Tarea 2. Análisis de algoritmos

FRANCISCO ACUÑA FRANCO, A01027294
ISAAC GARZA STRIMLINGAS, A01025798
SEBASTIÁN GÓMEZ GUTIÉRREZ, A01374843
SERGIO HERNÁNDEZ CASTILLO, A01025210
RODRIGO SIBAJA VILLARREAL, A01023121

Se implementaron dos árboles de estructura de datos: el AVL y el B. Se probaron diferentes elementos, en los métodos de inserción, borrado y búsqueda. Se creó gráficas comparativas en las que se obtuvo el resultado de que en el Árbol AVL y en el Árbol B. Después de comparar los resultados ejecución de ambas implementaciones podemos observar que ninguno es absolutamente mejor que otro, pues si bien el AVL es más estable que el B el árbol B es más rápido en algunos aspectos. Sin contar que no deberían usarse para lo mismo ya que no cumplen el mismo propósito. FInalmente en las comparaciones se pretende demostrar la utilidad de cada uno y así facilitar cuál es mejor utilizar.

ACM Reference Format:
Francisco Acuña Franco, Isaac Garza Strimlingas, Sebastián Gómez Gutiérrez, Sergio Hernández Castillo, and Rodrigo Sibaja Villarreal. 2020. Tarea 2. Análisis de algoritmos. 1, 1, Article 1 (March 2020), 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCCIÓN

Esta tarea consiste de realizar dos implementaciones de árboles de estructura de datos: la primera implementación de árboles es el AVL, una modificación del árbol binario de búsqueda, que trabaja exclusivamente en memoria, también se utiliza para conjuntos de datos más pequeños. Tiene incluido el auto balance de los datos del árbol a través de las operaciones de rotación simples y dobles.

El segundo árbol que se analizó fue el el B; este mismo se debe usar preferiblemente para manipularse en disco, debido al tamaño que puede llegar a alcanzar, ya que puede que la memoria no sea suficiente. Este árbol se implementó con el uso de archivos binarios, en el que se guardan nodos y cuándo se necesitan, se usaron en memoria.

Cada árbol se analizó a través de una Raspberry Pi y con diferentes cantidades de elementos: de diez a un millón en la inserción, diez elementos aleatorios en la búsqueda y diez en el borrado. Se crearon gráficas comparativas dependiendo del tiempo, para luego ser analizadas y sacar conclusiones diferentes de cada árbol implementado.

Authors' addresses: Francisco Acuña Franco, A01027294; Isaac Garza Strimlingas, A01025798; Sebastián Gómez Gutiérrez, A01374843; Sergio Hernández Castillo, A01025210; Rodrigo Sibaja Villarreal, A01023121.

XXXX-XXXX/2020/3-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 DESARROLLO

2.1 Características del sistema

La aplicación para probar los dos tipos de árboles fue programada en C++. Consecuentemente, la librería utilizada para medir los tiempos de ejecución fue "chrono". De esta librería se utilizó el método "highresolutionclock::now()", el cual mide tiempo, pero dependiendo de un punto de partida. Para medir los tiempos, definimos 3 variables: "start", "stop", y "duration". Cada vez que "start" es declarado, "highresolutionclock::now()" empieza a tomar el tiempo y solo para cuando se defina "stop". Finalmente, "duration" calcula cuánto tiempo hubo entre "start" y "stop".

2.2 Mecanismo utilizado para medir tiempos

El mecanismo utilizado para medir los tiempos fue una Raspberry Pi 3 Modelo B v1.2. Sus especificaciones relevantes son las siguientes: procesador de 64 bits/Quad Core de 1.2 GHz, memoria RAM de 1GB y una tarjeta microSD de 16 GB.

