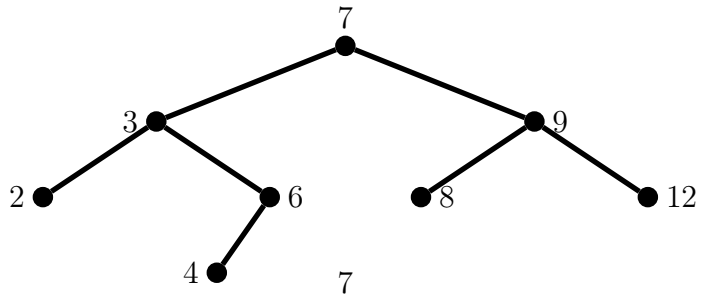
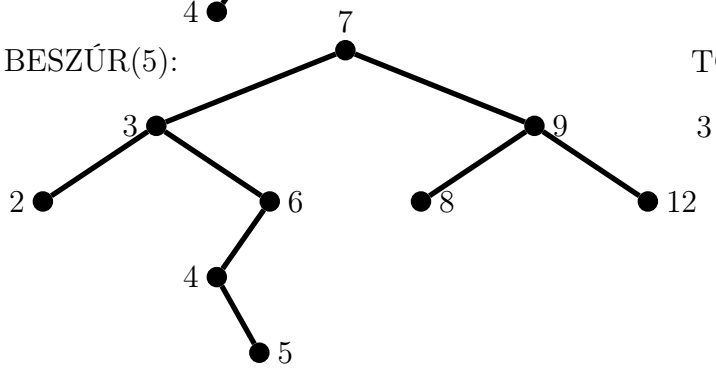
1. (a) Építsen beszúrásokkal bináris keresőfát az alábbi sorrendben érkező számokból:
$$7, 3, 2, 9, 8, 12, 6, 4.$$
(b) Járja be pre-, in-, és posztorder bejárással a kapott fát!
(c) Az (a) rész keresőfáján hajtsa végre az alábbi műveletsort: BESZÚR(5), TÖRÖL(2), TÖRÖL(7), TÖRÖL(6).

Megoldás: A végeredmény ez lesz:

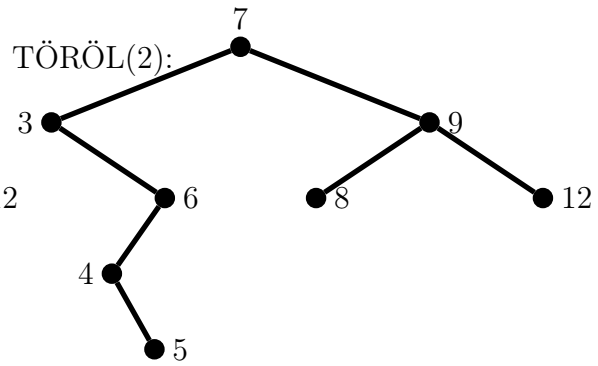


preorder: 7, 3, 2, 6, 4, 9, 8, 12
inorder: 2, 3, 4, 6, 7, 8, 9, 12
posztorder: 2, 4, 6, 3, 8, 12, 9, 7

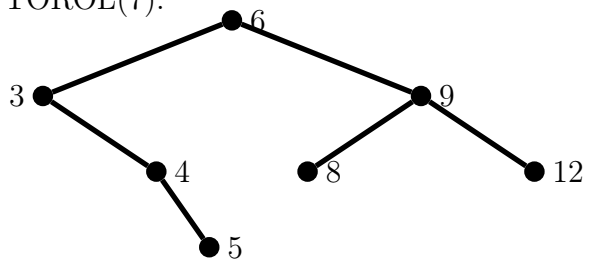
BESZÚR(5):



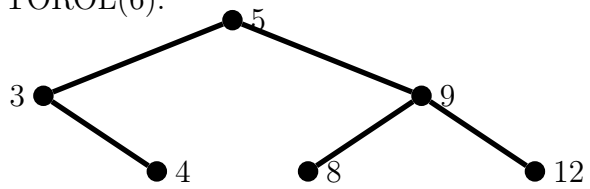
TÖRÖL(2):



TÖRÖL(7):



