7. BINÁRIS FÁK

Az előző fejezetekben már találkoztunk bináris fákkal. Ezt a központi fontosságú adatszerkezetet most vezetjük be a „saját helyén” és az általános fák szerepét szűkítve, csak a bináris fát mutatjuk be. Jelentőségét alapvetően az adja, hogy több olyan adatszerkezetet is használunk, amelyek speciális bináris fákként értelmezhetők. Ilyen adatstruktúra például a kupac vagy a bináris keresőfa.

7.1. A bináris fa absztrakt adattípus

A *bináris fát*, mint adattípust ebben a fejezetben lényegében csak két absztrakciós szinten definiáljuk, azokon sem teljes körűen. Az ADT szintre ezúttal nem fordítunk túl nagy figyelmet. Ennek az indoka kettős.

Az ADT szinten olyan meghatározást várunk egy típusra, amely nem utal szerkezeti összefüggésekre. A két ismert *matematikai definíció* közül az *egyik* a speciális gráfként vezeti be a bináris fát, tehát a gráfok révén a szerkezetről beszél, így ezt inkább az ADS szinthez sorolnánk. A *másik definíciót* már jobban ehhez a szinthez tartozónak lehet mondani. Ez úgy határozza meg a *t bináris fát*, mint ami vagy az *üres fa* (ezt ε jelöli), vagy pedig egy olyan (c, t_1, t_2) hármast, ahol c egy *csúcs*, t_1 és t_2 *bináris fák*; itt c-t a t fa *gyökerének* nevezzük. Ebben a *rekurzív* definícióban nem nehéz felismerni a *struktúrára* való utalást. Ezért is szokták t_1-et *bal*(t)-vel vagy 0(t)-vel, míg t_2-t *jobb*(t)-vel vagy 1(t)-vel jelölni. A bináris fának annyira erős attribútuma a szerkezet, hogy nem érdemes anélkül beszélni róla.

Az ADT szinthez tartozik a *műveletek* megadása is. Az itt szóba jövő műveletek elemi szinten teszik lehetővé a bináris fák felépítését, részeinek lekérdezését, a fa módosítását és lebontását. Ezek az elemi műveletek nem játszanak hangsúlyos szerepet a továbbiakban, ezért – ennél az egy adatszerkezetnél – nem is határozzuk meg pontosan a körüket.

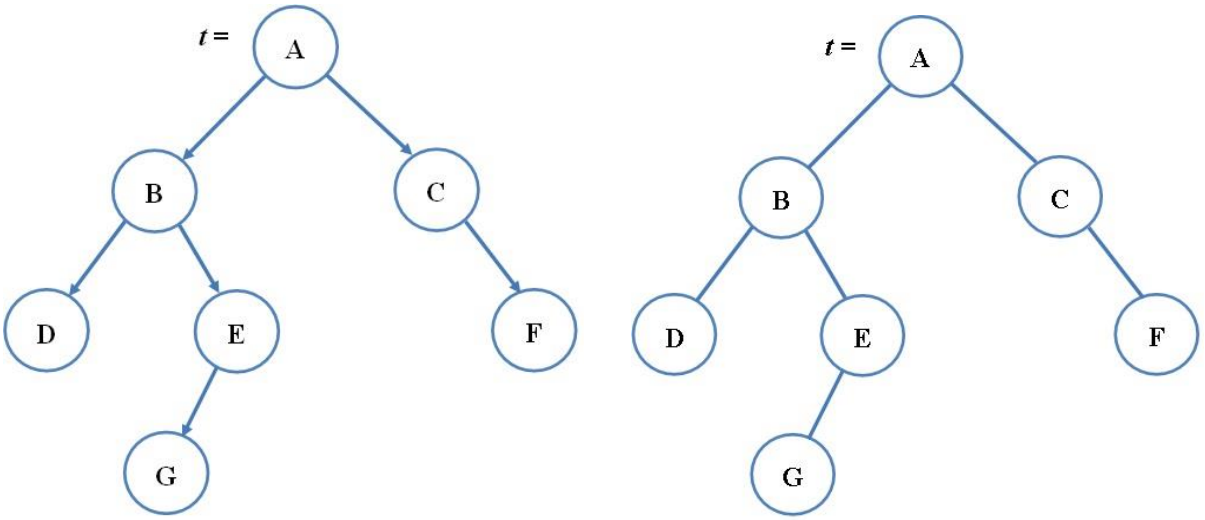
7.2. A bináris fa absztrakt adatszerkezet

Az ADS szinten megállapítjuk a szóban forgó típus *absztrakt szerkezetét*. A *bináris fa* struktúrája ezen a szinten lényegében a *matematikai fa-fogalommal* írható le.

