068 s	SI
16	322486	9000000	0.0000059605 s	SI
17	450057883	9000000	0.0000081062 s	NO
18	5000	9000000	0.0000028610 s	NO
19	61396	9000000	0.0000040531 s	NO
20	6337399	9000000	0.0000040531 s	SI

Promedio de tiempo real: **3.89813E-06 s**

Para n = 10000000

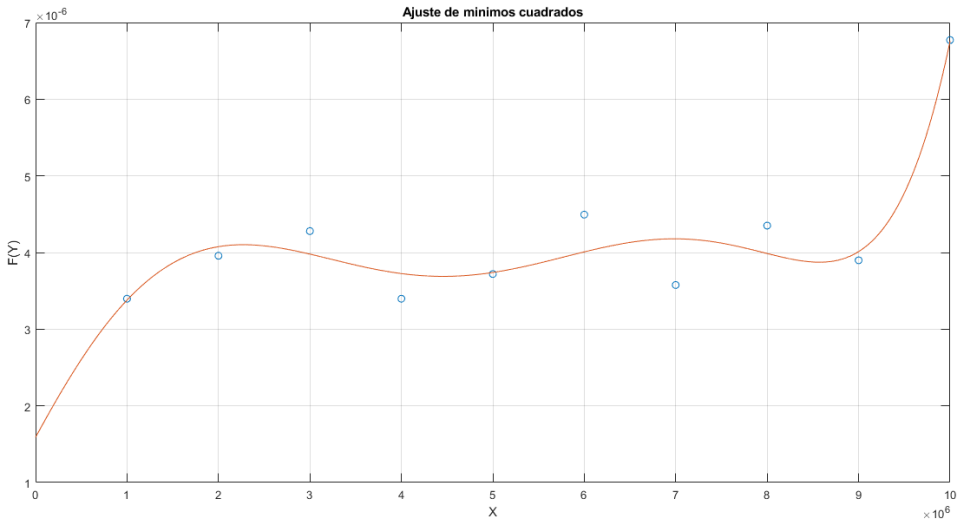
# Test	***Número por buscar***	***Tamaño de n***	***Tiempo real***	***Encontrado***
1	0	10000000	0.0000050068 s	SI
2	10393545	10000000	0.0000038147 s	SI
3	1295390003	10000000	0.0000309944 s	SI
4	1360839354	10000000	0.0000050068 s	NO
5	14700764	10000000	0.0000090599 s	NO
6	1493283650	10000000	0.0000059605 s	SI
7	152503	10000000	0.0000038147 s	NO
8	1843349527	10000000	0.0000038147 s	SI
9	187645041	10000000	0.0000188351 s	NO
10	1980098116	10000000	0.0000038147 s	NO
11	2109248666	10000000	0.0000040531 s	SI
12	2147445644	10000000	0.0000040531 s	NO
13	2147470852	10000000	0.0000050068 s	SI
14	214826	10000000	0.0000050068 s	NO
15	3128036	10000000	0.0000050068 s	SI
16	322486	10000000	0.0000059605 s	SI
17	450057883	10000000	0.0000050068 s	NO
18	5000	10000000	0.0000030994 s	NO
19	61396	10000000	0.0000040531 s	NO
20	6337399	10000000	0.0000040531 s	SI

Promedio de tiempo real**: 6.77109E-06 s**



Aproximación del comportamiento temporal

$$F(n) = n^6 - n^5 + n^4 - n^3 - n^2 + n + 0.1590$$



Tiempos para el peor caso

n	***Tiempo estimado***
50000000	$2x10^{-5}$ s
100000000	$1.85x10^{-6}$ s
500000000	$2.3x10^{-6}$ s
1000000000	$2.61x10^{-5}$ s
5000000000	$2.74x10^{-6}$ s

Idea para mejorar el algoritmo

Como la búsqueda binaria funciona únicamente en arreglos de números que estén ordenados, al momento de tomar una decisión de buscar un elemento hacia la izquierda o hacia la derecha, se va descartando la mitad del arreglo, y si se hace en forma paralela utilizando dos hilos, cada uno

buscando en una mitad del arreglo, seria exactamente la misma complejidad, por lo tanto no hay mejor del algoritmo mediante hilos.

Registro de tiempo

Búsqueda Exponencial

Teoría

El algoritmo de búsqueda exponencial, fue creado por *Jon Bentley* y *Andrew Chi-Chih Yao* en el año de 1976.

Una búsqueda exponencial es una combinación de dos métodos:

- Encontrar el rango en el que existe el elemento.
- Realizar una búsqueda binaria en ese rango.

Entonces para comenzar una búsqueda, primero encontramos el rango. Hacemos esto comprobando primero si el elemento deseado está en la primera posición. Si no es así, probamos con un tamaño de arreglo de 2, luego 4, luego 6 ... y así sucesivamente. Si el último elemento del arreglo parcial no es mayor que el elemento, realizamos una búsqueda binaria.

Análisis Teórico

Mejor caso

El mejor caso se presenta cuando el elemento a buscar está en la primera posición del arreglo.

Entonces la función temporal es f(n) = 1

Peor caso

El peor caso es cuando el elemento a buscar no se encuentra en el arreglo, entonces para calcular el rango por medio de potencias de dos, se hacen $6*((\log_2 n) + 1)$ veces donde se consideraron comparaciones, asignaciones y operaciones aritméticas. Para buscar el elemento se hacen $10*((\log_2 n) - 1)$, de igual forma solo considerando comparaciones, asignaciones y operaciones aritméticas.

Por tanto, la función queda como $f(n) = 1 + 6*((\log_2 n) + 1) + 10*((\log_2 n) - 1)$

Caso medio

La probabilidad que se le asigna será 1/n, puesto que puede estar el elemento buscado en cualquier posición, entonces la función queda como f(n) $1/n\ (1 + 6*((\log_2 n) + 1) + 10*((\log_2 n) - 1))$

Registro de tiempo

Para n = 1000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	1000000	0.0000019073 s	SI
2	10393545	1000000	0.0000040531 s	SI
3	1295390003	1000000	0.0000028610 s	NO
4	1360839354	1000000	0.0000028610 s	NO

5	14700764	1000000	0.0000040531 s	NO
6	1493283650	1000000	0.0000030994 s	NO
7	152503	1000000	0.0000028610 s	NO
8	1843349527	1000000	0.0000030994 s	NO
9	187645041	1000000	0.0000050068 s	NO
10	1980098116	1000000	0.0000028610 s	NO
11	2109248666	1000000	0.0000030994 s	NO
12	2147445644	1000000	0.0000028610 s	NO
13	2147470852	1000000	0.0000028610 s	NO
14	214826	1000000	0.0000030994 s	NO
15	3128036	1000000	0.0000030994 s	SI
16	322486	1000000	0.0000040531 s	SI
17	450057883	1000000	0.0000030994 s	NO
18	5000	1000000	0.0000030994 s	NO
19	61396	1000000	0.0000078678 s	NO
20	6337399	1000000	0.0000030994 s	SI

Promedio de tiempo real**: 0.0000150203 s**

Para n = 2000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	2000000	0.0000021458 s	SI
2	10393545	2000000	0.0000030994 s	SI
3	1295390003	2000000	0.0000030994 s	NO
4	1360839354	2000000	0.0000030994 s	NO
5	14700764	2000000	0.0000038147 s	NO
6	1493283650	2000000	0.0000021458 s	NO
7	152503	2000000	0.0000030994 s	NO
8	1843349527	2000000	0.0000030994 s	NO
9	187645041	2000000	0.0000038147 s	NO
10	1980098116	2000000	0.0000030994 s	NO
11	2109248666	2000000	0.0000019073 s	NO
12	2147445644	2000000	0.0000030994 s	NO
13	2147470852	2000000	0.0000030994 s	NO
14	214826	2000000	0.0000028610 s	NO
15	3128036	2000000	0.0000028610 s	SI
16	322486	2000000	0.0000050068 s	SI
17	450057883	2000000	0.0000021458 s	NO
18	5000	2000000	0.0000019073 s	NO
19	61396	2000000	0.0000030994 s	NO
20	6337399	2000000	0.0000040531 s	SI

Promedio de tiempo real**: 0.0000138282 s**

Para n = 3000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	3000000	0.0000019073 s	SI
2	10393545	3000000	0.0000030994 s	SI

