Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Área de Ingeniería en Computadores

Curso CE-1103 Algoritmos y Estructuras de Datos I

Proyecto #2:
"SpiderSearch Engine: Stage 2"

Realizado por
Steven Ortiz Álvarez

Carnet
2014098717

Cartago - Mayo 12, 2015

Índice

Contenido

Bibliotecas y Funciones

Bibliotecas utilizadas:

- IO: contiene las clases relacionadas con entrada y salida de datos
  Clases utilizadas:
  - File: se enlaza con el nombre físico de un archivo
    Funciones Utilizadas:
    - ❖ Length: cuenta cantidad de archivos
    - ❖ getName: obtiene el nombre de los archivos
    - ❖ endsWith: compara el tipo de archivo
    - ❖ getPath: obtiene la dirección completa del archivo
    - ❖ exist: verifica que exista el directorio
- Apache tika: contiene las clases relacionadas con el parseo de archivos
  Clases utilizadas:
  - metadata: obtiene información del archivo (autor, versión, etc.)
  - parser: pasea el archivo obtiene todo el texto
- math: contiene las clases relacionadas a las funciones matemáticas
  Funciones Utilizadas:
  - max: obtiene el máximo entre dos números

Métodos

En las estructuras de datos desarrolladas de se implementaron varios métodos:

- Insertar: inserta un nodo a la estructura
- Buscar: busca un nodo en la estructura
- Eliminar: elimina un nodo de la estructura
- Imprimir: muestras todos los nodos de la estructura
- Recorrer: recorre todos los nodos de la estructura
- Largo: cuenta la cantidad de nodos de la estructura

- Nombre: asigna un nombre a la estructura
- Altura: obtiene la altura de un nodo de la estructura
- Rotaciones: realiza un movimiento de posiciones de los nodos de la estructura para mantener el equilibrio de la misma

Estructuras de datos

Estructuras de datos desarrolladas:

- Árbol binario de búsqueda:

  En ciencias de la computación, un árbol es una estructura de datos ampliamente usada que emula la forma de un árbol (un conjunto de nodos conectados). Un nodo es la unidad sobre la que se construye el árbol y puede tener cero o más nodos hijos conectados a él. Se dice que un nodo a es padre de un nodo b, si existe un enlace desde a hasta b (en ese caso, también decimos que b es hijo de a). Sólo puede haber un único nodo sin padres, que llamaremos raíz. Un nodo que no tiene hijos se conoce como hoja.

  El árbol También se define como una estructura de datos no lineal. Esta estructura se usa principalmente para representar datos con una relación jerárquica entre sus elementos, como por ejemplo registros, árboles genealógicos y tablas de contenidos. Entre otros tenemos un tipo especial de de árbol que es, llamado árbol binario, que puede ser implementado fácilmente en la computadora.

- Árbol Heap Sort:

  El ordenamiento por montículos (heapsort en inglés) es un algoritmo de ordenamiento no recursivo, no estable, con complejidad computacional \Theta(n\log n)

  Este algoritmo consiste en almacenar todos los elementos del vector a ordenar en un montículo (heap), y luego extraer el nodo que queda como nodo raíz del montículo (cima) en sucesivas iteraciones obteniendo el conjunto ordenado. Basa su funcionamiento en una propiedad de los montículos, por la cual, la cima contiene siempre el menor elemento (o el mayor, según se haya definido el montículo) de todos los almacenados en él. El algoritmo, después de cada extracción, recoloca en el nodo raíz o cima, la última hoja por la derecha del último nivel. Lo cual destruye la propiedad heap del árbol. Pero, a continuación realiza un proceso de "descenso" del número insertado de forma que se elige a cada movimiento el mayor de sus dos hijos, con el que se intercambia. Este intercambio, realizado sucesivamente "hunde" el nodo en el árbol restaurando la propiedad montículo del arbol y dejándo paso a la siguiente extracción del nodo raíz.

  El algoritmo, en su implementación habitual, tiene dos fases. Primero una fase de construcción de un montículo a partir del conjunto de elmentos de entrada, y después, una fase de extracción sucesiva de la cima del montículo. La implementación del almacén de datos en el heap, pese a ser conceptualmente un árbol, puede realizarse en un vector de forma fácil. Cada nodo tiene dos hijos y por tanto, un nodo situado en la posición i del vector, tendrá a sus hijos en las posiciones 2 x i, y 2 x i +1 suponiendo que el primer elemento del vector tiene un índice = 1. Es decir, la cima ocupa la posición inicial del vector y sus dos hijos la posición segunda y tercera, y así, sucesivamente. Por tanto, en la fase de ordenación, el intercambio ocurre entre el primer elemento del vector (la raíz o cima del árbol, que es el mayor

elemento del mismo) y el último elemento del vector que es la hoja más a la derecha en el último nivel. El árbol pierde una hoja y por tanto reduce su tamaño en un elemento. El vector definitivo y ordenado, empieza a construirse por el final y termina por el principio.

