	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	M.I Tista Garcia Edgar
Asignatura:	Estructura de Datos y Algoritmo II
Grupo:	5
No de Práctica(s):	9
Integrante(s):	Calzada Martìnez Jonathan Omar
No. de Equipo de cómputo empleado:	35
No. de Lista o Brigada:	09
Semestre:	2019-2
Fecha de entrega:	22 de abril de 2019
Observaciones:	

CALIFICACIÓN: _____

Objetivos: El alumno conocerá e identificará las características de los árboles B.
Introducción:
Los árboles-Bo B-árboles son estructuras de datos de árbol que se encuentran comúnmente en las implementaciones de bases de datos y sistemas de archivos. Son árboles binarios de búsqueda en los cuales cada nodo puede poseer más de dos hijos.

DEFINICION

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.

Desarrollo:
Actividad 1). Análisis de una implementación de árbol B.

Para poder hacer el análisis del código se realizó con el primer ejemplo que se muestra en la actividad dos, en la cual instanciamos y creamos un objeto llamado b2, y llamamos al método agregar (Btree) de dicha clase, en ella se crea un constructor del mismo nombre y llamamos al método SetM que este se encuentra en la clase Bnode y le mandamos el orden del árbol.

Del objeto que se creó a la hora de instanciar, que en este caso fue b2 se llama al método add, en donde se coloca el nuevo elemento a insertar en nuestro árbol,

Al recibir la clave se pregunta si ya existe si es el caso se manda un mensaje, si no se llama a la función nodo hoja (lealfNode) junto con la raíz (root) y lo que nos regresa lo manda a la función agregar al nodo junto con la clave.

El método leafNode, verifica si existe, si existe este regresa el nodo, si no recorre el nodo por la derecha y por la izquierda para encontrar donde debe ir y este regresa la posición.

En el método de la clase Btree agregar al nodo (addToNode), este si el nodo es menor que m-1 este muestra el tramaño del nodo y llama a al método insertar para hacer la colocación. En caso de que este lleno y el nodo sea igual a m-1, este realiza la división celular.

En el método insertar se manda el nodo y la clave, en donde por medo de un while realiza la búsqueda de la posción en donde debe ir la clave a insertar, una ves insertarda se llama al método add y se manda la posición y la clave.

La división celular lo que hace es recibir el nodo y la clave. Y se realiza la división, en este m-2 se parte en dos y se guarda en h, y si el m es par se realiza la división sin problemas, ya que no hay residuo y se llama al método insertar. Se crea las llaves y los hijos.

Si el nodo es igual a la raíz se crean dos nuevos nodos en donde se le manda respectivamente sus llaves que les corresponden y el nodo original se limpia y se le manda la clave que quedo al hacer la división (la de en medio), en seguida re repite el procedimiento de arriba en donde si el modulo es igual a cero y la clave es menor al medio se llama al método insertar y se le manda el nuevo nodo y la clave, si es mayor o igual al medio este se le manda el nodonuevo2 y la clave.

Después si el nodo no es la hoja, entonces se saca el modulo y si es distinto de cero se crea el nodo hijo1 y el nodo hijo2, o si el modulo es igual a cero pres pregunta si la clave es mayor o menor al medio y se crean las claves los hijos, después si la clave es igual a la hoja se crea un nuevo nodo y se repite el procedimiento.

Actividad 2). Probando el programa (Ejemplos hechos en clase)
En el primer ejemplo se muestra el resultado de un árbol de orden 4 agregando las claves: 70,50,30,40,20,80,25,90,75,10,15
El segundo ejemplo se muestra el árbol de orden 5 agregando las claves: 20,40,10,30,15,35,7,2618,22,5,42,13,46,27,8,32,38,45,25

```
nodo key sizeeeeeeee0
nodo key sizeeeeeeee1
nodo key sizeeeeeeee2
valor de h   1
nodo key sizeeeeeeee2
nodo key sizeeeeeeee1
valor de h   1
nodo key sizeeeeeeee1
nodo key sizeeeeeeee2
valor de h   1
nodo key sizeeeeeeee2
nodo key sizeeeeeeee2
valor de h   1
valor de h   1
Nodo Raiz:
                50


Nodo Padre: 50
               Nodos:
               20 30
               80

Nodo Padre: 20 30
               Nodos:
               10 15
               25
               40

Nodo Padre: 80
               Nodos:
               70 75
               90

----------------------
```

```
nodo key sizeeeeeeee3
nodo key sizeeeeeeee2
nodo key sizeeeeeeee3
nodo key sizeeeeeeee2
valor de h   2
holis   25
valor de h   2
holis   25
Nodo Raiz:
                  25


Nodo Padre: 25
                  Nodos:
                  10 20
                  30 40

Nodo Padre: 10 20
                  Nodos:
                  5 7 8
                  13 15 18
                  22 24

Nodo Padre: 30 40
                  Nodos:
                  26 27
                  32 35 38
                  42 43 46

----------------------
```

Actividad 3). Menú de usuario.

```
do{
System.out.println("----------------------");
System.out.println("| 1). Agregar clave  |");
System.out.println("| 2). Imprimir arbol |");
System.out.println("| 3). Buscar Datos   |");
System.out.println("| 4). Eliminar nodo  |");
System.out.println("| 5). Salir          |");
System.out.println("----------------------");
op = sc.nextInt();
// BTree b3 = new BTree(4);
```

```
----------------------
| 1). Agregar clave  |
| 2). Imprimir arbol |
| 3). Buscar Datos   |
| 4). Eliminar nodo  |
| 5). Salir          |
----------------------
```

Actividad 4). Opción para eliminar

Conclusiones:

Se cumplieron los objetivos a medias ya que para poder crear el método para eliminación de claves tuve muchos problemas por lo que no pude completar la práctica. La actividad 1, 2 y 3 se cumplieron.

Los arboles b son estructuras de datos de tipo árbol que nos permiten implementar bases de datos y sistemas de archivos. Por la forma de cómo están construidos en más rápido localizar un elemento.
Con esta práctica se reforzó lo aprendido en clase de manera teórica y se logró entender la forma de cómo se implementa con el código, la parte de la división celular en el código fue la parte un poco más confusa per al final se logró analizar.