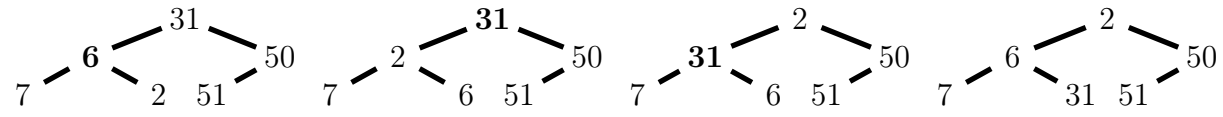
TÖRÖL(6):



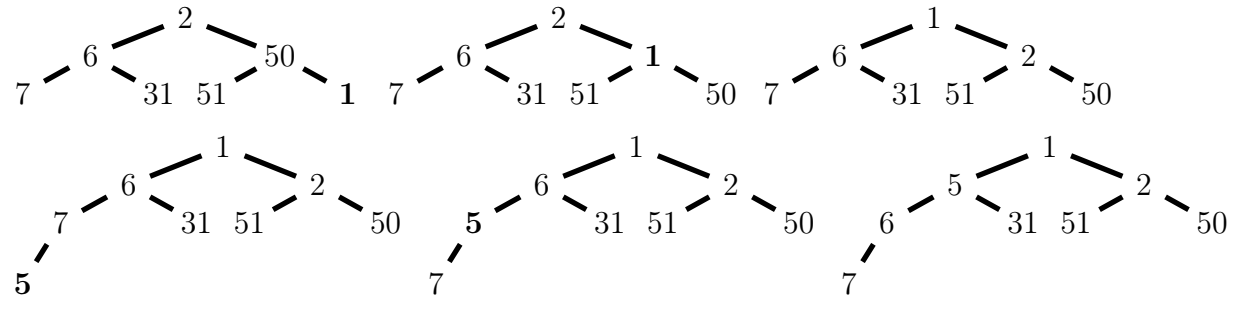
2. (a) Építsen kupacot az órán tanult lineáris idejű módszerrel az alábbi tömbből: $31, 6, 50, 7, 2, 51$.
(b) Szúrja be az így kapott tömbbe az 1, majd ezután az 5 számot.
(c) Hajtson végre két egymást követő MINTÖR-t az így kapott kupacon.

Megoldás:

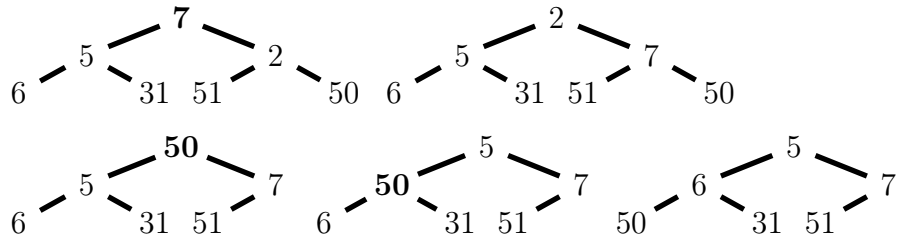
a)



b)



c)



3. Bizonyítsuk be, hogy ha az inorder bejárást egy bináris keresőfára futtatjuk, akkor az elemeket rendezett sorrendben kapjuk meg.

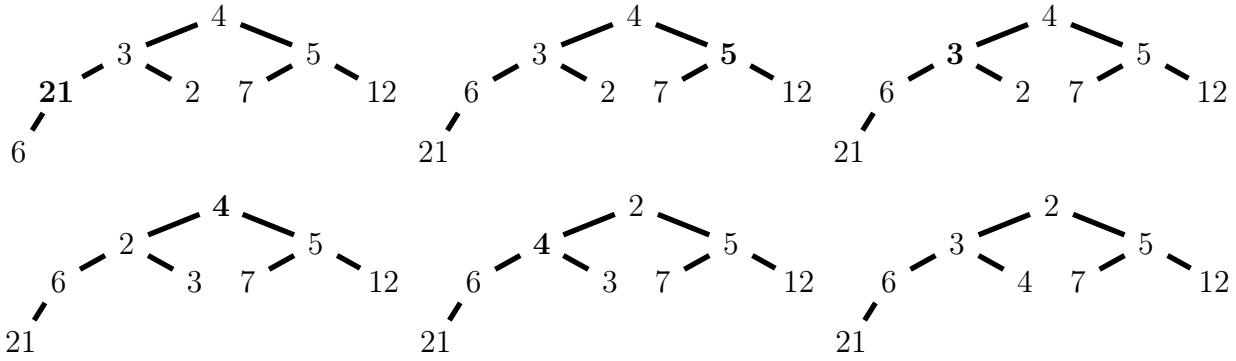
   Megoldás: Azt látjuk be, hogy ha x és y két elem a fában, melyekre teljesül, hogy $x < y$, akkor az inorder bejárás x-et írja ki előbb.

