Dos Árboles y una Búsqueda

Matías Salim Seda Auil
Universidad de Chile

Los árboles de búsqueda binaria (ABB) son una estructura de datos que permite ordenar un conjuntos de datos , añadir y extraer elementos del conjunto y realizar consultas sobre la existencia o no de un elemento en el conjunto de manera eficiente. Sin embargo, existen situaciones en donde la eficiencia de los ABB no es la más óptima. Por esta razón, existen distintas versiones de los ABB que intentan mejorar parte de las deficiencia de los ABB normales. Los árboles del tipo *single pointer tree* son ABB que presentan mejoras respecto a los árboles de búsqueda binaria normales.

Así, en el siguiente trabajo se calculó y analizó el costo promedio de una búsqueda infructuosa en un single pointer tree y se comparó el valor obtenido con el costo promedio de una búsqueda infructuosa en un ABB normal, obteniendo como principal resultado que, efectivamente, el costo promedio de una búsqueda infructuosa en un single pointer tree es, generalmente, es más bajo que el costo promedio de una búsqueda infructuosa en un árbol de búsqueda binario normal. Asimismo, la diferencia de eficiencia es más significativa mientras más llaves posee el árbol.

De esa forma, se concluyó que un single pointer tree no solo permite una eficiencia respecto a la cantidad de punteros presentes por llave sino que, además, las búsquedas infructuosas y, por tanto, la inserciones en un single pointer tree son más eficientes que en un ABB normal.

PRESENTACIÓN DEL PROBLEMA: UNA LLAVE, UN PUNTERO

Los famosos árboles de búsqueda binaria (ABB) son una poderosa estructura de datos que permite ordenar un conjuntos de datos , añadir y extraer elementos del conjunto y realizar consultas sobre la existencia o no de un elemento en el conjunto de manera eficiente. En particular, la búsqueda infructuosa de un elemento en un ABB, en promedio, toma un costo dado por la siguiente expresión:

$$C_n = 2(H_{n+1} - 1)$$

Con H_{n+1} el $(n+1)$-ésimo número armónico.

A pesar de todas las bondades de los árboles de búsqueda binaria, existen diversos escenarios y situaciones en donde la eficiencia de los ABB no es la más óptima. Por esta razón, existen variadas *versiones* de los ABB que intentan resolver estos problemas.

Una de estas versiones de los árboles de búsqueda binaria son los *single pointer trees*, llamados así ya que, en promedio, poseen un puntero por llave, en lugar de los dos que posee un ABB normal. Este tipo de árbol se caracteriza por que cada nodo interno almacena un par de llaves de rango consecutivo (esto es, tales que no hay ninguna otra llave del árbol que se ubique entre ellas), mientras las hojas pueden contener cero o una llaves. Un ejemplo de un *single pointer tree* es el siguiente:

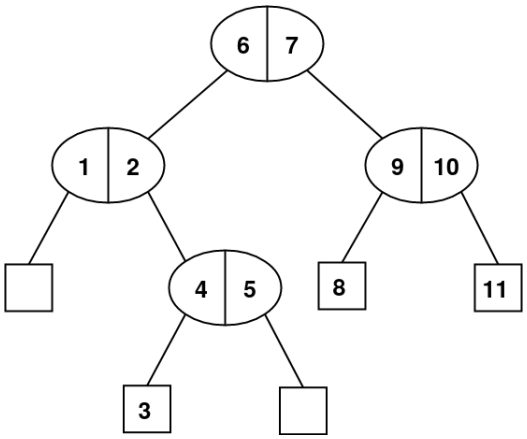

Figura 1. Ejemplo de un single pointer tree

Ahora, respecto a la inserción en un single pointer tree, cuando se inserta un elemento, si éste cae en medio de dos llaves que forman un par, se decide al azar a cuál de las dos llaves *expulsas* para que la llave recién insertada ocupe su lugar. La llave expulsada sigue su camino de inserción hacia abajo. Cuando cae una llave en una hoja vacía, ésta pasa a tener una llave, y cuando llega la segunda, se transforma en un nodo interno con dos hojas vacías como hijos.