Az a *definíció*, amelyre az előző pontban utaltunk, speciális gráfként határozza meg a bináris fát. Eszerint a fa olyan körmentes irányított gráf, amelyre igazak az alábbiak: pontosan egy csúcsba nem vezet él (ez a gyökér); a többi csúcsba pontosan egy él vezet és minden csúcs elérhető a gyökérből, mégpedig egyértelműen. Az egyes csúcsokból kivezető élek száma minden fára korlátos ($r > 0$). Az elnevezése ekkor: r-áris fa. Gyakorlatban az élek rendezett szelektornevekkel címkézettek. Az $r = 2$ érték esetén beszélünk *bináris fákról*.

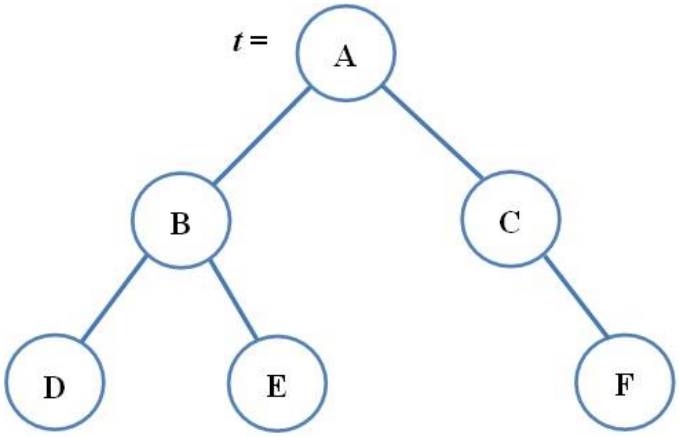
Ha a fenti *másik definíciót* vesszük alapul és meglátjuk benne a bináris fa *struktúráját*, akkor tekinthetjük azt az ADS szintre tartozó specifikációnak. Annyiban hasznosabb ez a rekurzív meghatározás, mint az előző gráf-alapú definíció, hogy a legtöbb fákon értelmezett algoritmus, amelyek általában rekurzívan működnek, jó illeszkedik a bináris fa *rekurzív szerkezetéhez*.

A 7.1. ábrán megadtuk a bináris fa két leggyakrabban előforduló ADS szintű képét. Az elsőn az élek irányítottak szülő-gyerek irányban. Ha a gyerek-szülő irányra is szükségünk lenne, akkor kétirányú éleket kellene felvenni, ilyen ábrázolással azonban nem ADS szinten alig találkozunk. A második, amelyen az élek irányítása nem szerepel, a bináris fa szokásos absztrakt ábrázolása. Az irányítás hiánya ellenére az élek kifejezik *mindkét irányú szülő-gyerek kapcsolatot*.



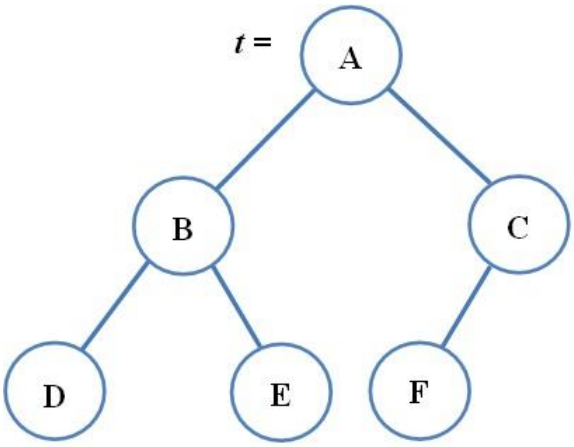
7.1. ábra. Bináris fa absztrakt adatszerkezet (két változat)

A következő ábrákon három *speciális* bináris fa látható. A *majdnem teljes bináris fának* csak az alsó szinten lehet hiánya: megengedett, hogy egy levélcsúcsnak *üresen* maradjon a helye. Például a 7.2. ábrán szereplő fában a C belső csúcsnak nincsen bal gyereke.



