Tarea 2: Análisis de Algoritmos

CARLA PÉREZ GAVILÁN DEL CASTILLO, A01023033
RUBÉN HERNÁNDEZ, A01024669
CHRISTIAN ALBERTO DALMA, A01423166

En el siguiente reporte se desarrollan implementaciones de un árbol B y un árbol AVL, con el objetivo de realizar una comparación en los tiempos de inserción, borrado y búsqueda entre ambos. De forma que se pueda evaluar las ventajas y desventajas de la implementación de una estructura de datos en disco (externa) o en memoria (interna). Así como identificar los factores que permiten la mejora de su eficiencia. Por ejemplo, cómo es que pequeñas alteraciones en su implementación pueden afectar al tiempo de ejecución.

CCS Concepts: • **AVL Tree**; • **Algorithm time complexity**; • **Algorithm Efficiency**; • **B Tree**; • **external and internal sorting**;

Additional Key Words and Phrases: sorting, data structure, file management, algorithms

ACM Reference Format:
Carla Pérez Gavilán Del Castillo, Rubén Hernández, and Christian Alberto Dalma. 2020. Tarea 2: Análisis de Algoritmos. 1, 1 (March 2020), 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCIÓN

Árbol B

Un árbol B es aquel que agrupo de forma ordenada a sus nodos en estrcuturas de tamaño N, que se les puede denominar páginas.

Todos los ábroles B deben de cumplir con las siguientes características:

- Siendo n, la cantidad mínimo de nodos por página; 2n será la cantidad máxima de nodos al que también se le denominará la letra N.
- Todos las páginas deben cumplir con la condición de tener al menos 2n nodos, a excpeción de la raíz que podrá tener como mínimo 2 nodo.
- Todos los nodos deben cumplir con un orden ascendente de izquierda a derecha. Por lo tanto todo nodo a la derecha de otro, es menor a este.
- Cada nodo apunta hacia otros dos páginas, la página izquierda tendrá nodos menores a este y la derecha mayores.
- Se le denomina nodo hoja a aquel cuyos apuntadores son todos nulos.

NOTA: Para que se pueda impementar una estructura como esta en disco se debe hacer uso de archivos binarios, de forma que en lugar de apuntadores seamos capacez de acceder a los hijos de un nodo padre por medio de índices que indiquen la posición en el archivo del nodo hijo.

Authors' addresses: Carla Pérez Gavilán Del Castillo, A01023033; Rubén Hernández, A01024669; Christian Alberto Dalma, A01423166.

XXXX-XXXX/2020/3-ART $15.00
https://doi.org/10.1145/1122445.1122456

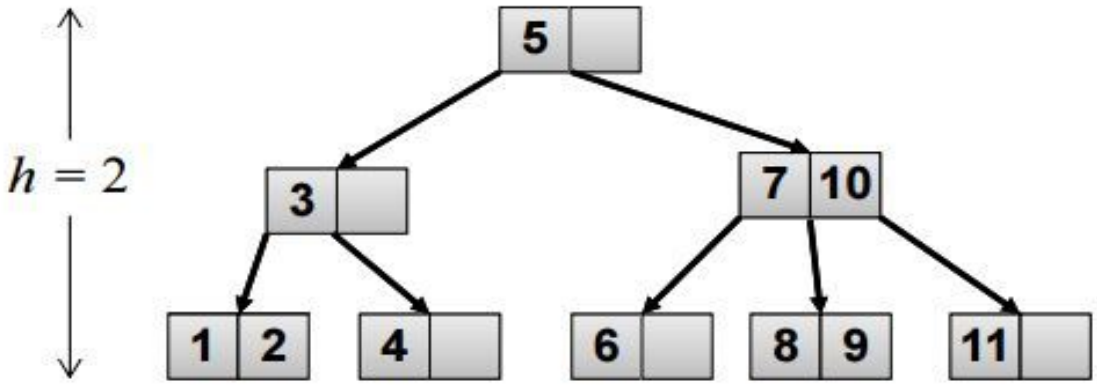

Fig. 1. Ejemplo de árbol B donde n=2, y por lo tanto N=4

Ábrbol AVL

Un árbol AVL consiste en un árbol binario de búsqueda, que debe cumplir con la condición de siempre estar balanceado. Es por ello que debe cumplir con las siguientes características:

- Por ser un ABB cumple con la condición de que todos sus subárboles izquierdos deben ser menores a los del subárbol derecho, es por ello que los nodos siempre permanecen en orden ascendiente de izquierda a derecha.
- El mantenerse balanceado consiste en que el factor de balanceo entre el árbol derecho y el árbol izquierdo (calculado por la diferencia entre la altura de uno menos la altura del otro) debe ser a lo sumo 1.
- Después de una inserción se debe retornar al estado de balanceo.

