Bináris fák bejárása

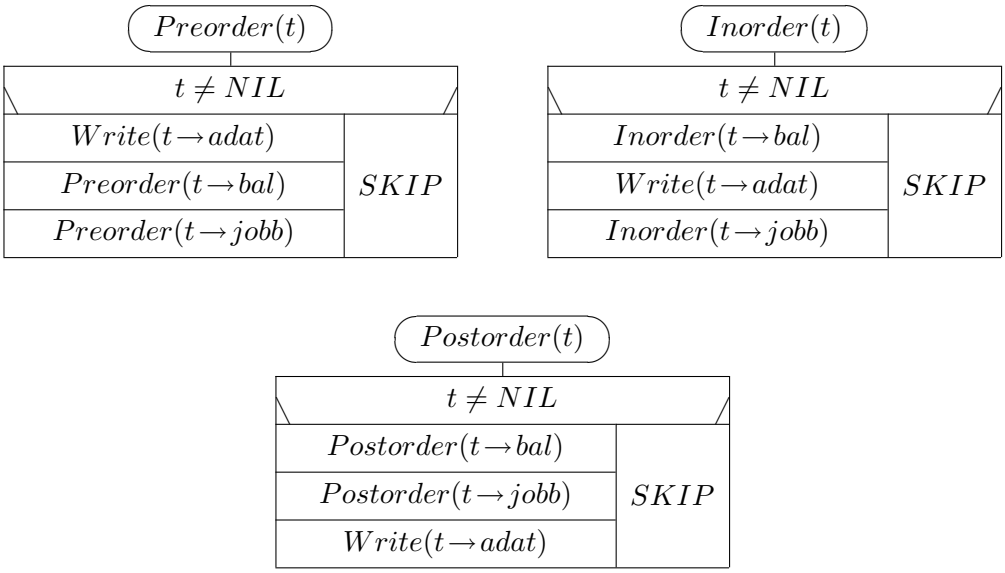
Kovács Péter
kpeter@inf.elte.hu

2008. december 12.

1. Feladat: Adott egy bináris fa a szokásos láncolt ábrázolással. Járjuk be a csúcsait *preorder*, *inorder* és *postorder* bejárás szerint (írjuk ki a csúcsokban tárolt értékeket).

Megoldás:

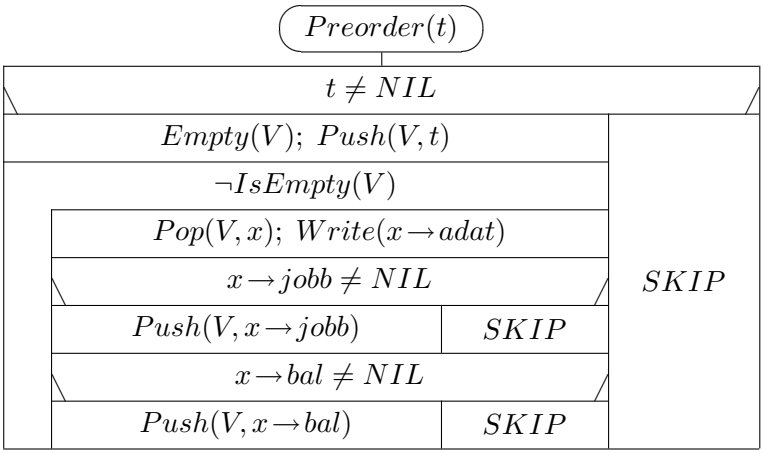
a) Rekurzív algoritmussal a szokásos módon:



