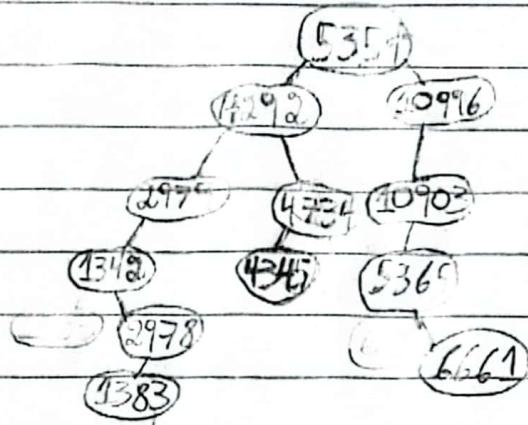


UT4 - PD2



Preorden: 5354, 4292, 2979, 1342, 2978, 1383, 4734, 4345, 10996, 10903, 5365, 6661.

Inorden: 1342, 1383, 2978, 2979, 4292, 4345, 4734, 5354, 5365, 6661, 10903, 10996.

Postorden: 1383, 2978, 1342, 2979, 4345, 4734, 4292, 6661, 5365, 10903, 10996, 5354.

UT2 - PD2

Factorial (n: int)

COM

Si $n = 1$ o $n = 0$ entonces $O(1)$
devolver 1 $O(1)$

Sino

devolver $n * \text{factorial}(n-1)$ $O(1)$

Fin si

FIN



• En la llamada recursiva se la base decrementeando n, en algún momento será 1 o 0.

• El orden del tiempo de ejecución es $O(N)$.

Papiror

SumaLineal(A, n)

COM

Si $n=1$ entonces $O(N)$

devolver $A[0] O(1)$

Sino

devolver sumaLineal(A, $n-1$) + $A[n-1]$ $O(1)$

Fin n

FIN

- En la llamada recursiva que se hace, este decremente n llegando n a ser 1 o 0.
- El orden del tiempo de ejecución es $O(N)$.

calcularPotencia(m, exp: int)

COM

Si $exp = 0$ entonces $O(N)$

devolver 1 $O(1)$

Si $exp = 1$ entonces $O(N)$

devolver m $O(1)$

Sino

devolver $m * \text{potencia}(m, exp-1)$ $O(1)$

Fin Si

FIN

- Flex**
- Si como en la llamada recursiva se hace decrementando el exponente, en algún momento verá 1.

- El tiempo de ejecución es $O(N)$, depende del exponente

Papirito

invertirarray(A, i, j)

COM

Si $i \geq j$ entonces

Devolver A

Sino

Intercambiar $A[i]$ y $A[j]$

Devolver invertirarray($A, i+1, j-1$)

Fin si

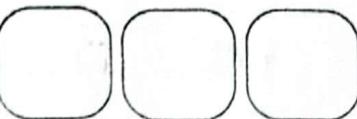
FIN

- Se comporta con la llamada recursiva i va aumentando y j disminuyendo hasta que se cumpla el caso base.

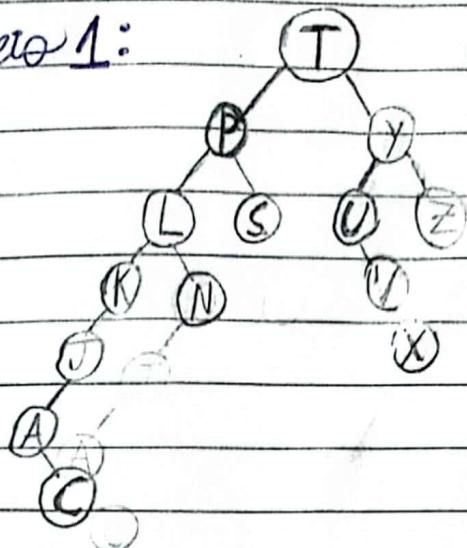
- El tiempo de ejecución es lineal, hace la mitad de iteraciones de la cantidad de elementos de la lista.