   Három eset van:

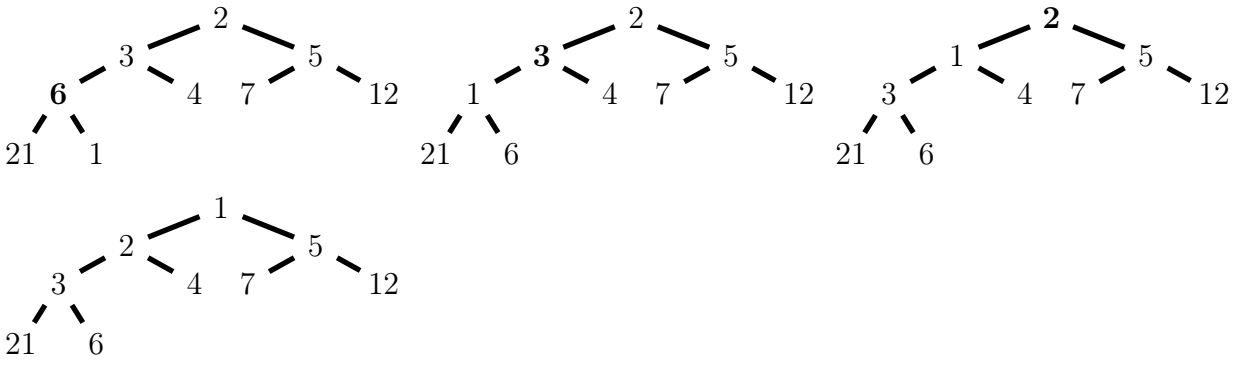
   - Az x csúcs az y csúcs részfájában van: ekkor $x < y$ miatt csak a bal oldali fában lehet, de ekkor az inorder bejárás y bal fáját és így az x csúcsot is y előtt járja be.
   - Az y csúcs az x csúcs részfájában van: ekkor $x < y$ miatt csak a jobb oldali fában lehet, de ekkor az inorder bejárás x jobb fáját és így az y csúcsot is x után járja be.
   - Egyik csúcs sem leszármazottja a másiknak: ekkor van egy olyan közös ősük, aminek egyik részfájában van x, másik részfájában pedig y, legyen ez ős z. De ekkor $x < y$ miatt x a bal, y pedig a jobb részfájában van z-nek és így az inorder bejárás a teljes bal fát (beleértve x-et) z előtt, a teljes jobb fát (beleértve y-t) z után járja be, vagyis x előbb lesz, mint y.

4. (a) Építsen kupacot az órán tanult lineáris idejű módszerrel az alábbi tömbből: $4, 3, 5, 21, 2, 7, 12, 6$.
   (b) Szúrja be az így kapott tömbbe az 1 számot.
   (c) Hajtson végre két egymást követő MINTÖR-t az így kapott kupacon.

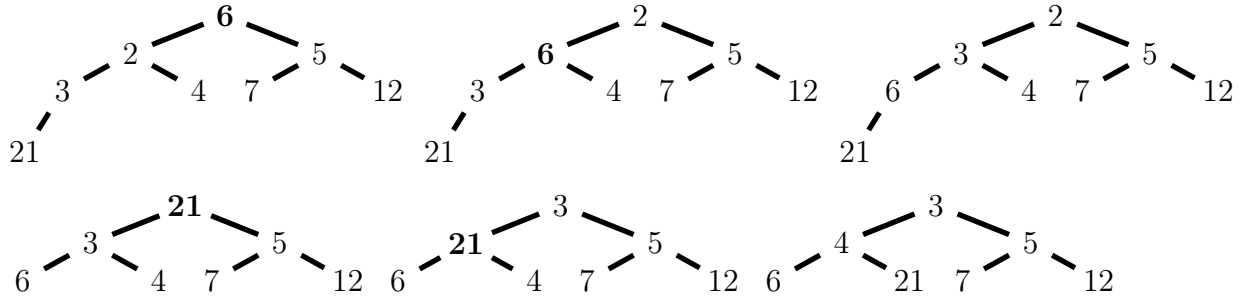
   Megoldás: a)



   b)



c)



5. Igazoljuk, hogy egy n elemből álló bináris kupac felépítése bármilyen a tanult kupacépítés módszerétől eltérő módszerrel is $\Omega(n)$ összehasonlítást igényel!