Así, dado los árboles de búsqueda binaria *normales* y los single pointer tree, gustaria comparar el costo promedio de búsqueda infructuosa de un elemento en cada tipo de árbol para determinar si realmente los árboles de tipo single pointer tree presentan mejoras considerables respecto a un ABB el relación a el proceso de búsqueda infructuosa de un elemento en un árbol.

RESOLUCIÓN DEL PROBLEMA: PROBABLEMENTE ES LA INSERCIÓN CORRECTA

Nótese que el árbol de la figura 1 presenta 11 llaves y, siguiendo la lógica de un árbol de búsqueda binario normal, el árbol presenta 12 puntos de inserción. En general, en un single pointer tree de n llaves, existen $n+1$ puntos de inserción. Respecto a a lógica de la inserción en un single pointer tree, la manera de insertar llaves genera que existan hojas donde es más probable insertar una llave que en otras. Así, se puede definir el peso de una hoja como la cantidad de puntos de inserción que están asociada a la hoja. En particular, respecto al árbol de ejemplo de la figura 1, los *pesos* de cada nodo se grafican en la siguiente figura:

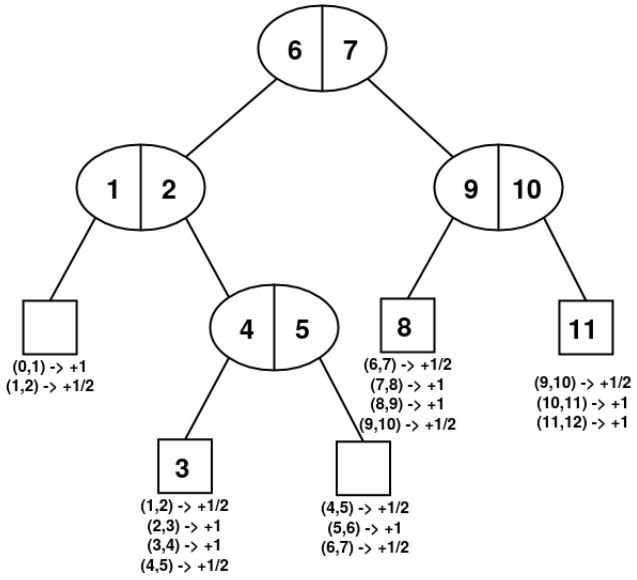

Figura 2. Ejemplo de un single pointer tree con los pesos de cada hoja

En la figura 2 se explicita de donde provienen los valores del peso de cada hoja. Nótese existen valores fraccionarios ($\frac{1}{2}$) que aportan al peso de una hoja. Esto es debido a que en el proceso de inserción existen casos donde se elige aleatoriamente que llave expulsar en un nodo interno. Dado que hay hojas con distintos pesos, se pueden definir seis tipos de hojas que pueden existir en uno de estos árboles.

1. Hoja inicial vacía.
2. Hoja inicial con una llave.
3. Hoja extrema vacía.
4. Hoja extrema con una llave.
5. Hoja interna vacía.
6. Hoja interna con una llave.

En el árbol de la figura 1, una hoja extrema vacía es una hoja del tipo como la hoja más a la izquierda en el árbol (la hoja izquierda del nodo interno $(1,2)$). Una hoja extrema con una llave es una hoja del tipo como la hoja

que contiene a la llave 11. Una hoja interna vacía es una hoja del tipo como la hoja derecha del nodo interno $(4, 5)$. Finalmente, una hoja interna con una llave es una hoja del tipo como la hoja que contiene a la llave 8. En particular, siempre existen dos hojas extremas y éstas son la hoja más a la derecha en el árbol y la hoja más a la izquierda en el árbol.

La hoja inicial vacía solo existe cuando se tiene un single pointer tree con 0 llaves. Al agregar una llave a este árbol vacío, la hoja inicial vacía se transforma en una hoja inicial con una llave. Luego, al agregar otra llave, la hoja inicial con una llave se transforma en un nodo interno con dos hojas extremas vacías. En este punto, al seguir agregando llaves, el crecimiento del árbol sigue la lógica planteada anteriormente respecto al proceso de inserción.

