

7. Gyakorlat (2019.03.26.)

Bináris fa láncolt ábrázolása

Két pointeres csúcs. Node.

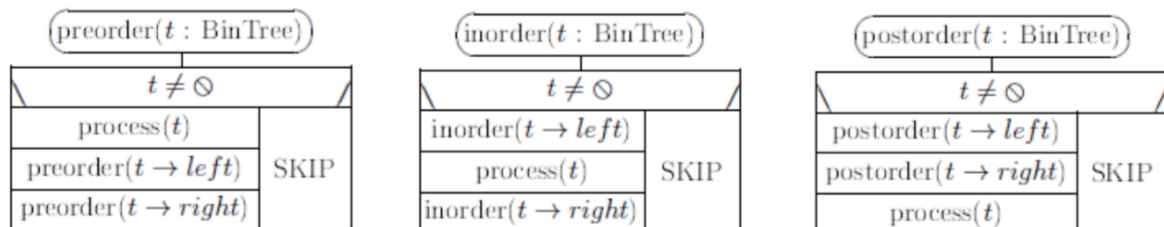
Node
+ $key : \mathcal{T}$ // \mathcal{T} valamilyen ismert típus
+ $left, right : Node^*$
+ $Node() \{ left = right = \emptyset \}$ // egycsúcsú fát képez belőle
+ $Node(x : \mathcal{T}) \{ left = right = \emptyset ; key = x \}$

Három pointeres csúcs. Node3.

Node3
+ $key : \mathcal{T}$ // \mathcal{T} valamilyen ismert típus
+ $left, right, parent : Node3^*$
+ $Node3(p : Node3^*) \{ left = right = \emptyset ; parent = p \}$
+ $Node3(x : \mathcal{T}, p : Node3^*) \{ left = right = \emptyset ; parent = p ; key = x \}$

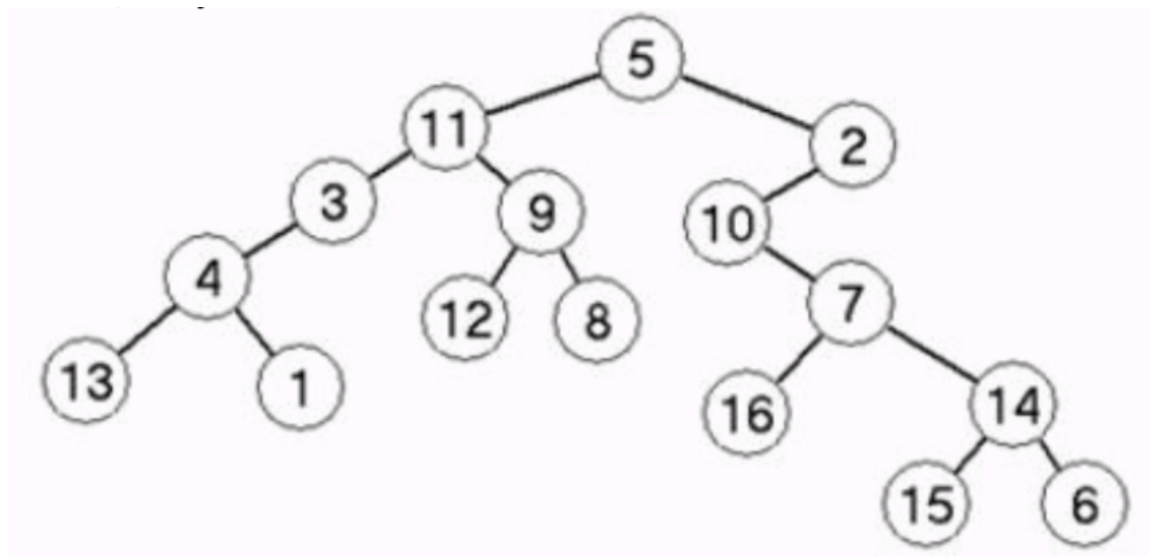
Jegyzetben szereplő három rekurzív algoritmus:

BinTree – absztrakt bináris fa típus, $t \rightarrow left$, $t \rightarrow right$ helyett szokás $left(t)$ és $right(t)$ jelölést is használni. Az üres fát szokták Ω -val jelöli, így $t = 0$ helyett $t = \Omega$ is használható. Láncolt ábrázolású bináris fák esetében a bejáró algoritmusok paramétere lehetne $t : Node^*$, vagy $t : Node3^*$.



