9. előadás

Speciális fák

Piros-fekete fa, kupac

Adatszerkezetek és algoritmusok előadás
2011. április 13.

Kósa Márk és Pánovics János
Debreceni Egyetem
Informatikai Kar

A piros-fekete fa definíciója

Piros-fekete fa

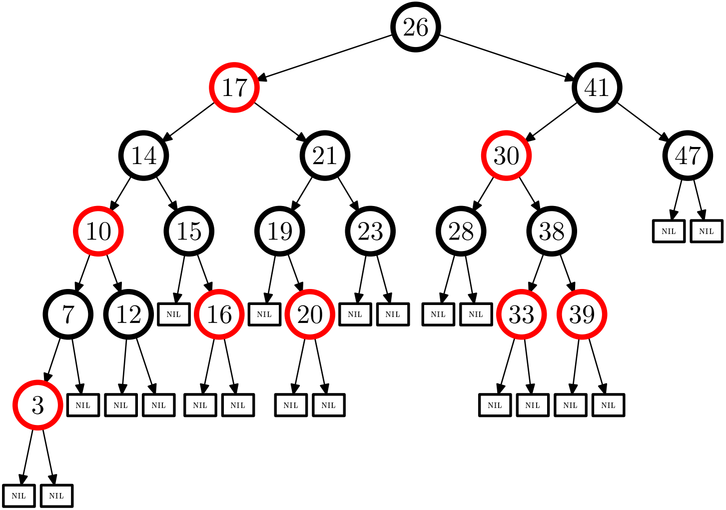
A piros-fekete fa olyan bináris keresőfa, amely a következő tulajdonságokkal rendelkezik:

1. Minden csomópontja piros vagy fekete.
2. A gyökere fekete.
3. Minden (NIL értékű) levele fekete.
4. Ha egy csomópont piros, akkor mindkét rákövetkezője fekete. (Más szavakkal kifejezve: nincs benne két egymást követő piros csomópont.)
5. Minden csomópont esetén az összes olyan úton, amely az adott csomópontból indul ki és levélig vezet, ugyanannyi a fekete csomópontok száma.

Megjegyzés

Egy n adatelemet tartalmazó piros-fekete fa magassága legfeljebb $2\log(n+1)$.

Példa piros-fekete fára

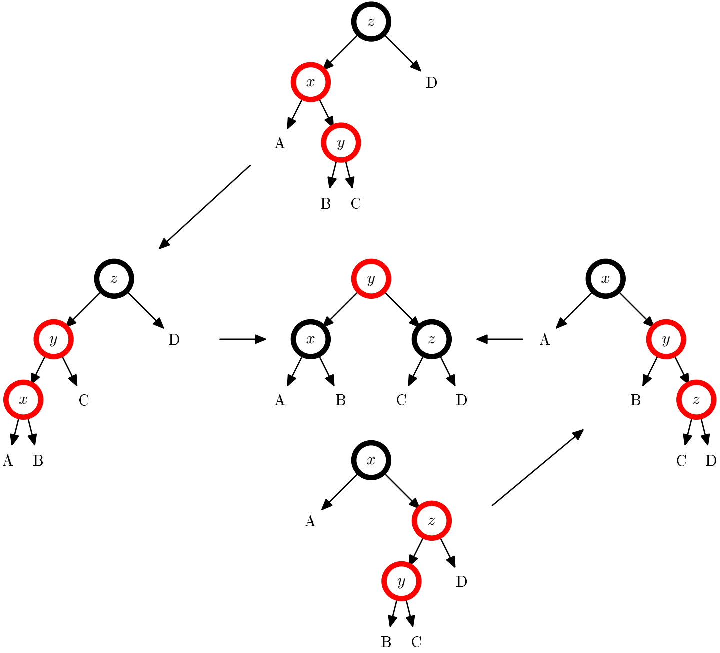


Beszúrás piros-fekete fába (Okasaki-módszer)

Egy piros-fekete fát úgy bővítünk, mint egy keresőfát: mindig levélelemmel, amelyet pirosra színezünk. A levélelemmel történő bővítést követően a következő esetek fordulhatnak elő:

1. A fa továbbra is rendelkezik a piros-fekete tulajdonságokkal. Ekkor nincs teendőnk, készen vagyunk.
2. Nem teljesül a 2-es tulajdonság, miszerint a gyökérelem fekete. Ez csak akkor fordulhat elő, ha éppen a gyökeret szúrtuk be, azaz a fa előzőleg üres volt. Ekkor átszínezzük a beszúrt (gyökér)elemet feketére, és készen vagyunk.
3. Nem teljesül a 4-es tulajdonság, miszerint nincs a fában két egymást követő piros csomópont. Ez csak akkor fordulhat elő, ha a beszúrt elem szülője is piros. Mivel a gyökér fekete, a beszúrt elemnek biztosan létezik nagyszülője, amelynek a 4-es tulajdonság miatt feketének kell lennie. Ekkor forgatásokat és átszínezéseket kell végrehajtanunk.