3 ÁRBOL AVL

Tiempo (microsegundos) de generación de cada árbol:

Elementos dentro de árbol	Tiempo de creación
10	24
100	423
1,000	4348
10,000	57487
100,000	500579
1,000,000	6906044

10 números generados con tiempo (microsegundos) de búsqueda y borrado:

10 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	23	3	No
2	20	2	No
3	19	2	Si
4	21	1	No
5	19	1	No
6	19	1	No
7	19	1	No
8	19	1	No
9	19	1	No
10	20	1	No
Total:	198	14	

100 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	37	4	No
2	24	4	No
3	23	3	No
4	23	2	No
5	54	3	No

6	30	3	No
7	26	3	No
8	25	3	No
9	28	3	No
10	20	3	No
Total:	290	31	

1000 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	69	5	No
2	25	4	No
3	29	3	No
4	23	3	No
5	28	3	No
6	29	3	No
7	32	3	No
8	28	3	No
9	27	3	No
10	23	3	No
Total:	313	33	

10000 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	32	9	No
2	24	5	No
3	30	5	No
4	29	5	No
5	28	5	Si
6	29	5	No
7	30	5	No
8	23	5	No
9	28	5	Si
10	28	5	No
Total	281	54	

100000 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	31	5	No
2	13	3	No
3	12	3	No
4	13	3	No
5	14	3	No
6	15	12	Si
7	12	3	No
8	12	3	No
9	13	3	Si
10	12	3	No

Total	147	41	

1000000 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	20	7	No
2	13	4	No
3	14	4	No
4	12	4	No
5	12	4	No
6	12	4	No
7	12	4	Si
8	12	4	No
9	12	4	No
10	12	3	Si
Total	131	42	

4 ÁRBOL B

Tiempo (microsegundos) de generación de cada árbol:

Elementos dentro de árbol	**Tiempo de creación**
10	23
100	194
1,000	1873
10,000	22441
100,000	218179
1,000,000	2582183

10 números generados con tiempo (microsegundos) de búsqueda y borrado:

10 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	1	34	No
2	1	18	No
3	0	14	No
4	1	14	No
5	1	13	No
6	1	14	No
7	0	1	Si
8	0	44	No
9	0	17	No
10	0	17	No
Total:	5	186	

100 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	1	57	No
2	1	1	Si
3	0	17	No
4	1	19	No

5	1	16	No
6	0	19	No
7	0	15	No
8	0	18	No
9	0	18	No
10	0	17	No
Total:	4	197	

1000 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	1	58	No
2	1	18	No
3	0	2	Si
4	1	19	No
5	1	20	No
6	1	17	No
7	1	18	No
8	1	18	No
9	1	17	No
10	1	17	No
Total:	9	204	

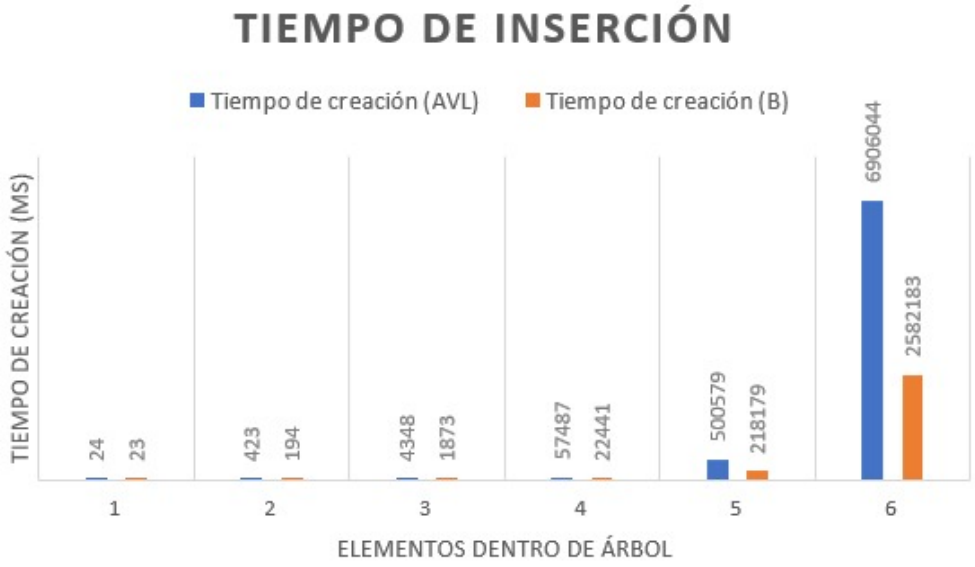
10000 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	2	123	No
2	1	75	No
3	1	2	No
4	1	63	Si
5	2	62	No
6	1	2	No
7	1	70	No
8	1	70	No
9	1	91	No
10	1	66	No
Total	12	624	