3	1295390003	3000000	0.0000030994 s	NO
4	1360839354	3000000	0.0000021458 s	NO
5	14700764	3000000	0.0000030994 s	NO
6	1493283650	3000000	0.0000028610 s	NO
7	152503	3000000	0.0000030994 s	NO
8	1843349527	3000000	0.0000021458 s	NO
9	187645041	3000000	0.0000040531 s	NO
10	1980098116	3000000	0.0000030994 s	NO
11	2109248666	3000000	0.0000028610 s	NO
12	2147445644	3000000	0.0000040531 s	NO
13	2147470852	3000000	0.0000030994 s	NO
14	214826	3000000	0.0000028610 s	NO
15	3128036	3000000	0.0000030994 s	SI
16	322486	3000000	0.0000040531 s	SI
17	450057883	3000000	0.0000040531 s	NO
18	5000	3000000	0.0000019073 s	NO
19	61396	3000000	0.0000028610 s	NO
20	6337399	3000000	0.0000028610 s	SI

Promedio de tiempo real**: 0.0000123977 s**

Para n = 4000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	4000000	0.0000030994 s	SI
2	10393545	4000000	0.0000040531 s	SI
3	1295390003	4000000	0.0000030994 s	NO
4	1360839354	4000000	0.0000030994 s	NO
5	14700764	4000000	0.0000028610 s	NO
6	1493283650	4000000	0.0000028610 s	NO
7	152503	4000000	0.0000030994 s	NO
8	1843349527	4000000	0.0000030994 s	NO
9	187645041	4000000	0.0000040531 s	NO
10	1980098116	4000000	0.0000021458 s	NO
11	2109248666	4000000	0.0000028610 s	NO
12	2147445644	4000000	0.0000030994 s	NO
13	2147470852	4000000	0.0000021458 s	NO
14	214826	4000000	0.0000028610 s	NO
15	3128036	4000000	0.0000028610 s	SI
16	322486	4000000	0.0000038147 s	SI
17	450057883	4000000	0.0000040531 s	NO
18	5000	4000000	0.0000030994 s	NO
19	61396	4000000	0.0000019073 s	NO
20	6337399	4000000	0.0000030994 s	SI

Promedio de tiempo real**: 0.0000133514 s**

Para n = 5000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*

1	0	5000000	0.0000021458 s	SI
2	10393545	5000000	0.0000028610 s	SI
3	1295390003	5000000	0.0000030994 s	NO
4	1360839354	5000000	0.0000028610 s	NO
5	14700764	5000000	0.0000028610 s	NO
6	1493283650	5000000	0.0000030994 s	NO
7	152503	5000000	0.0000028610 s	NO
8	1843349527	5000000	0.0000030994 s	NO
9	187645041	5000000	0.0000040531 s	NO
10	1980098116	5000000	0.0000030994 s	NO
11	2109248666	5000000	0.0000028610 s	NO
12	2147445644	5000000	0.0000040531 s	NO
13	2147470852	5000000	0.0000028610 s	NO
14	214826	5000000	0.0000030994 s	NO
15	3128036	5000000	0.0000028610 s	SI
16	322486	5000000	0.0000030994 s	SI
17	450057883	5000000	0.0000030994 s	NO
18	5000	5000000	0.0000028610 s	NO
19	61396	5000000	0.0000038147 s	NO
20	6337399	5000000	0.0000030994 s	SI

Promedio de tiempo real**: 0.0000131129 s**

Para n = 6000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	6000000	0.0000019073 s	SI
2	10393545	6000000	0.0000028610 s	SI
3	1295390003	6000000	0.0000030994 s	NO
4	1360839354	6000000	0.0000030994 s	NO
5	14700764	6000000	0.0000040531 s	NO
6	1493283650	6000000	0.0000030994 s	NO
7	152503	6000000	0.0000028610 s	NO
8	1843349527	6000000	0.0000030994 s	NO
9	187645041	6000000	0.0000040531 s	NO
10	1980098116	6000000	0.0000030994 s	NO
11	2109248666	6000000	0.0000019073 s	NO
12	2147445644	6000000	0.0000030994 s	NO
13	2147470852	6000000	0.0000030994 s	NO
14	214826	6000000	0.0000030994 s	NO
15	3128036	6000000	0.0000028610 s	SI
16	322486	6000000	0.0000040531 s	SI
17	450057883	6000000	0.0000030994 s	NO
18	5000	6000000	0.0000028610 s	NO
19	61396	6000000	0.0000030994 s	NO
20	6337399	6000000	0.0000030994 s	SI

Promedio de tiempo real**: 0.000014305 s**

Para n = 7000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	7000000	0.0000019073 s	SI
2	10393545	7000000	0.0000030994 s	SI
3	1295390003	7000000	0.0000028610 s	SI
4	1360839354	7000000	0.0000040531 s	NO
5	14700764	7000000	0.0000038147 s	NO
6	1493283650	7000000	0.0000030994 s	SI
7	152503	7000000	0.0000030994 s	NO
8	1843349527	7000000	0.0000038147 s	NO
9	187645041	7000000	0.0000100136 s	NO
10	1980098116	7000000	0.0000030994 s	NO
11	2109248666	7000000	0.0000021458 s	NO
12	2147445644	7000000	0.0000040531 s	NO
13	2147470852	7000000	0.0000030994 s	NO
14	214826	7000000	0.0000019073 s	NO
15	3128036	7000000	0.0000030994 s	SI
16	322486	7000000	0.0000040531 s	SI
17	450057883	7000000	0.0000040531 s	NO
18	5000	7000000	0.0000030994 s	NO
19	61396	7000000	0.0000028610 s	NO
20	6337399	7000000	0.0000038147 s	SI

Promedio de tiempo real**: 0.0000209808 s**

Para n = 8000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	8000000	0.0000178814 s	SI
2	10393545	8000000	0.0000059605 s	SI
3	1295390003	8000000	0.0000309944 s	SI
4	1360839354	8000000	0.0000069141 s	NO
5	14700764	8000000	0.0000090599 s	NO
6	1493283650	8000000	0.0000090599 s	SI
7	152503	8000000	0.0000040531 s	NO
8	1843349527	8000000	0.0000071526 s	NO
9	187645041	8000000	0.0000090599 s	NO
10	1980098116	8000000	0.0000050068 s	NO
11	2109248666	8000000	0.0000250340 s	NO
12	2147445644	8000000	0.0000159740 s	NO
13	2147470852	8000000	0.0000050068 s	NO
14	214826	8000000	0.0000050068 s	NO
15	3128036	8000000	0.0000059605 s	SI
16	322486	8000000	0.0000061989 s	SI
17	450057883	8000000	0.0000269413 s	NO
18	5000	8000000	0.0000059605 s	NO
19	61396	8000000	0.0000050068 s	NO
20	6337399	8000000	0.0000059605 s	SI