1. feladat

Egy konkrét bináris fa bejárása a tanult rekurzív algoritmusokkal, milyen sorrendben írják ki a kulcsokat?

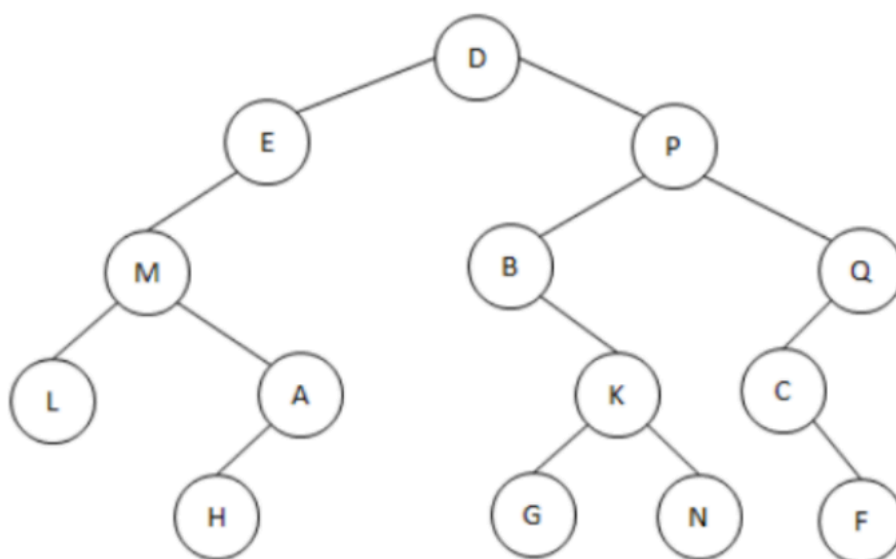


2. feladat

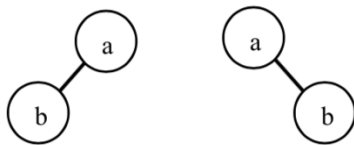
Egy nem teljes bináris fa preorder + inorder, vagy postorder + inorder bejárásból rekonstruáljuk, hogyan nézhetett ki a fa. Miért nem lehet rekonstruálni a preorder + postorder bejárásból?

PREORDER: D E M L A H P B K G N Q C F
INORDER: L M H A E D B G K N P C F Q

Megoldás:



Miért nem jó a preorder és postorder? Ha egy-gyerekes a csúcs, nem lehet tudni belőle, hogy az egy-gyerek melyik irányban van:

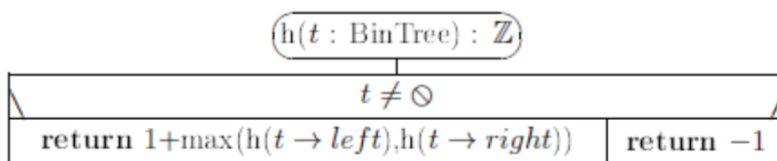


PRE: a b a b
 POST: b a b a

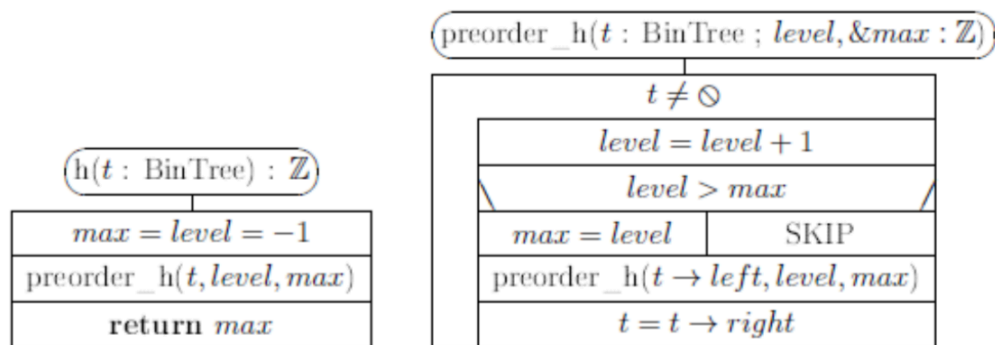
Rekurzív algoritmusok készítése bináris fákkal kapcsolatos feladatokhoz:

Előadáson szerepelt a magasság függvény kétféle módon: (a) a függvény visszatérési értéke a magasság, postorder bejárásra támaszkodva, (b) preorder bejárás+segédváltozó, ebben az esetben egy nem rekurzív algoritmus kerül a rekurzív fölé, ami gondoskodik a belső rekurzív algoritmus megfelelően paraméterezett meghívásáról és az abból kapott eredmény visszaadásáról. Így néznek ki:

(a)



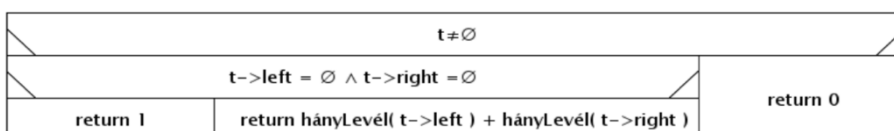
(b)



3. feladat

Hány levele vana fának?

hányLevél($t : \text{BinTree}$):N



4. feladat

Mi a legnagyobb kulcsa egy bináris fának? A fa esetleg lehet üres, azaz $t = 0$ eset is előfordulhat, Oldjuk meg (a) és (b) módszerrel is! Itt most (a) esetben is kell egy „indító” algoritmus, mert le kell ellenőrizni, hogy a fa nem üres-e? Ne használjuk a maximum kezdőértéknek a $-\infty$ értéket. Ha nem üres a fa, induljunk a gyökér kulcsával, ha üres a fa, jelezzünk hibát!

(a) nem használ segéd paramétert. Üres fát le kell ellenőrizni!

maxKulcs(t:BinTree):T

t = 0	
EmptyTreeError	return(maxBejár(t))

maxBejár(t:BinTree):T

t->left = 0 ∧ t->right = 0			
return t->key	t->left ≠ 0 ∧ t->right ≠ 0	t->left ≠ 0 ∧ t->right = 0	t->left = 0 ∧ t->right ≠ 0
	max1 := maxBejár(t-> left)	max1:=maxBejár(t-> left)	max1:=maxBejár(t-> right)
	max2 :=maxBejár(t-> right)		
	max1 := max(max1, max2)		
	return max(max1, t->key)		

Megjegyzések: maxBejár algoritmus előfeltétele, hogy t nem lehet üres! Ezért a rekurzív hívásokat szét kell bontani a t fa alakja szerint.

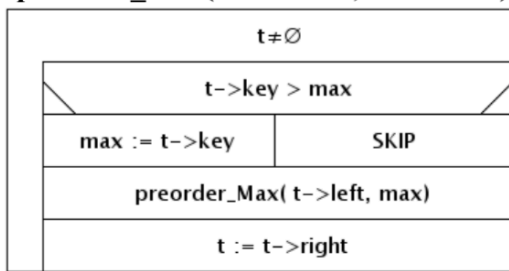
- t lehet levél, vagy belső pont
- belső pont esetei: két gyerekes, csak bal-, illetve jobb gyerekkel rendelkező.