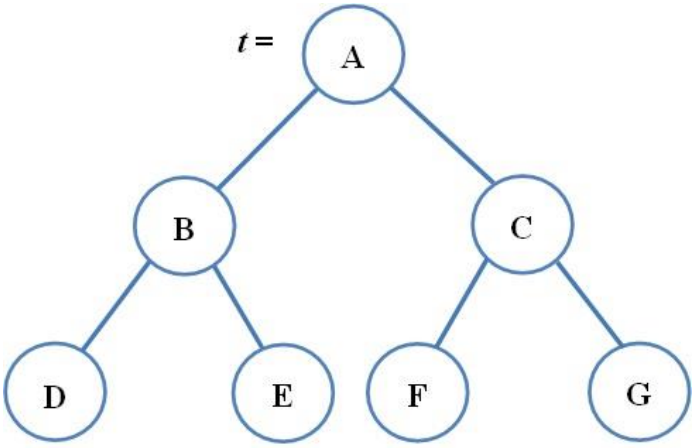
7.2. ábra. Majdnem teljes bináris fa (ADS)

A *majdnem teljes balra tömörített bináris fának* az alsó szintjén a levelek balról-jobbra hézagmentesen helyezkednek el. Ilyen tulajdonságú fát illusztrál a 7.3. ábra.



7.3. ábra. Majdnem teljes balra tömörített bináris fa (ADS)

A *teljes bináris fa* minden előforduló szintjének teljes a kitöltöttsége. Vezessük be a *famagasság* fogalmát, amely az első szinten nulla értéket vesz fel, vagyis a magasság értéke 1-gyel kisebb, mint a szintek száma. Ekkora $h = h(t)$ magasságú teljes bináris fa csúcsainak a száma 2^{h+1}-1. A 7.4. ábrán egy 3-szintű vagy, ami ugyanaz, egy 2-magasságú teljes bináris fa látható, amely 7 csúcsot tartalmaz.



7.4. ábra. Teljes bináris fa (ADS)

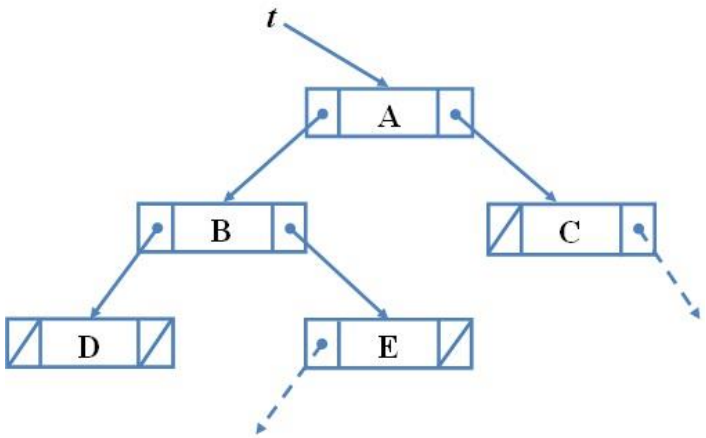
Speciális bináris fa a keresőfa és a kupac. A *bináris keresőfa* alakja általános, olyan jellegű, mint amilyen a 7.1. ábrán szerepel. A *kupac* – alakját tekintve – majdnem teljes és balra tömörített bináris fa. Mindkét fát jellemzik még a csúcsokban tárolt értékek között fennálló összefüggések is.

7.3. A bináris fa reprezentálása

A bináris fa esetén is *kétféle ábrázolást* mutatunk be. A láncolt ábrázolás az általános alakú fákhoz illeszkedik, míg a tömbös ábrázolás a teljes, illetve a majdnem teljes bináris fáknál jelenik meg.