100000 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	2	44	No
2	0	5	No
3	1	5	No
4	0	5	No
5	0	5	Si
6	0	5	No
7	0	5	No
8	1	5	No
9	0	5	Si
10	0	5	No

Total	4	119

1000000 elementos	Tiempo de búsqueda	Tiempo de borrado	Encontrado
1	2	42	No
2	0	12	No
3	1	10	No
4	1	9	No
5	1	9	No
6	0	9	No
7	0	9	No
8	0	9	No
9	0	9	No
10	0	1	Si
Total	5	119	

5 GRÁFICAS COMPARATIVAS



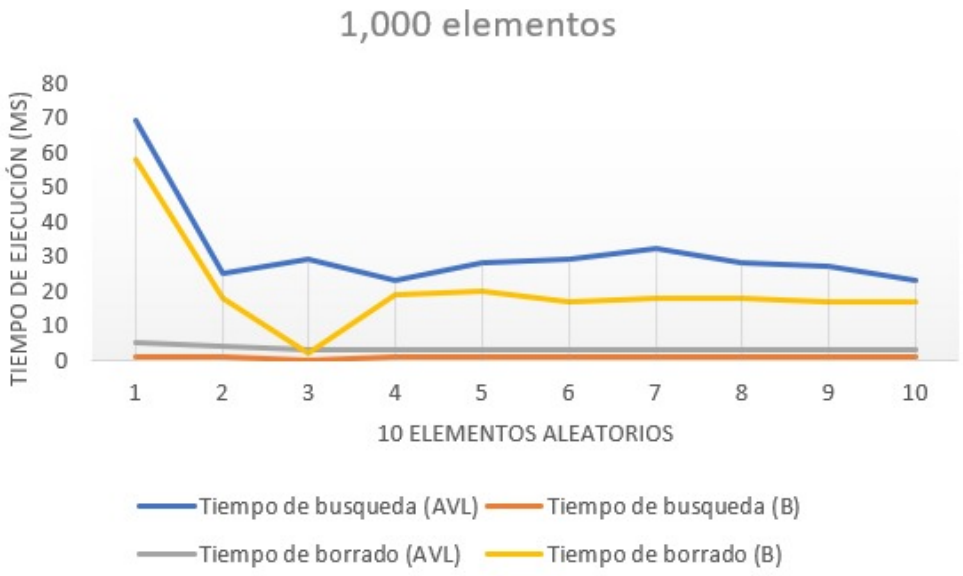
NOTA: 1 = 10, 2 = 100, 3 = 1,000, 4 = 10,000, 5 = 100,000, 6 = 1,000,000

Como podemos observar en esta gráfica, la diferencia de tiempos del árbol B y del AVL es de una brecha que se va abriendo más conforme a la cantidad de elementos insertados; el B demuestra ser más rápido que el AVL, esto nos demuestra que el crecimiento "horizontal" es más rápido que el "vertical".

Gráficas de búsqueda y eliminación:

Se puede apreciar que el tiempo de búsqueda de un Árbol AVL es mucho más lento que el de un árbol B, ya que en 10 elementos no tiene tantos hijo como el AVL. En tiempo de borrado, el AVL es más rápido y constante que el B, debido a la cantidad de balanceos que se generan en uno y otro.

