

Algoritmuselmélet

A kupac adatstruktúra, bináris keresőfák

Katona Gyula Y. / Vizer Máté

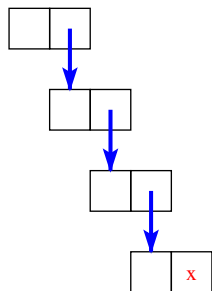
Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

Adatstruktúrák

Műveletek lépésszáma különböző adatstruktúrákban

	beszúr	keres	min	max	mintör	maxtör	töröl	módosít	épít
tömb	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(n)$
rendezett tömb	$O(n)$	$O(\log n)$	$O(1)$	$O(1)$	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n \log n)$
láncolt lista	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
kupac	$O(\log n)$	$O(n)$	$O(1)$	$O(n)$	$O(\log n)$	$O(n)$	$O(\log n)$	$O(\log n)$	$O(n)$
piros-fekete fa	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(n \log n)$

Bináris fa adatszerkezet

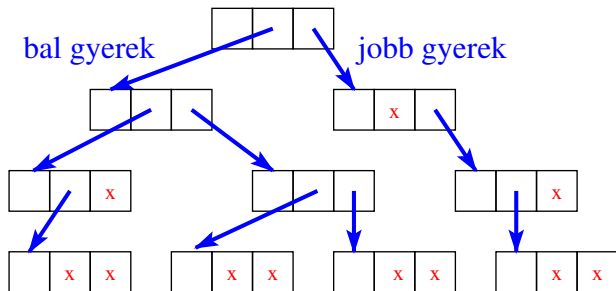


Láncolt lista

x: nincs további mutató

Alap műveletek: gyöker, bal-gyerek, jobb-gyerek

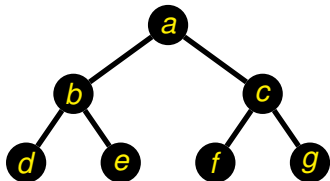
További műveletek: keres, beszúr, töröl



Bináris fa

Fa csúcsai \rightarrow $elem(x)$, $bal(x)$, $jobb(x)$ esetleg $szülő(x)$ és $részfa(x)$ (=az x gyökerű részfa csúcsainak száma)

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



$$bal(jobb(a)) = f,$$

$$szülő(szülő(d)) = a,$$

$$elem(bal(a)) = b,$$

$$részfa(a) = 7,$$

$$részfa(c) = 3.$$

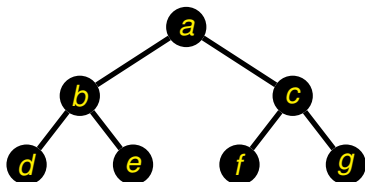
Bináris fa bejárásai

Tömbnél, láncolt listánál egyszerű a tárolt adatok felsorolása, a bináris fánál nem nyilvánvaló.

```
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
```



PREORDER: *abdecfg*

INORDER: *dbeafcg*

POSTORDER: *debfgca*

Bejárások animáció

Bináris fa bejárásainak lépésszáma

Tétel

A Bináris fa bejárásainak lépésszáma $O(n)$.

Bizonyítás.

Minden csúcsra 3 lépést végzek (valamilyen sorrendben):

- 1 Kírom a csúcsát,
- 2 megnézem mi a bal gyereke,
- 3 megnézem mi a jobb gyereke.

Amikor nincs gyereke, vagy csak 1 gyereke van, akkor is 1 lépés ennek megnézése.

Tehát a lépések száma $3n \in O(n)$.



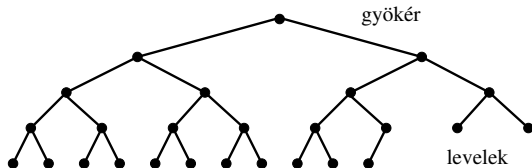
Kupac (Heap, Priority queue) adatszerkezet

Összehasonlítható elemek egy S véges részhalmazát szeretnénk tárolni, hogy a *beszúrás* és a *minimális elem törlése (mintör)* hatékony legyen. Feltesszük, hogy az elemek különbözőek.

Alkalmazások:

- Jobok indítása
- Gyors rendezési algoritmus
- Dijkstra gyorsabb megvalósítása

(majdnem) Teljes bináris fa:



A (majdnem) teljes fák levelei legfeljebb 2 szinten, a legalsón és esetleg a felette levőn helyezkednek el, és legfeljebb egy kivétellel minden belső csúcsuknak 2 fia van.

(A legalsó szint balra van „tömörítve”.)

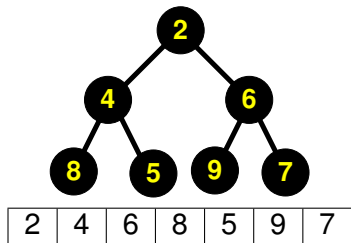
Bináris fa ábrázolása tömbbel, kupac definíciója

A fa csúcsai az $A[1 : n]$ tömb elemei.

(Itt most fontos, hogy az indexelés 1-től kezdődik.)

Az $A[i]$ csúcs bal fia $A[2i]$, a jobb fia pedig $A[2i + 1]$.