(b) A másik megoldási lehetőséget választva a fő eljárás ellenőrzi az előfeltételt, és ha a fa nem üres, akkor gondoskodik a max változó kezdőértékéről, majd elindít egy bejáró algoritmust. A bejárás végeztével max változó a legnagyobb értéket fogja tartalmazni, így azzal visszatér.

maxKulcs(t:BinTree) : T

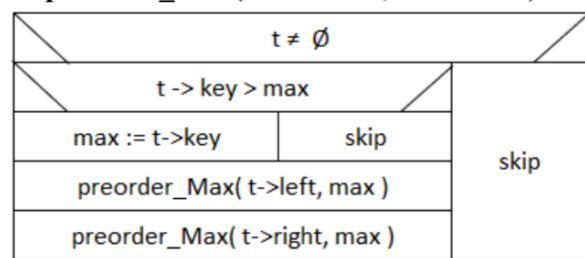
t \neq \emptyset	
max := t->key	EmptyTreeError
preorder_Max(t, max)	
return max	

preorder_Max(t: BinTree, &max : T)



Végrekurzió helyett ciklus (jegyzet ajánlása)

preorder_Max(t: BinTree, &max : T)



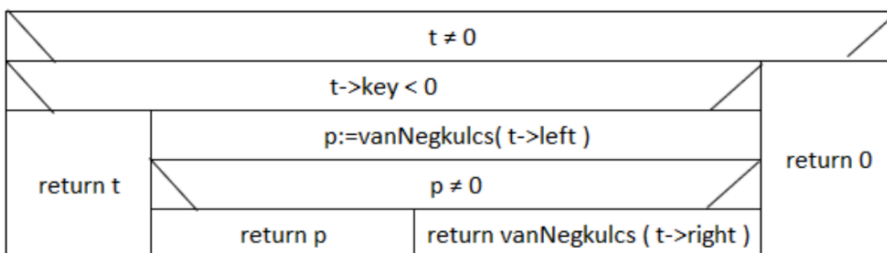
„Klasszikus preorder bejárással”

A bejáró algoritmus viszi magával a max változót, ha kell, módosítja. **FONTOS**, hogy a max cím szerint átadott paraméter legyen!

5. feladat

Keressünk egy negatív kulcsot a fában. Ha találunk, adjuk vissza a címét, ha nem találunk, adjunk 0 értéket vissza. Ügyeljünk a hatékonyságra, ha siker a keresés, minl hamarabb térjen vissza a rekurzió.

vanNegKulcs(t:BinTree): Node*



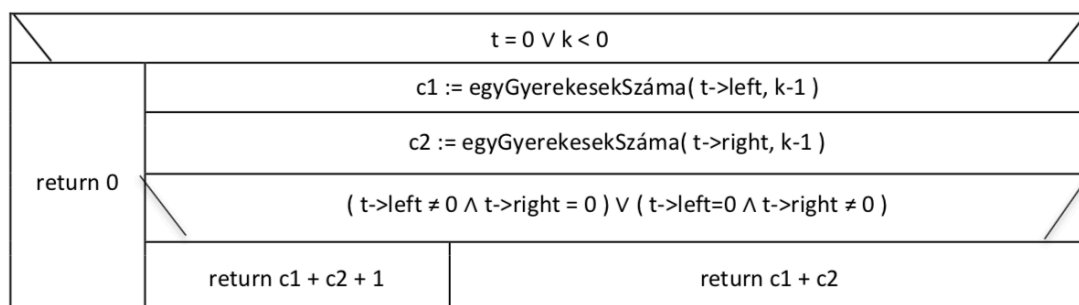
6. feladat

Bevezetjük a szint fogalmát. 0. szint a gyökér, majd 0. szint 2.szint stb. A feldolgozás csak egy adott szintig történik, vagy adott szinttől a teljes mélységig. Feladatok: (a) hány egy-gyerekes csúcsa van a fának a 0..k szinteken, vagy (b) a k-nál nagyobb szinteket?