Promedio de tiempo real**: 0.0000522137 s**

Para n = 9000000

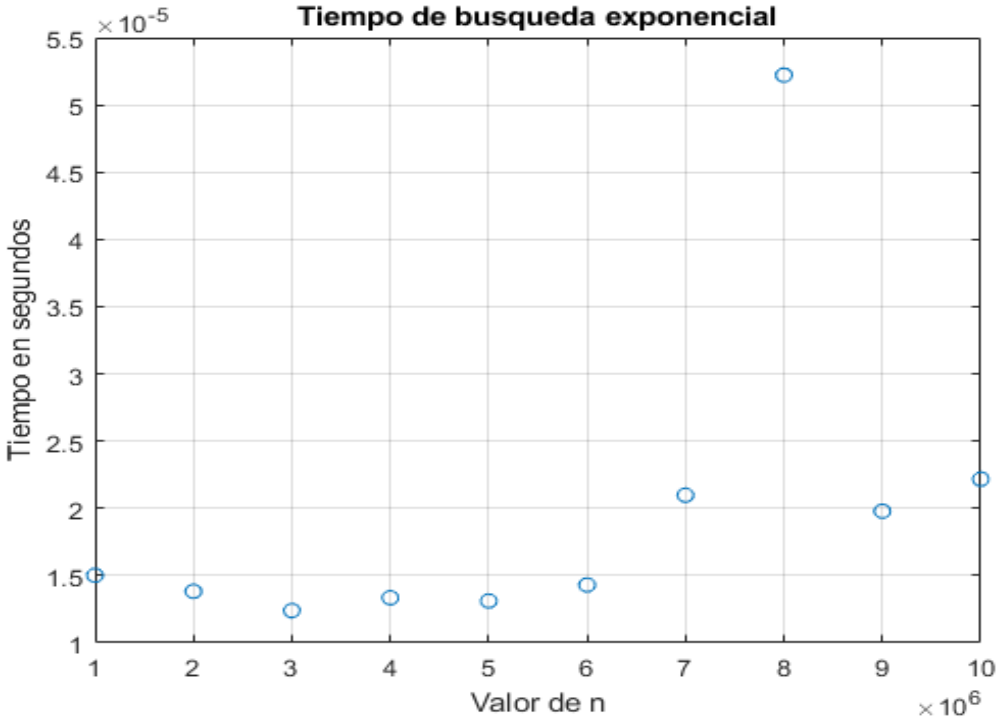
# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	9000000	0.0000028610 s	SI
2	10393545	9000000	0.0000030994 s	SI
3	1295390003	9000000	0.0000040531 s	SI
4	1360839354	9000000	0.0000069141 s	NO
5	14700764	9000000	0.0000040531 s	NO
6	1493283650	9000000	0.0000059605 s	SI
7	152503	9000000	0.0000021458 s	NO
8	1843349527	9000000	0.0000069141 s	SI
9	187645041	9000000	0.0000040531 s	NO
10	1980098116	9000000	0.0000028610 s	NO
11	2109248666	9000000	0.0000050068 s	NO
12	2147445644	9000000	0.0000038147 s	NO
13	2147470852	9000000	0.0000030994 s	NO
14	214826	9000000	0.0000028610 s	NO
15	3128036	9000000	0.0000030994 s	SI
16	322486	9000000	0.0000040531 s	SI
17	450057883	9000000	0.0000090599 s	NO
18	5000	9000000	0.0000028610 s	NO
19	61396	9000000	0.0000021458 s	NO
20	6337399	9000000	0.0000030994 s	SI

Promedio de tiempo real**: 0.0000197887 s**

Para n = 10000000

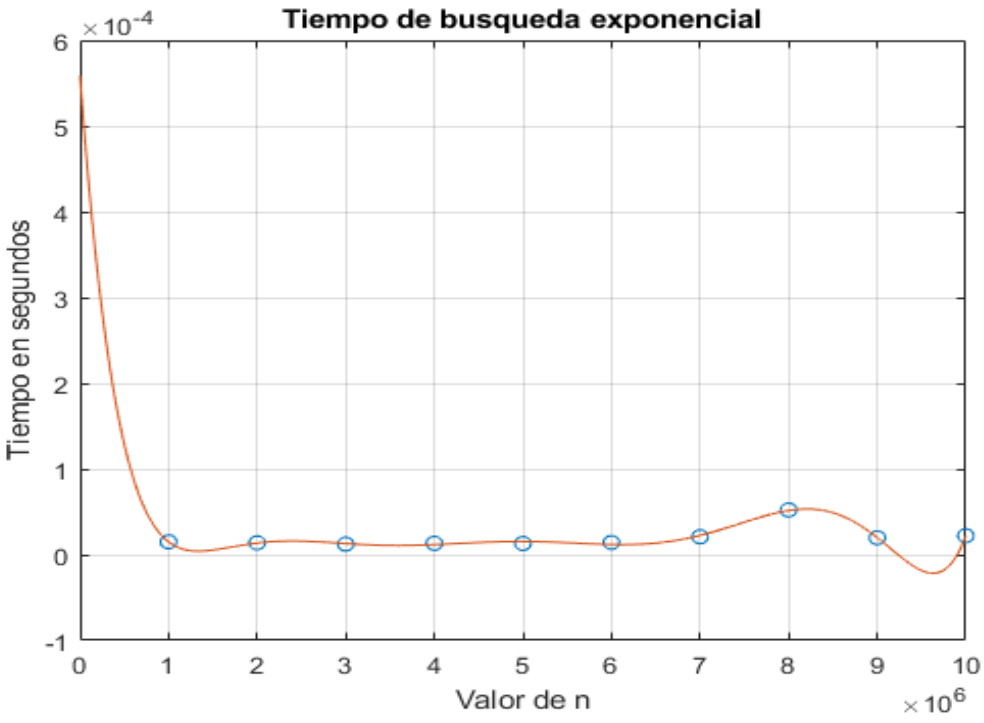
# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	10000000	0.0000028610 s	SI
2	10393545	10000000	0.0000028610 s	SI
3	1295390003	10000000	0.0000071526 s	SI
4	1360839354	10000000	0.0000050068 s	NO
5	14700764	10000000	0.0000050068 s	NO
6	1493283650	10000000	0.0000059605 s	SI
7	152503	10000000	0.0000038147 s	NO
8	1843349527	10000000	0.0000050068 s	SI
9	187645041	10000000	0.0000050068 s	NO
10	1980098116	10000000	0.0000040531 s	NO
11	2109248666	10000000	0.0000050068 s	SI
12	2147445644	10000000	0.0000040531 s	NO
13	2147470852	10000000	0.0000040531 s	SI
14	214826	10000000	0.0000030994 s	NO
15	3128036	10000000	0.0000040531 s	SI
16	322486	10000000	0.0000078678 s	SI
17	450057883	10000000	0.0000040531 s	NO
18	5000	10000000	0.0000030994 s	NO
19	61396	10000000	0.0000021458 s	NO
20	6337399	10000000	0.0000040531 s	SI

Promedio de tiempo real**: 0.000022173 s**



Aproximación del comportamiento temporal

$f(n) = 0.5589n^9 + 1 \times 10^{-3}$



Constante multiplicativa

Para el peor caso, considerando a N como 10.000.000, tenemos que la constante multiplicativa de cada operación básica es 1.971717×10^{-11}

Tiempos para el peor caso

n	***Tiempo estimado***
50000000	8.30×10^{-9} s
100000000	9.56×10^{-9} s
500000000	1.01×10^{-8} s
1000000000	1.08×10^{-8} s
5000000000	1.14×10^{-8} s

Idea para mejorar el algoritmo

La utilización de hilos para la búsqueda exponencial no es buena opción para mejorar el tiempo de procesamiento porque cuando se parte por pedazos la búsqueda, habrá que ejecutar cada una de ellas a pesar de que en alguna de ellas ya se encontró el elemento que se deseaba localizar, entonces no se mejoraría en nada el tiempo de ejecución.

Por tanto, al implementar una búsqueda con hilos solo haría que en la mayoria de los casos se empeoraría el tiempo, puesto que hace más operaciones por realizar las particiones.

Búsqueda de Fibonacci

Teoría

La búsqueda de Fibonacci es un método de búsqueda de una matriz ordenada que divide el arreglo en dos partes que tienen tamaños que son números de Fibonacci consecutivos.

La búsqueda primero encuentra el número de Fibonacci más pequeño mayor o igual que n, luego se tiene que comparar el número que se quiere buscar con el número menor que es del número de Fibonacci que se calculó, si los números coinciden devolverá el índice.

Análisis Teórico

Mejor caso

El mejor caso se presenta cuando el elemento a buscar está en la primera posición del arreglo cuando se efectúa la suma de -1 y el numero antecesor más pequeño de la serie de Fibonacci.

Entonces la función temporal es $f(n) = 5n - 8$

Ya que se tomó como operaciones básicas las comparaciones, asignaciones y aritméticas del cálculo del número Fibonacci y el de la búsqueda.

