Rendezések

1. Rendezze a $7, 3, 15, 1, 5, 4, 8, 2$ sorozatot gyorsrendezéssel úgy, hogy mindig a tömb első elemét választja particionáló elemnek!

   Megoldás: $(\mathbf{7}, 3, 15, 1, 5, 4, 8, 2) \longrightarrow (\mathbf{3}, 2, 1, 5, 4 \mid 7 \mid \mathbf{8}, 15) \longrightarrow$
   $(\mathbf{2}, 1 \mid 3 \mid \mathbf{5}, 4 \mid 7 \mid 8 \mid 15\ ) \longrightarrow (\ 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 7 \mid 8 \mid 15\ )$

2. Egy tömbön gyorsrendezést futtatva az első particionálás után az eredmény: $4, 2, 3, 1, 6, 8, 11$. Mi lehetett a particionáló elem?

   Megoldás: Olyan elemet kell keresni, hogy az előtte levők a nála kisebbek, az utána levők a nála nagyobbak. Tehát a 6, a 8 és a 11 is jó.

3. Igazolja, hogy nincs olyan összehasonlításokkal rendező algoritmus, amelynél akármi is a bemenet, minden elem legfeljebb 2020 összehasonlításban szerepel!

   Megoldás: Ha minden elem legfeljebb 2020 összehasonlításban szerepel, akkor az algoritmus az n elemmel összesen $2020n/2$ összehasonlítást végez. De tudjuk, hogy minden algoritmusnak kell legalább $\log n! = \Omega(n \log n)$ összehasonlítás, és $1010n \notin \Omega(n \log n)$, tehát valóban nincs ilyen algoritmus.

4. Adjon egy $O(n)$ időigényű algoritmust n olyan egész számból álló sorozat rendezésére, melynek elemei az
   (a) $\{1, \ldots, 3n\}$ halmazból vannak!
   (b) $\{1, \ldots, n^3 - 1\}$ halmazból vannak!

   Megoldás: (a) Használjunk ládarendezést $3n$ ládával! Ekkor a lépésszám $O(n + 3n) = O(n)$.

   (b) Az egyszerű ládarendezés itt nem elég, használjuk a radix rendezést! Képzeljük el a számokat az n alapú számrendszerben felírva. Ekkor mindegyik (legfeljebb) 3 jegyű. Ha ebben az alakban használjuk a radix rendezést, akkor 3 ládarendezés kell, mindegyiknél n láda, tehát az egész valóban $O(n)$ lépést használ.

   Megjegyzés: Az n alapú számrendszerbe való átírás is megoldható $O(n)$ aritmetikai művelettel, hiszen minden számot maradékosan el kell osztani n-nel, a maradék adja az utolsó jegyet, a hányadost újra elosztjuk n-nel, az itteni maradék adja a következő számjegyet, a hányados meg az elsőt.

5. A $G = (V,E)$ többszörös élet nem tartalmazó irányított gráf csúcshalmaza legyen $V = \{1, 2, \ldots, n\}$. Tegyük fel, hogy a gráf olyan éllistával adott, amelyben minden csúcsnál a szomszédok tetszőleges sorrendben vannak felsorolva. Adjon algoritmust, ami $O(|V|+|E|)$ lépésben olyan éllistát hoz létre, amiben a szomszédok minden csúcsnál növekvő sorrendben vannak!

   Megoldás: Adódna a ládarendezés, de ha minden csúcsra külön végrehajtjuk ($|V|$ ládával), akkor egy d-fokú csúcsnál $O(d + |V|)$, összesen $O(|E| + |V|^2)$ lesz a lépésszám, ami túl sok lehet.

   Ezért inkább az éllista alapján soroljuk fel az éleket (i, j) alakban. Ha ezeken egy radix rendezést hajtunk végre, akkor pont a kívánt sorrendben lesznek az élek, amiből a rendezett éllista már egyszerűen felírható. A lépésszám így: az élek felsorolása, illetve az új éllistába beírásuk $O(|V| + |E|)$, a rendezés, mindkétszer $|V|$ ládát használva, $O(|E| + 2|V|)$, tehát összesen is $O(|E| + |V|)$.

   2. megoldás: Rendezés helyett fordítsuk meg kétszer a gráfot. A megfordítás úgy történjen, hogy az éllistán végigmenve, amikor épp az i csúcs j szomszédjánál tartunk, akkor egy új éllistában a j csúcshoz írjuk be az i-t, mint szomszédot. Világos, hogy ezzel a fordított gráf éllistáját már rendezetten kapjuk. és ezért a második megfordítás után, amikor visszakapjuk az eredeti gráf éleit, annak az éllistája is rendezett lesz.

