Tarea II – Segunda Parte

Estudiante: Sebastián Montero Castro – B95016

Fecha de Entrega: Octubre 21, 2020

Resumen – En este trabajo se busca implementar algunas estructuras de datos y realizar pruebas específicas que proveerán información importante para analizar la eficiencia y comprobar que esta corresponde a la planteada en la teoría. Las estructuras a ser implementadas son listas ordenadas y árboles binarios para la primera parte y tablas hash y árboles rojinegros para la segunda parte.

Lista Ordenada

La primera estructura de datos que analizaremos es la lista ordenada. Esta consiste de nodos que contienen la información y cada uno de estos nodos tiene, a su vez, referencias al nodo siguiente y al nodo anterior.

En esta implementación en específico se utilizó un nodo centinela para facilitar y simplificar los métodos de inserción, borrado y en general la estructura de la lista. El nodo centinela se encuentra entre el primer y último elemento, conectándolos por medio de sus referencias. Las inserciones y borrados se realizan a través de este nodo, utilizando sus referencias para realizar la operación deseada. Es simple comprender su función una vez que se visualiza:

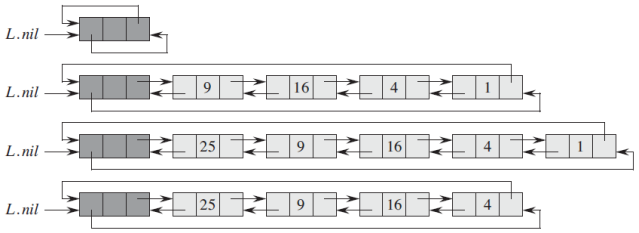

Figura 1. Lista Doble Ordenada Circular con Nodo Centinela. (Thomas, H.)

Una vez implementada la lista ordenada en C++, podemos proceder a realizar las pruebas correspondientes. Para estas crearemos una lista con n = 1 000 000 de nodos, los cuales tendrán datos según la prueba correspondiente.

Para la primera prueba, los datos en los nodos serán seleccionados aleatoriamente en un rango de 0 a *2n*. Una vez que tenemos la lista, procedemos a seleccionar elementos aleatorios en el mismo rango y buscarlos en la lista en un lapso de diez segundos. Al realizar esta primera prueba unas 3 veces, obtenemos los siguientes resultados:

Lista con elementos aleatorios	Cantidad de Búsquedas
Prueba 1	931
Prueba 2	861
Prueba 3	902

Al analizar los resultados, tenemos una media de aproximadamente 872 búsquedas. Debemos tomar en cuenta que estos resultados son dependientes de las especificaciones del computador en el que fueron procesados. Tendremos este número presente para futuras comparaciones y conclusiones.

La segunda prueba es similar a la primera. Tomamos una lista de un tamaño n = 1 000 000 y se llenan los nodos de manera 0, 1, … , n-1, en ese orden. Después, se seleccionan elementos al azar en el mismo rango del ejercicio anterior *(0 a 2n)* y se registran el número de búsquedas realizadas en 10 segundos. Al realizar esta prueba 3 veces, obtenemos como resultados:

Lista con elementos 0, … , n-1	Cantidad de Búsquedas
Prueba 1	962
Prueba 2	956
Prueba 3	972

Los resultados de esta prueba en comparación a la anterior son ligeramente más altos, con un promedio de 963 búsquedas. La búsqueda de un número aleatorio en una lista ya ordenada y sin elementos repetidos muy posiblemente presenta un tiempo de ejecución reducido debido a todos los factores que entran en juego en la búsqueda. Sin embargo, no hay prueba concreta que confirme esta teoría. Además, la diferencia entre búsquedas es relativamente baja, lo que lleva a pensar que la diferencia en tiempos de ejecución y efectividad es ínfima. Tomando los resultados de las pruebas y los datos conocidos como la complejidad de tiempo de búsqueda de las listas ordenadas, es decir O(n), podemos afirmar que los resultados fueron los esperados.

Árbol Binario

La segunda estructura de datos a ser analizada y estudiada será el árbol de búsqueda binario. Este consiste en un nodo raíz que guarda un elemento de información y contiene, a su vez, dos referencias a sus dos “hijos” (izquierda y derecha), los cuáles en sí funcionan como raíces, guardando elementos y sus propios hijos.

