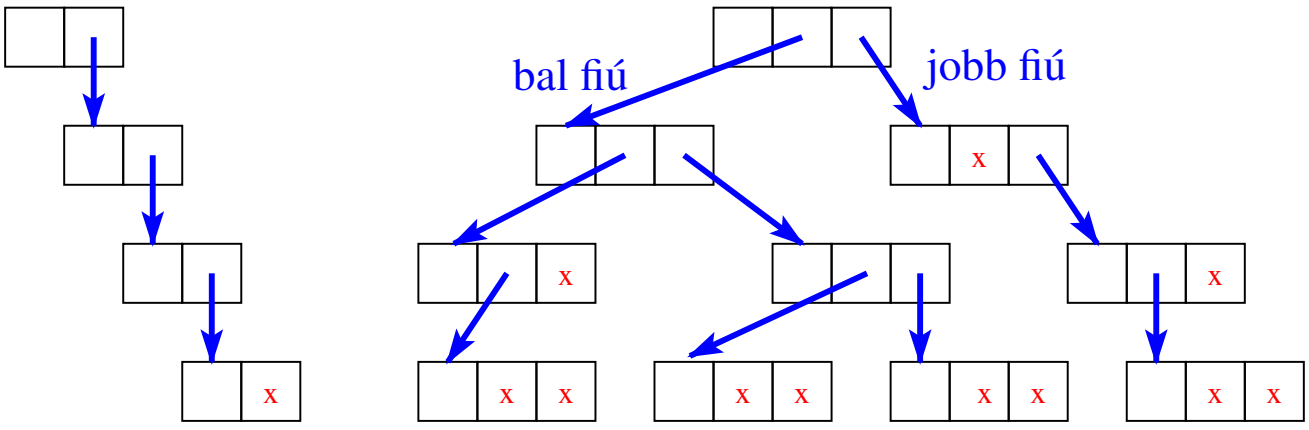
A számítástudomány alapjai

Katona Gyula Y.

Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

Bináris keresőfa, kupac

Bináris fa adatszerkezet

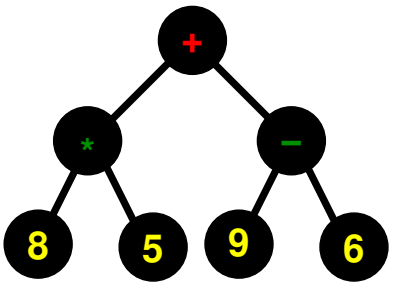


Láncolt lista Bináris fa

Műveletek: gyökér, bal-fiú, jobb-fiú, keres, beszúr, töröl

Bináris fa (Binary tree)

Fa csúcsai $\rightarrow$ $elem(x), bal(x), jobb(x)$ esetleg
$apa(x)$ és $reszfa(x)$ (=az x gyökerű fészfa csúcsainak száma)



ha x a gyökér, y pedig a 9-es csúcs, akkor

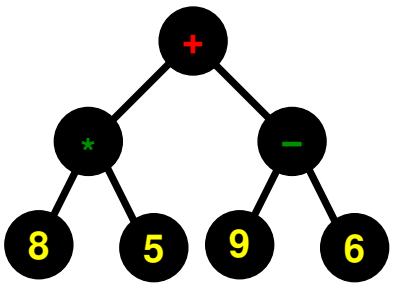
$$\begin{aligned} bal(jobb(x)) &= y, \\ apa(apa(y)) &= x, \\ elem(bal(x)) &= *, \\ reszfa(x) &= 7, \\ reszfa(*) &= 3. \end{aligned}$$

Bináris fa bejárásai

pre(x)
begin
látogat(x);
pre($bal(x)$);
pre($jobb(x)$)
end

in(x)
begin
in($bal(x)$);
látogat(x);
in($jobb(x)$)
end

post(x)
begin
post($bal(x)$);
post($jobb(x)$);
látogat(x)
end



ha x a gyökér, y pedig a 9-es csúcs, akkor

PREORDER: $+ * 85 - 96$

INORDER: $8 * 5 + 9 - 6$

POSTORDER: $85 * 96 - +$

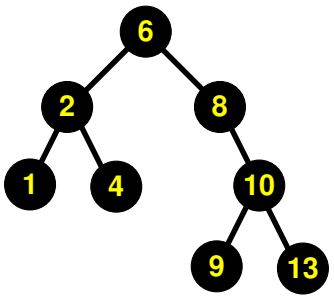
Lépésszám: cn

Bináris keresőfa (Binary search tree)

Tároljuk az U rendezett halmaz elemeit, hogy BESZÚR, TÖRÖL, KERES, MIN, (MAX, TÓLIG) hatékonyak legyenek.