Megoldás:

(a)

egyGyerekesekSzama(t:BinTree, k: Z):N



(b)

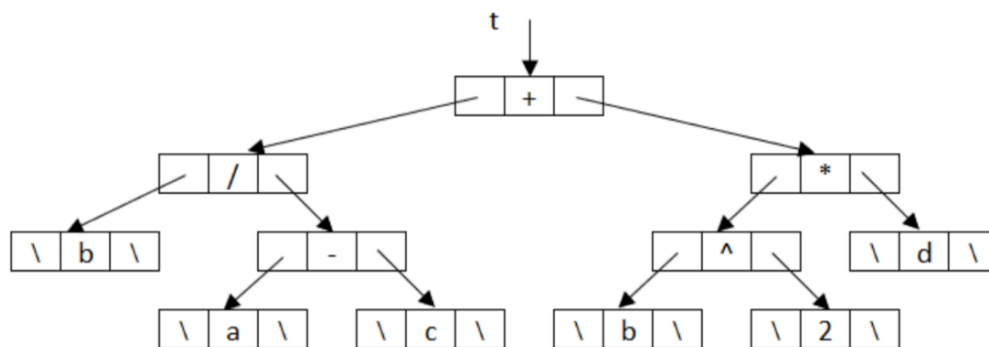
egyGyerekesekSzama(t:BinTree, k: Z):N

t = 0	
return 0	c1:= egyGyerekesekSzama(t->left, k-1)
	c2 := egyGyerekesekSzama(t->right, k-1)
	$k < 0 \wedge (t \rightarrow \text{left} \neq 0 \wedge t \rightarrow \text{right} = 0) \vee (t \rightarrow \text{left} = 0 \wedge t \rightarrow \text{right} \neq 0)$
	<div>return c1 + c2 + 1</div> <div>return c1 + c2</div>

7. feladat

Kifejezés fa. Az egy és két operandusú műveletekből álló kifejezést ábrázoljuk egy bináris fával. Például egy aritmetikai kifejezés fája:

A $b/(a-c)+b^2*d$ kifejezést ábrázoló kifejezés fa:



A kifejezés fa postorder bejárása pont a lengyel formát adja:

$b \ a \ c \ - \ / \ b \ 2 \ ^ \ d \ * \ +$

Házi feladatok:

1. Készítsünk rekurzív algoritmust, mely egy láncoltan ábrázolt bináris fa leveleiben található kulcsok minimumát adja vissza. Üres fára jelezzen hibát.

Ez egy „feltételes minimum keresés” feladat. Megoldható az ismert programozási tétel segítségével: $\text{minBejár}(t: \text{BinTree}, \&l: B, \&\text{min}: T)$ minLevelKulcs az $l := \text{hamis}$ kezdőértékkel elindítja a bejáró algoritmust, a min változó majd az első levélnél kap kezdőértéket, Amikor a bejárás az első levélhez ér, a logikai változó igazra áll, és innentől kezdve a leveleknél folyik a minimum kiválasztás. Egy elegánsabb, segéd paraméterek nélküli megoldás készíthető a maximális kulcsot meghatározó algoritmus ötletének felhasználásával.

2. Készítsünk rekurzív algoritmust, mely egy láncoltan ábrázolt bináris fáról megadja, hogy melyik az a legkisebb szint, ahol van levele a fának. Üres fára -1 – et adjon.
3. Adott egy láncoltan ábrázolt kifejezés fa. Készítsünk rekurzív algoritmust, mely kiírja a kifejezés teljesen zárójelezett alakját. Például a 7. feladat esetén: $((b/(a-c))+((b^2)*d))$ lenne a kimenet. (Hint: a megoldásban arra kell ügyelni, hogy a zárójeleket csak akkor írjunk, ha operátornál – azaz belső pontnál – járunk.)
4. Adott két láncoltan ábrázolt bináris fa: **t1** és **t2**. Készítsünk rekurzív logikai függvényt, mely eldönti, hogy a két fa alakja megegyezik-e. A tárolt kulcsoknak nem kell egyenlőnek lenniük, csak az alakjuknak!