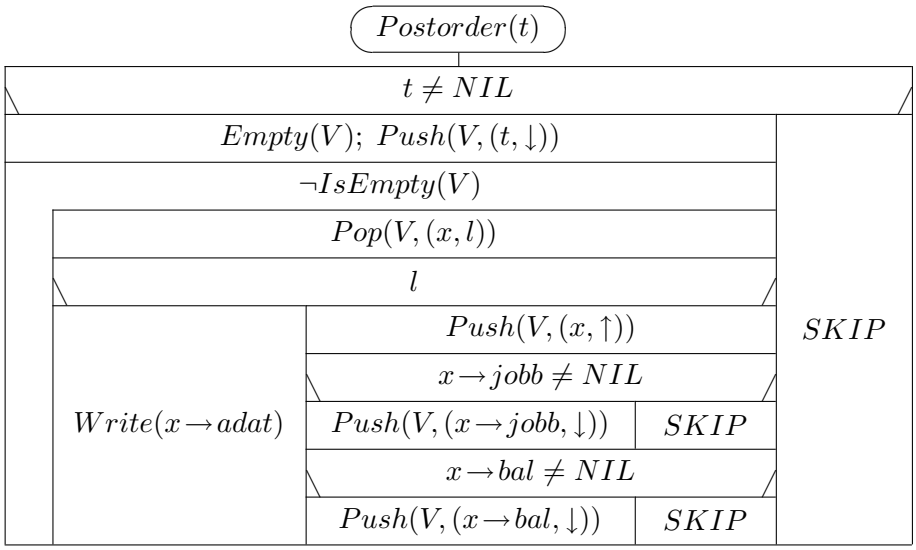
b) Próbáljunk meg iteratív algoritmusokat adni. Ezek nyilván bonyolultabbak lesznek, de a gyakorlatban sokszor hatékonyabbak.

Mindhárom bejárásra sokféle iteratív algoritmus adható, most a célunk az, hogy egy könnyen érthető általános módszert mutassunk be, amely mindegyikhez egyaránt használható. Ehhez induljunk ki a *preorder* bejárásból, és az arra adott algoritmust általánosítjuk majd.

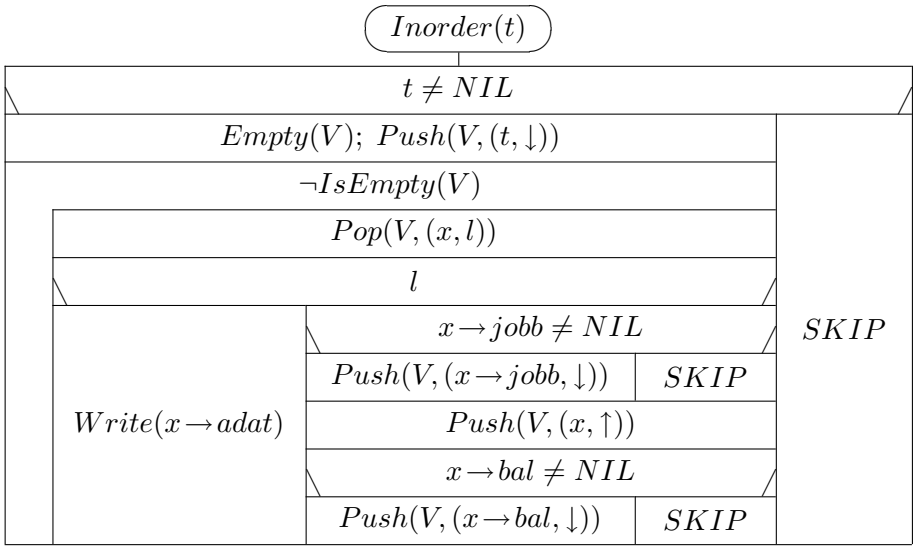
Az algoritmusban egy veremet használunk a soron következő csúcsok pointereinek tárolásához. Először beletesszük a gyökércsúcsot, majd amíg ki nem ürül a verem, minden lépésben kivesszük a legfelső elemét, kiírjuk, és a gyerekeit betesszük a verembe. (Először a nyilván a jobb-, utána a balgyerekét, hogy az utóbbit vegyük majd ki előbb.) Könnyen látható, hogy így valóban *preorder* bejárást kapunk.