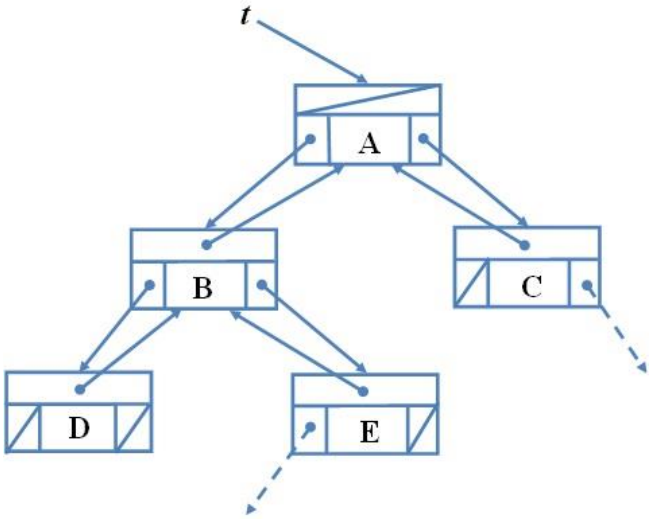
4.3.1. Láncolt ábrázolás

A bináris fa *pointeres ábrázolásában* is mutatók valósítják meg a rákövetkezési relációkat. A 7.5. ábrán olyan láncolt ábrázolás részletét láthatunk, amelyben csak a *bal* és *jobb gyerek pointerek* szerepelnek (a gyerek csúcsokból nem mutat vissza pointer a szülőre). .



7.5. ábra. Bináris fa pointeres ábrázolása (szülő pointerek nélkül, részlet)

A 7.6. ábrán szereplő láncolt ábrázolásban a *szülő pointerek* is megjelennek. A *gyökérelemben* ennek a pointernek kötelezően NIL az értéke.

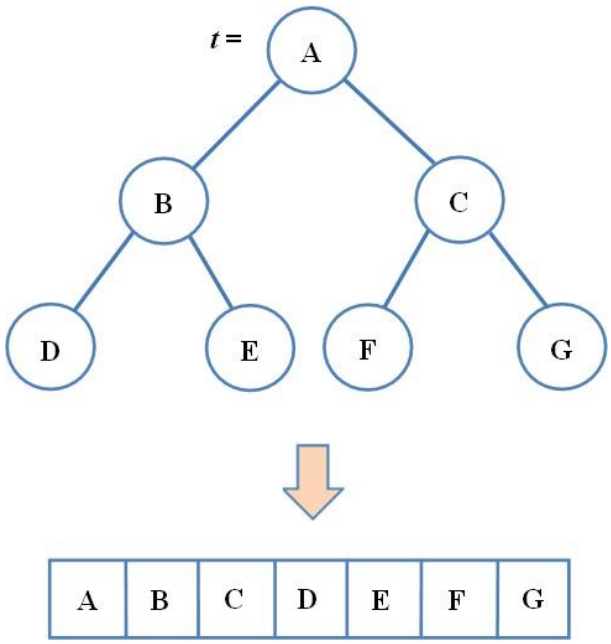


7.6. ábra. Bináris fa láncolt ábrázolása (szülő pointerekkel, részlet)

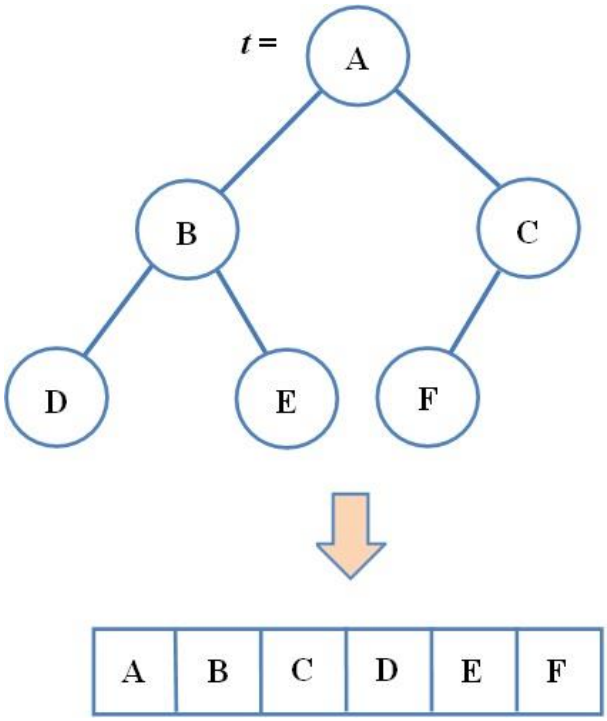
Mindkét pointeres ábra ugyanazt a részletet tartalmazza. Ez a részlet olyan, hogy a 7.1-4. ábra négy bináris fája bármelyikéhez illik (azzal a megjegyzéssel, hogy az E csúcs bal pointere az utolsó három fa esetében NIL.)