Respecto a los puntos de inserción por cada tipo de hoja, se puede mostrar que siempre se cumple lo siguiente:

- Una hoja inicial vacía siempre presenta 1 puntos de inserción
- Una hoja incial con una llave siempre presenta 2 puntos de inserción
- Una hoja extrema vacía siempre presenta $\frac{3}{2}$ puntos de inserción
- Una hoja extrema con una llave siempre presenta $\frac{5}{2}$ puntos de inserción.
- Una hoja interna vacía siempre presenta 2 puntos de inserción.
- Una hoja interna con una llave siempre presenta 3 puntos de inserción

Dada la cantidad de inserciones por cada tipo de hoja, para un single pointer tree de n llaves, se tiene que la probabilidad de insertar una llave en cada tipo de hoja es la siguiente:

- $P\{$ Inserción en una hoja inicial vacía$\} = \frac{1}{n+1}$
- $P\{$ Inserción en una hoja inicial con una llave$\} = \frac{2}{n+1}$
- $P\{$ Inserción en una hoja extrema vacía$\} = \frac{3}{2(n+1)}$
- $P\{$ Inserción en una hoja extrema con una llave$\} = \frac{5}{2(n+1)}$
- $P\{$ Inserción en una hoja interna vacía$\} = \frac{2}{n+1}$
- $P\{$ Inserción en una hoja interna con una llave$\} = \frac{3}{n+1}$

Ahora, sea $a^j_{n,k}$ el número esperado de hojas del tipo j en el nivel k de un single pointer tree de n llaves. Al agregar una nueva llave, la cantidad de hojas esperadas para cada tipo de hoja se modifica de la siguiente forma:

1. Hoja inicial vacía:

$$a^1_{n,k} \to a^1_{n,k} - \frac{1}{n+1} a^1_{n,k}$$

Para el caso de una hoja inicial vacía, en primer lugar, se tiene que el lado izquierdo de la expresión, es decir, $a^1_{n,k}$ representa la cantidad esperada de hojas iniciales vacías antes de agregar una nueva llave.

La flecha $\to$ representa el cambio en la cantidad esperada de hojas iniciales vacías cuando se agrega una nueva llave. Así, el lado derecho de la expresión, es decir, $a^1_{n,k} - \frac{1}{n+1} a^1_{n,k}$ representa la cantidad esperada de hojas iniciales vacías despúes de agregar una nueva llave.

Cuando se agrega una nueva llave, ésta pueda quedar una hoja inicial vacía o no. Si la llave queda en una hoja inicial vacía, la cantidad de hojas iniciales vacías disminuye. Por el otro lado, si la llave no queda en un hoja inicial vacía, la cantidad de hoja iniciales se mantiene. Así, La expresión $a^1_{n,k} - \frac{1}{n+1} a^1_{n,k}$ representa la cantidad esperada de hojas iniciales vacías dado los dos escenarios planteados.

2. Hoja inicial con una llave.

$$a^2_{n,k} \to a^2_{n,k} - \frac{2}{n+1}a^2_{n,k} + \frac{1}{n+1}a^1_{n,k}$$

Para el caso de una hoja inicial con una llave, en primer lugar, se tiene que el lado izquierdo de la expresión, es decir, $a^2_{n,k}$ representa la cantidad esperada de hojas iniciales con una llave antes de agregar una nueva llave.

La flecha $\to$ representa el cambio en la cantidad esperada de hojas iniciales con una llave cuando se agrega una nueva llave. Así, el lado derecho de la expresión, es decir, $a^2_{n,k} - \frac{2}{n+1}a^2_{n,k} + \frac{1}{n+1}a^1_{n,k}$, representa la cantidad esperada de hojas iniciales con una llave despúes de agregar una nueva llave.

Cuando se agrega una nueva llave, ésta pueda quedar una hoja inicial con una llave o no. Si la llave queda en una hoja inicial con una llave, la cantidad de hojas iniciales con una llave disminuye. Por el otro lado, si la llave no queda en un hoja inicial con una llave, la cantidad de hojas con una llave se mantiene. Además, en un último caso, la llave puede quedar en una hoja inicial vacía. En este último caso, la cantidad hojas iniciales con una llave aumenta. Así, la expresión $a^1_{n,k} - \frac{1}{n+1}a^1_{n,k}$ representa la cantidad esperada de hojas iniciales con una llave dado los escenarios planteados.