Beszúrás piros-fekete fába (Okasaki-módszer)



Beszúrás piros-fekete fába (Okasaki-módszer)

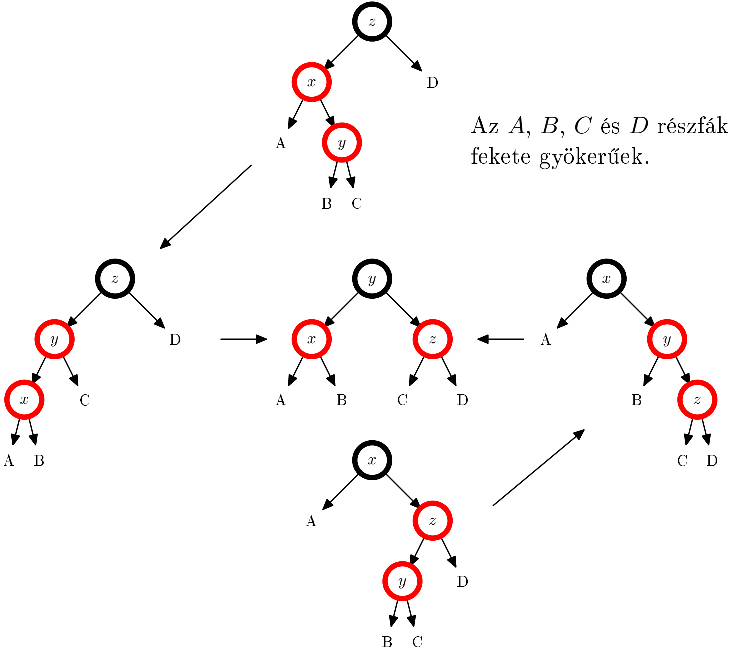
A fenti transzformáció (egy vagy két forgatás, valamint egy átszínezés) után az y szülőjéből (ha létezik) bármelyik levélbe vezető úton ugyanannyi fekete elem lesz, mint amennyi a transzformáció előtt volt. Az így kapott fa már vagy piros-fekete fa, vagy nem teljesül a 2-es tulajdonság (ha y a gyökérelem), vagy nem teljesül a 4-es tulajdonság (ha y szülője piros).

A fenti transzformációt tehát addig kell ismételnünk, amíg y szülője fekete nem lesz (ekkor nincs további teendőnk), vagy y a gyökér nem lesz. Utóbbi esetben átszínezzük y-t feketére, és készen vagyunk. (A gyökérelem feketére színezésével a gyökérből az egyes levelekbe vezető utak mindegyikén ugyanannyival nő a fekete csomópontok száma, tehát az 5-ös tulajdonság továbbra is fennáll.)

Beszúrás piros-fekete fába (CLRS-módszer)

A CLRS-módszer abban különbözik az Okasaki-módszertől, hogy hogyan kezeli azt az esetet, amikor a beszúrás után nem teljesül a 4-es tulajdonság. Ekkor két esetet különböztetünk meg attól függően, hogy a beszúrt elem nagybácsija (a szülőjének a testvére) piros-e vagy fekete. Tételezzük fel először, hogy fekete! Ekkor hasonló forgatásokat hajtunk végre, mint az Okasaki-módszer esetén, viszont utána az y csomópont lesz fekete, míg a két gyermeke (x és z) piros. Ezáltal biztosan teljesülni fog a 4-es tulajdonság, így nincs szükség további forgatásokra és átszínezésekre (természetesen a 2-es tulajdonság is teljesül).

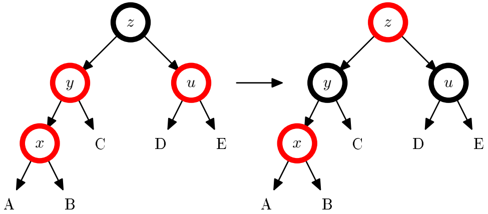
Beszúrás piros-fekete fába (CLRS-módszer)



Az A, B, C és D részfák fekete gyökerűek.