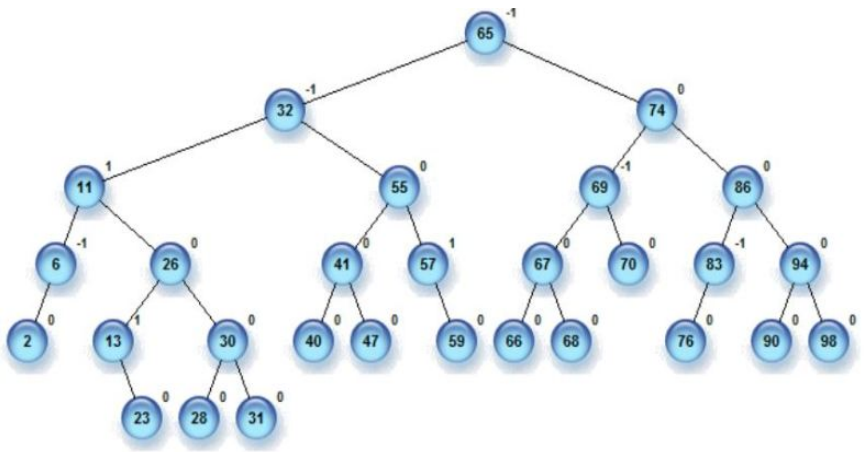

Fig. 2. Ejemplo de árbol AVL donde se muestran los factores de balanceo

2 DESARROLLO

Con el objetivo de realizar una comparación adecuada de la eficiencia de ambas estructuras, se deasarrolló una implementación y se realizaron mediciones de tiempos para ambos. Los resultados se presentan a continuación, mostrados en segundos.

2.1 Caracterísiticas del sistema utilizado para mediciones

Se utilizó una raspberry pi 3 modelo B+, con las siguientes características:

- CPU y GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz de cuatro núcleos, 1.4GHz
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps)
- 4 puertos USB, 1 hdmi
- RAM: 1GB LPDDR2 SDRAM
- GPIO de 40 pines

2.2 Mecanismo utilizado para medir tiempos de ejecución

Se utiliza la librería perteneciente a std denominada chrono. Esta cuenta con varios relojes. Se utilizó el high resolution clock que utiliza el reloj más preciso del sistema. Para más información ver cita número [1] en la sección de referencias bibliográficas.

2.3 Árbol AVL

Table 1. 10 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.005	0.0183
1	0.0019	0.0029
2	0.0014	0.0035
3	0.0023	0.0032
4	0.0018	0.0031
5	0.0028	0.0025
6	0.0016	0.0034
7	0.0019	0.003
8	0.0021	0.0033
9	0.002	0.0028
Total	0.0183	0.046

Table 2. 100 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.0021	0.0094
1	0.0031	0.0082
2	0.0024	0.007
3	0.0027	0.0062
4	0.0022	0.0041
5	0.0018	0.0036
6	0.00179	0.0031
7	0.0025	0.0035
8	0.003	0.0034
9	0.007	0.0083
Total	0.0285	0.0568

Table 3. 1,000 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.024	0.0195
1	0.013	0.018
2	0.005	0.0072
3	0.02	0.0291
4	0.09	0.0165
5	0.008	0.0154
6	0.0021	0.0099
7	0.002	0.0068
8	0.001	0.0102
9	0.028	0.007
Total	0.1931	0.3151

Table 4. 10,000 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.022	0.086
1	0.28	0.042
2	0.26	0.046
3	0.31	0.049
4	0.36	0.069
5	0.23	0.043
6	0.34	0.06
7	0.33	0.051
8	0.3	0.084
9	0.24	0.042
Total	0.287	0.489

Table 5. 100,000 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.049	0.092
1	0.077	0.153
2	0.056	0.093
3	0.062	0.102
4	0.027	0.069
5	0.047	0.084
6	0.064	0.112
7	0.032	0.066
8	0.058	0.156
9	0.15	0.068
Total	0.622	0.995

Table 6. 1,000,000 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.38	0.289
1	0.47	0.337
2	0.41	0.61
3	0.35	0.218
4	0.32	0.141
5	0.4	0.36
6	0.36	0.353
7	0.37	0.57
8	0.53	0.71
9	0.43	0.81
Total	4.02	4.39

2.4 Árbol B

Table 7. 10 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.00002	0.00003
1	0.00043	0.00057
2	0.00053	0.00077
3	0.00033	0.00044
4	0.00072	0.00088
5	0.00055	0.00079
6	0.00003	0.00013
7	0.00031	0.00046
8	0.00021	0.00034
9	0.00046	0.00062
Total	0.00359	0.00503