4.3.2. Tömbös ábrázolás

A *teljes* és a *majdnem teljes balra tömörített* bináris fa sajátossága az, hogy ha az elemeit szintfolytonosan érintjük, akkor egyetlen hiányzó csúcsot sem fedezünk fel, mert – egy bizonyos határpontig a levelek szintjén – minden csúcs szerepel a fában. Ha a csúcsokat *szintfolytonos* sorrendben egy tömbbe helyezzük, akkor *hiánymentes* kitöltéshez jutunk, ahogyan a 7.7. és a 7.8. ábrán látható.



7.7. ábra. Teljes bináris fa tömbös ábrázolása



7.8. ábra. majdnem teljes bináris fa tömbös ábrázolása

A tömbben a szülő-gyerek és a gyerek-szülő kapcsolatok az alábbi egyszerű képletek által kezelhetők maradnak:

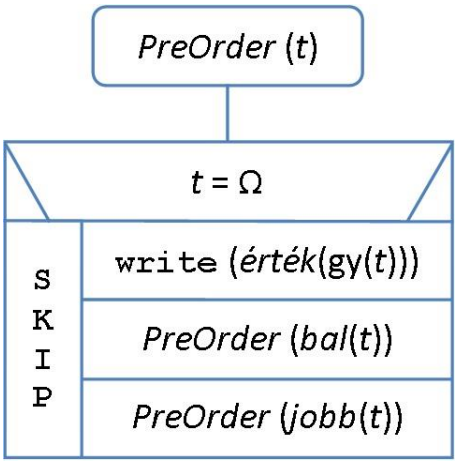
$$ind(bal(c)) = 2*ind(c) \qquad \text{és} \qquad ind(jobb(c)) = 2*ind(c) + 1, \text{ illetve}$$

$$ind(szülő(c)) = \lfloor ind(c) \rfloor$$

Az első két összefüggés értelemszerűen *nem-levél* csúcsokra alkalmazható, míg a harmadik képlet a *gyökértől különböző* csúcsokra érvényes.

7.4. A bináris fa bejárásai

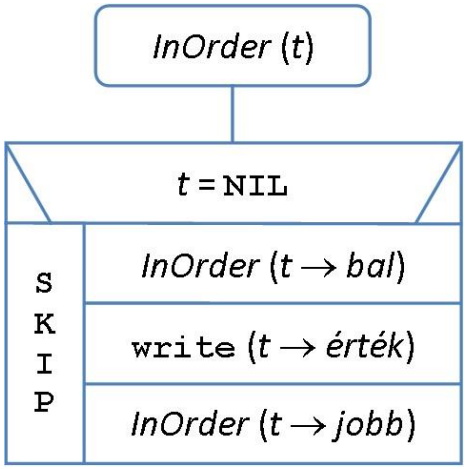
A fák jellegzetes algoritmusai a *bejárások*. A bináris fa rekurzív jellegéhez illeszkedően *rekurzív módon* könnyű definiálni a három féle bejárást, amelyek abban különböznek, hogy a *gyökérelemet* mikor érintjük a bejárás során.