Beszúrás piros-fekete fába (CLRS-módszer)

Mi történik akkor, ha a beszúrt elem nagybácsija piros? Ebben az esetben a beszúrt elem szülőjét és annak testvérét (a nagybácsit) feketére színezzük, a szülőjüket pedig pirosra. Forgatást ilyenkor nem kell végrehajtani. Az egyik lehetséges esetet szemlélteti a következő ábra:



Könnyen látható, hogy az átszínezés után nem változik a gyökérből a levelekbe vezető utakon a fekete elemek száma. Előfordulhat viszont, hogy nem teljesül a 2-es vagy a 4-es tulajdonság. Az eljárást tehát mindaddig kell ismételnünk, amíg (i) z szülője fekete nem lesz (ekkor készen vagyunk), (ii) z a gyökér nem lesz (amit átszínezünk feketére, és készen vagyunk), vagy (iii) z nagybácsija fekete nem lesz (ekkor végrehajtunk egy vagy két forgatást, és készen vagyunk).

Törlés piros-fekete fából

Kósa Márk
Pánovics János

Egy piros-fekete fából ugyanúgy törlünk, mint egy keresőfából. Az a csomópont, amelyet eltávolítottunk a fából, nem feltétlenül az a csomópont, amely a törölt adatelemet tartalmazta. A piros-fekete tulajdonságok helyreállításához az eltávolított csomópontot kell figyelembe vennünk. Legyen ez a csomópont v, a szülője pedig $p(v)$!

Az eltávolított csomópont (v) legalább egyik gyermekének levélnek kell lennie. Ha v-nek van egy nem levél gyermeke, akkor a helyét az a bizonyos gyermek, különben pedig egy levélelem veszi át. Legyen u az a gyermek, amelyik v helyére kerül a törlés után! Ha u levél, akkor tudjuk, hogy fekete.

Ha v piros, akkor készen vagyunk, mivel egyetlenegy piros-fekete tulajdonságot sem sértettünk. Tehát tegyük fel, hogy v fekete!

Törlés piros-fekete fából

A gyökérből a levelekbe vezető azon utak, amelyek keresztülmennek v-n, eggyel kevesebb fekete csomópontot fognak tartalmazni, mint a többi gyökér-levél út a fában, és ez megsérti az 5-ös tulajdonságot. Ha $p(v)$ és u is piros, akkor a 4-es tulajdonságot is megsértjük, de látni fogjuk, hogy az 5-ös tulajdonság helyreállítása a 4-es tulajdonságot is helyreállítja további teendők nélkül, ezért mi most az 5-ös tulajdonság helyreállítására koncentrálunk.

Képzeljük el, hogy egy fekete tokent rendelünk u-hoz! Ez a token azt jelzi, hogy az ezen a csomóponton átmenő, levélig vezető utak eggyel kevesebb fekete csomópontot tartalmaznak, mint kellene. (Kezdetben ez azért van, mert v-t kitöröltük.) A tokent a fában egyre feljebb visszük, amíg ki nem alakul egy olyan helyzet, amelyben az 5-ös tulajdonságot helyreállíthatjuk. Ezt a tokent egy kis fekete négyzettel jelöljük az ábrákon. Ha a tokennel rendelkező csomópont fekete, akkor azt duplán fekete csomópontnak nevezzük. (A token csak egy fogalmi eszköz, fizikailag nem jelenik meg az adatszerkezetben.)

Törlés piros-fekete fából (1. eset)

1. eset: Ha a tokennel rendelkező csomópont piros, vagy a fa gyökere (vagy mindkettő), akkor egyszerűen színezzük át feketére, és készen vagyunk. Vegyük észre, hogy ez a 4-es tulajdonságot (nincs két egymást követő piros csomópont) helyreállítja. Az 5-ös tulajdonságot is helyreállítja a következő okból. A token azt jelezte, hogy a gyökérből az ezen a csomóponton áthaladó, levélig vezető utaknak egy újabb fekete csomópontra lenne szükségük ahhoz, hogy a többi gyökér-levél útnak megfeleljenek. Azáltal, hogy ezt a piros csomópontot feketére színeztük, pontosan azon utakhoz adtunk egy újabb fekete csomópontot, amelyekben eggyel kevesebb volt a kelleténél.