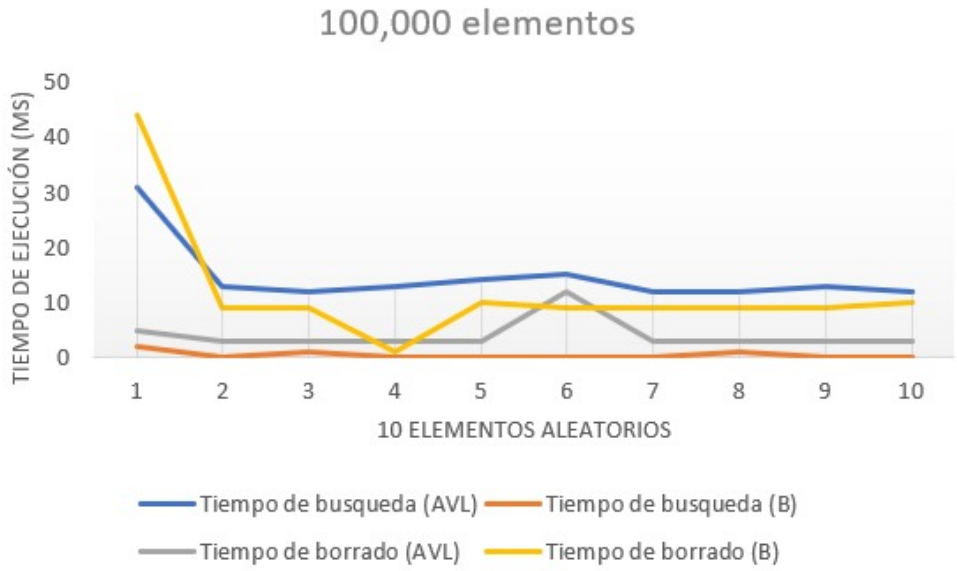
Se puede apreciar que el tiempo de búsqueda de un Árbol AVL es mucho mas lento que el de un árbol B ya que en 100 elementos no tiene tantos hijo como el AVL, cabe notar que la brecha es aún más grande. En tiempo de borrado, el AVL es más rápido y constante que el B, debido a la cantidad de balanceos que se generan en uno y otro, pero en si la brecha entre los dos, es más cercana.



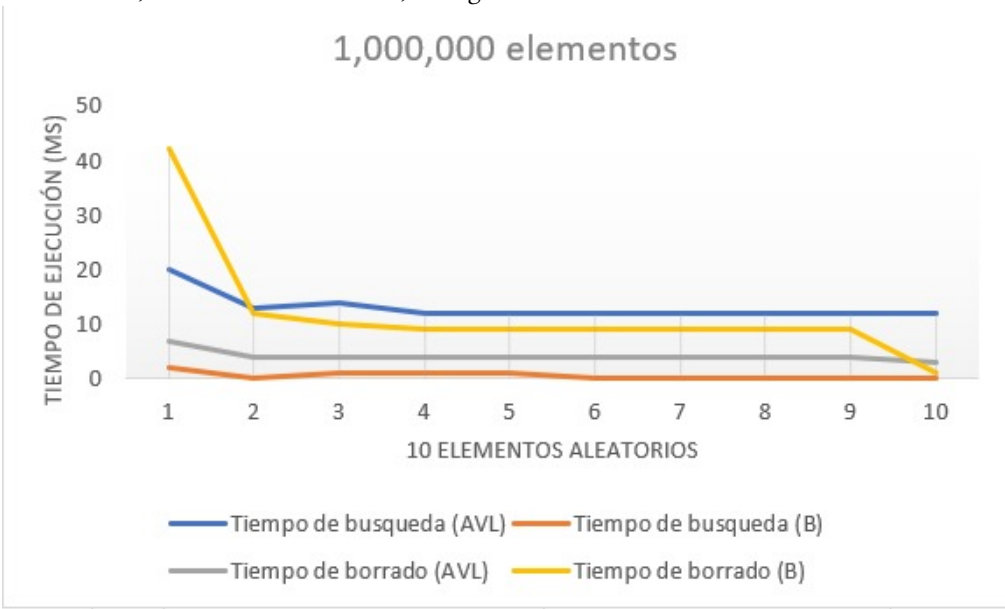
Se puede apreciar que el tiempo de búsqueda de un Árbol AVL es mucho más lento que el de un árbol B, ya que en 1000 elementos no tiene tantos hijo como el AVL, cabe notar que la brecha es aún más grande. En tiempo de borrado, el AVL es más rápido y constante que el B, debido a la cantidad de balanceos que se generan en uno y otro, pero en si la brecha entre los dos, es más cercana. Cabe notar que las líneas del B y del AVL se parecen mucho, pero en cuanto a la comparación del borrado con insrción.