Table 8. 100 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.0066	0.0093
1	0.0062	0.0099
2	0.0089	0.0134
3	0.0083	0.0102
4	0.0077	0.0098
5	0.0058	0.0088
6	0.0062	0.0091
7	0.0089	0.0105 8
0.0072	0.0094	
9	0.008	0.0111
Total	0.0732	0.1015

Table 9. 1,000 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.043	0.066
1	0.053	0.079
2	0.076	0.089
3	0.034	0.061
4	0.048	0.091
5	0.061	0.064
6	0.049	0.093
7	0.038	0.091
8	0.051	0.077
9	0.047	0.052
Total	0.51	0.763

Table 10. 10,000 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	0.23	0.53
1	0.39	0.61
2	0.41	0.57
3	0.26	0.58
4	0.29	0.51
5	0.32	0.69
6	0.47 0.79	
7	0.28	0.49
8	0.46	0.67
9	0.44	0.65
Total	3.55	6.09

Table 11. 100,000 elementos

Número	Tiempo de búsqueda	Tiempo de eliminación
0	1.05	2.44
1	2.33	5.62
2	1.58	2.94
3	1.77	4.01
4	3.01	6.11
5	2.58	5.06
6	1.33	3.29
7	3.10	6.58
8	2.54	5.99
9	1.66	3.47
Total	20.95	45.51

2.5 Gráficas comparativas

Table 12. Comparación de inserciones de elementos

Número	Árbol AVL	Árbol B
10	0.03	0.00051
100	0.52	0.0409
1,000	3.44	0.2933
10,000	36.60	5.3321
100,000	292.02	221.0882
1,000,000	500.47	Segmentation Fault

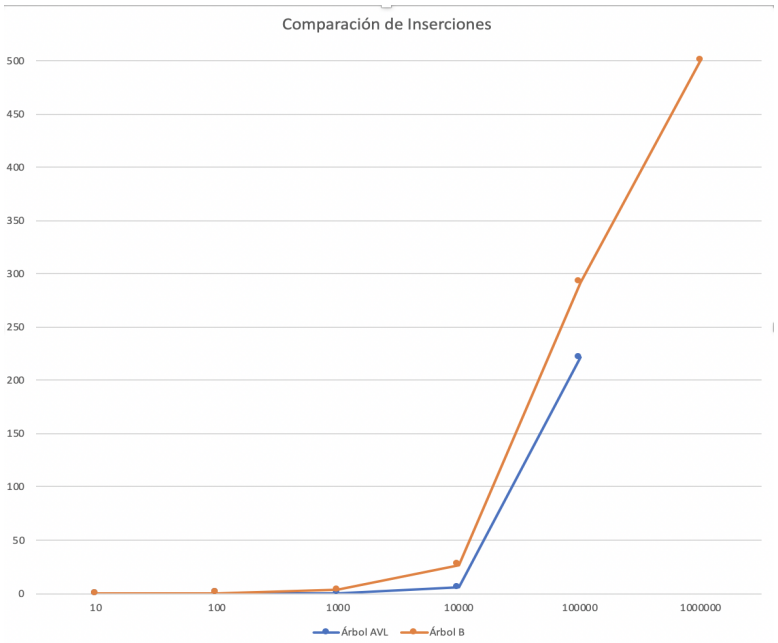

Fig. 3. Gráfica comparativa de los tiempos de inserción de Árbol B vs Árbol AVL

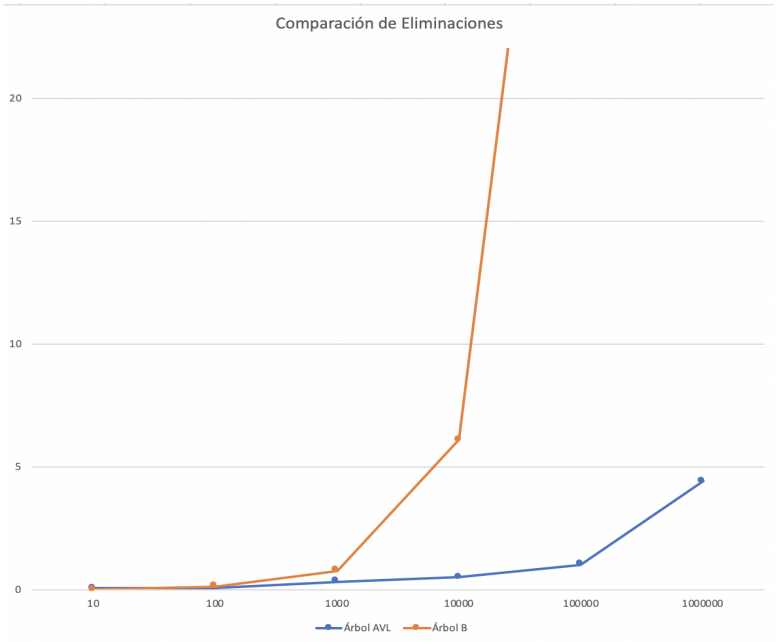

Fig. 4. Gráfica comparativa de los tiempos de eliminación de Árbol B vs Árbol AVL