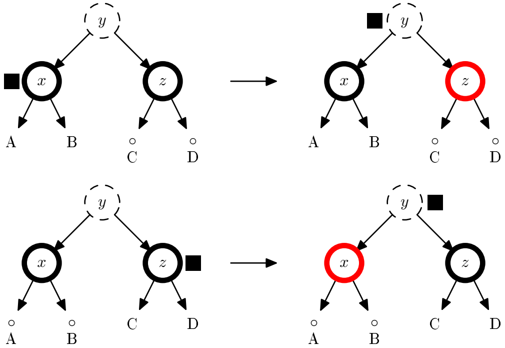
Ha a token a gyökérben van, és a gyökér fekete, a tokent egyszerűen eldobjuk. Ebben az esetben minden gyökér-levél úton eggyel csökkentettük a fekete csomópontok számát, így az 5-ös tulajdonság továbbra sem sérül.

A további esetekben feltételezhetjük, hogy a tokennel rendelkező csomópont fekete, és nem a gyökér.

Törlés piros-fekete fából (2. eset)

2. eset: Ha a duplán fekete csomópont testvére és mindkét unokaöccse fekete, akkor a testvért pirosra színezzük, a tokent pedig egy csomóponttal feljebb visszük a gyökér irányába.

Az alábbi ábrán, amely a két lehetséges alesetet mutatja, az *y* körüli szaggatott vonal jelzi, hogy ezen a ponton nem érdekel minket *y* színe, az *A*, a *B*, a *C* és a *D* fölötti kis karikák pedig azt jelzik, hogy az adott részfa gyökere fekete.



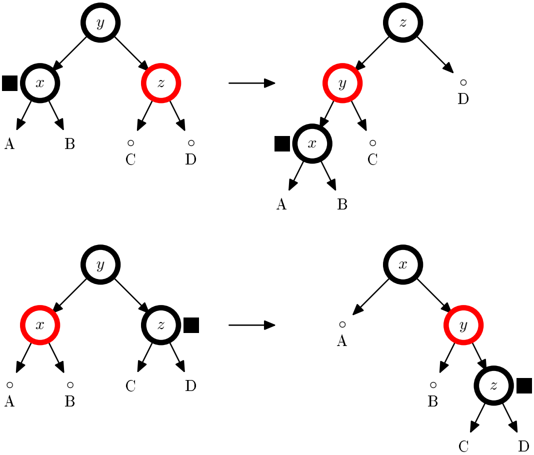
Törlés piros-fekete fából (2. eset)

A testvér pirosra színezése kitöröl egy fekete csomópontot a belőle elérhető levelekhez vezető utakból, így azokon az utakon ugyanannyi fekete csomópont lesz, mint amennyi a duplán fekete csomópontból elérhető levelekhez vezető utakon. A tokent felvisszük az y szülőbe, jelezve, hogy *minden* y alatti út most eggyel kevesebb fekete csomópontot tartalmaz, mint kellene. Nem oldottuk meg a problémát, csak egy szinttel feljebb toltuk a gyökér felé.

Ezt a műveletet nyilván csak akkor hajthatjuk végre, ha mindkét unokaöcs fekete, hiszen különben egymást követő piros csomópontokat kapnánk.

Törlés piros-fekete fából (3. eset)

3. eset: Ha a duplán fekete csomópont testvére piros, akkor egy forgatást és egy színcserét kell végrehajtani. A két lehetséges esetet a következő ábra mutatja:



Törlés piros-fekete fából (3. eset)

Ez a lépés nem változtatja meg a gyökérből a levelekhez vezető utakon a fekete csomópontok számát, de garantálja, hogy a duplán fekete csomópont testvére fekete lesz, amelynek következtében vagy a 2., vagy a 4. eset fog előállni.

Úgy tűnhet, hogy rontottunk a helyzeten, mivel a token *távolabb* került a gyökértől, mint korábban volt. Mivel azonban a duplán fekete csomópont szülője már piros, ezért ha a 2. eset állt elő, akkor a tokent egy piros csomópontba fogjuk továbbítani, amely aztán feketévé alakul, és készen leszünk. Ha pedig a 4. eset áll elő, akkor – ahogy mindjárt látni fogjuk – mindig eltűnik a token, és befejeződik a művelet. Ez a „visszalépés” tehát annak a jele, hogy már majdnem készen vagyunk.