   Megoldás: Ha már fel van építve a kupac, a legkisebb elem további összehasonlítások nélkül megtalálható (a kupac gyökerében). Tehát a kupacépítéshez legalább annyi összehasonlítás szükséges, mint a legkisebb elem kiválasztásához, ami $n - 1$.

6. Adott egy n elemet tartalmazó kupac és egy k kulcs. Keressük meg a kupac k-nál kisebb kulcsú elemeit! Ha m ilyen elem van, az algoritmus $O(m)$ elemi lépést használhat.

   Megoldás: A fa preorder bejárásához hasonló módszert adunk, azzal kiegészítve, hogy megfelelő körültekintéssel egész részfák bejárását megtakarítjuk (vö. *branch and bound* technika). Amikor a kupac egy elemét meglátogatjuk, hasonlítsuk össze k-val. Ha kisebb, írjuk ki ezt az elemet, és folytassuk a bejárást az esetleges gyermekeire. Ha nem, ne folytassuk, mert a kupac-tulajdonság miatt semelyik elem nem jöhet szóba a részfából. A költség $O(m)$, hiszen egy k-nál kisebb elemnek legfeljebb a két fiát látogathatjuk meg „feleslegesen”.

7. 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.

---

8. A kezdetben üres kupacba egyenként szúrunk be n elemet. Igazolja, hogy előfordulhat, hogy a beszúrások során végzett összehasonlítások száma $\Omega(n \log n)$.

   Megoldás: Igen, ha például $n, n-1, n-2, \ldots, 1$ sorrendben szúrunk be, akkor az összehasonlítások száma

$$\sum_{i=1}^{n} \lfloor \log i \rfloor \geq \sum_{i=1}^{n} ((\log i) - 1) = \left(\sum_{i=1}^{n} \log i \right) - n \geq \left(\sum_{i=n/2}^{n} \log \frac{n}{2} \right) - n = \frac{n}{2} \log \frac{n}{2} - n \in \Omega(n \log n).$$

9. Egy halmazból n elem kupacban van elhelyezve, bármely kettő összehasonlítható. Bizonyítsuk be, hogy a legnagyobb elem megkereséséhez $\Omega(n)$ összehasonlítás szükséges!

   Megoldás: A legnagyobb elem nyilván a levelek között helyezkedik el. Ahhoz, hogy rábizonyítsuk egy levélre, hogy a többi levélnél nagyobb, minden további levélnek legalább egy másik kupacbeli elemnél kisebbnek kell bizonyulnia. Mivel a kupac-tulajdonság semelyik levélre nem garantál ilyesmit, legalább *levelek száma* $- 1 = \Omega(n)$ új összehasonlítás szükséges.

10. Adjunk hatékony algoritmust egy kupac tizedik legkisebb elemének a megtalálására. Elemezzük a módszer költségét.

Megoldás: A tizedik elem távolsága a gyökértől legfőljebb 9. Az ilyen elemek száma legfőljebb $2^{10} - 1 = 1023$ és ezek a kupacot reprezentáló tömb első 1023 helyén vannak. A konstans sok elem közül a tizedik legkisebb kikeresése **konstans** időt igényel: Ha 10-szer végrehajtunk egy minimumkeresést, akkor $1022 + 1021 + \ldots + 1013 = 10175$ műveletet végzünk, ami nem függ a kupac méretétől.

Megjegyzés: Ennél gyorsabb, ha észrevesszük, hogy a tömb első 1023 eleme egy 9 mélységű kupac. Ha ebben 10-szer végrehajtjuk a MINTÖR műveletet, akkor csak $3 \cdot 90 = 270$ műveletet végzünk. (Ha az eredeti kupacban végeznénk 10 MINTÖRt, az $3 \log n$ lenne.)

11. Egy orvosi rendelőben a regisztrációnál kell bejelentkezni, ahol az ott dolgozók eldöntik, hogy a beteg az épp rendelő két orvos közül A-hoz vagy B-hez kell kerüljön, vagy bármelyikükhöz kerülhet. Ezen kívül, a beutaló ismeretében, a beteghez egy sürgősséget kifejező számot is rendelnek. Amikor valamelyik orvos végzett egy beteggel, akkor a betegek közül, akiket nem csak a másik orvos láthat el behívja a legnagyobb sürgősségi számút. Tegyük fel, hogy a kiosztott sürgősségi számok egymástól különbözőek. Írjon le egy olyan adatszerkezetet, ami abban az esetben, ha n beteg várakozik, akkor a regisztráción az új beteg beillesztését, illetve az orvosoknak a következő beteg kiválasztását és az adatstruktúra frissítését $O(\log n)$ lépésben lehetővé teszi.