Definíció (Keresőfa-tulajdonság)

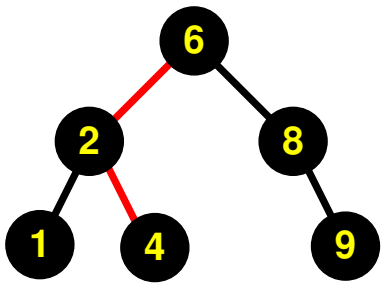
Tetszőleges x csúcsra és az x baloldali részfájában levő bármely y csúcsra igaz, hogy $elem(y) \leq elem(x)$. Hasonlóan, ha z egy csúcs az x jobb részfájából, akkor $elem(x) \leq elem(z)$.



Házi feladat: Igazoljuk, hogy egy bináris keresőfa elemeit a fa inorder bejárása *növekvő sorrendben* látogatja meg.

Egy kényelmes megállapodás: a továbbiakban feltesszük, hogy nincsenek ismétlődő elemek a keresőfában.

Algoritmusok



KERES(4, S)

KERES(s,S): Összehasonlítjuk s-et S gyökerével s'-vel.

- Ha $s = s'$, akkor megtaláltuk.
- Ha $s < s'$, akkor balra megyünk tovább.
- Ha $s > s'$, akkor jobbra megyünk.

Ugyanezt az utat járjuk be a KERES(5, S) kapcsán, de azt nem találjuk meg.

Lépésszám: cl, ahol l a fa mélysége

MIN: mindig balra lépünk, amíg lehet

MAX: mindig jobbra lépünk, amíg lehet

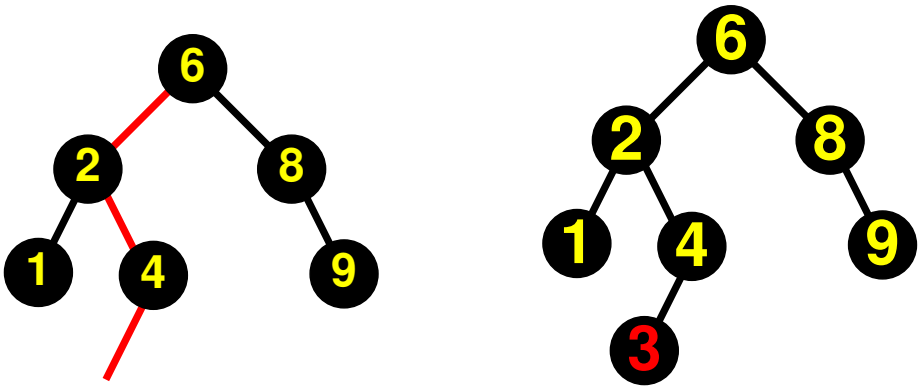
Lépésszám: cl

TÓLIG(a, b, S): KERES(a, S) $\Longrightarrow$ INORDER a-tól b-ig

Lépésszám: cn

BESZÚR

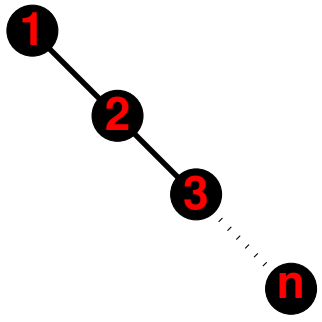
BESZÚR(s, S): KERES(s, S)-sel megkeressük hova kerülne és új levelet adunk hozzá, pl. BESZÚR(3, S):



Lépésszám: cl

Faépítés beszúrásokkal

Ha pl. az $1, 2, \ldots, n$ sorozatból építünk fát így, akkor ezt kapjuk:
Az építés költsége: $2 + 3 + \ldots + (n-1) = cn^2$



Tétel

Ha egy véletlen sorozatból építünk fát naiv beszúrásokkal, akkor az építés költsége átlagosan $c(n \log_2 n)$. A kapott fa mélysége átlagosan $c \log_2 n$.