Törlés piros-fekete fából (4. eset)

4. eset: Végül maradt az az eset, ahol a duplán fekete csomópontnak fekete a testvére, és legalább egy piros unokaöccse van. Legyen egy x csomópont közeli unokaöccse x testvérének a bal oldali gyermeke, ha x bal oldali gyermek, és x testvérének a jobb oldali gyermeke, ha x jobb oldali gyermek; és legyen x távoli unokaöccse x másik unokaöccse. (Az ábrán x közeli unokaöccse közelebb van x-hez, mint a távoli.)

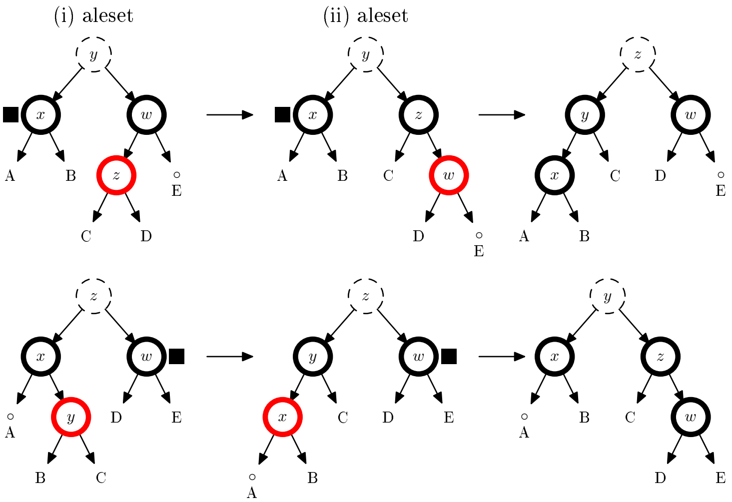
Két aleset létezik:

(i) A duplán fekete csomópont távoli unokaöccse fekete, tehát a közeli unokaöccse piros.

(ii) A távoli unokaöcs piros, tehát a közeli unokaöcs bármilyen színű lehet.

Ahogy a következő ábrán látható, az (i) alesetet egy forgatással és egy színcserével a (ii) alesetre transzformáljuk, a (ii) alesetet pedig egy újabb forgatással és színcserével oldjuk meg. A két sor szimmetrikus, attól függően, hogy a duplán fekete csomópont bal vagy jobb oldali gyermek-e.

Törlés piros-fekete fából (4. eset)



Ebben az esetben előállítunk egy extra fekete csomópontot, a tokent eldobjuk, és készen vagyunk. Ahogy az ábrán látható, a token alatti levelekhez vezető utakon a fekete csomópontok száma eggyel nő, míg a többi útvonalon változatlan marad, és a többi piros-fekete tulajdonság sem sérül.

A kupac definíciója

Kupac

A kupac olyan fa, amely rendelkezik a kupac tulajdonsággal: a gyökérelemet kivéve bármely adatelemének a kulcsa kisebb vagy egyenlő az adatelem szülőjének a kulcsánál. Az ilyen fában a legnagyobb kulcsú elem mindig a gyökérelem, ezért max-kupacnak is nevezzük. Ha megfordítjuk a relációt, akkor a gyökérelem lesz a legkisebb kulcsú elem, ekkor min-kupacot kapunk.

Megjegyzés

A kupac egyes elemeiben a gyermek csomópontok számára nézve általában nincs megszorítás. A kupac adatszerkezetnek rengeteg változata létezik attól függően, hogy hány gyermek csomópontja lehet az egyes elemeknek.

Kósa Márk
Pánovics János

Bináris kupac

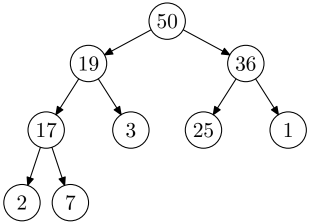
Bináris kupac

A bináris kupac olyan bináris fa, amely a kupac tulajdonságon kívül az alak tulajdonsággal is rendelkezik: a fa teljes bináris fa, azaz minimális magasságú, és ha a legalsó szint nincs teljesen kitöltve, akkor azon a szinten a csomópontok balról jobbra kerülnek feltöltésre.

Megjegyzés

A kupac tulajdonság nem határozza meg a gyermek csomópontok sorrendjét, ezért azok tetszőlegesen felcserélhetők, hacsak meg nem sértik az alak tulajdonságot.