UT4 - PD1



Ejercicio 1:



a) d) "S" es una hoja

- b) a) El árbol tiene siete hojas
b) "V" no es descendiente de "S",
están en subárboles distintos.
c) "K" no es hermano de "J", "J"
es descendiente de "K".
d) "A" y "X" no están al mismo
nivel.

Para imprimir los nodos en orden lexicográfico basta
con recorrer el árbol en Inorden.

Inorden: A, C, J, K, L, N, P, S, T, U, V, X, Y, Z

Ejercicio 2:

MiFunerón: devuelve un tipo entero

COM

A \leftarrow -1; B \leftarrow -1 O(1)

Si Hijo Izquierdo \neq nulo entonces O(N)

A \leftarrow Hijo Izquierdo. MiFunerón O(1)

Fin M

O(N)

Si Hijo Derecho \neq nulo entonces O(N)

B \leftarrow Hijo Derecho. MiFunerón O(1)

Fin M

Devolver máximo (A, B) + 1 O(1)

FIN

Flex

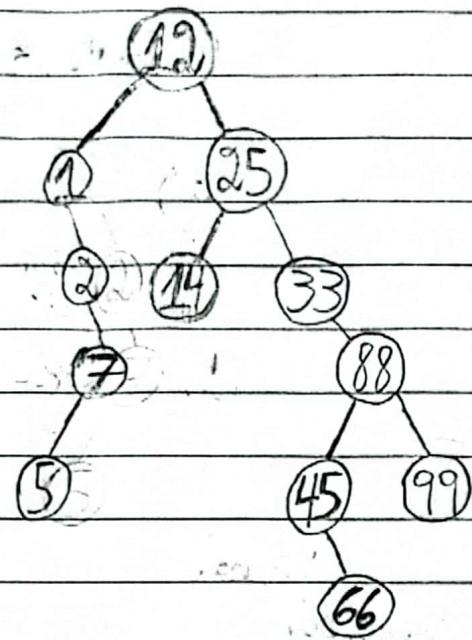
1) La altura del árbol

2) El orden del tiempo de ejecución del
algoritmo es O(N)

Papínam

Ejercicio 3

1)

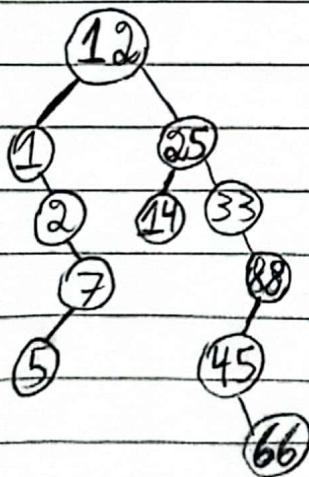


Preorden: 12, 1, 2, 7, 5, 25, 14, 33, 88, 45, 66, 99

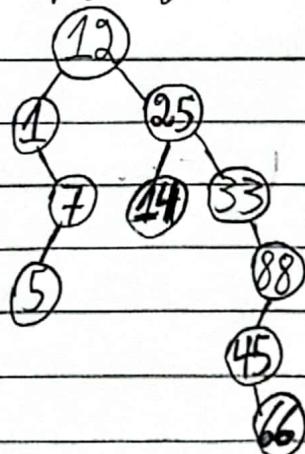
Inorden: 1, 2, 5, 7, 12, 14, 25, 33, 45, 66, 88, 99

Postorden: 5, 7, 2, 1, 66, 45, 99, 88, 14, 33, 25, 12

Eliminar 99



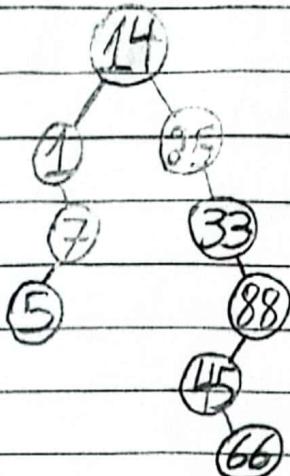
Eliminar 2



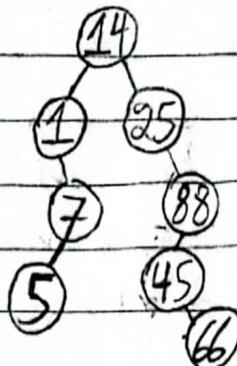
Flex

Papiror

Eliminar 14



Eliminar 33



Preorden: 14, 1, 7, 5, 25, 88, 45, 66.