Rendezés bináris keresőfával

- Építsünk keresőfát beszúrásokkal.
  Költsége: Legrosszabb esetben cn^2, átlagosan $c(n \log_2 n)$.
- INORDER bejárással kilistázzuk az elemeket
  Költsége: cn.

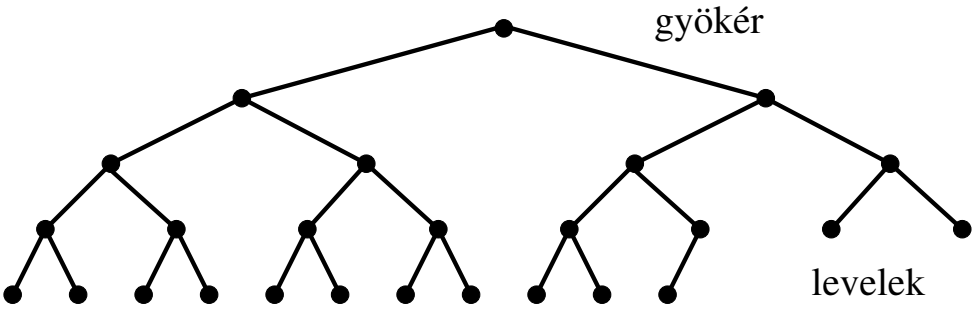
Költsége összesen: Legrosszabb esetben $cn^2 + cn \approx cn^2$, átlagosan $c(n \log_2 n) + cn \approx c(n \log_2 n)$.

Kupac (Heap) adatszerkezet

Egész számok egy S véges részhalmazát szeretnénk tárolni, hogy a *beszúrás* és a minimális elem törlése (*mintör*) hatékony legyen.
Alkalmazások:

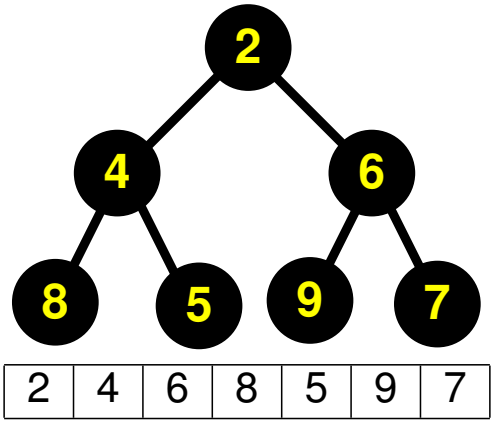
- Jobok indítása
- Több rendezett halmaz összefésülése
- Gyors rendezési algoritmus

Teljes bináris fa:



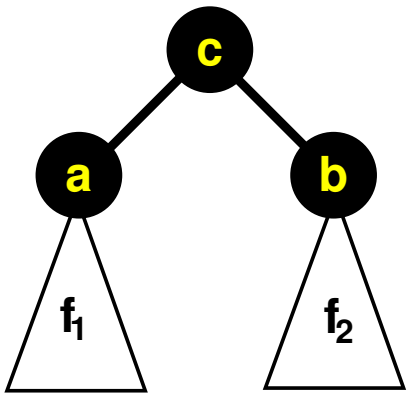
Bináris fa ábrázolása tömbbel

A fa csúcsai az $A[1:n]$ tömb elemei.
Az $A[i]$ csúcs bal fia $A[2i]$, a jobb fia pedig $A[2i+1]$.
$\Longrightarrow$ $A[j]$ csúcs apja $A[\lfloor j/2 \rfloor]$



Kupac tulajdonság: apa $<$ fia

Kupacépítés



f_1 és f_2 kupacok

kupacol(f)
{ Ha $\min\{a,b\} < c$, akkor $\min\{a,b\}$ és c helyet cserél
Ha a c elem a-val cserélt helyet, akkor *kupacol*(f_1), ha b-vel, akkor *kupacol*(f_2) }

c addig megy lefelé, amig sérti a kupac tulajdonságot.
Lépésszám: Ha l a fa szintjeinek száma, akkor $\leq l-1$ csere és $\leq 2(l-1)$ összehasonlítás

kupacépítés(f)
{ Az f fa v csúcsaira lentről felfelé, jobbról balra *kupacol*(v). }

Az előbbi sorrend a kupac tömb reprezentációjában épp a jobbról balra sorrendet jelenti.