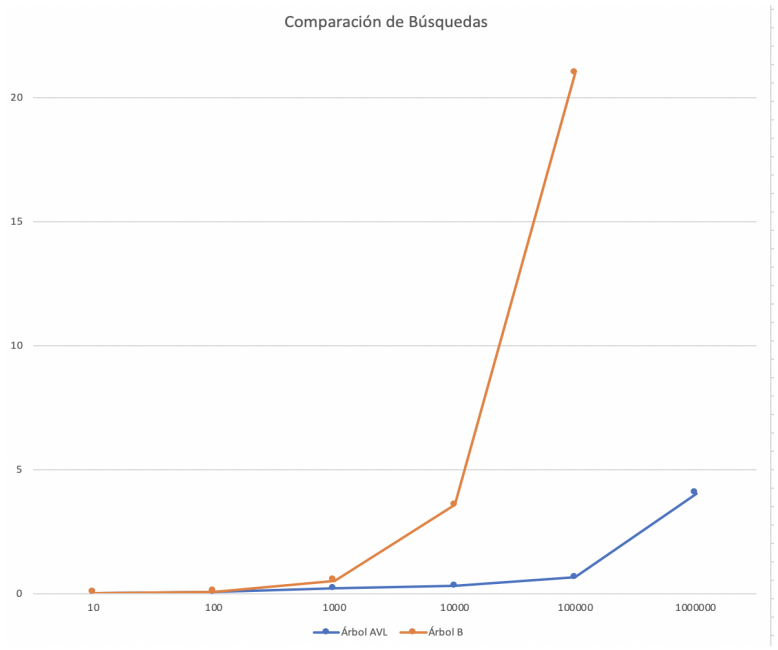

Fig. 5. Gráfica comparativa de los tiempos de búsqueda de Árbol B vs Árbol AVL

3 CONCLUSIONES

La eficiencia de una estructura de datos sobre otra dependerá completamente de la cantidad de nodos que se insertan. Debido a su implementación en disco, el árbol B es ideal para cuando se manejan grandes cantidades de nodos. Esto se debe a que en dicho caso, no se puede usar únicamente los registros de memoria para almacenar todos los nodos. No obstante, el árbol AVL (como se muestra en las tablas y en la comparación gráfica) es mucho más eficiente al tratarse de búsquedas e inserciones.

No obstante, a pesar de que confirmamos las declaraciones anteriores, la implementación ineficiente del árbol B hace imposible verificar que este tipo de estructura es capaz de manejar grandes cantidades de nodos. No obstante, si lograramos desarrollar una agloritmo de manera más eficiente, quizás utilizando un arreglo de valores dentro de la estructura del nodo que permitiese guardar los nodos hijos de cada nodo padre, podríamos simplificar el acceso.

4 REFERENCIAS BIBLIOGRÁFICAS

- Acceso aleatorio a un fichero [Sitio Web]. (4 de febrero de 2007). Recuperado el 22 de marzo de 2020 de http://www.chuidiang.org/clinux/ficheros/acceso-aleatorio-ficheros.php
- Función fseek ANSI C [Sitio Web]. (9 de septiembre de 2000). Recuperado el 22 de marzo de 2020 de http://c.conclase.net/librerias/?ansifun=fseek
- Gil, R. (14 de marzo de 2020). 5 easy steps to getting started using Raspberry Pi [Sitio Web]. Recuperado el 22 de marzo de 2020 de https://www.imore.com/how-get-started-using-raspberry-pi
- Introduction of B-Tree[Sitio Web]. (2018). Recuperado el 22 de marzo de 2020 de https://www.geeksforgeek of-b-tree-2/
- Vaca. C. (9 de febrero de 2011). Estructuras de Datos y Algoritmos. Recuperado el 22 de marzo de https://www.infor.uva.es/ cvaca/asigs/doceda/tema4.pdf
- Velasco, R. (02 de julio de 2018). Análisis: Raspberry Pi 3 Modelo B+ [Sitio Web]. Recuperado el 22 de marzo de 2020 de https://hardzone.es/reviews/perifericos/analisis-raspberry-pi-3-modelo-b/