Las pruebas realizadas para analizar la eficiencia del árbol son las mismas que las utilizadas en la lista ordenada. Nuevamente se utilizará el valor de n = 1 000 000. La primer prueba

consistirá, igual que la anterior primer prueba, en insertar n elementos aleatorios en un rango de 0 a 2n, esta vez en un árbol. Procederemos a medir la cantidad de búsquedas realizadas en 10 segundos y anotaremos los resultados. Cabe destacar que, a diferencia de la lista, el árbol tiene búsquedas iterativas y recursivas en esta implementación. Es por esto que se analizarán ambos métodos de búsqueda y se tomarán los mejores resultados entre estos. Al realizar búsquedas en el árbol con elementos aleatorios tenemos que en 10 segundos:

Árbol con elementos aleatorios 0…2n (Búsqueda recursiva)	Cantidad de Búsquedas
Prueba 1	9607880
Prueba 2	9094770
Prueba 3	9071366

Árbol con elementos aleatorios 0…2n (Búsqueda iterativa)	Cantidad de Búsquedas
Prueba 1	10202785
Prueba 2	10303302
Prueba 3	10025140

En esta ocasión en específico, la búsqueda iterativa tuvo mejores resultados. Sin embargo, como podemos notar, no son resultados extremadamente alejados en relación del uno con el otro. Una posible razón para la mayor eficiencia del iterativo puede ser el gran tiempo constante que toma el frecuente llamado de métodos en el paradigma recursivo. Por ahora tomaremos los resultados iterativos para realizar el análisis.

Ahora, para realizar la segunda prueba, debemos insertar elementos en orden de 0 a n en un árbol. Esto presenta una gran dificultad debido al tiempo de ejecución tan exagerado que toma la inserción de elementos en orden. Para poder realizar esta prueba, debemos crear una alternativa que reduzca el tiempo de ejecución significativamente.

Como buscamos insertar elementos de manera ascendiente, el elemento a insertar SIEMPRE será el hijo derecho del nodo actual. Esto se debe a que el elemento a insertar siempre será mayor, lo cual en un árbol corresponde a una inserción a la derecha.

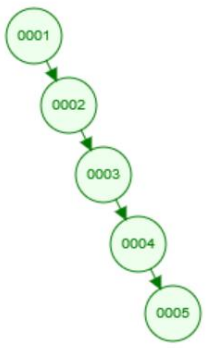

Figura 2. Árbol Binario de Búsqueda con elementos insertados ascendentemente.

También, el árbol tendrá una referencia auxiliar llamada “último”, la cual apuntará siempre al último nodo insertado. Esto ahorrará el recorrido del árbol constante y simplemente le asignará a este último nodo el nuevo elemento como su hijo derecho, actualizando la referencia del último nodo. Una vez implementada esta solución, podemos realizar las búsquedas de elementos aleatorios de 0 a 2n en 10 segundos, lo que nos da los siguientes resultados:

Árbol con elementos ordenados de 0…n (Búsqueda iterativa)	Cantidad de Búsquedas
Prueba 1	2276
Prueba 2	2257
Prueba 3	2257

Como podemos notar a simple vista, la diferencia de cantidad de búsquedas entre el árbol ordenado y el árbol aleatorio es absolutamente abismal. Mientras que el árbol con elementos aleatorios realiza millones de búsquedas, el ordenado llega ligeramente a los 2000. Esto se debe a que la búsqueda en un árbol ordenado se asimila bastante a la de una lista ordenada, pues los elementos se encuentran “en línea recta”.

Después de realizar todas estas pruebas, podemos llegar a múltiples conclusiones. La primera y más significativa es la superioridad del árbol cuando se habla eficiencia a la hora de buscar elementos. El árbol es una estructura de datos ligeramente más dinámica que la lista ordenada, la cual es muy unidimensional y limitada.

Era esperable que el árbol tuviera más búsquedas que la lista, pues la complejidad de tiempo del árbol binario de búsqueda es O(h), donde h se refiere a la propia altura del árbol. Esto, en definitiva, presenta una clara superioridad al tiempo de búsqueda de la lista, la cual en general posee O(n).

Por supuesto, existen estructuras de datos más complejas y eficientes en algunas áreas, las cuales se verán más adelante. Es evidente que el árbol es una estructura más completa que la lista, sin embargo, cada una tiene su uso específico, ventajas y desventajas.

Árbol Rojinegro

El árbol rojinegro es una estructura de datos que tiene similitudes con el árbol binario y, a su vez, resuelve algunas de las dificultades que presenta este. Funciona como un árbol binario, teniendo raíz y nodos hijos a la izquierda y derecha de cada nodo. La diferencia,

sin embargo, es que esta estructura implementa un sistema de colores (rojo y negro, como dice en el nombre) que se asignan a los nodos. Estos colores son cambiantes y determinan el balance del árbol por medio de un sistema de verificación de casos.