3. Hoja extrema vacía.

$$a^3_{n,k} \to a^3_{n,k} - \frac{3}{2(n+1)}a^3_{n,k} + \frac{5}{2(n+1)}a^4_{n,k-1} + \frac{2}{n+1}a^2_{n,k-1}2$$

Para el caso de una hoja extrema vacía, en primer lugar, se tiene que el lado izquierdo de la expresión, es decir, $a^3_{n,k}$ representa la cantidad esperada de hojas extremas vacías antes de agregar una nueva llave.

La flecha $\to$ representa el cambio en la cantidad esperada de hojas extremas vacías cuando se agrega una nueva llave. Así, el lado derecho de la expresión, es decir, $a^3_{n,k} - \frac{3}{2(n+1)}a^3_{n,k} + \frac{5}{2(n+1)}a^4_{n,k-1} + \frac{2}{n+1}a^2_{n,k-1}2$ representa la cantidad esperada de hojas extremas vacías despúes de agregar una nueva llave.

Cuando se agrega una nueva llave, ésta pueda quedar una hoja extrema vacía o no. Si la llave queda en una hoja extrema vacía, la cantidad de hojas extremas vacías disminuye. Por el otro lado, si la llave no queda en un hoja extrema vacía, la cantidad de hojas extremas vacías se mantiene. Además, la llave puede quedar en una hoja inicial con una llave. En este caso, la cantidad hojas extrema vacías aumenta. Del mismo modo, la llave puede quedar en una hoja extrema con una llave. En este caso, la cantidad hojas extrema vacías aumenta. Así, la expresión $a^3_{n,k} - \frac{3}{2(n+1)}a^3_{n,k} + \frac{5}{2(n+1)}a^4_{n,k-1} + \frac{2}{n+1}a^2_{n,k-1}$ representa la cantidad esperada de hojas extremas vacías dado los dos escenarios planteados.

4. Hoja extrema con una llave.

$$a^4_{n,k} \to a^4_{n,k} - \frac{5}{2(n+1)}a^4_{n,k} + \frac{3}{2(n+1)}a^3_{n,k}$$

Para el caso de una hoja extrema con una llave, en primer lugar, se tiene que el lado izquierdo de la expresión, es decir, $a^4_{n,k}$ representa la cantidad esperada de hojas extremas con una llave antes de agregar una nueva llave.

La flecha $\to$ representa el cambio en la cantidad esperada de hojas extremas con una llave cuando se agrega una nueva llave. Así, el lado derecho de la expresión, es decir, $a^4_{n,k} - \frac{5}{2(n+1)}a^4_{n,k} + \frac{3}{2(n+1)}a^3_{n,k}$ representa la cantidad esperada de hojas extremas con una llave despúes de agregar una nueva llave.

Cuando se agrega una nueva llave, ésta pueda quedar una hoja extrema con una llave o no. Si la llave queda en una hoja extrema vacía, la cantidad de hojas extremas con una llave disminuye. Por el otro lado, si la llave no queda en un hoja extrema con una llave, la cantidad de hojas extremas vacías se mantiene. Además, en un último caso, la llave puede quedar en una hoja extrema vacía. En este último caso, la cantidad hojas extrema con una llave aumenta. Así, la expresión $a^4_{n,k} - \frac{5}{2(n+1)}a^4_{n,k} + \frac{3}{2(n+1)}a^3_{n,k}$ representa la cantidad esperada de hojas iniciales con una llave dado los escenarios planteados.

5. Hoja interna vacía.

$$a^5_{n,k} \to a^5_{n,k} - \frac{2}{n+1}a^5_{n,k} + \frac{5}{2(n+1)}a^4_{n,k-1} + \frac{3}{n+1}a^6_{n,k-1}2$$

Para el caso de una hoja interna vacía, en primer lugar, se tiene que el lado izquierdo de la expresión, es decir, $a^5_{n,k}$ representa la cantidad esperada de hojas internas vacías antes de agregar una nueva llave.