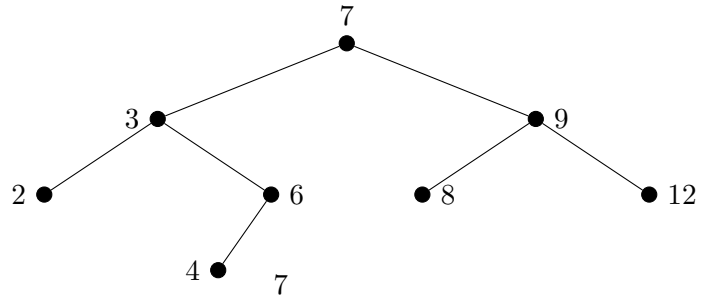
   A megfordítás során csak kiolvassuk és beírjuk az éleket, tehát ennek lépésszáma arányos az éllista méretével, tehát $O(|E| + |V|)$.

6. Egy bináris keresőfában csupa különböző egész számot tárolunk. Mely x egész számokra lehetséges, hogy egy KERES(x) hívás során a keresési út mentén a 20, 18, 3, 15, 5, 8, 9 kulcsokat látjuk ebben a sorrendben?

Megoldás: Nyilván a 20 van a fa gyökerében. Mivel utána a $18 < 20$ jön majd, nyilván balra léptünk, azaz $x < 20$. Hasonlóan a $3 < 18$ miatt megint balra léptünk, tehát $x < 18$. Ekkor egy 3-nál nagyobb szám következett, tehát $x > 3$, és így tovább, $x < 15$, $x > 5$, $x > 8$. Összegezve, biztos teljesül, hogy $8 < x < 15$. Azaz $x = 9, 10, 11, 12, 13, 14$ mindegyike lehet, az első esetben a keresés sikeres volt, a többiben nem.

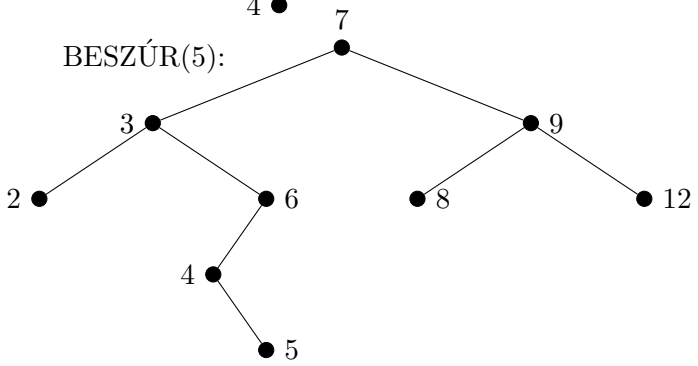
7. (a) Építsen beszúrásokkal bináris keresőfát az alábbi sorrendben érkező számokból: 7, 3, 2, 9, 8, 12, 6, 4.
(b) Járja be pre-. in-, és posztorder bejárással a kapott fát!
(c) Az (a) rész keresőfáján hajtsa végre az alábbi műveletsort: BESZÚR(5), TÖRÖL(2), TÖRÖL(7), TÖRÖL(6).

Megoldás: A végeredmény ez lesz:

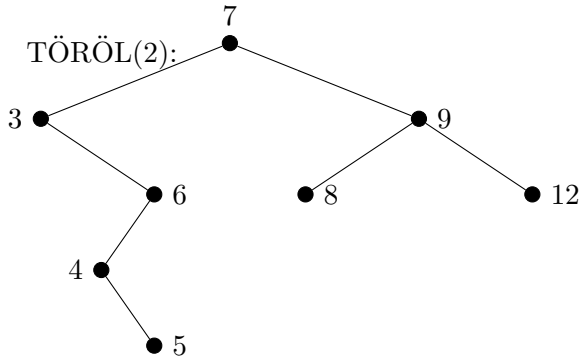


preorder: 7, 3, 2, 6, 4, 9, 8, 12
inorder: 2, 3, 4, 6, 7, 8, 9, 12
posztorder: 2, 4, 6, 3, 8, 12, 9, 7

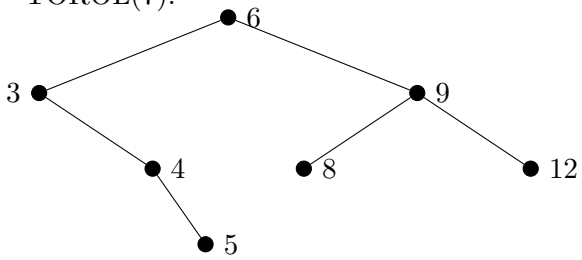
BESZÚR(5):



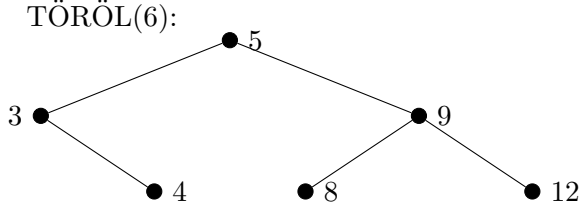
TÖRÖL(2):



TÖRÖL(7):



TÖRÖL(6):



8. Egy bináris fa csúcsai 0 és 9 közötti egész számokkal vannak megcímkézve. Az inorder bejárás során a címkék sorrendje: 9, 3, 1, 0, 4, 2, 7, 6, 8, 5, a posztorder bejárásnál pedig 9, 1, 4, 0, 3, x, 7, 5, y, 2. Mi lehet az x és mi az y?