Se puede apreciar que el tiempo de búsqueda de un Árbol AVL es mucho más lento que el de un árbol B, ya que en 10,000 elementos no tiene tantos hijo como el AVL, cabe notar que la brecha se disminuyó entre no y otro. En tiempo de borrado, el AVL es más rápido y constante que el B,

debido a la cantidad de balanceos que se generan en uno y otro, pero en si la brecha entre los dos, es más cercana. La gráfica deltiempo del borrado del B, se puede notar que tiene un comportamiento normal.



Se puede apreciar que el tiempo de búsqueda de un Árbol AVL es mucho más lento que el de un árbol B, ya que en 100,000 elementos no tiene tantos hijo como el AVL, cabe notar que la brecha se disminuyó entre uno y otro. En tiempo de borrado, el AVL es más rápido y constante que el B, debido a la cantidad de balanceos que se generan en uno y otro, pero en si la brecha entre los dos, es más cercana, inclusive en momentos, se llegan a cruzar.



Se puede apreciar que el tiempo de búsqueda de un Árbol AVL es mucho más lento que el de un árbol B, ya que en 1,000,000 elementos no tiene tantos hijo como el AVL, cabe notar que las gráficas se hicieron más constantes. En tiempo de borrado, el AVL es más rápido y constante que el B, debido a la cantidad de balanceos que se generan en uno y otro, pero en si la brecha entre los dos, es más cercana; al final el B se volvió más rápido que el AVL.

6 CONCLUSIONES

De acuerdo a los resultados obtenidos, los tiempos de ejecución de operaciones fueron cambiando de acuerdo al número de elementos insertados, en la mayoría de los casos manteniéndose constante , excepto en los tiempos de generación de elementos donde los tiempos del AVL crecían más rápido que los del B con la introducción de más elementos. Los resultados de búsqueda en ambos árboles fueron constantes, pero no con el mismo tiempo, las búsquedas en el árbol B fueron más rápidas. Por otra parte, El tiempo de borrado para el AVL fue rápido y de manera constante, mientras que el árbol B variaba mucho en el tiempo de búsqueda.

7 REFERENCIAS

- Benchoff, B. (2016). Introducing The Raspberry Pi 3. Retrieved March 19, 2020, from *https://hackaday.com/2 the-raspberry-pi-3/*
- GeeksforGeeks. (2017). Chrono in C . Retrieved March 19, 2020, from*https://www.geeksforgeeks.org/chrono- in-c/*
- GeeksforGeeks. (2019). Delete Operation in B-Tree. Retrieved March 17, 2020, from*https://www.geeksforgee operation-in-b-tree/*
- GeeksforGeeks. (2019, August 7). AVL Tree: Set 2 (Deletion). Retrieved March 20, 2020, from https://www.geeksforgeeks.org/avl-tree-set-2-deletion/