- Árbol AVL:

  El árbol AVL toma su nombre de las iniciales de los apellidos de sus inventores, Georgii Adelson-Velskii y Yevgeniy Landis. Lo dieron a conocer en la publicación de un artículo en 1962, An algorithm for the organization of information (Un algoritmo para la organización de la información).

  Los árboles AVL están siempre equilibrados de tal modo que para todos los nodos, la altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa. Gracias a esta forma de equilibrio (o balanceo), la complejidad de una búsqueda en uno de estos árboles se mantiene siempre en orden de complejidad O(log n). El factor de equilibrio puede ser almacenado directamente en cada nodo o ser computado a partir de las alturas de los subárboles.

  Para conseguir esta propiedad de equilibrio, la inserción y el borrado de los nodos se ha de realizar de una forma especial. Si al realizar una operación de inserción o borrado se rompe la condición de equilibrio, hay que realizar una serie de rotaciones de los nodos.

  Los árboles AVL más profundos son los árboles de Fibonacci.

- Árbol Splay:

  Un Árbol biselado o Árbol Splay es un Árbol binario de búsqueda auto-balanceable, con la propiedad adicional de que a los elementos accedidos

recientemente se accederá más rápidamente en accesos posteriores. Realiza operaciones básicas como pueden ser la inserción, la búsqueda y el borrado en un tiempo del orden de O(log n). Para muchas secuencias no uniformes de operaciones, el árbol biselado se comporta mejor que otros árboles de búsqueda, incluso cuando el patrón específico de la secuencia es desconocido. Esta estructura de datos fue inventada por Robert Tarjan y Daniel Sleator.

Todas las operaciones normales de un árbol binario de búsqueda son combinadas con una operación básica, llamada biselación. Esta operación consiste en reorganizar el árbol para un cierto elemento, colocando éste en la raíz. Una manera de hacerlo es realizando primero una búsqueda binaria en el árbol para encontrar el elemento en cuestión y, a continuación, usar rotaciones de árboles de una manera específica para traer el elemento a la cima. Alternativamente, un algoritmo "de arriba a abajo" puede combinar la búsqueda y la reorganización del árbol en una sola fase.

- Árbol Roji-Negro:

Un árbol rojo-negro es un tipo abstracto de datos. Concretamente, es un árbol binario de búsqueda equilibrado, una estructura de datos utilizada en informática y ciencias de la computación. La estructura original fue creada por Rudolf Bayer en 1972, que le dio el nombre de “árboles-B binarios simétricos”, pero tomó su nombre moderno en un trabajo de Leo J. Guibas y Robert Sedgewick realizado en 1978. Es complejo, pero tiene un buen peor caso de tiempo de ejecución para sus operaciones y es eficiente en la práctica. Puede buscar, insertar y borrar en un tiempo O(log n), donde n es el número de elementos del árbol.

Un árbol rojo-negro es un árbol binario de búsqueda en el que cada nodo tiene un atributo de color cuyo valor es rojo o negro. En adelante, se dice que un nodo es rojo o negro haciendo referencia a dicho atributo.

Además de los requisitos impuestos a los árboles binarios de búsqueda convencionales, se deben satisfacer las siguientes reglas para tener un árbol rojo-negro válido:

- Todo nodo es o bien rojo o bien negro.
- La raíz es negra.
- Todas las hojas (NULL) son negras.
- Todo nodo rojo debe tener dos nodos hijos negros.
- Cada camino desde un nodo dado a sus hojas descendientes contiene el mismo número de nodos negros.
-

Estas reglas producen una regla crucial para los árboles rojo-negro: el camino más largo desde la raíz hasta una hoja no es más largo que dos veces el camino más corto desde la raíz a una hoja. El resultado es que dicho árbol está aproximadamente equilibrado.