Megoldás: 3 kupacot használunk, az elsőbe azoknak a betegeknek a sürgősségi számai kerülnek, akiket csak az A orvos kezelhet, a másodikba azokéi, akiket csak a B orvos kezelhet, a harmadikba pedig azok, akiket A és B is kezelhet. Itt a növekvő sorrend, ami alapján a kupacot felépítjük, a sürgősségi számok szerinti csökkenő sorrendnek felel meg (tehát pl. a kupac gyökerébe a legnagyobb sürgősségi szám kerül).
A regisztráción az új beteg beillesztése úgy zajlik, hogy a beteg sürgősségi számát beszúrjuk (a szokásos kupacbeli beszúrással) a neki megfelelő kupacba, ami $O(\log n)$ lépés, hiszen a három kupacban összesen legfeljebb n elem van, tehát külön-külön is.
Mindkét orvos a következő betege kiválasztásához $O(1)$ lépésben kiolvassa és összehasonlítja az általa kezelhető betegek sürgősségi számait tartalmazó két kupac gyökerében lévő elemeket. Ezután a kettő közül amelyik a nagyobb, az annak megfelelő beteget hívja be az orvos, majd az adott kupacon (a regisztációs szoftver segítségével) végrehajt egy MINTÖR-t, amihez $O(\log n)$ lépés elég, így a beteg kiválasztásához is. A MINTÖR után a kupactulajdonság helyreáll, így a következő beteg kiválasztása is ugyanígy történhet.

12. Egy halmaz adott n eleméből (bármely két elem összehasonlítható) kupacot szeretnénk építeni, melyre (a kupac tulajdonság mellett) teljesül, hogy minden x csúcs bal részfájának az elemei kisebbek mint az x jobb részfájának az elemei. Igazoljuk, hogy ehhez legalább $\Omega(n \log n)$ lépés szükséges!

Megoldás: Tegyük fel, hogy van egy algoritmus a kívánt struktúra létrehozására, akkor az eredményül kapott fából a PREORDER éppen egy rendezett listát adna $O(n)$ lépésben. Vagyis legalább annyi lépés kell a kívánt struktúra létrehozására, mint a rendezésre, ami $\Omega(n \log n)$.

13. Adott n pont a síkon, melyek páronként mindkét koordinátájukban különböznek. Bizonyítsuk be, hogy egy és csak egy bináris fa létezik, melynek csúcsaiban az adott n pontot tároljuk, és az első koordináta szerint a keresőfa tulajdonsággal, a második szerint pedig a kupac tulajdonsággal rendelkezik. (*Vigyázat:* a kupac tulajdonságba nem értendő bele, hogy a fa teljes bináris fa legyen, mint amilyet a tanult „kupacépítő" algoritmus létrehoz.)

Megoldás: A kupac-tulajdonság miatt a gyökérben az a P pont van, amelynek a legkisebb a második koordinátája. A keresőfa-tulajdonság miatt a bal részfában azok a pontok vannak, amelyeknek az első koordinátája kisebb P-énél, a jobb részfában pedig azok, amelyek első koordinátája nagyobb P-énél. Ez alapján az állítás indukcióval egyszerűen adódik.

14. Egy bináris keresőfában csupa különböző egész számot tárolunk. Lehetséges-e, hogy egy KERES(x) hívás során a keresési út mentén a $20, 18, 3, 15, 5, 8, 9$ kulcsokat látjuk ebben a sorrendben? Ha nem lehetséges, indokolja meg miért nem, ha pedig lehetséges, határozza meg az összes olyan x egész számot, amire ez megtörténhet.

Megoldás: Mivel a második kulcs kisebb, mint az első, ezért $x < 20$. Hasonlóan $x < 18$. A következő kulcs viszont nagyobb, ezért $x > 3$. Így folytatva: $x < 15$, $x > 5$, $x > 8$. Az utolsó kulcs 9, így lehet, hogy épp ez volt a keresett elem. Ha nem, akkor x nincs a fában. Ennek alapján, ha $8 < x < 15$, akkor pont ezeket a kulcsokat látjuk.