7.9. ábra. Bináris fa preorder bejárása (ADT/ADS)

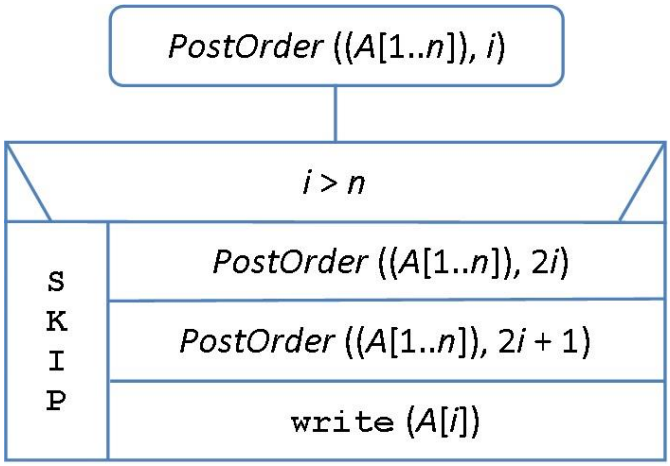
Ha a gyökérelemet először, a bal és jobb oldali részfa bejárásai előtt közvetlenül érintjük, akkor *preorder* bejárásról beszélünk. Ennek algoritmusát a 7.9. ábra mutatja, méghozzá ADT/ADS szinten.

Amikor a gyökérelemet a bal és jobb oldali részfa bejárásai között érjük el, akkor *inorder* bejárást valósítunk meg. A 7.10 ábrán látható ez a bejárási stratégia, a fa láncolt ábrázolását véve alapul.



7.10. ábra. Bináris fa inorder bejárása (pointeres reprezentáció)

Ha a gyökérelemhez a bal és jobb oldali részfa bejárásai után jutunk el, akkor *postorder* bejárást valósítunk meg. A 7.11 ábrával illusztrált eljárást speciálisan a tömbös ábrázolásra adtuk meg.



7.11. ábra. Bináris fa postorder bejárása (tömbös reprezentáció)

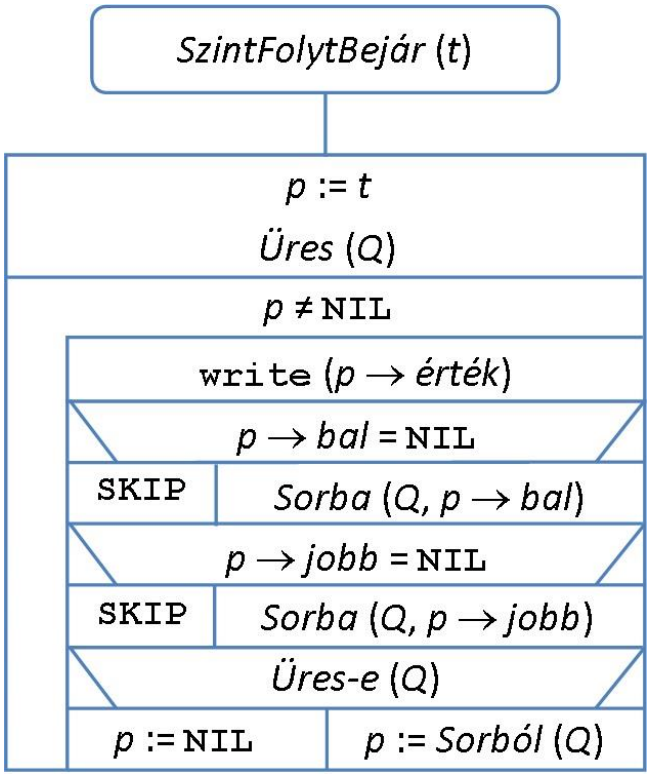
Érdekes feladat egy bináris fa *szintfolytonos bejárásának* a megvalósítása. A csúcsok szintfolytonos sorrendű elérése ugyanis – szemléletesen szólva – ortogonális arra az irányra, amelyet az élek meghatároznak. Szintfolytonos bejárást készíthetünk egy sor adatszerkezet alkalmazásával. Az algoritmust néhány *szabály* alapján ezután már könnyen kialakíthatjuk.

(1) A bejárás során a gyökérelem legyen az első érintett csúcs.

(2) Amint egy csúcsot elértünk, akkor a gyerekeit helyezzük el a sorban.

(3) A bejárás következő elemét vegyük a sorból.