La flecha $\to$ representa el cambio en la cantidad esperada de hojas internas vacías cuando se agrega una nueva llave. Así, el lado derecho de la expresión, es decir, $a^5_{n,k} - \frac{2}{n+1}a^5_{n,k} + \frac{5}{2(n+1)}a^4_{n,k-1} + \frac{3}{n+1}a^6_{n,k-1}2$ representa la cantidad esperada de hojas internas vacías despúes de agregar una nueva llave.

Cuando se agrega una nueva llave, ésta pueda quedar una hoja interna vacía o no. Si la llave queda en una hoja interna vacía, la cantidad de hojas internas vacías disminuye. Por el otro lado, si la llave no queda en un hoja interna vacía, la cantidad de hojas internas vacías se mantiene. Además, la llave puede quedar en una hoja extrema con una llave. En este caso, la cantidad hojas internas vacías aumenta. Del mismo modo, la llave puede quedar en una hoja interna con una llave. En este caso, la cantidad hojas internas vacías aumenta. Así, la expresión $a^5_{n,k} - \frac{2}{n+1}a^5_{n,k} + \frac{5}{2(n+1)}a^4_{n,k-1} + \frac{3}{n+1}a^6_{n,k-1}2$ representa la cantidad esperada de hojas internas vacías dado los escenarios planteados.

6. Hoja interna con una llave.

$$a^6_{n,k} \to a^6_{n,k} - \frac{3}{n+1}a^6_{n,k} + \frac{2}{n+1}a^5_{n,k}$$

Para el caso de una hoja interna con una llave, en primer lugar, se tiene que el lado izquierdo de la expresión, es decir, $a^6_{n,k}$ representa la cantidad esperada de hojas extremas con una llave antes de agregar una nueva llave.

La flecha $\to$ representa el cambio en la cantidad esperada de hojas internas con una llave cuando se agrega una nueva llave. Así, el lado derecho de la expresión, es decir, $a^6_{n,k} - \frac{3}{n+1}a^6_{n,k} + \frac{2}{n+1}a^5_{n,k}$ representa la cantidad esperada de hojas internas con una llave despúes de agregar una nueva llave.

Cuando se agrega una nueva llave, ésta pueda quedar una hoja interna con una llave o no. Si la llave queda en una hoja interna con una llave , la cantidad de hojas internas con una llave disminuye. Por el otro lado, si la llave no queda en un hoja interna con una llave, la cantidad de hojas interna con una llave se mantiene. Además, en un último caso, la llave puede quedar en una hoja interna vacía. En este último caso, la cantidad hojas internas con una llave aumenta. Así, la expresión $a^6_{n,k} - \frac{3}{n+1}a^6_{n,k} + \frac{2}{n+1}a^5_{n,k}$ representa la cantidad esperada de hojas internas con una llave dado los escenarios planteados.

Lo seis puntos anteriores explican como varía la cantidad de hojas de un tipo en un nivel en particular al agregar una nueva llave. Para tener una descripción más general que abarque todos los niveles, se puede aplicar función generatriz a las seis expresiones para conocer las distribuciones de hojas de cada tipo. Así, las distribuciones mencionadas derivan de los seis puntos anteriores y vienen dadas por la siguientes igualdades:

1. Hoja inicial vacía

$$a^1_{n+1}(z) = a^1_n(z) - \frac{1}{n+1}a^1_n(z)$$

2. Hoja inicial con una llave.

$$a^2_{n+1}(z) = a^2_n(z) - \frac{2}{n+1}a^2_n(z) + \frac{1}{n+1}a^1_n(z)$$

3. Hoja extrema vacía.

$$a^3_{n+1}(z) = a^3_n(z) - \frac{3}{2(n+1)}a^3_n(z) + \frac{5z}{2(n+1)}a^4_n(z) + \frac{4z}{n+1}a^2_n(z)$$

4. Hoja extrema con una llave.

$$a^4_{n+1}(z) = a^4_n(z) - \frac{5}{2(n+1)}a^4_n(z) + \frac{3}{2(n+1)}a^3_n(z)$$

5. Hoja interna vacía.

$$a^5_{n+1}(z) = a^5_n(z) - \frac{2}{n+1}a^5_n(z) + \frac{5z}{2(n+1)}a^4_n(z) + \frac{6z}{n+1}a^6_n(z)$$