Inorden: 1, 5, 7, 14, 25, 45, 66, 88.

Postorden: 5, 7, 1, 66, 45, 88, 25, 14.

UT2 - PD4

5.20:

Algoritmo Una (list: int[], x: int)

COM

comienzo \leftarrow 0

fin \leftarrow list.largo - 1

Mientras (comienzo \leq fin) Leer

medio \leftarrow comienzo + (fin - comienzo)

Si (lista[medio] = x) entonces

devolver medio

Si (lista[medio] < x) entonces

comienzo = medio + 1

Fin

fin = medio - 1

Fin M

En mientas
devolver -1

Fin

Flex

Papiror

5.23

algoritmo Plata [7]

(M. L. L.)

$n \leftarrow \text{lito_long}$

candidato $\leftarrow \text{lito}[0]$

contador $\leftarrow 1$

Para (int $i = 1; i < n; i++$) hacer

Si ($\text{lito}[i] = \text{candidato}$) entonces

contador ++

Líne

contador --

Fin i

Si contador = 0 entonces

candidato = $\text{lito}[i]$

contador = 1

Fin M

Fin para

contador $\leftarrow 0$

Para (int $i = 0; i < n; i++$) hacer

Si ($\text{lito}[i] = \text{candidato}$) entonces

contador ++;

Fin i

Fin para

Si contador $\geq n/2$, hacer

devolver candidato

Líne

devolver -1

Fin M

FIN



Papiror

UT4-TA7

De tipo entero Tipo Elemento AB. algoritmo Unos()

COM

$x \leftarrow -1; y \leftarrow -1;$

Si hijo Izquierdo \neq nulo entonces
 hijo Izquierdo.algoritmo Unos()

Fin 12

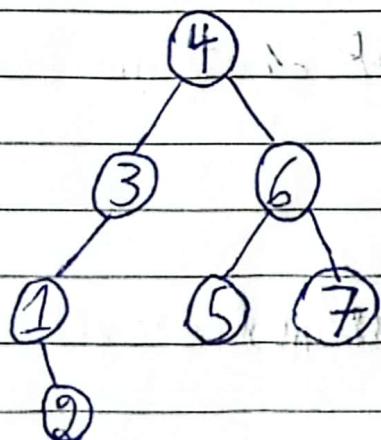
Si hijo Derecho \neq nulo entonces
 hijo Derecho.algoritmo Unos()

Fin 11

Devolver maximo($x+1, y+1$)

FIN

Resultado: 3, devuelve la altura del árbol



Flex

Ejercicio #2

Si tipo entero tipo Elemento AB. algoritmo doy()

com

$x \leftarrow 0; y \leftarrow 0;$

Si ($leyizq = \text{nulo}$ y $leyder = \text{nulo}$) entonces
desarrollar 0

Fin si

Si $leyizq <> \text{nulo}$ entonces

$x \leftarrow leyizq.algoritmo doy()$

Fin si

Si $leyder <> \text{nulo}$ entonces

$y \leftarrow leyder.algoritmo doy()$

Fin si

desarrollar $x + y + 1$

FIN

Resultado: 6, devuelve total los nodos del árbol

Ejercicio #3

Si tipo entero tipo Elemento AB. algoritmo frey()

com

$x \leftarrow 0; y \leftarrow 0;$

Si $leyizq <> \text{nulo}$ entonces

$x \leftarrow leyizq.algoritmo$

Fin si

Si $leyder <> \text{nulo}$ entonces

$y \leftarrow leyder.algoritmo frey()$

Fin si

desarrollar $x + y + (\text{entero})$ etiqueta

FIN

Resultado: 28, la suma de todos los nodos.