(4) Az eljárás addig folytatódik, ameddig a sor ki nem ürül.



7.12. ábra. Bináris fa szintfolytonos bejárása
sor adatszerkezet segítségével (pointeres reprezentáció)

A 7.12. ábrán láthatjuk a teljes eljárást, amely *üres fára* is helyesen működik. Az *iteráció feltételét* ($p \neq$ NIL) ezzel összhangban választottuk meg és a sor kiürülésekor az eljárás ennek megfelelően (p = NIL) állítja be a *terminálás feltételét*.

Az algoritmus jellemzője az is, hogy a *Q* sorba nem a csúcsok tartalmát, hanem csak azok *pointereit* helyezi be.

Mind a négy fabejárásra teljesül az, hogy *műveletigényük* a bináris fa csúcsainak számában *lineáris*, ugyanis a bináris fa minden csúcsát pontosan egyszer érintik.

7.5. Rekurzív algoritmusok bináris fákon

Az imént szereplő fabejáró algoritmusok közül az első három rekurzív hívási szerkezettel rendelkezik. A működés megfogalmazása illeszkedik a struktúra rekurzív szemléletéhez. Számos további, bináris fákon értelmezett algoritmussal találkozhatunk a különféle feladatok megoldása során, amelyeket szintén *rekurzív* módon a legkönnyebb leírni.

Példaként tekintsük azt a feladatot, amely egy bináris fa *leveleinek megszámolását* tűzi ki. (Ebben felismerhetnénk az adott tulajdonságú elemek megszámolásának sztenderd feladatát, de most gondolkodjunk e nélkül.) A megoldást ismét fogalmazzuk meg néhány szabályban. Rekurzív megoldást keresünk.

(1) Az *üres* bináris fa leveleinek száma nulla.

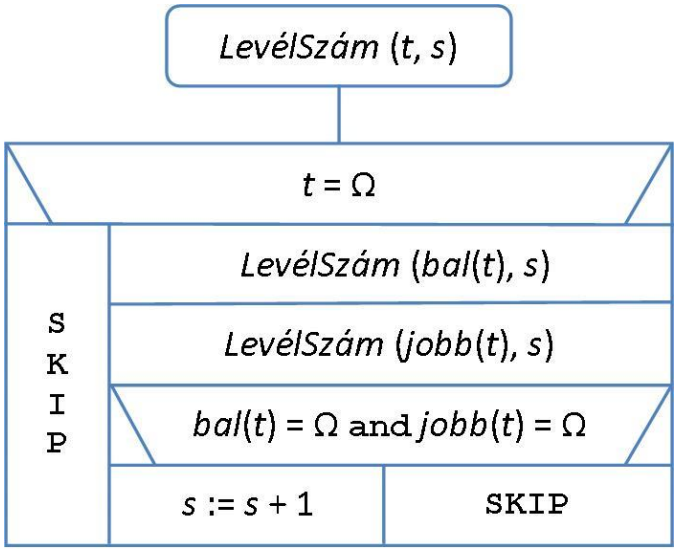
(2) Az *egyetlen csúcsból* álló bináris fa egy levelet tartalmaz.

(3) Egy *összetett* bináris fa leveleinek száma megegyezik a gyökérpont alatti *bal* és *jobb oldali részfa* levél-számának *összegével*.

Az utolsó pontban szereplő összetett bináris fát az a tulajdonság azonosítja, hogy a gyökerének legalább az egyik oldali részfája nem üres.