Peor caso

El peor caso es cuando el elemento a buscar no se encuentra en el arreglo, entonces para calcular el numero Fibonacci que es menor a n, se hacen 5n - 8 veces. Para buscar el elemento se hacen $11*((\log_2 n) + 3)$

Por tanto, la función queda como $f(n) = 5n - 8 + 11*((\log_2 n) + 3)$

Caso medio

La probabilidad que se le asigna será 1/n, puesto que puede estar el elemento buscado en cualquier posición, entonces la función queda como $f(n)\ 1/n\ (5n - 8 + 11*((\log_2 n) + 3))$

Registro de tiempo

Para n = 1000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	1000000	0.0000028610 s	SI
2	10393545	1000000	0.0000028610 s	SI
3	1295390003	1000000	0.0000030994 s	NO
4	1360839354	1000000	0.0000030994 s	NO
5	14700764	1000000	0.0000030994 s	NO
6	1493283650	1000000	0.0000030994 s	NO
7	152503	1000000	0.0000030994 s	NO
8	1843349527	1000000	0.0000030994 s	NO
9	187645041	1000000	0.0000038147 s	NO
10	1980098116	1000000	0.0000030994 s	NO
11	2109248666	1000000	0.0000030994 s	NO
12	2147445644	1000000	0.0000028610 s	NO
13	2147470852	1000000	0.0000030994 s	NO
14	214826	1000000	0.0000030994 s	NO
15	3128036	1000000	0.0000028610 s	SI
16	322486	1000000	0.0000028610 s	SI
17	450057883	1000000	0.0000028610 s	NO
18	5000	1000000	0.0000030994 s	NO
19	61396	1000000	0.0000028610 s	NO

20	6337399	1000000	0.0000028610 s	SI

Promedio de tiempo real**: 0.0000131129 s**

Para n = 2000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	2000000	0.0000030994 s	SI
2	10393545	2000000	0.0000030994 s	SI
3	1295390003	2000000	0.0000030994 s	NO
4	1360839354	2000000	0.0000028610 s	NO
5	14700764	2000000	0.0000040531 s	NO
6	1493283650	2000000	0.0000028610 s	NO
7	152503	2000000	0.0000028610 s	NO
8	1843349527	2000000	0.0000030994 s	NO
9	187645041	2000000	0.0000040531 s	NO
10	1980098116	2000000	0.0000019073 s	NO
11	2109248666	2000000	0.0000030994 s	NO
12	2147445644	2000000	0.0000028610 s	NO
13	2147470852	2000000	0.0000019073 s	NO
14	214826	2000000	0.0000028610 s	NO
15	3128036	2000000	0.0000030994 s	SI
16	322486	2000000	0.0000040531 s	SI
17	450057883	2000000	0.0000030994 s	NO
18	5000	2000000	0.0000028610 s	NO
19	61396	2000000	0.0000028610 s	NO
20	6337399	2000000	0.0000028610 s	SI

Promedio de tiempo real**: 0.0000131129 s**

Para n = 3000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	3000000	0.0000030994 s	SI
2	10393545	3000000	0.0000030994 s	SI
3	1295390003	3000000	0.0000028610 s	NO
4	1360839354	3000000	0.0000030994 s	NO
5	14700764	3000000	0.0000040531 s	NO
6	1493283650	3000000	0.0000030994 s	NO
7	152503	3000000	0.0000030994 s	NO
8	1843349527	3000000	0.0000019073 s	NO
9	187645041	3000000	0.0000028610 s	NO
10	1980098116	3000000	0.0000030994 s	NO
11	2109248666	3000000	0.0000030994 s	NO
12	2147445644	3000000	0.0000030994 s	NO
13	2147470852	3000000	0.0000030994 s	NO
14	214826	3000000	0.0000028610 s	NO
15	3128036	3000000	0.0000040531 s	SI
16	322486	3000000	0.0000040531 s	SI
17	450057883	3000000	0.0000038147 s	NO

18	5000	3000000	0.0000028610 s	NO
19	61396	3000000	0.0000030994 s	NO
20	6337399	3000000	0.0000038147 s	SI

Promedio de tiempo real**: 0.0000119208 s**

Para n = 4000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	4000000	0.0000028610 s	SI
2	10393545	4000000	0.0000040531 s	SI
3	1295390003	4000000	0.0000030994 s	NO
4	1360839354	4000000	0.0000030994 s	NO
5	14700764	4000000	0.0000040531 s	NO
6	1493283650	4000000	0.0000028610 s	NO
7	152503	4000000	0.0000040531 s	NO
8	1843349527	4000000	0.0000030994 s	NO
9	187645041	4000000	0.0000050068 s	NO
10	1980098116	4000000	0.0000030994 s	NO
11	2109248666	4000000	0.0000078678 s	NO
12	2147445644	4000000	0.0000038147 s	NO
13	2147470852	4000000	0.0000030994 s	NO
14	214826	4000000	0.0000028610 s	NO
15	3128036	4000000	0.0000038147 s	SI
16	322486	4000000	0.0000040531 s	SI
17	450057883	4000000	0.0000128746 s	NO
18	5000	4000000	0.0000038147 s	NO
19	61396	4000000	0.0000040531 s	NO
20	6337399	4000000	0.0000030994 s	SI

Promedio de tiempo real**: 0.0000162124 s**

Para n = 5000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	5000000	0.0000028610 s	SI
2	10393545	5000000	0.0000038147 s	SI
3	1295390003	5000000	0.0000030994 s	NO
4	1360839354	5000000	0.0000030994 s	NO
5	14700764	5000000	0.0000038147 s	NO
6	1493283650	5000000	0.0000040531 s	NO
7	152503	5000000	0.0000028610 s	NO
8	1843349527	5000000	0.0000030994 s	NO
9	187645041	5000000	0.0000040531 s	NO
10	1980098116	5000000	0.0000028610 s	NO
11	2109248666	5000000	0.0000030994 s	NO
12	2147445644	5000000	0.0000040531 s	NO
13	2147470852	5000000	0.0000030994 s	NO
14	214826	5000000	0.0000040531 s	NO
15	3128036	5000000	0.0000040531 s	SI

16	322486	5000000	0.0000050068 s	SI
17	450057883	5000000	0.0000040531 s	NO
18	5000	5000000	0.0000028610 s	NO
19	61396	5000000	0.0000030994 s	NO
20	6337399	5000000	0.0000028610 s	SI

Promedio de tiempo real**: 0.0000140666 s**

Para n = 6000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	6000000	0.0000050068 s	SI
2	10393545	6000000	0.0000100136 s	SI
3	1295390003	6000000	0.0000030994 s	NO
4	1360839354	6000000	0.0000040531 s	NO
5	14700764	6000000	0.0000050068 s	NO
6	1493283650	6000000	0.0000038147 s	NO
7	152503	6000000	0.0000050068 s	NO
8	1843349527	6000000	0.0000040531 s	NO
9	187645041	6000000	0.0000078678 s	NO
10	1980098116	6000000	0.0000040531 s	NO
11	2109248666	6000000	0.0000030994 s	NO
12	2147445644	6000000	0.0000040531 s	NO
13	2147470852	6000000	0.0000059605 s	NO
14	214826	6000000	0.0000050068 s	NO
15	3128036	6000000	0.0000061989 s	SI
16	322486	6000000	0.0000050068 s	SI
17	450057883	6000000	0.0000059605 s	NO
18	5000	6000000	0.0000040531 s	NO
19	61396	6000000	0.0000040531 s	NO
20	6337399	6000000	0.0000038147 s	SI

Promedio de tiempo real**: 0.0000269413 s**

Para n = 7000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	7000000	0.0000028610	SI
2	10393545	7000000	0.0000028610	SI
3	1295390003	7000000	0.0000050068 s	SI
4	1360839354	7000000	0.0000040531 s	NO
5	14700764	7000000	0.0000040531 s	NO
6	1493283650	7000000	0.0000040531 s	SI
7	152503	7000000	0.0000040531 s	NO
8	1843349527	7000000	0.0000028610 s	NO
9	187645041	7000000	0.0000040531 s	NO
10	1980098116	7000000	0.0000028610 s	NO
11	2109248666	7000000	0.0000019073 s	NO
12	2147445644	7000000	0.0000030994 s	NO
13	2147470852	7000000	0.0000030994 s	NO

14	214826	7000000	0.0000028610 s	NO
15	3128036	7000000	0.0000040531 s	SI
16	322486	7000000	0.0000040531 s	SI
17	450057883	7000000	0.0000040531 s	NO
18	5000	7000000	0.0000030994 s	NO
19	61396	7000000	0.0000028610 s	NO
20	6337399	7000000	0.0000040531 s	SI

