UNIVERSIDAD INTERAMERICANA DE PANAMÁ
Facultad de Ingeniería, Arquitectura y Diseño
Escuela de Ingeniería en Sistema Computacionales
Estructura de Datos II

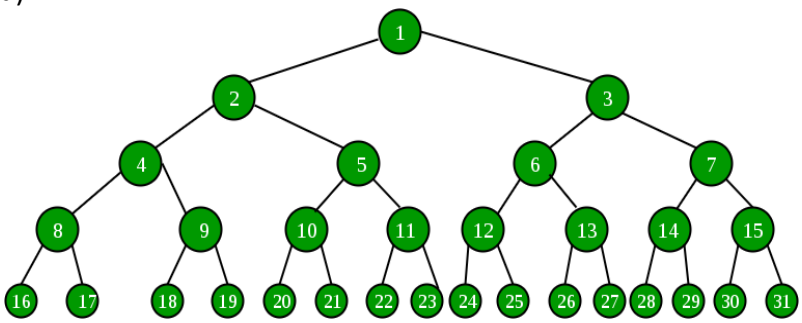
Profesor: Leonardo
EsquedaExamen Parcial #1

Nombre: Keisy Morales **ID:** 4-807-2495

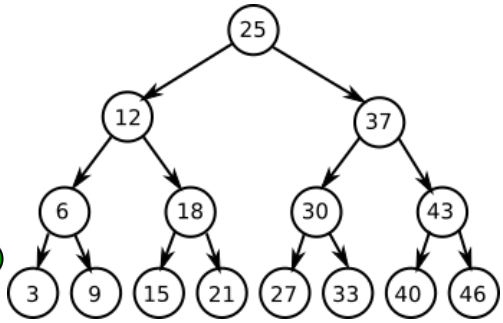
Teoría:

1) Realice el recorrido preorden, inorden y postorden de los árboles siguientes:

a)



b)



A) **Preorden:** 1,2,4,8,16,17,9,18,19,5,10,20,21,11,22,23,3,6,12,24,25,13,26,27,7,14,28,29,15,30,31
Inorden: 16,8,17,4,18,9,19,2,20,10,21,5,22,11,23,1,24,12,25,6,26,13,27,3,28,14,29,7,30,15,31
Postorden: 16,17,8,18,19,9,4,20,21,10,22,23,11,5,2,24,25,12,26,27,13,6,28,29,14,30,31,15,7,3,1

B)

```
SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

Arbol ABB

1.-Insertar nodo
2.-Inorden
3.-Preorden
4.-Postorden
5.-Buscar
6.-Salir

Elige una opcion -> 2
3
6
9
12
15
18
21
25
27
30
33
37
40
43
46
```

Inorden: 3,6,9,12,15,18,21,25,27,30,33,37,40,43,46.

SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
Arbol ABB

1.-Insertar nodo
2.-Inorden
3.-Preorden
4.-Postorden
5.-Buscar
6.-Salir

Elige una opcion -> 3
25
12
6
3
9
18
15
21
37
30
27
33
43
40
46
```

Preorden: 25, 12, 6, 3, 9, 18, 15, 21,37, 30, 27, 33, 43, 40, 46

SALIDA CONSOLA DE DEPURACIÓN TERMINAL

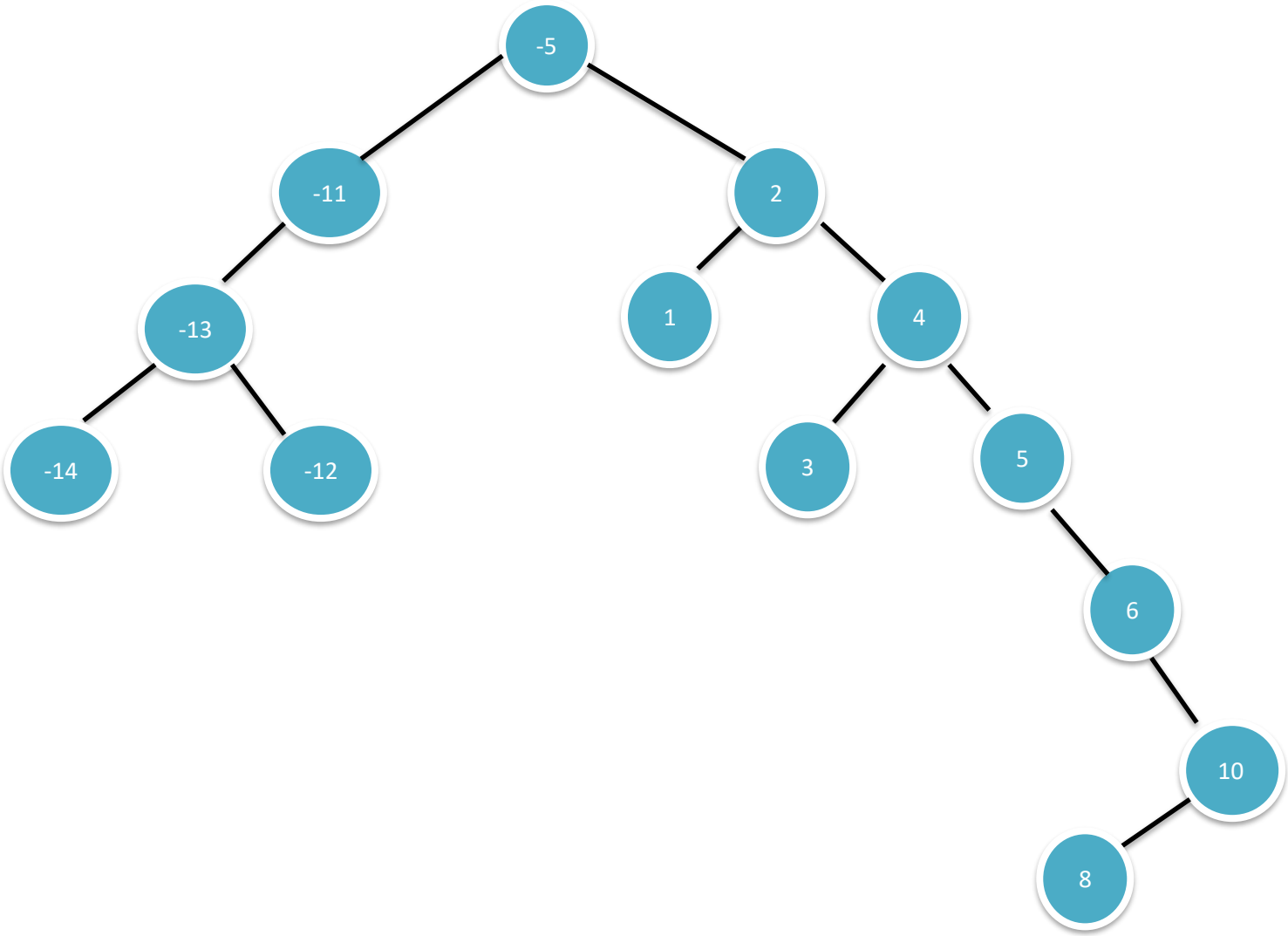
```
Arbol ABB