- Árbol B:

La idea tras los árboles-B es que los nodos internos deben tener un número variable de nodos hijo dentro de un rango predefinido. Cuando se inserta o se elimina un dato de la estructura, la cantidad de nodos hijo varía dentro de un nodo. Para que siga manteniéndose el número de nodos dentro del rango predefinido, los nodos internos se juntan o se parten. Dado que se permite un rango variable de nodos hijo, los árboles-B no necesitan rebalancearse tan frecuentemente como los árboles binarios de búsqueda auto-balanceables. Pero, por otro lado, pueden desperdiciar memoria, porque los nodos no permanecen totalmente ocupados. Los límites (uno superior y otro

inferior) en el número de nodos hijo son definidos para cada implementación en particular. Por ejemplo, en un árbol-B 2-3 (A menudo simplemente llamado árbol 2-3), nodo sólo puede tener 2 ó 3 nodos hijo.

Un árbol-B se mantiene balanceado porque requiere que todos los nodos hoja se encuentren a la misma altura.

Los árboles B tienen ventajas sustanciales sobre otras implementaciones cuando el tiempo de acceso a los nodos excede al tiempo de acceso entre nodos. Este caso se da usualmente cuando los nodos se encuentran en dispositivos de almacenamiento secundario como los discos rígidos. Al maximizar el número de nodos hijo de cada nodo interno, la altura del árbol decrece, las operaciones para balancearlo se reducen, y aumenta la eficiencia. Usualmente este valor se coloca de forma tal que cada nodo ocupe un bloque de disco, o un tamaño análogo en el dispositivo. Mientras que los árboles B 2-3 pueden ser útiles en la memoria principal, y además más fáciles de explicar, si el tamaño de los nodos se ajustan para caber en un bloque de disco, el resultado puede ser un árbol B 129-513.

Los creadores del árbol B, Rudolf Bayer y Ed McCreight, no han explicado el significado de la letra B de su nombre. Se cree que la B es de balanceado, dado que todos los nodos hoja se mantienen al mismo nivel en el árbol. La B también puede referirse a Bayer, o a Boeing, porque sus creadores trabajaban en los Boeing Scientific Research Labs por ese entonces.

Algoritmos

Algoritmos desarrollados:

- Buscar archivos en una carpeta: crea una lista de los archivos y los recorre y solo guarda los que sean necesarios por medio del filtro de archivos.
- Parsear texto: toma un archivo de texto y extrae el texto y lo almacena en un string.

Problemas Conocidos

Problemas conocidos:

- Crear la aplicación web para permitir el uso de la aplicación en múltiples usuarios
- Extraer archivos de un sitio web
- Extraer archivos de una carpeta compartida

Actividades

Actividades realizadas:

Fecha	**Labor**	**Horas**
24/4/2015	Nodo simple	2
25/4/2015	Nodo simple	2
26/4/2015	Lista simple enlazada	2
27/4/2015	Lista simple enlazada	2
28/04/2015	Nodo Árbol Binario	2
29/05/2015	Nodo Árbol Binario	2
30/5/2015	Nodo Árbol Avl	2
1/5/2015	Árbol Avl	2
2/5/205	Buscar archivos en carpeta	2
3/5/2015	Parsear pdf	2
4/5/2015	Crear servidor	2
5/5/2015	Crear cliente	2
6/5/2015	Javadoc nodo simple	2
7/5/2015	Javadoc lista simple	2
8/15/2015	Javadoc nodo árbol binario	2
9/15/2015	Javadoc árbol binario	2
10/5/2015	Javadoc nodo árbol avl	2
11/5/2015	Javadoc árbol avl	2
12/5/2015	documentación	2
	Total	**38 horas**

Problemas Encontrados

Se encontraron muchos problemas que provocaron un mal funcionamiento del programa SpiderSearch Engine: Stage 2, entre ellos cabe destacar que no se lograba crear un XML de manera correcta, ya que en muchas ocasiones el programa se caía y ocurre una excepción de Java, para solucionar este problema se procedió a trabajar con la biblioteca JespXML que ayuda a tener un mejor manejo de estos archivos, JespXML se encarga de crear y leer archivos XML por una serie de métodos ya establecidos en la biblioteca, lo que provocó un manejo adecuado y mejorado de estos archivos. Se recomienda mejorar la lectura de el mismo, ya que para lograr leer un “Tag” estos no pueden tener el mismo nombre, por lo tanto no se tener una lectura de los hijos distintos con el mismo nombre. Aun así la biblioteca JespXML está muy bien creada y facilita el manejo de archivos XML en Java.