Promedio de tiempo real**: 0.000015974 s**

Para n = 8000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	8000000	0.0000030994 s	SI
2	10393545	8000000	0.0000028610 s	SI
3	1295390003	8000000	0.0000038147 s	SI
4	1360839354	8000000	0.0000040531 s	NO
5	14700764	8000000	0.0000040531 s	NO
6	1493283650	8000000	0.0000040531 s	SI
7	152503	8000000	0.0000028610 s	NO
8	1843349527	8000000	0.0000030994 s	NO
9	187645041	8000000	0.0000038147 s	NO
10	1980098116	8000000	0.0000030994 s	NO
11	2109248666	8000000	0.0000030994 s	NO
12	2147445644	8000000	0.0000030994 s	NO
13	2147470852	8000000	0.0000028610 s	NO
14	214826	8000000	0.0000028610 s	NO
15	3128036	8000000	0.0000038147 s	SI
16	322486	8000000	0.0000038147 s	SI
17	450057883	8000000	0.0000040531 s	NO
18	5000	8000000	0.0000030994 s	NO
19	61396	8000000	0.0000038147 s	NO
20	6337399	8000000	0.0000030994 s	SI

Promedio de tiempo real**: 0.0000138282 s**

Para n = 9000000

# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	9000000	0.0000030994 s	SI
2	10393545	9000000	0.0000028610 s	SI
3	1295390003	9000000	0.0000040531 s	SI
4	1360839354	9000000	0.0000040531 s	NO
5	14700764	9000000	0.0000030994 s	NO
6	1493283650	9000000	0.0000040531 s	SI
7	152503	9000000	0.0000028610 s	NO
8	1843349527	9000000	0.0000030994 s	SI
9	187645041	9000000	0.0000040531 s	NO
10	1980098116	9000000	0.0000028610 s	NO
11	2109248666	9000000	0.0000030994 s	NO

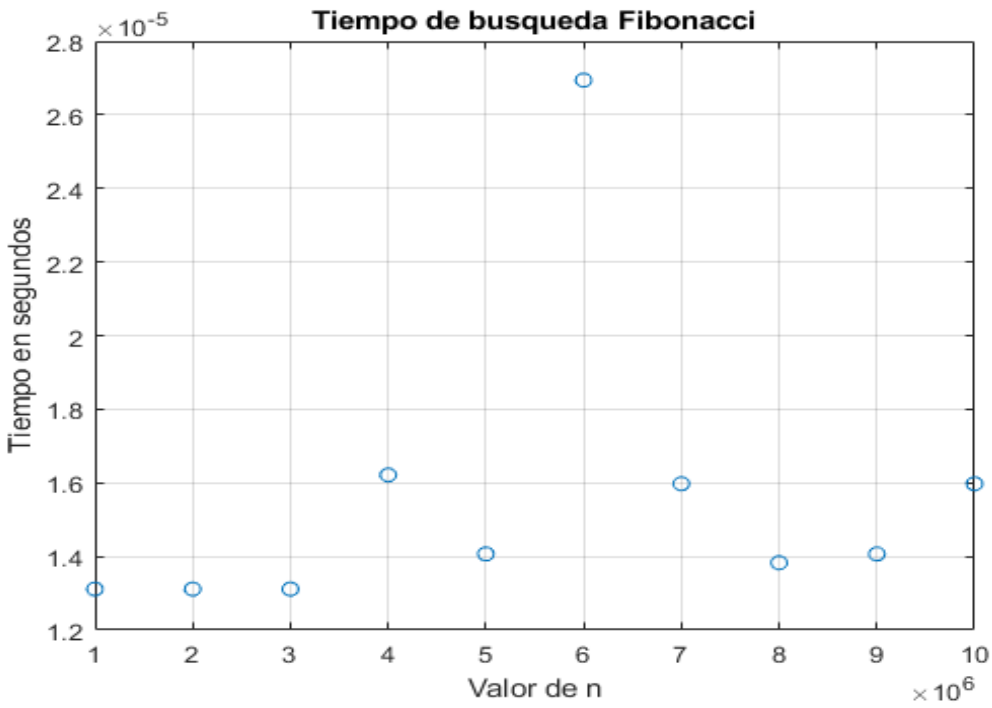
12	2147445644	9000000	0.0000030994 s	NO
13	2147470852	9000000	0.0000030994 s	NO
14	214826	9000000	0.0000030994 s	NO
15	3128036	9000000	0.0000030994 s	SI
16	322486	9000000	0.0000050068 s	SI
17	450057883	9000000	0.0000040531 s	NO
18	5000	9000000	0.0000030994 s	NO
19	61396	9000000	0.0000028610 s	NO
20	6337399	9000000	0.0000030994 s	SI

Promedio de tiempo real**: 0.0000140666 s**

Para n = 10000000

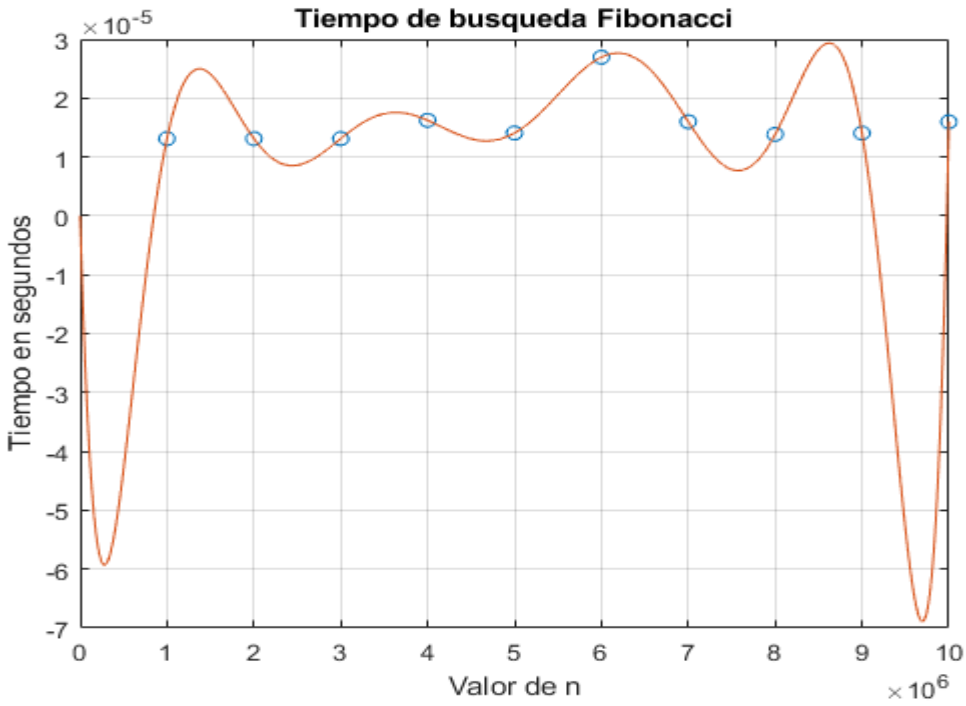
# Test	*Número por buscar*	*Tamaño de n*	*Tiempo real*	*Encontrado*
1	0	10000000	0.0000038147 s	SI
2	10393545	10000000	0.0000040531 s	SI
3	1295390003	10000000	0.0000040531 s	SI
4	1360839354	10000000	0.0000040531 s	NO
5	14700764	10000000	0.0000028610 s	NO
6	1493283650	10000000	0.0000040531 s	SI
7	152503	10000000	0.0000040531 s	NO
8	1843349527	10000000	0.0000038147 s	SI
9	187645041	10000000	0.0000040531 s	NO
10	1980098116	10000000	0.0000050068 s	NO
11	2109248666	10000000	0.0000040531 s	SI
12	2147445644	10000000	0.0000040531 s	NO
13	2147470852	10000000	0.0000061989 s	SI
14	214826	10000000	0.0000040531 s	NO
15	3128036	10000000	0.0000038147 s	SI
16	322486	10000000	0.0000050068 s	SI
17	450057883	10000000	0.0000050068 s	NO
18	5000	10000000	0.0000030994 s	NO
19	61396	10000000	0.0000040531 s	NO
20	6337399	10000000	0.0000040531 s	SI

Promedio de tiempo real**: 0.000015974 s**



Aproximación del comportamiento temporal

$f(n) = -0.5621n^{10} + 1 \times 10^{-9}$



Constante multiplicativa

Para el peor caso, considerando a N como 10.000.000, tenemos que la constante multiplicativa de cada operación básica es 1.564018×10^{-11}

