Estructura de Datos y Algoritmos II

Grupo: 5

Semestre: 2019-2

Profesor: M.C. Tista GarcÃa Edgar

Integrantes:

- Calzada Martínez Jonathan Omar
- Jiménez Hernández Juan Carlos
- Garcia Lazcano Carlos David

0.1. TEMA

Algoritmos Árboles binarios de búsqueda equilibrados (AVL) y arboles B más.

0.2. Objetivo general:

Que el alumno implemente árboles las estructuras de datos de tipo árbol y que desarrolle sus habilidades en la programaciún orientada a objetos a través de la aplicaciún del concepto de árboles como estructuras de datos no lineales

0.3. Objetivo particular:

Lograr implementar los algoritmos de árbol binario balanceado y b+

0.4. Introducción:

Los Datos Abstractos se tratan de conjuntos de datos y / o objetos al que se le realizan operaciones. El TDA nos da un un entorno en el que cual se puede realizar las operaciones autorizadas, abstrayéndose de la manera en cómo están implementadas dichas operaciones.

El paradigma de orientación a objetos permite el encapsulamiento de los datos y las operaciones mediante la definición de clases e interfaces, lo cual permite ocultar la manera en cómo ha sido implementado el TDA y solo permité el acceso a los datos a través de las operaciones provistas por la interfaz.

0.5. Planteamiento del problema:

El estudio de los arboles b y b+ es un tema de importancia, ya que podría facilitar mucho la forma en la que se resuelven problemas computacionales en relaciún a las bases de datos para poder ejecutar las querys más rápido. El sistema responde más rápido porque no tiene que hacer validaciones por cada elemento, sino más bien por bloques, si el criterio especificado entra en el rango de este para sustraerlo. Estudiar y entender los arboles Binarios de búsqueda equilibrados y los arboles B+.

0.6. Hiótesis:

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.

0.7. Â¿Qué es un árbol Binario?

En los tipos abstractos de datos lineales existen exactamente un elemento previo y otro siguiente (excepto para el primero y el último, si los hay); en las estructuras no lineales, como conjuntos o árboles, este tipo de secuencialidad no existe, aunque en los árboles existe una estructura jerárquica, de manera que un elemento tiene un solo predecesor, pero varios sucesores. Una exploraciún algo amplia en el campo de la ciencia de la computaciún nos lleva a situaciones en que las representaciones lineales son inadecuadas, tanto en sentido conceptual como práctico. Un paso importante lo representan los árboles binarios, y el siguiente vendrá dado con el estudio de la nociún general de árbol. En cap/'itulos posteriores, lo extenderemos hasta llegar a los grafos.

Un árbol impone una estructura jerárquica sobre una colecciún de objetos. Ejemplos claros de utilizaciún de árboles se presentan tanto dentro como fuera del área de computaciún (indices de libros, árboles genealúgicos, etc.); en Informática constituyen una de las estructuras m/as utilizadas, con aplicaciones que van desde los árboles sintácticos utilizados para la representaciún y/o interpretaciún de términos de un lenguaje o expresiones aritméticas, pasando por los arboles de activaciún de procedimientos recursivos, hasta la representaciún de datos que se desea mantener ordenados con un tiempo de acceso relativamente bajo. En general, se usarán árboles siempre que se quiera representar informaciún jerarquizada, cuando esta converja en un solo punto.

Se define un árbol binario como un conjunto finito de elementos (nodos) que bien esta vacío o esta formado por una raíz con dos arboles binarios disjuntos, es decir, dos descendientes directos llamados subarbol izquierdo y subarbol derecho. Los árboles binarios)tienen una especial importancia. Las aplicaciones de los arboles binarios son muy variadas ya que se les puede utilizar para representar una estructura en la cual es posible tomar decisiones con dos opciones

en distintos puntos.

árbol binario de búsqueda.

Los árboles binarios se utilizan frecuentemente para representar conjuntos de datos cuyos elementos se identifican por una clave única. Si el árbol está organizado de tal manera que la clave de cada nodo es mayor que todas las claves su subarbol izquierdo, y menor que todas las claves del subarbol derecho se dice que este árbol es un árbol binario de búsqueda.

En 1972 se crearon los árboles B+ R.Bayer y E.McCreight. Esto inicio por la necesidad de mantener indices en el almacenamiento externo para los accesos a bases de datos , esto quiere decir que, con el problema de la lentitud de estos dispositivos se quiere aprovechar la gran capacidad de almacenamiento para mantener una cantidad de información muy alta organizada de manera que se pueda acceder a de forma más rápida a las claves.