Conclusiones y Recomendaciones

Al ser Java un lenguaje de programación orientada a objetos hace que el manejo del código sea mejorado, esto quiere decir que facilita a tener un programa más estructurado, simple, y limpio. Es por esto que el manejo de bibliotecas externas, la creación de nuestras propias clases con sus respectivos métodos hace que sea más fácil manejar nuestros proyectos.

Respectivamente con este proyecto se observa como la organización y el trabajo en equipo es fundamental para lograr el éxito, sin embargo no se pudo alcanzar las metas propuestas para este primer proyecto programado, siendo así la falta de comunicación y organización claves en el transcurso del mismo, se recomienda una mayor investigación para próximos proyectos.

Bibliografía

Villalobos, J. (2015). *Tutoriales de Java y Estructuras de datos - YouTube*. [Online] Youtube.com. Available at: https://www.youtube.com/playlist?list=PL1DEB0E818EB4AFC6 [Accessed 29 Apr. 2015].

Deitel, P., Deitel, H. and Vidal Romero Elizondo, A. (2008). *Java Como Programar*. México, D.F.: Pearson educacion.

García de Jalón de la Fuente, J. (1999). *Aprenda Java como si estuviera en primero*. San Sebastián: Universidad de Navarra. Escuela Superior de Ingenieros Industriales.

Jaimes, J. (2012). *Código de Java - Arboles binarios de búsqueda*. [online] Lawebdelprogramador.com. Available at: http://www.lawebdelprogramador.com/codigo/Java/2257-Arboles-binarios-de-busqueda.html [Accessed 2 May 2015].

Users.dcc.uchile.cl, (2015). *Computaci*[?]*n I*. [online] Available at: http://users.dcc.uchile.cl/~lmateu/CC10A/Apuntes/arboles/ [Accessed 11 May 2015].

V., J. and V., J. (2015). *Sockets en Java (cliente y servidor) | Codigoprogramacion*. [online] Codigoprogramacion.com. Available at: http://codigoprogramacion.com/cursos/java/103-sockets-en-java-con-cliente-y-servidor.html#.VVKgjeS3XOx [Accessed 11 May 2015].

Forosdelweb.com, (2015). *Foros del Web*. [online] Available at: http://www.forosdelweb.com/f45/recorrer-directorio-con-subcarpetas-1051479/ [Accessed 11 May 2015].

Anon, (2015). [online] Available at: http://web.fdi.ucm.es/profesor/fpeinado/courses/oop/LPS-17Hilos.pdf [Accessed 12 May 2015].

Tika.apache.org, (2015). *Apache Tika - Apache Tika*. [online] Available at: https://tika.apache.org/ [Accessed 12 May 2015].

GitHub Gists, (2012). *kinjouj/Sample.java*. [online] Available at: https://gist.github.com/kinjouj/2507727 [Accessed 12 May 2015].

Tutorialspoint.com, (2015). *TIKA - Extracting PDF*. [online] Available at: http://www.tutorialspoint.com/tika/tika_extracting_pdf.htm [Accessed 12 May 2015].

Programcreek.com, (2015). *Java Code Example for org.apache.tika.parser.Parser*. [online] Available at: http://www.programcreek.com/java-api-examples/index.php?api=org.apache.tika.parser.Parser [Accessed 12 May 2015].

Chuidiang.com, (2015). *Sincronización de hilos*. [online] Available at: http://www.chuidiang.com/java/hilos/sincronizar_hilos_java.php [Accessed 12 May 2015].

Es.wikipedia.org, (2015). *Heapsort*. [online] Available at: http://es.wikipedia.org/wiki/Heapsort [Accessed 10 May 2015].

Es.wikipedia.org, (2015). *Árbol AVL*. [online] Available at: http://es.wikipedia.org/wiki/%C3%81rbol_AVL [Accessed 10 May 2015].

Es.wikipedia.org, (2015). *Árbol biselado*. [online] Available at: http://es.wikipedia.org/wiki/%C3%81rbol_biselado [Accessed 10 May 2015].

Es.wikipedia.org, (2015). *Árbol rojo-negro*. [online] Available at: http://es.wikipedia.org/wiki/%C3%81rbol_rojo-negro [Accessed 10 May 2015].

Es.wikipedia.org, (2015). *Árbol-B*. [online] Available at: http://es.wikipedia.org/wiki/%C3%81rbol-B [Accessed 10 May 2015].

Html.rincondelvago.com, (2015). *Árboles binarios*. [online] Available at: http://html.rincondelvago.com/arboles-binarios.html [Accessed 10 May 2015].