1.-Insertar nodo
2.-Inorden
3.-Preorden
4.-Postorden
5.-Buscar
6.-Salir

Elige una opcion -> 4
3
9
6
15
21
18
12
27
33
30
40
46
43
37
25
```

Postorden: 3, 9,6, 15, 21, 18, 12, 27, 33, 30, 40, 46, 43, 37,25

2) **Distribuya los nodos -5,2,-11,4,-13,5,3,-14,1,6,10,-12,8 en un árbol binario y determine ¿Cuáles sonlos nodos hoja? ¿Cuáles son nodos de 2 hijos?**



Respuesta: los nodos hoja son:-14,-12, 1, 3, 8 y los nodos con 2 hijos son: -13,-5, 2, 4.

3) **Mencione dos (2) de las propiedades de un árbol binario.**

R:

1) **El número máximo de nodos en el nivel 'l' de un árbol binario es 2^l:** Aquí el nivel es el número de nodos en la ruta desde la raíz hasta el nodo (incluyendo raíz y el nodo). El nivel de la raíz es 0.
   Esto se puede probar por inducción.
   Para raíz, l = 0, número de nodos = $2^0 = 1$
   Suponga que el número máximo de nodos en el nivel 'l' es 2^l
   Dado que en el árbol binario cada nodo tiene como máximo 2 hijos, el siguiente nivel tendría el doble de nodos, es decir $2 * 2^l$
2) **El número máximo de nodos en un árbol binario de altura 'h' es $2^h – 1$:** Aquí, la altura de un árbol es el número máximo de nodos en el camino de la raíz a la hoja. La altura de un árbol con un solo nodo se considera como 1.
   Este resultado se puede derivar del punto 2 anterior. Un árbol tiene nodos máximos si todos los niveles tienen nodos máximos. Entonces, el número máximo de nodos en un árbol binario de altura h es $1 + 2 + 4 + .. + 2^{h-1}$. Esta es una serie geométrica simple con h términos y la suma de esta serie es $2^h – 1$.
   En algunos libros, la altura de la raíz se considera 0. En esta convención, la fórmula anterior se convierte en $2^{h+1} – 1$

Práctica:

1) **Realice un algoritmo lógico el cual explique y demuestre el proceso de inserción de los elementosdel problema teórico número dos (2) en un árbol binario.**

Algoritmo de inserción de elementos de problema 2 en un árbol binario

Paso 1:
Inserción del elemento -5 como nodo principal:
- Capturamos el valor 5 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.
-En este caso el root de nuestro árbol binario es None ya que es el valor default al ser creado, entonces el valor -5 pasa como valor el root node.

Paso 2:
Inserción del elemento 2:
- Capturamos el valor 2 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.
Como ya el root no es None entonces pasa a verificar si el valor 2 es menor o mayor que el valor del root en este caso es mayor entonces pasara a tomar el valor de nodo a la derecha y como ya no tiene valor pasa a ser 2.

Paso 3:
Inserción del elemento -11:
- Capturamos el valor -11 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.
Como ya el root no es None entonces pasa a verificar si el valor -11 es menor o mayor que el valor del root -5 en este caso es menor entonces pasara a tomar el valor de nodo a la izquierda y como no tiene valor pasa a ser -11.

Paso 4:
Inserción del elemento 4:
- Capturamos el valor 4 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.
Como ya el root no es None entonces pasa a verificar si el valor 4 es menor o mayor que el valor del root -5 en este caso es mayor entonces pasara a tomar el valor de nodo a la derecha, invocamos nuevamente como en los caso anteriores el insert pero en este caso al pasarle el mismo objeto y el valor que tiene el nodo a la derecha tenemos que ese tiene como valor 2 entonces vuelve a hacer la misma pregunta si es mayor o menor pero en este caso de 2, 4 es mayor por lo que pasa a guardarse como nodo de la derecha de 2.

Paso 5:
Inserción del elemento -13:
- Capturamos el valor -13 por el usuario
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar
Como ya el root no es None entonces pasa a verificar si el valor -13 es menor o mayor que el valor del root -5 en este caso es menor entonces pasara a tomar el valor de nodo a la izquierda, invocamos nuevamente como en los casos anteriores el insert pero en este caso al pasarle el mismo objeto y el valor que tiene el nodo a la izquierda tenemos que ese tiene como valor -11 entonces vuelve a hacer la misma pregunta si es mayor o menor pero en este caso de -11, -13 es menor por lo que pasa a guardarse como nodo de la izquierda de -11.

Paso 6:
Inserción del elemento 5:
- Capturamos el valor 5 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.
Como ya el root no es None entonces pasa a verificar si el valor 5 es menor o mayor que el valor del root -5 en este caso es mayor entonces pasara a tomar el valor de nodo a la derecha, invocamos nuevamente como en los casos anteriores el insert pero en este caso al pasarle el mismo objeto y el valor que tiene el nodo a la derecha tenemos que ese tiene como valor 2 entonces vuelve a hacer la misma pregunta si es mayor o menor pero en este caso de 2, 5 es mayor por lo que llama el insert en la derecha nuevamente, hasta llega al nodo None donde lo aloja lo cual es luego del 4 donde se realiza la pregunta si es mayor o menor que el 4, 5 es mayor por lo que pasa a guardarse como nodo de la derecha de 4.

Paso 7:
Inserción del elemento 3:
- Capturamos el valor 3 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.