Los métodos y estructuras de datos nos permiten realizar búsquedas dentro de los datos en un tiempo de orden O(log2n). Por lo que tenemos el caso de los árboles binarios de búsqueda equilibrada .Mientras que en memoria interna el tiempo de acceso a n datos almacenados en distintas partes de la memoria es independiente de las direcciones que estos ocupen, en memoria externa es fundamental el acceder a datos situados en el mismo bloque para hacer que el tiempo de ejecución disminuye debido a que el tiempo depende del tiempo de acceso del dispositivo externo, si se disminuye el número de accesos a disco el tiempo resultante de ejecución de nuestra búsqueda se verá fuertemente recortado. Uno de los factores determinantes en el tiempo de ejecución es el número total de accesos, de forma que aunque dicho numero mero pueda ser acotado por un orden de eficiencia es muy importante tener en cuenta el número real ya que el tiempo para realizar un acceso es suficientemente alto como para que dos algoritmos pese a tener un mismo orden, puedan tener en un caso un tiempo real de ejecución aceptable y en otro inadmisible.

De esta forma, si construimos un árbol binario de búsqueda equilibrado en disco, los accesos a disco serán para cargar en memoria uno de los nodos, es decir, para poder llevar a memoria una cantidad de informaciún suficiente como para poder decidir entre dos ramas. Los árboles de múltiples ramas tienen una altura menor que los árboles binarios pues pueden contener más de dos hijos por nodo, además de que puede hacerse corresponder los nodos con las páginas en disco de forma que al realizar un único acceso se leen un número alto de datos que permiten elegir un camino de búsqueda no entre dos ramas, sino en un número considerablemente mayor de ellas. Además, este tipo de árboles hace más fácil y menos costoso conseguir equilibrar el árbol.

0.8. Aplicaciones

Los árboles son mayormente utilizados para el sistema de archivos así como para poder facilitar la realización en bases de datos.

0.9. Metodología

Para poder realizar el árbol Binario de búsqueda equilibrado. Se comenzado a analizar el algoritmo realizado para la práctica nueve de laboratorio en donde de ahí partimos para realizar el árbol binario de búsqueda balanceado.

Para el algoritmo de el árbol B más se analizú de igual manera el árbol B más proporcionado por el profesor y de ahí se partiú para poder implementar el árbol B+.

0.10. Conclusiones:

Calzada Martínez Jonathan Omar:

No sé logró llegar a los objetivo, debido a un mal análisis del problema ya que al principio no sabíamos por dónde empezar y esto nos afectó bastante por lo que nos hizo perder tiempo y tardar en empezar a codificar el programa. Sin embargo sí se pudo implementar programa de el árbol binario de búsqueda equilibrado por lo que se aprendió de manera teórica y práctica y esto nos ayudó a fortalecer nuestras habilidades. Otra cosa que se aprendió fue el manejo de C(sharp) y nos dimos cuenta de que no difiere mucho de el lenguaje java, esto inició como una prueba y terminó siendo el lenguaje con el que se implementó el árbol de búsqueda equilibrado.

García Lazcano Carlos David:

Puesto que ya se contaba con un conocimiento acerca de ambos árboles, se supone que se nos debía de facilitar la implementación, pero no fue así, se batallo demasiado con el árbol b+, incluso se llegó a tener varios modelos, pero ninguno funcional. Debido a ciertos factores no se logró lo esperado; puede que de tener una mejor planificación se hubiesen cumplido los objetivos principales; no es que no hayamos trabajado en equipo, mas bien lo que nos faltó a cada quien fue tomar en ciertas ocasiones determinación. Por otro lado aprendimos un poco de C(sharp), no difiere de Java, al menos hasta donde pudimos trabajar, y está bien puesto que debemos de manejar varios lenguajes de programación. Debemos de mejorar nuestra aptitudes en cuanto a dirigir a nuestros compañeros de vez en cuando.

Hernandez Jimenez Juan Carlos:

No pudimos lograr al objetivo por diferentes razones. Si bien no logramos terminar en forma y tiempo, pusimos nuestro empenio, también conocíamos como funcionaban teóricamente, pero el desarrollo se complicó demasiado en la última recta del proyecto, todo lo demás se hizo de una manera muy buena, se realizaron los diagramas de UML , el uso de clases, las instancias, solo se tuvieron tropiezos que nos impidieron terminar bien por completo el código. Aprendimos C(sharp) que eso nos ayuda a nuestra formación de ingenieros y al dominio de los tantos lenguajes de programación que debemos de aprender a lo largo de la carrera.

0.11. Bibliografia

@ELECTRONICmar, author = Exposito Lopez Daniel, Abraham García Soto, Martin Gomez Antonio Jose, year = 2019, title = `http://decsai.ugr.es/ jfv/ed1/tedi/cdrom/docs`, 22 de Abril de 2019

- Joyanes A.,Luis (2008). Estructuras de datos en java.Madrid,Espania.pp,403
- Lenguaje Java Avanzado. Archivo PDF Pag. 34-40.
- http://www.utm.mx/ jahdezp/archivos
- https://users.dcc.uchile.cl/ bebustos/apuntes/cc30a/TDA/
- https://es.slideshare.net/rennybatista/java-colecciones
- Tutorial de Latex. Múltiples páginas. https://www.latex-tutorial.com/