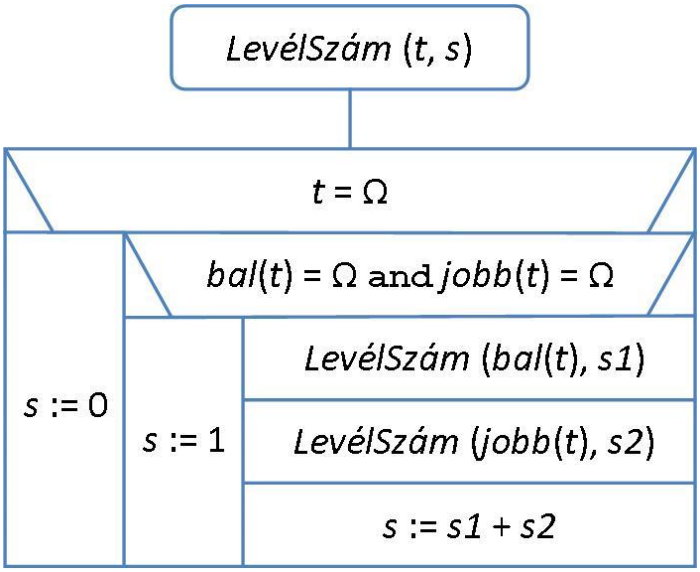
A megoldó algoritmus három változatát láthatjuk a következő ábrákon. A 7.13. ábrán szereplő eljárás egy *globális* input-output típusú számláló változót tartalmaz a paraméterlistáján. Az eljárás külső hívása helyén, a vívás előtt ezt a számlálót a nulla kezdőértékkel *inicializálni* kell.

Amikor a rekurzív *hívások* láncolata egy *levélhez* ér, akkor az eljárás még meghívja önmagát a bal és a jobb oldali üres részfákra, ami nem módosítja a számláló értékét. Ez után kerül lekérdezésre az, hogy üres volt-e mindkét oldali részfa. Ha igen, akkor a számláló értékét növelni kell eggyel, mert levél csúcsnál tart az eljárás.



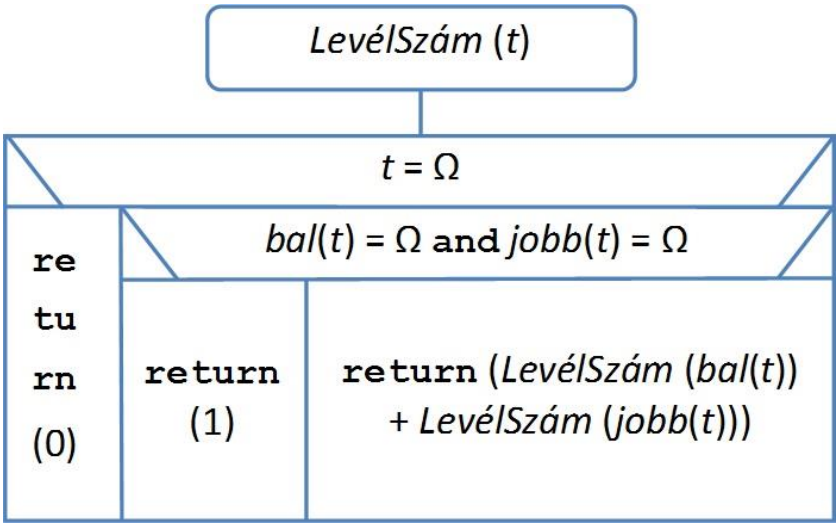
7.13. ábra. Bináris fa leveleinek száma
globális számlálóval

Az eljárás következő változata a 7.14. ábrán, amely *lokális számláló* kimenő paraméterrel működik, tisztább szerkezetűnek mondható. Az algoritmus logikája megfelel előzetes megfontolásunk (1)-(3) pontjainak. A külső hívás helyén *s*-et nem kell inicializálni.



7.14. ábra. Bináris fa leveleinek száma
lokális számlálóval

A 7.15. ábrán bemutatott *függvényeljárás* követi az előző változat világos szerkezetét. A visszaadott függvényérték használata és a *return* utasítás szerkezete tömör, matematikai írásmódot tesz lehetővé.



7.15. ábra. Bináris fa leveleinek száma
rekurzív függvény visszatérő értékével

Megjegyezzük, hogy számos olyan gyakori egyszerű feladat, amelyekkel jobbára tömbökre megfogalmazva találkozunk (például adott tulajdonságú elem keresése vagy a maximum kiválasztás feladata), értelmezhetők más struktúrákra, így bináris fákra is.

A bináris keresőfákról önálló fejezet szól ebben a tananyagban. A keresőfa műveletei is megfogalmazhatók rekurzív módon, egy helyen meg is adjuk ezt a változatot, azonban a műveletek iteratív változatát részesítjük előnyben (lásd: 11. fejezet).