Papelería

Ejercicio 4

Se tipo entero tipo Elemento AB. algoritmo Cuadro (comparable unaEti)

COM

$A \leftarrow 0$

Si unaEti < etiqueta entonces

Si hijoIzq < null entonces

$A \leftarrow \text{hijoIzq. algoritmoCuadro(unaEti)}$

Fin Si

Fin M

Si unaEti > etiqueta entonces

Si hijoDer < null entonces

$A \leftarrow \text{hijoDer. algoritmoCuadro(unaEti)}$

Fin Si

Fin M

Si unaEti = etiqueta entonces

$A \leftarrow 1$

Fin M

Devolver A

Fin

Resultado son unaEti = 1, la etiqueta 1 está en el árbol, si no devuelve 0.

Flex

Papiror

Ejercicio 5.1

encuentraLinealUno (nombreAtributo: string,
valorAtributo: object)

COM

Si $thy = \text{null}$ entonces
devolver falso

Fin M

ValorNodo = $thy.\text{etiqueta}().\text{getValorAtributo}(\text{nombreAtributo})$

Si $\text{valorNodo} = \text{null}$ entonces
devolver verdadero

Fin M

Si $\text{valorNodo} > 0$ entonces

devolver $thy.getValorNodo().\text{encuentraLinealUno}(\text{nombreAtributo}, \text{valorAtributo})$

Fin M

Devolver $thy.getValorNodo().\text{encuentraLinealUno}(\text{nombreAtributo}, \text{valorAtributo})$

FIN

Siempre recorrerá el árbol completo. $\Theta(n)$

Flex

Papiror

Ejercicio 5.2

encuentraLinealDot (nombreAtributo: String)

valorAtributo: Object

encontrado: Boolean

com

Si thy = null o encontrados entonces
devolver encontrado

Fin si

valorNodo = thy.elementAt(1).getValorAtributo (valorAtributo)

Si valorNodo = valorAtributo entonces
devolver true

Fin si

Si valorNodo > valorAtributo entonces

encontrado = thy.elementAt(1).encuentraLinealDot
(nombreAtributo, valorAtributo, encontrado)

Lino

encontrado = thy.elementAt(1).encuentraLinealDot
(nombreAtributo, valorAtributo, encontrado)

Fin si

Devolver encontrado

FIN

Flex

Ejercicio 5.3

encuentraNodeTree(nombreAtributo: String,
valorAtributo: Object)

COM

Si $\text{this} = \text{nulo}$ entonces
devolver nulo

Fin si

node $\overset{\text{def}}{=} \text{this.getKeys}\text{[0]}$. encuentraNodeTree
(nombreAtributo, valorAtributo)

Si $\text{node} \neq \text{nulo}$ entonces
devolver node $\overset{\text{def}}{=}$

Fin si

nodeDer $\overset{\text{def}}{=} \text{this.getKeys}\text{[1]}$. encuentraNodeTree
(nombreAtributo, valorAtributo)

Si $\text{nodeDer} < \text{nulo}$ entonces
devolver nodeDer

Fin si

Devolver nulo

FIN

Flex

Papirer

UT4 - PD3

T Árbol BB Insertar(un Elemento) Ejercicio #1

LN: comprueba primero si la raíz del árbol es nula, en caso de serla, se le agrega como raíz un elemento, si no se llama al método Insertar de la clase TElementoAB pasando a un Elemento como parámetro

Precondiciones: - El elemento será del tipo TElementoAB.