Megoldás: Látszik, hogy a második listából a 6 és a 8 hiányzik, ezek egyike az x, a másik az y.

Próbáljuk rekonstruálni a fát! A posztorder utolsó eleme a fa gyökere, tehát ez a 2. Az inorderben a 2 előttiek vannak a bal részfában, a 2 utániak a jobban. A 2 gyerekeit megint a posztorderből olvashatjuk ki, a 3 a bal oldalra került elemek közül az utolsó, tehát ez a 2 bal gyereke, a jobb pedig az y. Az y értékétől függetlenül a 7 az y bal részfájában lesz, hiszen az indorder sorrendben megelőzi. Nem lehet $y = 6$, mert akkor az inorder sorrend szerint a bal részfája csak a 7-ből állna, és nem állhatna a posztorder sorrendben előtte az x. Tehát csak $x = 6$ és $y = 8$ lehet. Ez pedig valóban egy jó megoldás, a 2 jobb gyereke az $y = 8$, ennek két fia a 7 és az 5, a 7-nek valamelyik oldali gyereke a 6.

Megjegyzés: ha látni akarjuk, hogy ez tényleg jó megoldás, azt is ellenőrizni kell, hogy a bal részfa sorrendjei sem mondanak ellent egymásnak.

9. Határozza meg azokat a bináris fákat, amelyekben a preorder bejárás szerinti sorrend éppen a posztorder bejárás által adott sorrend fordítottja!

Megoldás: Ha egy csúcsnak két gyereke van, akkor ezek sorrendje ugyanaz a preorder és a posztorder bejárás szerint is. Tehát a fa egyetlen út lehet csak (nincs benne elágazás). Ebben az esetben pedig függetlenül attól, hogy az a gyerek melyik oldalon van, a preorder a fán lefelé haladva sorolja fel az elemeket, a posztorder meg felfelé.

10. Az A keresőfában n egész számot, a B-ben pedig m-et tárolunk. Rendezzük az $n+m$ elemet $O(n+m)$ lépésben!

    Megoldás: Tudjuk, hogy az inorder bejárás növekvő sorrendben adja meg a tárolt elemeket. Olvassuk így ki mindkét fából a tárolt számokat, és a két növekvő listát fésüljük össze. A kiolvasás (bejárás) lépésszáma arányos a fa méretével, az összefésülés $n+m-1$ összehasonlítást és $n+m$ mozgatást igényel, összesen tehát a lépésszám $O(n+m)$.

11. Igazolja, hogy minden olyan algoritmus, ami csak összehasonlításokkal fel tud építeni egy bináris keresőfát n elem esetén $\Omega(n \log n)$ összehasonlítást használ!

    Megoldás: Mivel egy bináris keresőfából az elemek növekvő sorrendjének előállításához már nem kell további összehasonlítás (csak az inorder bejárás), ezért az összes szükséges összehasonlítás a fa elkészítésekor történik. A rendezéshez kell $\Omega(n \log n)$ összehasonlítás, tehát a fa építéséhez is szükség volt ennyire.

12. Adott egy n csúcsú bináris keresőfa, melyben csupa különböző elemeket tárolunk. Ennek minden v csúcsára meg akarjuk határozni, hogy a v gyökerű részfában hány darab v-nél kisebb elem van tárolva. Adjon algoritmust, ami ezt a feladatot $O(n)$ lépésben megoldja!

    Megoldás: Egy keresőfa v gyökerű részfájában v-nél kisebb elemek csak a v bal részfájában lehetnek, a feladatunk minden csúcsra a bal részfa méretének meghatározása. Helyette dinamikus programozással határozzuk meg minden v csúcsra a bal részfa $B[v]$ és a jobb részfa $J[v]$ méretét is! Posztorder sorrendben menjünk végig a csúcsokon. Ha v levél, akkor $B[v] = J[v] = 0$. Ha v nem levél és x a bal, y a jobb gyereke, akkor $B[v] = B[x] + J[x] + 1$ és $J[v] = B[y] + J[y] + 1$. Amennyiben x vagy y nem létezik, akkor $B[v]$ vagy $J[v]$ értéke 0.

    Végül a $B[v]$ értékekre van szükségünk.

    A lépésszám: a bejárás lineáris, minden csúcsnál konstans sok további műveletet végzünk, ezért összesen $O(n)$.