Como ya el root no es None entonces pasa a verificar si el valor 3 es menor o mayor que el valor del root -5 en este caso es mayor entonces pasara a tomar el valor de nodo a la derecha, invocamos nuevamente como en los casos anteriores el insert pero en este caso al pasarle el mismo objeto y el valor que tiene el nodo a la derecha tenemos que ese tiene como valor 2 entonces vuelve a hacer la misma pregunta si es mayor o menor pero en este caso de 2, 3 es mayor por lo que llama el insert en la derecha nuevamente, hasta llega al nodo None, pasa y el siguiente es el nodo con valor 4, 3 es menor que 4 por lo que pasa a la izquierda y este tiene None por lo que lo aloja allí y se vuelve un nodo hoja del nodo 4.

Paso 8:
Inserción del elemento -14:
- Capturamos el valor -14 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.
Como ya el root no es None entonces pasa a verificar si el valor -14 es menor o mayor que el valor del root -5 en este caso es menor entonces pasara a tomar el valor de nodo a la izquierda, invocamos nuevamente como en los caso anteriores el insert pero en este caso al pasarle el mismo objeto y el valor, el valor del siguiente nodo es el -11 por lo que -14 es menor y continuaría a la izquierda, luego esta -13 entonces comprobamos -14 es menor que -13 por lo que pasa a la izquierda y el siguiente nodo es None por lo que lo guarda allí y se convierte en un nodo hoja del -13.

Paso 9:
Inserción del elemento -12:
- Capturamos el valor -12 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.
Como ya el root no es None entonces pasa a verificar si el valor -12 es menor que el valor del root -5 en este caso es menor entonces pasara a tomar el valor de nodo a la izquierda, invocamos nuevamente como en los caso anteriores el insert pero en este caso al pasarle el mismo objeto y el valor, el valor del siguiente nodo es el -11 donde -12 es menor, luego está el nodo -13 entonces comprobamos -12 es mayor que -13 por lo que pasa a la derecha e invocamos el insert de nuevo, en este -12 es mayor por lo que se guarda a la derecha y como ya es None el nodo lo guarda directamente

Paso 10:
Inserción del elemento 1:
- Capturamos el valor 1 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro

árbol binario y el valor a ingresar, el nodo principal tiene -5 por lo que tomamos la derecha ya que 1 es mayor, ese nodo tiene el nodo 2 entonces el 1 es menor que este por lo que pasa a la izquierda y ya en ese nodo es None por lo que se guarda directamente.

Paso 11:

Inserción del elemento 6:

- Capturamos el valor 6 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.

Como ya el root no es None entonces pasa a verificar si el valor 6 es menor o mayor que el valor del root -5 en este caso es mayor entonces pasara a tomar el valor de nodo a la derecha, invocamos nuevamente como en los caso anteriores el insert pero en este caso al pasarle el mismo objeto y el valor que tiene el nodo a la derecha tenemos que ese tiene como valor 2 entonces vuelve a hacer la misma pregunta si es mayor o menor pero en este caso de 2, 6 es mayor por lo que llama el insert en la derecha nuevamente.

El siguiente nodo tiene 4 por lo que 6 es mayor y pasa a la derecha donde se encuentra el nodo 5 y se repite nuevamente la pregunta si el 6 es mayor o menor que 5 como es mayor se coloca a la derecha de este nodo donde es None y lo almacena directamente.

Paso 12:

Inserción del elemento 10:

- Capturamos el valor 10 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.

Como ya el root no es None entonces pasa a verificar si el valor 10 es menor que el valor del root -5 en este caso es mayor entonces pasara a tomar el valor de nodo a la derecha, invocamos nuevamente como en los caso anteriores el insert pero en este caso al pasarle el mismo objeto y el valor que tiene el nodo a la derecha tenemos que ese tiene como valor 2 entonces vuelve a hacer la misma pregunta si es mayor o menor pero en este caso de 2, 10 es mayor por lo que llama el insert en la derecha nuevamente.

El siguiente nodo tiene 4 por lo que 10 es mayor y pasa a la derecha nuevamente. Donde el siguiente nodo es 5 por lo que el 10 es mayor que este y seguiría a la derecha.

Por último, está el nodo 6 por lo que 10 sigue siendo mayor y pasa a la derecha y por ser None se guarda directamente.

Paso 13:

Inserción del elemento 8:

- Capturamos el valor 8 por el usuario.
- A nuestro árbol binario le asignamos por medio del método insert pasándole como parámetro el root de nuestro árbol binario y el valor a ingresar.

Como ya el root no es None entonces pasa a verificar si el valor 8 es menor o mayor que el valor del root -5 en este caso es mayor entonces pasara a tomar el valor de nodo a la derecha, invocamos nuevamente como en los caso anteriores el insert pero en este caso al pasarle el mismo objeto y el valor que tiene el nodo a la derecha tenemos que ese tiene como valor 2 entonces vuelve a hacer la misma pregunta si es mayor o menor pero en este caso de 2, 8 es mayor por lo que llama el insert en la derecha nuevamente.

El siguiente nodo tiene 4 por lo que 8 es mayor y pasa a la derecha nuevamente. Donde el siguiente nodo es 5 por lo que el 8 es mayor que este y seguiría a la derecha.

Sigue el 6 donde 8 sigue siendo mayor por lo que pasa a la derecha y el siguiente es 10 entonces 8 es menor que 10 por lo que pasa a la izquierda y se guarda ya que el siguiente es None.

2) Desarrolle un programa en el lenguaje python en donde cargue por defecto los elementos del árbol: 1 –B y realice los recorridos: preorden, inorden y postorden

arbol1.py

C: > Users > jeffr > Desktop > Estructura de Datos II > arbol1.py > Node > Postorder