Postcondiciones: - Si no tiene raíz, se la agrega
- En todo contrario, llama al método Insertar de TElementoAB

Lenguaje:

COM

Si raíz = nulo entonces
raíz ← un Elemento

Falso

raíz. Insertar(un Elemento)

Fin si

FIN



TElemento AB Insertar(un Elemento)

COM

Si etiqueta = un Elemento. etiqueta entonces
devolver false

Fin si

Si etiqueta > un Elemento. etiqueta entonces

Si hijo Izq = nulo entonces

hijo Izq ← un Elemento

devolver true

Lino

devolver hijo Izq. insertar(un Elemento)

Fin si

Lino

Si hijoDer = nulo entonces

hijoDer = elemento

devolver true

Lino

devolver hijoDer. insertar(un Elemento)

Fin si

Fin si

Fin N

LN: Verifica que el elemento no esté agregado al arbol, luego compara los etiquetas para ver la posición del dato. Si la etiqueta del elemento pasado como parámetro es mayor a la etiqueta del nodo actual en el que se invoca el método, luego comprueba si el hijo izquierdo es nulo, si lo es, entonces el hijo Izq es el elemento mandado por parámetro, sino se llama con recursividad de insertar el hijo izquierdo del actual pasándole el elemento como parámetro.



En el caso que el elemento actual no tiene un hijo derecho, se agrega el elemento pasado por parámetro como hijo derecho, uno hacemos la llamada recursiva e insertar del hijo derecho del actual pasándole el elemento como parámetro.

Precondición: Ninguna

Postcondición: - Si el elemento ya está en el árbol, no va a modificar el árbol.
- Si no se encuentra en el árbol, lo va a modificar añadiendo un nuevo elemento en el árbol.

Ejercicio #2

Árbol BB contarHoja()

LN: Si la raíz es nula, devolverá 0, uno llama al método contarHoja de TElementoAB

Precondición: Ninguna

Postcondición: - Devuelve 0 si la raíz es nula
- Invoca al método contarHoja de TElementoBB devolviendo la cantidad de hojas.

Flex

Papiror

Lenguaje

COM

cantidad ← 0

Si fijoizq > nulo entonces 0(1)

cantidad += fijoizq. contarKojay()

Si fijoDer > nulo entonces 0(1)

cantidad += fijoDer. contarKojay() 0(1)

Fin M

Si fijoizq = nulo y fijoDer = nulo entonces 0(1)
cantidad ++ 0(1)

Fin M

Devolver cantidad 0(1)

FIN

Ejercicio #3 Tablón BB

UN: Si la raiz es nula, devuelve 0. En caso contrario se llama al método suma de Elementos AB

Precondiciones: - Ninguna

Postcondiciones: - Si no tiene raiz, devuelve 0

- Si no invoca al método suma devolviendo la suma de los valores de los nodos del árbol

Lenguaje: sumar()

COM

Si raiz = nulo entonces
devolver 0

Suma

devolver raiz.suma() Página

FIN Fin M

Flex

Ejercicio #4

ArbolBB

LN: Si la raíz del arbol es nula, devolver 0. En caso contrario llamar a el método recursivo cantidadNodosEnNivel de TElementoAB.

Precondición: Un entero como parámetro que no se pase del nivel del arbol.

Postcondición: - Si no tiene raíz, devolver 0
- Llama al método recursivo cantidadNodosEnNivel de TElementoAB.

seudocódigo:

TArbolBB cantidadNodosEnNivel (nivel buscado)

COM

Si raiz = nula entonces
devolver 0

fin

devolver raiz. cantidadNodosEnNivel (nivel buscado)

fin si

FIN

EX

LN: El algoritmo lleva por parámetro el nivel a buscar y se hace a él devolver la cantidad de nodos que se encuentran en el nivel del árbol. Se recorre el árbol y se usa un contador para llevar un registro de los nodos encontrados en el nivel especificado. Se utilizará una función recursiva que visitará cada nodo del árbol, en cada visita comprobará si el nivel actual es igual que el nivel buscado. En el caso que el nivel actual coincida con el nivel buscado, se incrementa el contador de nodos en ese nivel. Si el nivel actual no coincide con el nivel buscado, la función recursiva seguirá visitando los hijos del nodo actual.

Una vez que se hayan visitado todos los nodos del árbol, el algoritmo devolverá el valor del contador.