Tiempos para el peor caso

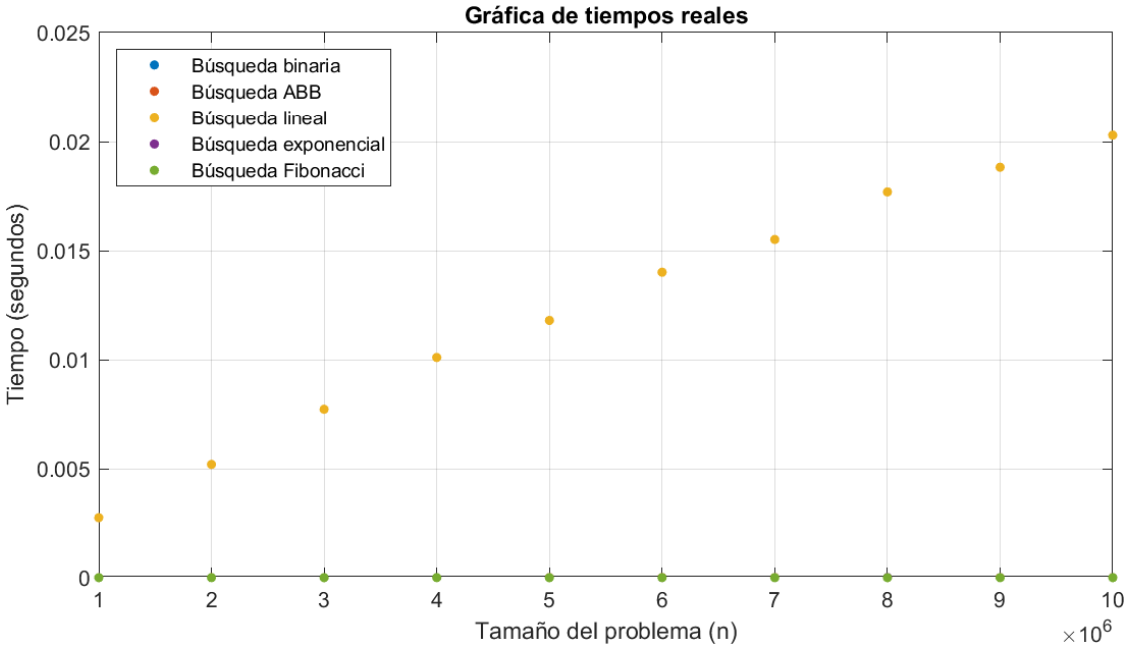
n	*Tiempo estimado*
50000000	1.16 s
100000000	2.40 s
500000000	1.29 s
1000000000	2.66 s
5000000000	14.62 s

Idea para mejorar el algoritmo

La utilización de hilos para la búsqueda Fibonacci no es buena opción para mejorar el tiempo de procesamiento porque cuando se parte por pedazos la búsqueda, habrá que ejecutar cada una de ellas a pesar de que en alguna de ellas ya se encontró el elemento que se deseaba localizar, entonces no se mejoraría en nada el tiempo de ejecución.

Por tanto, al implementar una búsqueda con hilos solo haría que en la mayoría de los casos se empeoraría el tiempo, puesto que hace más operaciones por realizar las particiones.

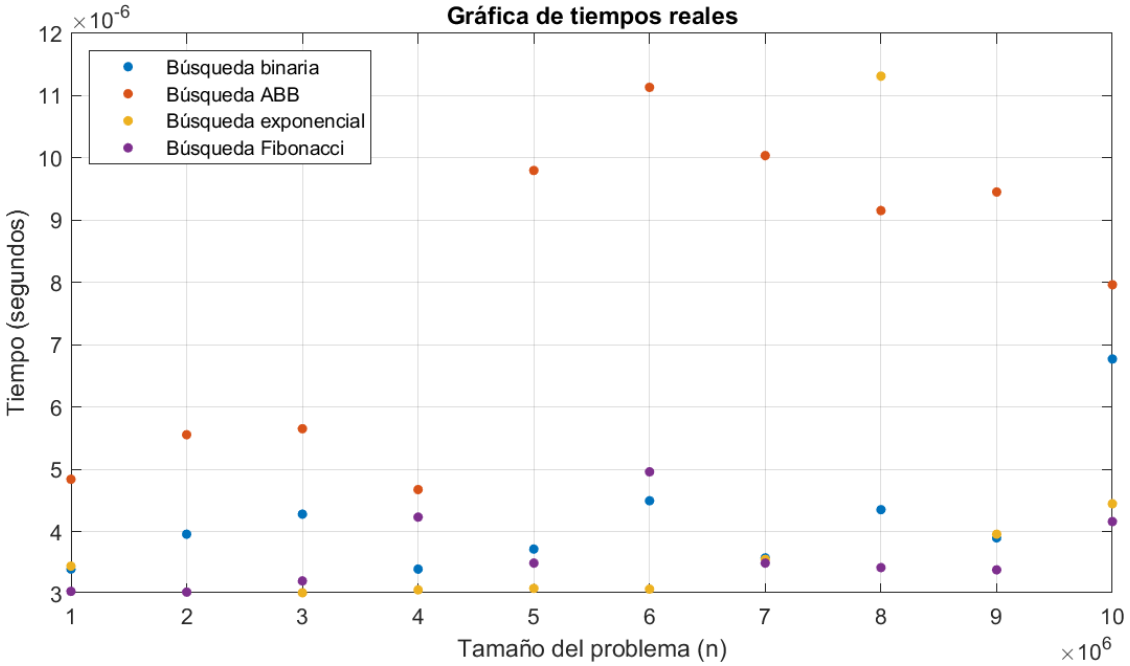
General



Los resultados son fáciles de ver: aunque todos los algoritmos son increíblemente rápidos y dan resultados de eficiencia para el máximo valor de N considerado, es claro que la búsqueda lineal

tiene el peor rendimiento de todos los algoritmos tal que evita que el resto de la información se observe con claridad.

Para visualizar el comportamiento de los otros algoritmos, la siguiente imagen oculta los datos de la búsqueda lineal.



Un poco sorprendente es la comparación del árbol binario de búsqueda, que se encuentra sobre las funciones de los demás algoritmos. En cambio, podemos observar un comportamiento interesante entre la búsqueda exponencial y la búsqueda Fibonacci (los dos algoritmos más rápidos), porque inicialmente la búsqueda exponencial se mantiene ligeramente debajo de Fibonacci, y después los roles se intercambian.

Conclusiones

Se realizó el análisis de cinco algoritmos de búsqueda conocidos para obtener la función de complejidad temporal para los tres casos esenciales: mejor, peor y promedio. Posteriormente, se implementaron en ANSI C buscando distintos valores en un arreglo de tamaño N, donde el valor del parámetro N cambiaba para cada prueba. El cálculo del tiempo real de ejecución se obtuvo con temporizadores de Linux. Posteriormente, se aproximaron funciones polinomiales para los tiempos promedio para cada N con MATLAB, y se obtuvieron gráficas de cada función resultante. Finalmente, se modificó el código de la Búsqueda Lineal para hacer uso de hilos POSIX y mejorar el rendimiento del algoritmo.

Cuestionario

1. **¿Cuál de los 5 algoritmos es más fácil de implementar?** La búsqueda lineal tiene una implementación muy intuitiva, pues sólo se itera desde el inicio del arreglo hasta el final para encontrar el número.
2. **¿Cuál de los 5 algoritmos es el más difícil de implementar?** En contraste con la búsqueda lineal, el árbol de búsqueda binaria involucra la creación de una estructura de datos y sus funciones elementales; en este caso, la creación de nodos y la inserción en el árbol.