Példa bináris max-kupacra:

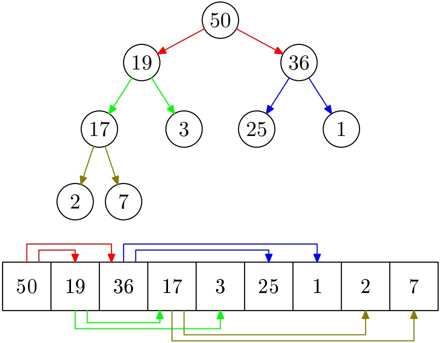


A bináris kupac reprezentációja

Az alak tulajdonság miatt a bináris kupacot leggyakrabban egy tömbbel reprezentáljuk. Nincs szükség mutatókra, mivel bármely adatelem szülőjének és gyermekeinek az indexe egyszerű számtani műveletekkel meghatározható az adatelem indexéből. Ha a tömb indexelése 1-ről indul, akkor az a_i elem

- gyermekei az a_{2i} és az a_{2i+1},
- szülője az $a_{\left[\frac{i}{2}\right]}$

elem lesz (ha létezik).



Kupacrendezés

A kupac segítségével viszonylag egyszerűen implementálható egy olyan helyben rendező algoritmus, amely általános esetben majdnem olyan gyors, mint a gyorsrendezés, a legrosszabb esetben viszont gyorsabb annál.

Az ötlet

Először is bináris max-kupaccá alakítjuk a rendezendő tömböt. Ezután kicseréljük a tömb első (legnagyobb) elemét az utolsóval, amely így a helyére kerül. Helyreállítjuk a bináris max-kupacot az utolsó elem elhagyásával kapott résztömbben, majd kicseréljük az első elemet az utolsó előttivel, és így tovább...

Megjegyzés

Az algoritmus alatt a tömb eleje tartalmazza a kupacot, a vége pedig a már rendezett résztömböt.

A kupacrendezés algoritmusa

Az algoritmus bemeneteként adott adatszerkezetet A-val, elemeinek a számát n-nel jelöljük.

procedure KUPACRENDEZÉS(A)
    KUPACOSÍT(A)
    $vég \leftarrow n$
    while $vég > 1$ **do**
        CSERÉL(A, 1, $vég$)
        SZITÁL(A, 1, $vég - 1$)
        $vég \leftarrow vég - 1$
    end while
end procedure

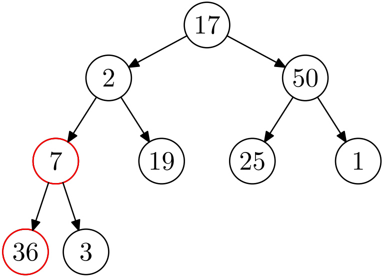
procedure KUPACOSÍT(A)
    $start \leftarrow \lfloor n/2 \rfloor$
        ▷ *start* kezdetben az utolsó nem levél elem indexe
    while $start \geq 1$ **do**
        SZITÁL(A, $start$, n)
        $start \leftarrow start - 1$
    end while
end procedure

A szitálás algoritmusa

```
procedure SZITÁL(A, bal, jobb)
    gyökér ← bal
    while gyökér ∗ 2 ≤ jobb do
        gyerek ← gyökér ∗ 2
                                ▷ gyerek a gyökér bal oldali gyermeke
        csere ← gyökér
            ▷ csere a gyökér azon gyermeke, amelyikkel ki kell őt cserélni
        if A[csere] < A[gyerek] then
            csere ← gyerek
        end if
        if gyerek < jobb and A[csere] < A[gyerek + 1] then
            csere ← gyerek + 1
        end if
        if csere ≠ gyökér then
            CSERÉL(A, gyökér, csere)
            gyökér ← csere
        else
            return
        end if
    end while
end procedure
```

Példa kupacrendezésre

Kupacosítás:

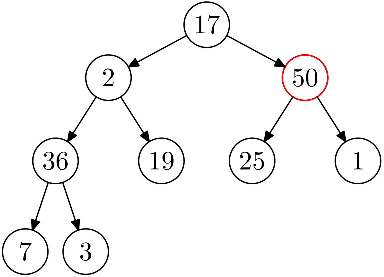


17	2	50	7	19	25	1	36	3

Kósa Márk
Pánovics János

Példa kupacrendezésre

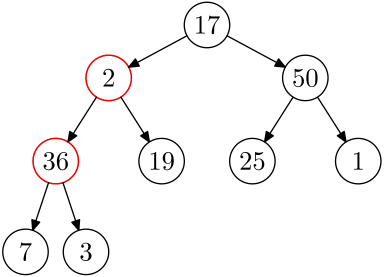
Kupacosítás:



17	2	50	36	19	25	1	7	3

Példa kupacrendezésre

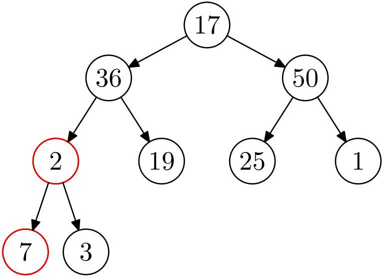
Kupacosítás:



17	2	50	36	19	25	1	7	3

Példa kupacrendezésre

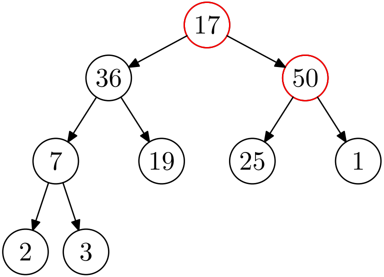
Kupacosítás:



17	36	50	2	19	25	1	7	3

Példa kupacrendezésre

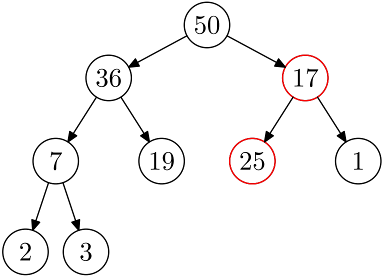
Kupacosítás:



17	36	50	7	19	25	1	2	3

Példa kupacrendezésre

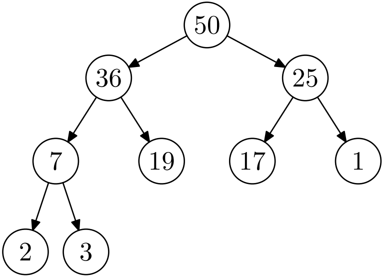
Kupacosítás:



50	36	17	7	19	25	1	2	3

Példa kupacrendezésre

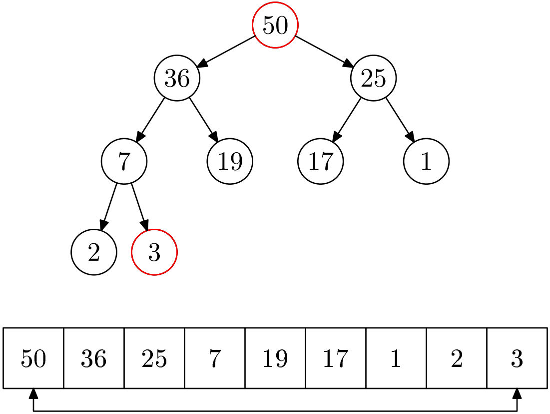
Kupacosítás:



50	36	25	7	19	17	1	2	3

Példa kupacrendezésre

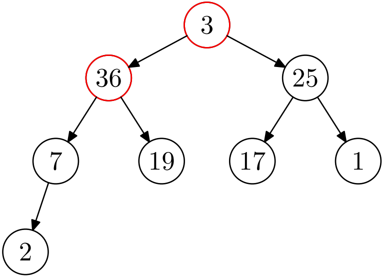
Rendezés:



Kósa Márk
Pánovics János

Példa kupacrendezésre

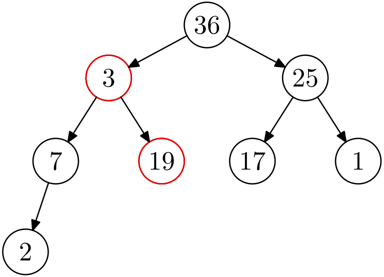
Rendezés:



3	36	25	7	19	17	1	2	50

Példa kupacrendezésre

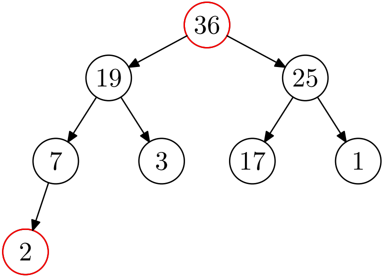
Rendezés:



36	3	25	7	19	17	1	2	50

Példa kupacrendezésre

Rendezés:

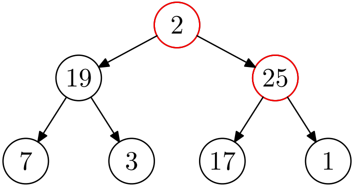


36	19	25	7	3	17	1	2	50

Kósa Márk
Pánovics János

Példa kupacrendezésre

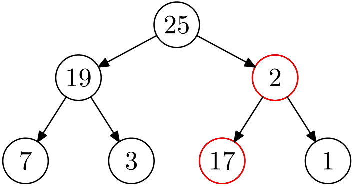
Rendezés:



2	19	25	7	3	17	1	36	50

Példa kupacrendezésre

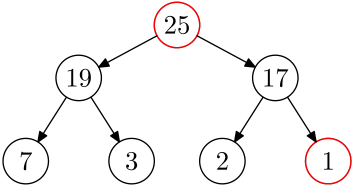
Rendezés:



25	19	2	7	3	17	1	36	50

Példa kupacrendezésre

Rendezés:



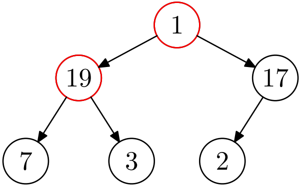
25	19	17	7	3	2	1	36	50

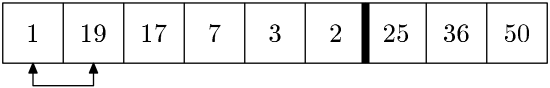
Kósa Márk
Pánovics János

Kósa Márk
Pánovics János

Példa kupacrendezésre

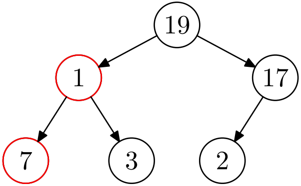
Rendezés:





Példa kupacrendezésre

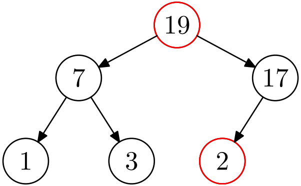
Rendezés:



19	1	17	7	3	2	25	36	50

Példa kupacrendezésre

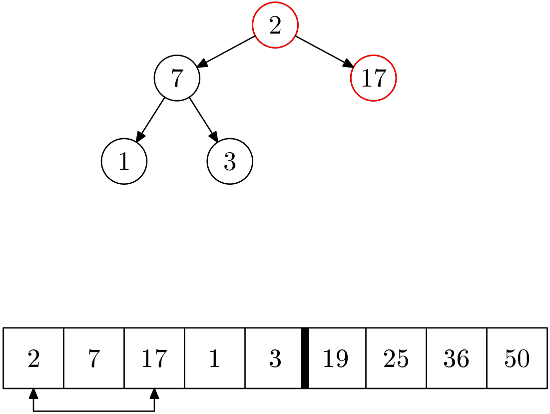
Rendezés:



19	7	17	1	3	2	25	36	50

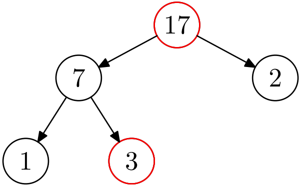
Példa kupacrendezésre

Rendezés:



Példa kupacrendezésre

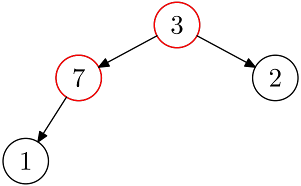
Rendezés:



17	7	2	1	3	19	25	36	50

Példa kupacrendezésre

Rendezés:

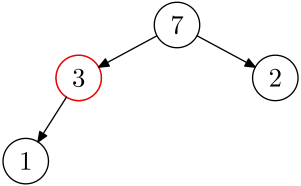


3	7	2	1	17	19	25	36	50

Kósa Márk
Pánovics János

Példa kupacrendezésre

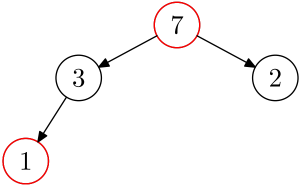
Rendezés:



7	3	2	1	17	19	25	36	50

Példa kupacrendezésre

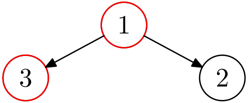
Rendezés:



7	3	2	1	▌	17	19	25	36	50

Példa kupacrendezésre

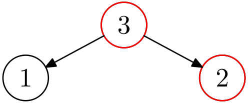
Rendezés:



1	3	2	7	17	19	25	36	50

Példa kupacrendezésre

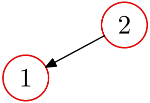
Rendezés:



3	1	2	7	17	19	25	36	50

Példa kupacrendezésre

Rendezés:

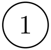


2	1	3	7	17	19	25	36	50

Kósa Márk
Pánovics János

Példa kupacrendezésre

Rendezés:



1	2	3	7	17	19	25	36	50