PRÁCTICA 9

ÁRBOLES PARTE II

## Alumno: Robles Martínez Héctor

## Número de Cuenta UNAM: 317356548

## Semestre: Tercer semestre de Ingeniería en Computación (2016)

## Semestre Actual: 2021-1

## Profesor: Jorge A. Solano G.

Objetivo:

Conocer e identificar las características de los árboles-B.

Instrucciones:

• Implementar en lenguaje Python el programa propuesto de la práctica 9.

• Crear una función que permita mostrar el árbol creado a partir del código del punto

anterior.

• Crear la gráfica de la complejidad que tiene el algoritmo de inserción en un árbol B.

Resultados obtenidos:

class node():

name = ""

dad = ""

son = None

def \_\_init\_\_(self, name, dad = '\*', son = None):

self.name = name

self.dad = dad

self.son = str(son)

def \_\_str\_\_(self):

string = "|\t" + self.name + "\t|\t" + self.dad + "\t|\t" + self.son + "\t|"

return string

class operation():

a = None

b = None

c = None

root = None

def \_\_init\_\_(self, a, b, c):

self.a = a

self.b = b

self.c = c

self.root = self

def \_\_str\_\_(self, mode = 1):

string = self.a.name + self.c.name + self.b.name

return string

def string(self, mode = 1):

if mode == 1:

string = self.a.name + self.b.name + self.c.name

elif mode == 2:

string = self.a.name + self.c.name + self.b.name

elif mode == 3:

string = self.c.name + self.a.name + self.b.name

return string

class graph():

grade = []

nodes = [[None], [None]]

flag = True

count = 0

def \_\_init\_\_(self, grade = 0, nodes = [[], []], count = 0):

self.nodes = nodes

self.grade = grade

self.count = count

def findPos(self, number):

self.count += 1

if self.nodes[1].count(number) < 5:

return number

else:

self.findPos(number + 1)

def insert(self, op):

self.count += 1

maxSize = 5

self.nodes[0].append(op)

size = len(self.nodes[0])

if size == 1:

self.nodes[1].append("")

else:

self.nodes[1].append(self.nodes[0][self.findPos(size//5)])

def string(self, mode = 1, count = 0):

string = ""

for i in range(len(self.nodes[0])):

count += 1

pos = 0

ps = 0

node = self.nodes[0][i]

string += node.string() + " hijos: "

if node in self.nodes[1][pos:]:

ps = self.nodes[1].index(node)

pos += ps

string += (self.nodes[0][ps]).string(mode)

try:

while node in self.nodes[1][pos:]:

count += 1

string += (self.nodes[0][ps+1]).string(mode)

pos += 1

ps += 1

except:

pass

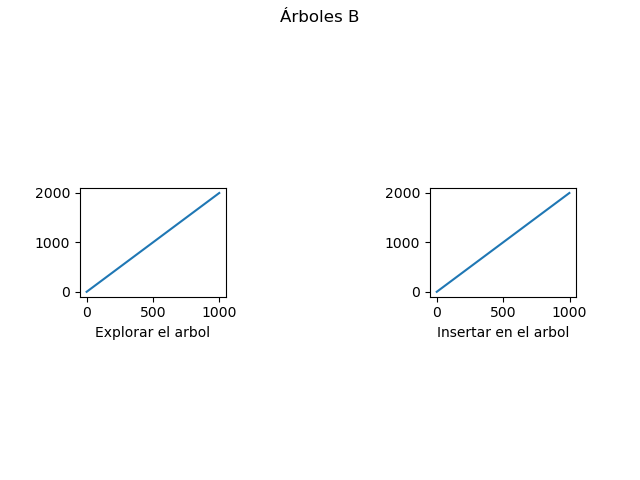
pos += 1

else:

pos = -1

string += "\n"

return string, count



Conclusiones:

Los árboles B y en general los grafos pueden ser representados mediante listas de adyacencia.