3. **¿Cuál de los 5 algoritmos fue el más difícil de modelar en su variante con hilos?** El equipo tuvo muchos problemas tratando de implementar hilos en prácticamente cada algoritmo, con excepción de la búsqueda lineal. Fue muy difícil intentar mejorar la complejidad de varios algoritmos que ya son muy eficientes.
4. **¿Cuál de los 5 algoritmos en su variante con hilos resultó ser más rápido? ¿Por qué?** La búsqueda lineal se vio beneficiada por una implementación con hilos. Poder dividir el arreglo en varias secciones que se ejecutan simultáneamente aceleró el algoritmo.
5. **¿Cuál de los 5 algoritmos en su variante con hilos no representó alguna ventaja? ¿Por qué?** Como se mencionó, el resto de los algoritmos de búsqueda no obtienen una mejora de rendimiento constante. Como los rangos de búsqueda constantemente son modificados para buscar el elemento más rápido, no es necesario implementar hilos para realizar esa tarea.
6. **¿Cuál algoritmo tiene menor complejidad temporal?** La búsqueda Fibonacci tiene la menor complejidad de los cinco algoritmos
7. **¿Cuál algoritmo tiene mayor complejidad temporal?** La búsqueda lineal tiene la mayor complejidad temporal de los cinco algoritmos
8. **¿El comportamiento experimental de los algoritmos era el esperado? ¿Por qué?** Particularmente para la búsqueda lineal, binaria y en árbol binario, dado que el equipo ya los había implementado con anterioridad, pues teníamos experiencia analizando el algoritmo de manera práctica.
9. **¿Sus resultados experimentales difieren mucho de los análisis teóricos realizados? ¿A qué se debe?** Sí hay variación entre los resultados obtenidos debido a que nuestros análisis son sólo estimaciones que no consideran aspectos como el hardware utilizado para las pruebas. Sin embargo, las diferencias no son suficientes para descartar los modelos.
10. **En la versión de hilos, ¿usar n hilos dividió el tiempo en n? ¿Lo hizo n veces más rápido?** Únicamente implementamos con hilos la búsqueda lineal, ya que es la única búsqueda en donde se puede obtener una mejora, tuvimos que dividir el arreglo en 2 y cada hilo realiza la búsqueda lineal en su arreglo, por lo tanto, teóricamente hay una mejora del 100%
11. **¿Cuál es el porcentaje de mejora que tiene cada uno de los algoritmos en su variante con hilos? ¿Es lo que esperabas? ¿Por qué?** Únicamente tiene mejora el algoritmo de búsqueda lineal, la mejora teórica es del 100% ya que únicamente programamos un hilo más, cada hilo busca en una mitad del arreglo.
12. **¿Existió un entorno controlado para realizar las pruebas experimentales? ¿Cuál fue?** Sí, las pruebas fueron realizadas en la misma computadora, con el mismo entorno de Linux para Windows 10 (WSL), con las mismas operaciones de entrada y salida para la obtención de los resultados de tiempo.
13. **Si sólo se realizara el análisis teórico de un algoritmo antes de implementarlo, ¿podrías asegurar cuál es el mejor?** Sí, porque un análisis es realizado para poder tomar buenas decisiones, y aunque definir a un algoritmo como "el mejor" puede ser un poco problemático, el análisis nos permite modificar esa definición a "el mejor para resolver nuestro problema"
14. **¿Qué tan difícil fue realizar el análisis teórico de cada algoritmo?** Hubo una confusión al momento de la primera revisión dado que se obtuvieron los resultados requeridos con cota O, y porque la función para el algoritmo ABB podría ser mejor representada en función de la altura del árbol.
15. **¿Qué recomendaciones darían a nuevos equipos para realizar esta práctica?** Familiarización con los hilos POSIX para que puedan realizar los programas y las pruebas correctamente. Y de nuevo, la utilización de scripts para facilitar el desarrollo de la práctica

Bibliografía

"Searching Algorithms", GeeksforGeeks, 2017. [En línea]. Disponible en: https://www.geeksforgeeks.org/searching-algorithms/

T. H. Cormen, Introduction to Algorithms, Massachusetts: MIT Press, 2009.

Gibbs. M. (s.f). Exponential Search in Java: Algorithm, Implementation & Analysis [Online]. Disponible en: https://study.com/academy/lesson/exponential-search-in-java-algorithm-implementation-analysis.html

Anexos

Búsqueda Lineal

Algoritmo sin hilos

```
/*
      ____________________________________________________________
     / \                                                          \.
    |   |    + Titulo: Busqueda Lineal                            |.
    \_  |    + Author: Luis Fernando Resendiz Chavez               |.
        |    + Version: 1.0                                        |.
        |    + Fecha: 20 noviembre 2020                            |.
        |    + Descripcion:                                        |.
        |       Buscar un elemento en un arreglo utilizando        |.
        |       el algoritmo de busqueda lineal.                   |.
        |    + Entrada:                                            |.
        |       - Arreglo de numeros enteros                       |.
        |       - Longitud de dicho arreglo                        |.
        |       - Elemento a buscar                                |.
        |    + Salida:                                             |.
        |       - SI: en caso de que el elemento SI                |.
        |           haya sido encontrado en el arreglo.            |.
        |       - NO: en caso de que el elemento NO                |.
        |           haya sido encontrado en el arreglo.            |.
        |   ____________________________________________________|___
        |  /                                                      /.
        \_/_____________________________________________________/.
*/



/* =^..^= *** =^..^= ***** LIBREARIAS UTILIZADAS ***** =^..^= ***
=^..^= */
#include <stdio.h>
#include <stdlib.h>

/* =^..^= *** =^..^= ***** DEFINICIONES ***** =^..^= *** =^..^= */
```

```c
#define TRUE 1
#define FALSE 0

/* =^..^= *** =^..^= ***** PROTOTIPOS DE FUNCIONES ***** =^..^= ***
=^..^= */
int busquedaLineal(int n, int *A, int x);

/* =^..^= *** =^..^= ***** FUNCION PRINCIPAL ***** =^..^= *** =^..^= */
int main() {
    // Se crean n para el tamano del arreglo, x elemento a buscar y un
iterador =^..^= */
    int n, x, i;
    // Se reciben dichas variables =^..^= */
    scanf("%d %d", &n, &x);
    // Se crea un arreglo dinamico de n elementos enteros =^..^= */
    int *A = malloc(sizeof(int) * n);
    // Se reciben los datos del arreglo =^..^= */
    for(i = 0; i < n; ++i)
        scanf("%d", &A[i]);

    // Se ejecuta la funcion busquedaLineal y se imprime dependiendo el
resultado =^..^= */
    busquedaLineal(n, A, x) ?
        printf("YES\n") : printf("NO\n");
    return  0;
}


/* =^..^= *** =^..^= ***** FUNCION BUSQUEDA LINEAL ***** =^..^= ***
=^..^= */
int busquedaLineal(int n, int *A, int x) {
    // Se crea una variable iteradora =^..^= */
    int i;
    // Se comparan todas las posiciones del arreglo en busca del
elemento x =^..^= */
    for(i = 0; i < n; ++i)
        // Si lo encontramos, retornamos TRUE =^..^= */
        if(A[i] == x)
            return TRUE;

    // Si no lo encontramos, retornamos FALSE =^..^= */
    return FALSE;
}
```

Algoritmo con hilos

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

struct data {
        int *array;
        int size;
        int target;

};

void *LinearSearch(void *args){
        struct data *linear = (struct data*) args;
        int answer = 1, j;

        for(j=0; j<(linear->size); j++){
        if(linear->array[j]==(linear->target)){
                        answer=0;
                        pthread_exit((void*)&answer);
                        printf("%i",linear->target);
        }
        }
        pthread_exit((void*)&answer);
}



void main(int argc, char *argv[]){
        printf("%d",argc);
        int *nums = NULL, i = 0, n = 0, buscado = 0;
        int eid[5];
        struct data *d = NULL;
        pthread_t hilos[5];

        d = malloc(sizeof(struct data));

        n = atoi(argv[1]);
        buscado = atoi(argv[2]);
        nums = malloc(sizeof(int)*n);

        if(nums==NULL)
                exit(2);
        printf("hola");
```

```
        //Se leen los nums desde la consola o un archivo
        for(i=0;i<n;i++){
                scanf("%d", &nums[i]);
        }

        d->array = nums;
        d->size = n;
        d->target = buscado;

        for(i=0;i<5;i++){

                pthread_create(&hilos[i],NULL,LinearSearch,(void*)d);
        }

        for(i=0;i<5;i++){
                pthread_join(hilos[i],(void*)&eid[i]);
        }
        return;
}
```

Búsqueda Binaria

Algoritmo sin hilos

Búsqueda en Árbol binario

Algoritmo sin hilos