Para esta estructura de datos se realizarán las mismas pruebas de búsquedas que en las anteriores. Al realizar búsquedas (tanto iterativas como recursivas) en un árbol con elementos aleatorios, estos fueron los resultados:

Árbol Rojinegro con elementos aleatorios 0…2n (Búsqueda recursiva)	Cantidad de Búsquedas
Prueba 1	9343575
Prueba 2	9938097
Prueba 3	9205827

Árbol Rojinegro con elementos aleatorios 0…2n (Búsqueda iterativa)	Cantidad de Búsquedas
Prueba 1	10376242
Prueba 2	10199019
Prueba 3	10188623

Al igual que en el árbol rojinegro, la búsqueda iterativa obtuvo mejores resultados (10254628 búsquedas en promedio) y por ende será utilizada para responder las interrogantes con respecto a eficiencia.

Ahora, se insertarán elementos 0 a n, igual que en la sección de árbol binario, y se realizarán búsquedas iterativas en 10 segundos. Estos fueron los resultados:

Árbol Rojinegro con elementos ordenados de 0…n (Búsqueda iterativa)	Cantidad de Búsquedas
Prueba 1	18165384
Prueba 2	17069107
Prueba 3	16862714

A simple vista podemos observar la mejora entre búsquedas en un árbol con elementos aleatorios y uno con elementos ordenados. Estos resultados son distintos a los del árbol binario debido a la naturaleza del rojinegro, el cual balancea sus nodos y permite un acceso menos lineal.

Además, es evidente la mejora tan significativa en la cantidad de búsquedas del árbol rojinegro en comparación al árbol binario (y consecuentemente a la lista). Como el árbol rojinegro en su base es simplemente un binario con mejores de eficiencia, es esperado que los tiempos y resultados mejoren.

Esta superioridad se ve evidenciada incluso más en la cantidad de búsquedas en un árbol con elementos ordenados. El árbol binario no pasó de 3000 búsquedas, mientras que el rojinegro fácilmente promedio 17365735 búsquedas (¡una diferencia impresionante!).

Tablas de Dispersión

Las tablas de dispersión son estructuras de datos las cuales, por medio de funciones especificadas, son capaces de localizar datos guardados de manera eficiente. Cada dato guardado tiene, por decirlo de una manera, una llave que permite que sea rescatado de manera rápida.

Realizando las mismas pruebas que en las estructuras anteriores tenemos siguientes resultados:

Tablas Hash con elementos aleatorios 0…2n	Cantidad de Búsquedas
Prueba 1	18210153
Prueba 2	18599256
Prueba 3	18942038

A continuación observaremos los resultados con elementos insertados de manera ordenada y compararemos finalmente los tiempos de búsqueda de todas las estructuras en conjunto.

Tablas Hash con elementos ordenados 0…n	Cantidad de Búsquedas
Prueba 1	17321525
Prueba 2	17721766
Prueba 3	17965203

Como podemos observar, las tablas hash presentan los mejores números de búsquedas entre todas las estructuras de datos analizadas. La diferencia es considerable en comparación a los árboles binarios ordenados y las listas pero esta se reduce en comparación a los rojinegros. Las tablas hash con elementos aleatorios, en comparación a una con elementos ordenados, presentan una ligera ventaja en cuanto a búsquedas.

Entonces, realizando una comparación final, podemos ver a simple vista que las estructuras de datos analizadas tienen distintas funcionalidades, algunas más eficientes que otras dependiendo de su complejidad. Por supuesto, la lista simple es fácil de entender y de manejar pero su falta de accesibilidad y su rigidez hace que sea poco viable en términos de búsqueda en comparación a árboles binarios, los cuales en sí son menos eficientes que los rojinegros y tablas hash.

Comprendiendo esto podemos notar que el método de guardado de información más eficiente de los evaluados en este proyecto, juzgando exclusivamente por los resultados es el de las Tablas Hash. Este resultado es esperado, debido a que por su naturaleza y simplicidad los tiempos de búsqueda son casi inmediatos, con una complejidad de tiempo de O(1) cuando se tiene una fórmula eficiente y apropiada.

Bibliografía

Cormen, T., Leiserson, C., Rivest, R. & Stein, C.. (2009). Introduction To Algorithms. Massachusetts, EEUU: Massachusetts Institute of Technology.