6. Hoja interna con una llave.

$$a^6_{n+1}(z) = a^6_n(z) - \frac{3}{n+1}a^6_n(z) + \frac{2}{n+1}a^5_n(z)$$

Nótese que definiendo el vector $\vec{a_n}$ como:

$$\vec{a_n}(z) = \begin{bmatrix} a^1_n(z) \\ a^2_n(z) \\ a^3_n(z) \\ a^4_n(z) \\ a^5_n(z) \\ a^6_n(z) \end{bmatrix}$$

y definiendo la matriz $H(z)$ como:

$$H(z) = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 4z & -\frac{3}{2} & \frac{5z}{2} & 0 & 0 \\ 0 & 0 & \frac{3}{2} & -\frac{5}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{5z}{2} & -2 & 6z \\ 0 & 0 & 0 & 0 & 2 & -3 \end{bmatrix}$$

las seis ecuaciones asociades a las funciones generatrices para $a^i_n(z)$ con $1 \leq i \leq 6$ pueden ser escritas de la siguiente forma matricial:

$$\vec{a_n}(z) = (I + \frac{1}{n+1}H(z))\vec{a_{n-1}}(z)$$

Nótese que en un single pointer vacío, solo existe una hoja inicial vacía en el nivel 0. Por lo tanto, la condición inicial $\vec{a_0}(z)$ del vector $\vec{a_n}(z)$ es tal que:

$$\vec{a_0}(z) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Con la información conocida, se puede calcular un valor explicito para $\vec{a_n}(z)$. Recuérdese que el vector $\vec{a_n}(z)$ contiene la información sobre la distribución de los distintos tipos de hojas en todos los niveles de un árbol de n llaves.

Ahora, dado el vector $\vec{a_n}(z)$, gustaría conocer la función generatriz $p_n(z)$ para el costo de hacer una búsqueda infructuosa y así poder conocer el costo promedio de una búsqueda infructuosa en un single pointer tree.

Definiendo un vector $\vec{c}(z)$ tal que:

$$\vec{c}(z) = \begin{bmatrix} 1 \\ 2 \\ \frac{3}{2} \\ \frac{5}{2} \\ 2 \\ 3 \end{bmatrix}^T$$

Se tiene que la distribución de probabilidad $p_n(z)$ que contiene la información sobre la búsqueda infructuosa de un elemento en un single pointer tree viene dada por la siguiente expresión:

$$p_n(z) = \frac{1}{n+1}\vec{c}(z) \cdot \vec{a_n}(z)$$

Finalmente, para conocer el valor $\hat{C_n}$, es decir, el costo promedio de una búsqueda infructuosa en un single pointer tree se tiene que conocer expresiones explícitas para $\vec{a_n}(z)$ y $p_n(z)$. Resolver manualmente las distribuciones $\vec{a_n(z)}$ y $p_n(z)$ es una tarea compleja y, por tanto, se utilizará la ayuda de un programa de álgebra computacional.

Con los calculos realizados computancionalmente[(1)], se tiene que el valor $\hat{C_n}$, es decir, el costo promedio de una búsqueda infructuosa en un single pointer tree es el siguiente:

$$\hat{C_n} = \frac{12}{7}H_{n+1} - \frac{10564n + 6889}{3920(n+1)}$$

ANÁLISIS DE LA SOLUCIÓN: EL ORDEN DE LAS BÚSQUEDAS

Nótese que en primer lugar, respecto a la fracción $\frac{10564n+6889}{3920(n+1)}$ asociada al valor $\hat{C_n}$, se tiene que:

$$\lim_{n\to\infty} \frac{10564n + 6889}{3920(n+1)} = \frac{10564}{3920} \approx 2{,}6948$$

Por lo tanto, dado que $O(H_n) = \log n$, se tiene que $O(\hat{C_n}) = \log n$. Ahora, recordando que el valor C_n, es decir, el costo esperado de una búsqueda infructuosa de un elemento en un ABB normal, viene dada por la expresión:

$$C_n = 2(H_{n+1} - 1)$$

Al igual que en un single pointer tree, se tiene que el costo promedio de una búsqueda infructuosa tiene un crecimiento logarítmico, es decir, se cumple que:

$$O(C_n) = O(\hat{C_n}) = \log n$$

Nótese que tanto C_n como $\hat{C_n}$ son de orden $\log n$. Sin embargo, gustaría tener una comparación más precisa de los dos valores. Así, tener un primer acercamiento a una comparación más precisa, la siguiente figura grafica tanto la función C_n como la función $\hat{C_n}$.