```
/*      Título: Búsqueda en Árbol Binario de Búsqueda
        Algoritmo que busca un dato en un árbol BST

        Fecha: 23/11/20
        Autor: Luis Armando Ramírez Espinosa
*/


/* BIBLIOTECAS UTILIZADAS */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../tiempo.h"

/* ESTRUCTURA DE DATOS PARA EL ÁRBOL DE BÚSQUEDA*/
struct Node
{
  int num;
  struct Node *left;
  struct Node *right;
};
```

```c
/* PROTOIPOS DE FUNCIONES */
int searchInTree (struct Node *, int num);
struct Node *createNode (int);
struct Node *insertNum (struct Node *, int);
void balanceTree (struct Node *);
void buildTree (struct Node **, int);
void test (struct Node *);

/* FUNCIÓN PRINCIPAL DEL PROGRAMA */
int
main (int argc, char **argv)
{
  int n = 0;
  int num = 0;

  //La práctica solicita la búsqueda de los siguientes números
  //en el árbol de búsqueda binaria.
  int queries[20] =
    { 322486, 14700764, 3128036, 6337399, 61396, 10393545, 2147445644,
    1295390003, 450057883, 187645041, 1980098116, 152503, 5000,
    1493283650, 214826, 1843349527, 1360839354, 2109248666,
    2147470852, 0
  };
  int searchResult = 0;
  struct Node *root = NULL;

  //Si el usuario no ingresa el número de elementos a guardar N,
  //el programa es terminado.
  if (argc != 2)
    return 0;

  //Conversión del parámetro N a un entero.
  n = atoi (argv[1]);

  //Construcción del árbol de búsqueda binario
  buildTree (&root, n);
  //root = balanceTree (root, n);

  //El programa muestra el parámetro N, y procede a buscar
  //cada número de las solicitudes en el árbol, escribiendo
  //si se encontró o no.
  printf ("%d\n", n);
  for (int i = 0; i < 20; i++)
    {
      searchResult = searchInTree (root, queries[i]);

      if (searchResult == 0)
        printf ("%d: not in tree\n", queries[i]);
      else
        printf ("%d: in tree\n", queries[i]);
    }

  return 0;
}
```

```c
/*      IMPLEMENTACIÓN DE FUNCIONES    */

/*
 * searchInTree (struct Node *, int)
 * Recibe el árbol sobre el cual realizar la
 * búsqueda, y un entero que representa el dato
 * a buscar.
 * El árbol navega por el subarbol derecho si el
 * dato en el nodo es menor que la llave, o por
 * el izquierdo si es mayor, y se detiene hasta
 * llegar a una hoja
*/
int
searchInTree (struct Node *root, int num)
{
  while (root != NULL)
    {
      if (num > root->num)
        root = root->right;
      else if (num < root->num)
        root = root->left;
      else
        return 1;
    }

  return 0;
}

/*
 * createNode (int)
 * Recibe un entero que representa el dato a colocar
 * en el nodo, devuelve un apuntador al nuevo nodo.
 * Crea un nodo.
*/
struct Node *
createNode (int num)
{
  struct Node *temp = (struct Node *) malloc (sizeof (struct Node));
  temp->num = num;
  temp->left = NULL;
  temp->right = NULL;

  return temp;
}

/*
 * insertNum (struct Node*, int)
 * Recibe un apuntador a un nodo, y un dato entero.
 * Inserta un nuevo número/dato al árbol.
*/
struct Node *
insertNum (struct Node *node, int num)
{
  if (node == NULL)
    return createNode (num);
```

```
  if (num < node->num)
    node->left = insertNum (node->left, num);
  else if (num > node->num)
    node->right = insertNum (node->right, num);

  return node;
}

/*
 * buildTree (struct Node **, int)
 * Recibe el nodo raíz del árbol, y el número de
 * datos a leer.
 * Lee los datos a ingresar al árbol
*/
void
buildTree (struct Node **root, int n)
{
  int num = 0;

  for (int i = 0; i < n; i++)
    {
      scanf ("%d", &num);
      *root = insertNum (*root, num);
    }
}
```

Búsqueda exponencial

Algoritmo sin hilos

```
//*****************************************************************//Qu
intana Martínez Erick
//Curso: Análisis
 de algoritmos//(C) noviembre 2020
//ESCOM-IPN
//Buscar un elemento en un arreglo utilizando el algoritmo de búsqueda
exponencial.
//versión 1.1

//*****************************************************************//**
*****************************************************************//
LIBRERIAS INCLUIDAS
//*****************************************************************
#include <stdio.h>
#include <stdlib.h>
/* si se encuentra el elemento a buscar está en laprimera posición
retornara 1 si no ira a lafunción busqueda binaria para buscar el
numero*/
int busquedaExponencial(int *A, int n, int x){
     int anterior = 0, actual = 1;
     if(A[0] == x)
        return 1;
```

```
     while((actual < n) && (A[actual] < x)){
         anterior = actual;
         actual *= 2;
     }

     return busquedaBinaria(A, n, x, anterior, actual);
}

/*Si se encuentra el numero se retorna 1, en caso contrario devolvera 0
*/
int busquedaBinaria(int *A, int n, int x, int inf, int sup){
    int limInf = inf;
    int limSup = sup;
    int centro = (limSup+limInf)/2;
    int bandera = 0;

    while((limInf <= limSup) && (bandera == 0)){
       if(A[centro] == x)
         bandera = 1;
       else if(A[centro] < x)
         limInf = centro+1;
       else
         limSup = centro-1;

     centro = (limSup+limInf)/2;
    }

    return bandera;
}
//PROGRAMA PRINCIPAL
//*****************************************************************
int main() {
    int n, x, i;

    scanf("%d %d", &n, &x);

    // Se crea un arreglo dinamico de n elementos enteros
    int *A = malloc(sizeof(int) * n);

    // Se reciben los datos del arreglo
    for(i = 0; i < n; ++i)
      scanf("%d", &A[i]);

    int encontrado;encontrado = busquedaExponencial(A,n,x);

    printf("\n%d",encontrado);

    return 0;
}
```

Búsqueda de Fibonacci

Algoritmo sin hilos

```
//******************************************************************//Qu
intana Martínez Erick
//Curso: Análisis de algoritmos
//(C) noviembre 2020
//ESCOM-IPN
//Buscar un elemento en un arreglo utilizando el algoritmo de búsqueda
Fibonacci.
//versión
1.1//******************************************************************/
/******************************************************************//LIB
RERIAS
INCLUIDAS//*****************************************************************
*****#include <stdio.h>
#include <stdlib.h>

/*Si se encuentra el número se retorna el índice, en caso
contrariodevolverá -1 */
int busquedaFibonacci(int *A, int n, int x){
    int i;
    int fibMMm2 = 0;
    int fibMMm1 = 1;
    int fibM = fibMMm2 + fibMMm1;

      while (fibM < n){
        fibMMm2 = fibMMm1;
        fibMMm1 = fibM;
        fibM  = fibMMm2 + fibMMm1;
      }

      int offset = -1;

      while(fibM > 1){
        int i = min(offset+fibMMm2, n-1);

        if (A[i] < x){
          fibM  = fibMMm1;
          fibMMm1 = fibMMm2;
          fibMMm2 = fibM - fibMMm1;
          offset = i;
         }

         elseif (A[i] > x){
           fibM  = fibMMm2;
           fibMMm1 = fibMMm1 - fibMMm2;
           fibMMm2 = fibM - fibMMm1;
         }
```

```
          else return i;
       }

       if(fibMMm1 && A[offset+1]==x)
          return offset+1;

       return -1;
}

int min(int x, int y){
   return (x<=y)? x : y;
}

//PROGRAMA PRINCIPAL
int main(){
    int n, x, i;

    scanf"%d %d", &n, &x);

    // Se crea un arreglo dinamico de n elementos enteros
    int *A = malloc(sizeof(int) * n);

    // Se reciben los datos del arreglo
    for(i = 0; i < n; ++i)
       scanf("%d", &A[i]);

    int encontrado;

    encontrado = busquedaFibonacci(A,n,x);

    printf("\n%d",encontrado);

    return 0;
}
```