Facultad de Ingeniería
UNAM

Maestro: Tista Garcia Edgar MI.

Asignatura: Estructura de Datos y Algoritmos II

Práctica 8: Árboles
Parte 1

Integrantes:
Mejía Alba Israel Hipólito
Rosario Morales Oscar Daniel

Grupo:8

Objetivos:

El estudiante conocerá e identificará las características de la estructura no-lineal de un árbol binario.

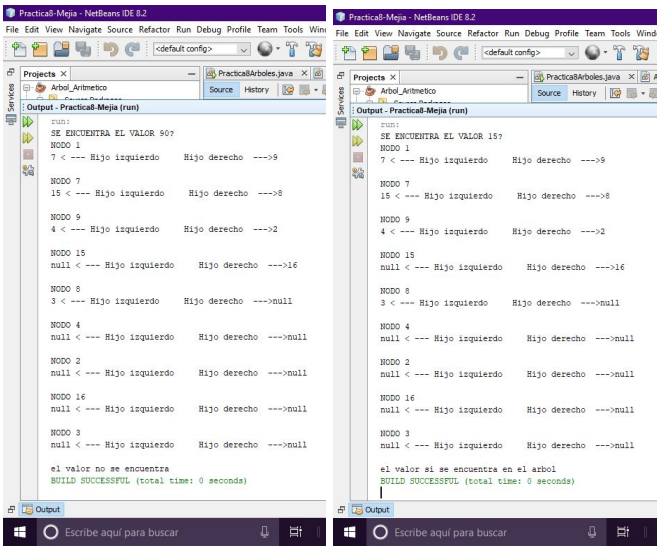
Desarrollo:

Ejercicio 1.Implementación de un árbol binario.
a)Realiza un análisis de la implementación anterior, incluyendo los comentarios que te parezcan relevantes acerca de los elementos que integran las clases.Dibuja el árbol que se realiza en la clase principal.

Para la implementación del árbol binario, se necesitan 2 clases aparte de la clase principal, la primera es la clase ArbolBin, la cual será la encargada de representar el árbol binario pero que en si solo es contiene al nodo raíz y este consecuentemente contiene a los demás subárboles que se forman de sus hijos.
La clase ArbolBin cuenta con 3 constructores, el primero crea un árbol con el valor null en el nodo raíz, el segundo recibe un valor entero y posteriormente crea un nuevo nodo que será el nodo raíz con el valor recibido, el tercer constructor recibe un nodo el cual será asignado como nodo raíz.
Esta clase cuenta con el método add, el cual en el primer ejercicio está implementado de la siguiente manera: el método recibe 3 parámetros, nodo padre , nodo hijo , y un valor entero que puede ser 0 ó 1, internamente el método con una estructura de control if(), si el valor entero es un 0 entonces el nodo hijo que recibió será asignado como hijo izquierdo del nodo padre, y si por el contrario el valor es un 1, entonces el nodo hijo que recibió será el nuevo nodo derecho del nodo padre antes mencionado.
Otro método de esta misma clase ArbolBin, es el breadFirst(), el cual hará un recorrido por capas dentro del árbol, funciona de manera parecida al que vimos en prácticas anteriores, el nodo del que se empezará a hacer el recorrido será por defecto el nodo raíz, posteriormente creará una cola donde se irán almacenando los nodos por capas, encolará primero la raíz, después while() la cola no está vacía entonces desencolará el nodo que se encuentre en la cola(que en este caso sería la raíz) y la marcará como visitada y la imprimirá en pantalla, posteriormente si ninguno de los nodos hijos de el primer nodo marcado como visitado son diferentes de null, encolará ambos nodos repitiendo, y volverá al proceso del while().
La segunda clase que se encuentra en esta parte del ejercicio es la clase Nodo, la cual está constituida por 3 atributos, un valor entero, y dos nodos(izquierdo y derecho), como en la clase pasada esta también tiene 3 constructores que son muy parecidos, el primero creará un nodo con el valor de null en ambos nodos hijos, el segundo recibirá un valor entero el cual se le asignará como su valor predeterminado, y los nodos hijos tendrán por defecto un valor null, el tercer