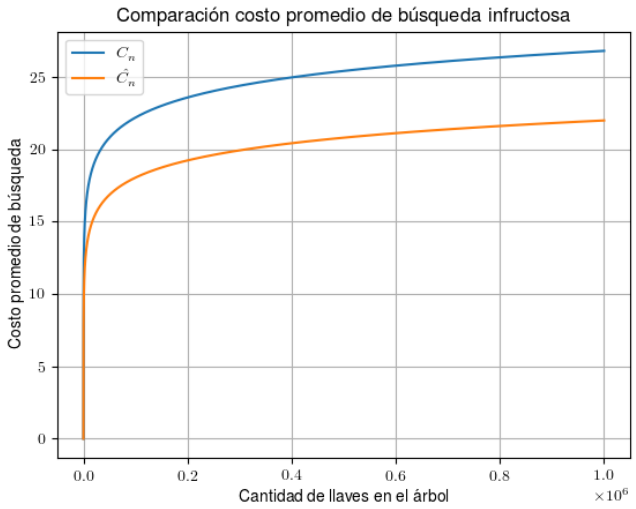

Figura 3. Comparación de costo promedio de búsqueda infructuosa entre un ABB normal y un single pointer tree

En la figura 3 se puede observar que, inicialmente, la diferencia del costo de búsqueda infructuosa entre un árbol de búsqueda binario normal y un single pointer tree es nula. Sin embargo, a medida que crece n, es decir, a medida que aumenta la cantidad de llaves presente en los árboles, la diferencia del costo de búsqueda entre los dos tipos de árboles empieza a ser más significativa.

Para seguir ahondando en la comparación entre los valores C_n y $\hat{C}_n$, es necesario conocer cómo se comporta la diferencia de los valores, es decir, que tipo de crecimiento posee la diferencia recién planteada. Para realizar lo anterior, la siguiente figura muestra un valor que representa la diferencia entre los costos esperados para una búsqueda infructuosa entre los dos tipos de árboles.

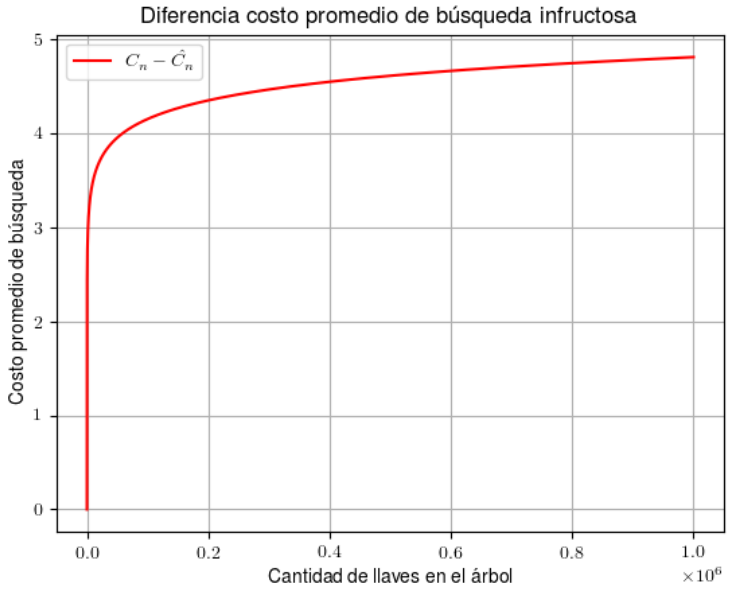

Figura 4. Diferencia de costo promedio de búsqueda infructuosa entre un ABB normal y un single pointer tree

La figura 4 pareciese mostrar que la diferencia de costo promedio de búsqueda infructuosa entre un ABB normal y un single pointer tree es de orden logarítmico. Para confirmar la hipótesis recién mencionada, se puede definir la función d_n tal que:

$$d_n = C_n - \hat{C}_n = 2(H_{n+1} - 1) - \left(\frac{12}{7}H_{n+1} - \frac{10564n + 6889}{3920(n+1)}\right)$$

Desarrollando la expresión asociada a d_n se tiene que:

$$d_n = \frac{2}{7}H_{n+1} - 2 + \frac{10564n + 6889}{3920(n+1)}$$

No es difícil observar que $O(d_n) = \log n$. Por lo tanto la diferencia de los costos esperados para una búsqueda infructuosa entre los dos tipos de árboles posee un crecimiento logarítmico.

