TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

Unidad 1
Lenguajes y Autómatas

ALUMNOS- Christiam Iván Rodríguez Moreno
NO.CONTROL: 13480619
CARRERA: Ingeniería en Sistemas Comunicacionales
ASESOR INTERNO: Juan Pablo Rosas Baldazo

Cd. Guadalupe, N.L. Mayo, 2018

Índice

Introducción

Este proyecto tiene el fin de codificar un árbol binario que realiza una evaluación de expresiones que incluyan diferentes expresiones tales como(+,-,*) entre otro tipo de expresiones así como el programa debe de implementar nodos que empiezan desde arriba y vayan bajando poco a poco hasta poder llegar a la obtención de ecuaciones

Descripcion

Primero que nada tuvimos que realizar una investigación ya que no entendíamos bien el como funcionaba un árbol binario por motivos que el entender el como funcionaba era un poco complejo para la creación de dicho árbol, después de la investigación se utilizo una re-ingeniera de un código que se encontró en la web del **Ing Luis Garcia**

Después de crear la re-ingeniera se hizo un pequeño pseudocodigo para la modificación del árbol binario

Pseudocodigo

define la clase(dato):

se ingresa el nodo left

se ingresa el nodoright

se declara la variable dato

class arbol():

//declaracion de variables

self:int

self.root

insert(self, a, dato):

si a == None:

si a = node(dato)

si es falso:

d = a.dato

si dato < d:

a.left = self.insert(a.left, dato)

si es falso:

a.right = self.insert(a.right, dato)

la variable a retorna

def inorder(self, a):

si a == Null:

retorna a null

si es falso:

self.inorder(a.left) \\se defina el metodo in orden

se imprime(a.dato)

self.inorder(a.right)

def preorder(self, a):

si a == Null:

return Null

Si es falso:

se imprime(a.dato)

self.preorder(a.left)

self.preorder(a.right)

def postorder(self, a):

si a == Null:

retorna a Null

si es falso:

self.postorder(a.left)

self.postorder(a.right)

se imprime(a.dato)

def buscar(self, dato, a):

si a == Null:

retorna Null

si es falso :

si dato == a.dato:

retorna a.dato

si es falso:

si dato < a.dato:

```
retorna self.buscar(dato, a.left)
si es falso:
retorna self.buscar(dato, a.right)
SI tree = arbol()
Donde Verdadero es :
os.system("cls")
se imprime("Arbol ABB")
opc = input("\n1.-Insertar nodo \n2.-Inorden \n3.-Preorden \n4.-Postorden \n5.-
Buscar \n6.-Salir
\n\nElige una opcion -> ")
si opc == '1':
nodo = input("\nIngresa el nodo -> ")
Si nodo.isdigit():
nodo = int(nodo)
tree.root = tree.insert(tree.root, nodo)
si es falso:
se imprime("\nIngresa solo digitos...")
Si opc == '2':
Si tree.root == None:
se imprime("Vacio")
si es falso:
tree.inorder(tree.root)
elif opc == '3':
if tree.root == None:
print("Vacio")
else:
tree.preorder(tree.root)
elif opc == '4':
if tree.root == None:
```

```
print("Vacio")
else:
tree.postorder(tree.root)
elif opc == '5':
nodo = input("\nIngresa el nodo a buscar -> ")
if nodo.isdigit():
nodo = int(nodo)
if tree.buscar(nodo, tree.root) == None:
print("\nNodo no encontrado...")
else:
print("\nNodo encontrado -> ",tree.buscar(nodo, tree.root), " si existe...")
else:
print("\nIngresa solo digitos...")
elif opc == '6':
print("\nElegiste salir...\n")
os.system("pause")
break
else:
print("\nElige una opcion correcta...")
print()
os.system("pause")
```

Experimentación

Se tuvo algunos problemas al entender el código ya que era la primera vez que se practicaba con un árbol que implementara diferentes tipo de ordenes se tuvo que investigar en diferentes códigos para poder encontrar un código que se pudiera adaptar a nuestras necesidades

Conclusion

Se pueden obtener los resultados de el código pero tienes que tener cuidado a la hora de aplicar la lógica ya que es algo complicado en este tipo de código ya que cualquier error lógico puede dañar el resultado del orden al final se logro llevar a cabo el dicho resultado

Referencias

https://gist.github.com/codigosdeprogra/12be086b79730718d8fe530d300983b7