método, recibirá los 3 parámetros, un valor entero, un nodo izquierdo y nodo derecho, cada valor será asignado con su correspondiente. La clase nodo tiene 2 métodos los cuales son muy similares setIzq() y setDer(), ambos reciben un nodo, y es asignado ya sea como hijo izquierdo o derecho del nodo al que se le esté aplicando el método.
La tercera clase es el main, donde se realizarán los ejemplos de la práctica, aquí se crean los nodos y posteriormente se instancia el árbol, con n1 como nodo raíz, y se añaden los nodos correspondientes, para ver el resultado se utiliza el método breadFirst(), para mostrar el árbol completo, nosotros modificamos un poco ese método para que aparte de que mostrará todos los nodos que se encuentran, muestre de cada nodo sus hijos izquierdo y derecho, para tener una idea un poco más clara de cómo está estructurado.
Si bien la implementación del ejercicio 1 es sencilla, debido a que se utilizan dos constructores de la clase nodo para crear estos mismos, una vez creados los nodos, solo se necesitaron dos nodos, para agregarlos a un tercero como nodos hijos, siendo el nodo padre el n1, para poder crear el árbol, se utilizó el constructor que recibe un nodo, el cual fue n1, en este paso fue sencillo y más fácil agregar los nodos al árbol, debido a que se sabía que nodo sería el padre y si el nodo a agregar sería un hijo izquierdo o derecho.
En los incisos siguientes donde se solicita crear un método para agregar nodos, nos encontramos con esa situación, de si el usuario indicará el nodo padre y si el nodo hijo sería izquierdo o derecho o si el usuario solo agregaria el nodo al árbol sin importarle esas cuestiones.

B)Agrega el método para realizar la búsqueda en dicha clase.
Antes que nada cabe destacar que para el correcto funcionamiento de la búsqueda, es necesario hacer la operación de agregar nodos y ya despues ahora si pedir que se busque.
Nosotros utilizamos el BFS, ya que pudimos darnos cuenta que como hace el recorrido desde el nodo raíz significa que visitará todos los nodos, entonces solo añadimos una condición de que si el valor que se quiere buscar se encuentra como valor de algún nodo dentro del árbol devuelva un valor “true”, en caso contrario devolverá un “false”. La implementación de el método para realizar la búsqueda en el árbol binario la hicimos dentro de la clase principal del ejercicio 1.



Ejercicio 2. Menú de usuario, agrega una clase para elaborar un menú en el que el usuario pueda elegir entre trabajar en árboles binarios y binarios de búsqueda.

Del Menú de usuario solo funcionan arboles binarios, en el cual las opciones de agregar tiene la opción de agregar nodo raíz, o nodos normales, en la desarrollo solo logramos hacer que se agregue un nodo raíz, y sus dos hijos, se pueden crear más nodos pero no se pueden agregar al árbol binario, la opción de BFS funciona y la de Busqueda tambien, pero dentro del ejercicio 1.

4.-Conclusiones:

Mejía Alba Israel Hipólito:
Después de realizar la práctica, puedo ver cómo a pesar de tener los conocimientos claros sobre las estructuras de árboles, sus recorridos y operaciones sencillas como agregar un nuevo nodo, al pasarlo a código se nos complicó en múltiples ocasiones, donde a pesar de intentar distintos métodos, no logramos todas las operaciones deseadas de la practica.De lo que más aprendí esta práctica fue sobre las colas de tipo queue, con las cuales podemos hacer recorridos sobre cada uno de nuestros nodos del árbol y de esta manera poder buscar un elemento del mismo.
Por otro lado al hacer nuestro intento de agregar elementos al árbol, en sí lo que hicimos fue crear uno nuevo sin darnos cuenta, donde primero crearemos la raíz e íbamos agregando los nodos bajando por subniveles, sin embargo, de manera estricta no podemos agregar nuevos nodos al árbol que nos dio en el código de la fotocopia, de cualquier manera me impresiono que terminamos creando un nuevo árbol.

Rosario Morales Oscar Daniel:
Si bien el concepto y teoría de este tema es relativamente sencillo, la implementación del algoritmo en el lenguaje de programación java se nos complicó, ya que aunque planteamos varios caminos en la parte de agregar nodos, pensamos como primera opción hacer una lista de nodos y después de esta lista elegir qué nodos se agregarían al árbol, también buscamos la forma de que en el momento que el usuario quisiera agregar un nodo, pudiera elegir el nombre del nodo pero no encontramos la forma de que la cadena de texto que el usuario tecleara la pudiéramos tomar como nombre del objeto *nodo*, por último camino decidimos solo crear los nodos y que automáticamente se agregan al árbol, pero solo pudimos agregar el nodo raíz que es el que crea o genera el árbol y sus dos hijos consecutivos, ya que los nodos que se quisieran agregar despues de esos 3 nodos deberian de agregarse lo más a la izquierda y lo más abajo del arbol posible, para esto habíamos pensado en una función que trabajara en el método add en el cual se creará otro nodo que funcionara como root temporal, y que fuera moviéndose hacia abajo y de izquierda a derecha, este modo sería como una especie de pivote sobre el cual se agregan los nodos consecutivos, es decir se agregarían un nodo como hijo izquierdo y posteriormente el hijo derecho y después bajaría al hijo izquierdo donde haría la misma función, regresando siempre en donde un nodo tenga algún hijo con valor null. En si el objetivo de la practica no se cumplio pero seguiremos trabajando sobre el avance que tenemos ya que esta es la base de nuestro según proyecto y trataremos de cumplir con todos los puntos del mismo.