DISCUSIONES Y CONCLUSIONES: LA IMPORANCIA DE LA BALANZA

En primer lugar, es interesante destacar que, la estructura de un single pointer tree es es una de las razones principales de la mejora de la eficiencia respecto a una búsqueda infructuosa. Al tener nodos internos con dos llaves, un single pointer evita tener muchos de los peores casos de un árbol de búsqueda binario normal. Por ejemplo, en un ABB normal, la permutación 1, 2, 3, ..., n genera un árbol que, esencialmente, es una lista enlazada. En ese caso, la búqueda infructuosa de un elemento es de orden $O(n)$. En cambio, en un single pointer tree esos casos extremos no existen o, en el peor de los casos, son listas mucho más cortas. Por tanto, en promedio, los single pointer trees

están más *balanceados* que los árboles de búsqueda binaria normales. De esa forma, al estar, en general, mejor balanceados, el costo promedio de una búsqueda infructuosa en un single pointer tree tiende a ser menor que en un ABB.

También es interesante destacar que esta *cualidad* de los single pointer tree de estar mejor balanceados es más relevante mientras más llaves posee el árbol. Lo recién mencionado es claro por lo observado en la figura 4 pero tiene sentido al pensar que si se tiene una permutación 1, 2, 3, ..., n con un n muy grande, la diferencia entre la altura genera por un árbol de búsqueda binaria normal y un single pointer tree dada la permutación será bastante grande, en comparación a si se tiene un n pequeño. En particular, se puede notar que si la permutación 1, 2, 3, ..., n genera un ABB normal de altura n, la misma permutación generará un single pointer tree de, apróximadamente, $\frac{n}{2}$. En ese sentido, no es difícil notar que la diferencia entre una función $f(x) = x$ y una función $g(x) = \frac{x}{2}$ es proporcional a x. La siguiente figura muestra las funciones mencionadas anteriormente.

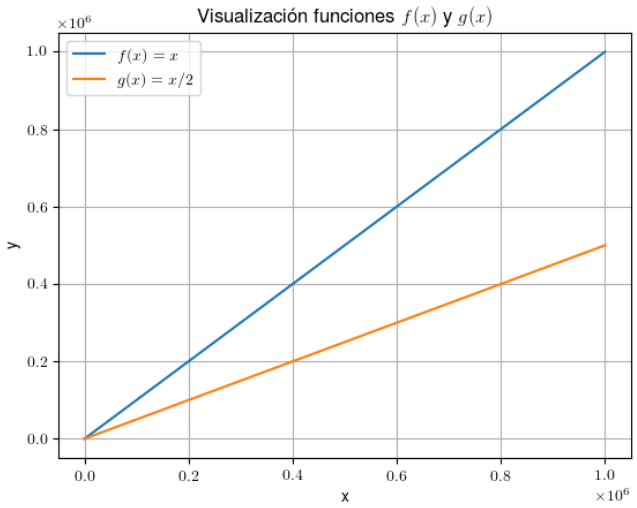

Figura 5. Comparación funciones $f(x) = x$ y $g(x) = \frac{x}{2}$

Así, finalmente, un single pointer tree no solo permite una eficiencia respecto a la cantidad de punteros presentes por llave sino que, además, en promedio, las búsquedas infructuosas y, por tanto, la inserciones en un single pointer tree son más eficientes que en un ABB normal y esta eficiencia es más notoria mientras más cantidad de llaves posee el árbol. Por lo tanto, para grandes cantidades de datos, reemplazar un árbol de búsqueda binaria normal por un single pointer tree trae mejoras en la eficiencia de manera significativa.

ANEXOS

(1) Se adjunta archivo con los cálculos realizados en Maple.