```python
class Node:
    def __init__(self, dato):
        self.left = None
        self.right = None
        self.dato = dato
# Insert Node
    def insert(self, dato):
        if self.dato:
            if dato < self.dato:
                if self.left is None:
                    self.left = Node(dato)
                else:
                    self.left.insert(dato)
            elif dato > self.dato:
                if self.right is None:
                    self.right = Node(dato)
                else:
                    self.right.insert(dato)
        else:
            self.dato = dato
# Print the Tree
    def PrintTree(self):
        if self.left:
            self.left.PrintTree()
        print( self.dato),
        if self.right:
            self.right.PrintTree()
# Inorder
# Left -> Root -> Right
    def inorder(self, root):
        resultado = []
        if root:
            resultado = self.inorder(root.left)
            resultado.append(root.dato)
            resultado = resultado + self.inorder(root.right)
        return resultado
# Preorder
# Root -> Left ->Right
    def Preorder(self, root):
        resultado = []
        if root:
            resultado.append(root.dato)
            resultado = resultado + self.Preorder(root.left)
            resultado = resultado + self.Preorder(root.right)
        return resultado
```

```python
    # Postorder
    # Left ->Right -> Root
    def Postorder(self, root):
        resultado = []
        if root:
            resultado = self.Postorder(root.left)
            resultado = resultado + self.Postorder(root.right)
            resultado.append(root.dato)
        return resultado

root = Node(25)
root.insert(12)
root.insert(37)
root.insert(6)
root.insert(18)
root.insert(30)
root.insert(43)
root.insert(3)
root.insert(9)
root.insert(15)
root.insert(21)
root.insert(27)
root.insert(33)
root.insert(40)
root.insert(46)
print("Arbol ABB")
print("Imprimir Arbol")
root.PrintTree()
print("Recorrido Inorden")
print(root.inorder(root))
print("Recorrido Preorden")
print(root.Preorder(root))
print("Recorrido Postorden")
print(root.Postorder(root))
```

SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS C:\Users\jeffr> & C:/Python3/python.exe "c:/Users/jeffr/
Arbol ABB
Imprimir Arbol
3
6
9
12
15
18
21
25
27
30
33
37
40
43
46
Recorrido Inorden
[3, 6, 9, 12, 15, 18, 21, 25, 27, 30, 33, 37, 40, 43, 46]
Recorrido Preorden
[25, 12, 6, 3, 9, 18, 15, 21, 37, 30, 27, 33, 43, 40, 46]
Recorrido Postorden
[3, 9, 6, 15, 21, 18, 12, 27, 33, 30, 40, 46, 43, 37, 25]
```