Próbáljuk meg ezt az ötletet általánosítani a *postorder* bejárásra. Ilyenkor a ciklusmagban egy veremből kivett csúcsot csak a bal- és jobboldali részfáinak bejárása után írhatunk ki, ezért először ezt a csúcsot is vissza kell tennünk a verembe. Amikor viszont legközelebb kivesszük, tudnunk kell, hogy a gyerekeit már bejártuk, ezért ki lehet írni. Ezt megoldhatjuk úgy, hogy a veremben minden csúcs mellett egy logikai értéket is tárolunk, amely jelzi, hogy ki lehet-e már írni a csúcsot (vagyis azt, hogy először vagy másodszor került-e verembe). Kezdetben a $(t, \downarrow)$ pár kerül a verembe, a ciklusmagban pedig az aktuális csúcs gyerekeit mindig $\downarrow$ értékkel tesszük a verembe, a csúcsot pedig $\uparrow$ értékkel tesszük vissza. (Ez az ötlet a gráfbejárások során használt színezésnek felel meg.)

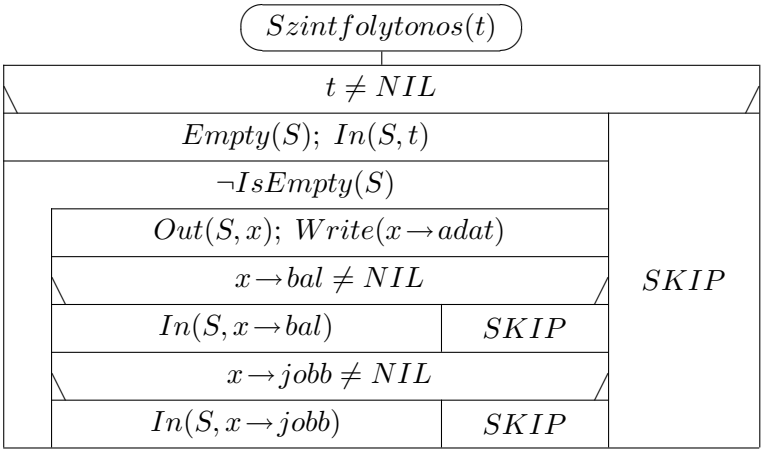


Vegyük észre, hogy ezzel egy általános elvet kaptunk, amely használható a másik két bejárás esetén is. A három *Push* művelet sorrendjének módosításával megkapjuk a *preorder* és az *inorder* bejárást is. Például:



2. Feladat: Adott egy bináris fa a szokásos láncolt ábrázolással. Járjuk be a csúcsait *szintfolytonos* bejárás szerint (írjuk ki a csúcsokban tárolt értékeket).

Megoldás: A feladatot iteratív algoritmussal célszerű megoldani (ugyanis most a fát nem *mélységi*, hanem *szélességi* értelemben akarjuk bejárni). Használjunk ehhez egy sort, amely a soron következő csúcsokat tartalmazza. Először beletesszük a gyökércsúcsot, majd minden iterációban kivesszük a sor első elemét, kiírjuk, és a sor végére betesszük a gyerekeit. Így az algoritmus minden lépésében teljesül, hogy a sor egy k-adik szint utolsó valahány, és a $(k+1)$-edik szint első valahány elemét tartalmazza, és a szintfolytonos bejárás szerint soron következő elem a sor első eleme. Tehát az algoritmus:



Vegyük észre, hogy ez az algoritmus nagyon hasonlít az iteratív *preorder* bejárásra. A különbség csak az, hogy ott sor helyett vermet használtunk, továbbá, hogy ennek megfelelően a bal- és jobbgyereket fordított sorrendben tettük az adatszerkezetbe.

Megjegyzések:

- Az iteratív algoritmusok tár- és műveletigénye a fa csúcsainak számában lineáris, aminél aszimptotikusan jobb nyilván nem adható.
- A (bináris) fák *szintfolytonos* bejárása a gráfok *szélességi* bejárásának speciális esete, a *preorder*, illetve *postorder* bejárás pedig a gráfok *mélységi* bejárásának felel meg. (Még pontosabban: a *preorder* sorrend a *mélységi számozás* sorrendjének, a *postorder* pedig a *befejezési számozás* sorrendjének.)