Kupac mélysége

Bináris fában:

1. szint: 1 pont
2. szint: 2 pont
3. szint: 2^2 pont

$\vdots$

$l-1$-edik szint: 2^{l-2} pont

l-edik szint: > 1 és $\leq 2^{l-1}$ pont

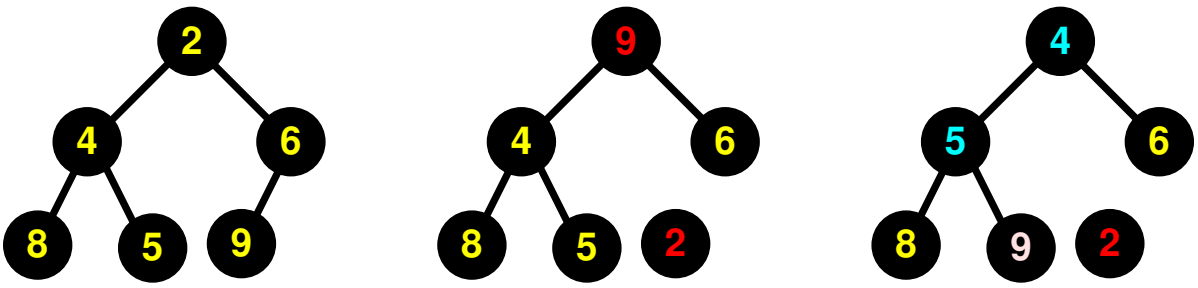
$\Longrightarrow n \geq 1 + \sum_{i=0}^{l-2} 2^i = 2^{l-1} \Longrightarrow l \leq 1 + \log_2 n$

Tétel

Kupacépítés költsége: cn

MINTÖR

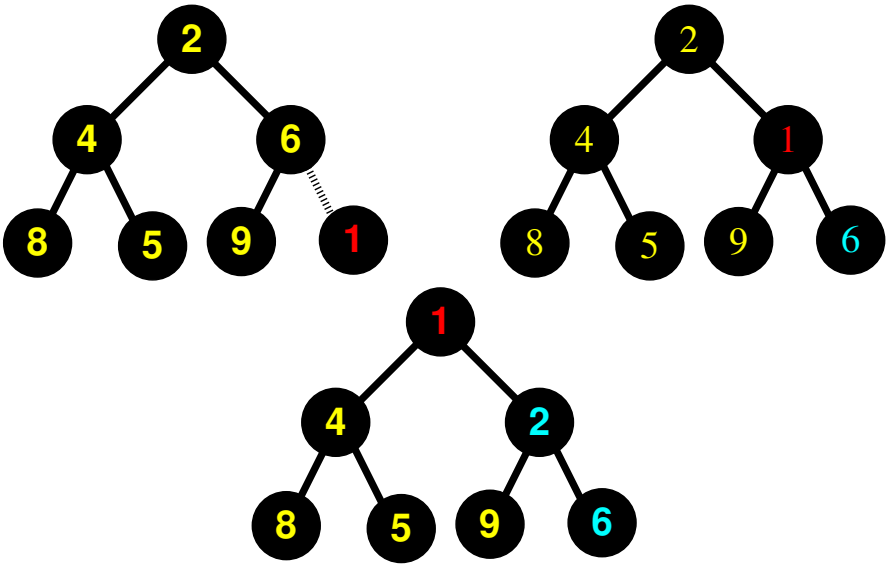
A minimális elem az f gyökerében van, ezt töröljük.
Helyére tesszük a fa utolsó szintjének jobb szélső elemét, majd *kupacol*(f).



Költség: $cl = c(\log_2 n)$

BESZÚR

Új levelet adunk a fához (ügyelve a teljességre), ide tesszük az s elemet. Ezután s-et mozgatjuk felfelé, mindig összehasonlítjuk az apjával.



Költség: $cl = c(\log_2 n)$

A kupacos rendezés (heap sort)

Először kupacot építünk, utána n darab MINTÖR adja nem csökkenő sorrendben az elemeket.
[J. W. J. Williams és R. W. Floyd, 1964]

Költség: $c_1 n + c_2(n \log_2 n) = c_3(n \log_2 n)$

Legjobb ismert rendező algoritmus.
Pontos implementációval:
$2n\lfloor \log_2 n \rfloor + 3n$ (összehasonlítások száma) és $n\lfloor \log_2 n \rfloor + 2,5n$ (cserék száma).