Precondiciones: - El método se llama en un objeto TElementoAB válido

- El árbol binario de búsqueda no es nulo.
- El nivel pasado como parámetro debe ser un número entero no negativo y que no sea más del nivel máximo.

Postcondiciones: - El algoritmo devolverá un número entero no negativo que representa la cantidad de nodos del árbol que se encuentran en el nivel especificado.

- No se modifican los datos del árbol binario de búsqueda.

Flex

Papirer

seudocódigo

Elemento AB cantidadNodoEnNivel (nivelBuscado)

COM

Si this = null entonces
devolver 0

Fin si

Si nivelBuscado = 0 entonces
devolver 1

Fin si

cantidadIzq = hijoIzq.cantidadNodoEnNivel (nivelBuscado - 1)
cantidadDer = hijoDer.cantidadNodoEnNivel (nivelBuscado - 1)

Devolver cantidadIzq + cantidadDer

FIN



UT4 - TA4

Ejercicio #1

TArbolBB cantidadHojas

Precondiciones: - Ningunas

Postcondiciones: - Devuelve 0 si la raiz es nula
- Llama recursivamente al metodo cantidadHojas de TElementoAB

seudocódigo:

TArbolBB cantidadHojas()

COM

Si raiz = nulo entonces
devolver 0

Lino

devolver raiz.cantidadHoja()

Fin M

FIN

TElementoAB cantidadHojas

Precondiciones: - El método se llama en un objeto TElementoAB válido

- El árbol binario de búsqueda no debe ser nulo

Flex

Postcondiciones: - Devuelve un contador que representa la cantidad de hojas en el árbol binario de búsqueda

Papiror

Sueldos

TElemento AB: cantidad Hoja()

COM

cantidad < 0

Si th_1 . obtener Hoja() <> nulo entonces

cantidad += hijo Izq. cantidad Hoja()

Fin 1

Si th_1 . obtener HojaDer() <> nulo entonces

cantidad += hijo Der. cantidad Hoja()

Fin 2

Si th_1 . obtener Hoja() = nulo y th_1 . obtener HojaDer() = nulo entonces

cantidad = 1

Fin 3

devolver cantidad

FIN

Nivel

Precondiciones: - El método recibe una etiqueta de tipo comparable.

- El árbol debe estar estar correctamente construido

Postcondiciones: - El método devuelve el nivel del nodo que tiene la etiqueta proporcionada como parámetro.

Flex

- Si no se encuentra el nodo con la etiqueta dada, devuelve 0.

Papirro

Arbol tree

predicción unPrefijo:Strong)

COM

resultado ← nueva lista

Si nulo = nulo y unPrefijo < nulo y unPrefijo en "Cose
entonces"

raiz = predicción(unPrefijo, resultado)

Fin

FIN

Nodo Tree

predicción(unPrefijo:Strong, unaLista:LinkedList)

COM

nodo ← buscar(NodoTree(unPrefijo))

Si nodo < nulo entonces

predicción(unPrefijo, (Linked List(strong)) unaLista, nodo)

Fin de

FIN



Siendo este

TElementoAB
Nivel (una Etiqueta)



COM

Si una Etiqueta = etiqueta entonces
devolver 0

Lino

Si una Etiqueta < etiqueta entonces

Si thy obtenerHijoIgual(>) nulo entonces

devolver 1 + thy obtenerHijoIgual(). nivel
(una Etiqueta).

Lino

devolver 0

Fin si

Lino

Si thy obtenerHijoDer() <> nulo entonces

devolver 1 + thy obtenerHijoDer(). nivel (una Etiqueta)

Lino

devolver 0

Fin si

FIN..

TArbolBB

Precondiciones: Ninguna

Flex

Postcondiciones: - Si la raiz es nula, devolverá 0

- Llama al método nivel de TElementoAB.

Papirito

Levaduras:

TArbolBB nivel (una etiqueta)

com

La raiz = nulo entonces

devolver 0

Lono

devolver raiz. nivel (una etiqueta)