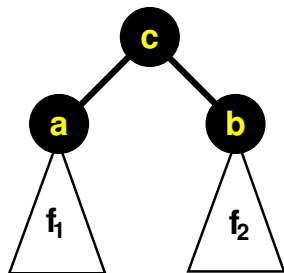
$\Rightarrow A[j]$ csúcs apja $A[\lfloor j/2 \rfloor]$



Definíció (Kupac)

A *(majdnem)* teljes bináris fában tárolt elemek kupacot alkotnak, ha minden csúcsra teljesül a **kupac tulajdonság**:
a szülőben tárolt elem < gyerekekben tárolt elemek.

Kupacépítés



f_1 és f_2 teljesíti a kupac tulajdonságot

kupacépítés(f)

{ Az f fa v csúcsaira letről felfelé, jobbról balra $kupacol(v)$. }

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

kupacol(f)

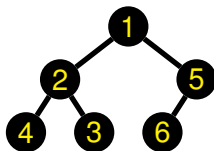
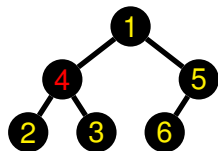
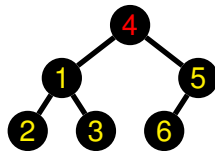
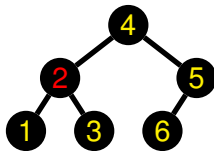
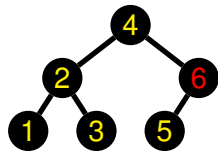
{ Ha $\min\{a, b\} < c$, akkor $\min\{a, b\}$ és c helyet cserél

Ha a c elem a -val cserélt helyet, akkor $kupacol(f_1)$, ha b -vel, akkor $kupacol(f_2)$ }

c addig megy lefelé, amíg sérti a kupac tulajdonságot.

Lépésszám: Ha ℓ a fa szintjeinek száma, akkor $\leq \ell - 1$ csere és $\leq 2(\ell - 1)$ összehasonlítás

Kupacépítés példa



Kupacépítés animáció

Kupac mélysége

(majdnem) teljes bináris fában:

1. szint: 1 pont

2. szint: 2 pont

3. szint: 2^2 pont

⋮

$\ell - 1$ -edik szint: $2^{\ell-2}$ pont

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

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

Tétel

Kupac mélysége: $O(\log n)$

Kupacépítés költsége

Lemma

$$\sum_{j=1}^{\ell-1} \frac{j}{2^j} < 2$$

$$\begin{array}{ccccccc} \frac{1}{2} & & & & & & \\ \frac{1}{4} & & \frac{1}{4} & & & & \\ \frac{1}{8} & & \frac{1}{8} & & \frac{1}{8} & & \\ \vdots & & & & & & \\ \frac{1}{2^{\ell-1}} & & \frac{1}{2^{\ell-1}} & & \frac{1}{2^{\ell-1}} & \cdots & \frac{1}{2^{\ell-1}} \\ \hline < 1 & < \frac{1}{2} & < \frac{1}{4} & \cdots & \leq \frac{1}{2^{\ell-1}} & \Leftarrow < 2 \end{array}$$

A kupacépítés költsége

Az i -edik szinten levő v csúcsra $kupacol(v)$ költsége legfeljebb $\ell - i$ cseré és legfeljebb $2(\ell - i)$ összehasonlítás.

A cserék száma ezért összesen legfeljebb

$$\ell - 1 + 2^1(\ell - 2) + 2^2(\ell - 3) + \dots + 2^{\ell-2} \cdot 1 = \sum_{i=1}^{\ell-1} (\ell - i)2^{i-1}.$$

$\ell - i = j$ (azaz $i = \ell - j$) helyettesítéssel

$$\sum_{j=1}^{\ell-1} j2^{\ell-j-1} = \sum_{j=1}^{\ell-1} j \frac{2^{\ell-1}}{2^j} = 2^{\ell-1} \sum_{j=1}^{\ell-1} \frac{j}{2^j} < 2^\ell \leq 2n.$$

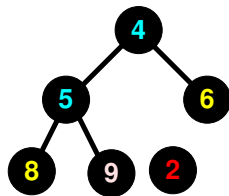
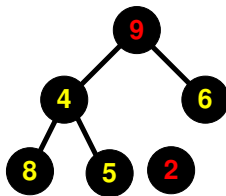
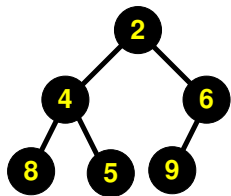
Összes művelet: $\leq 6n$

Tétel

A kupacépítés költsége: $O(n)$

MINTÖR

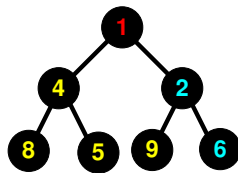
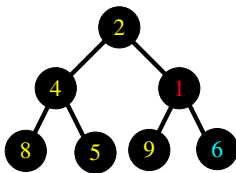
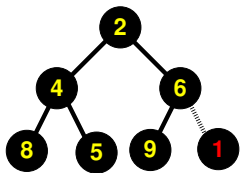
A minimális elem az f gyökerében van, ezt töröljük.
Helyére tesszük a fa utolsó szintjének jobb szélső elemét, majd
 $kupacol(f)$.



Költség: $O(\ell) = O(\log_2 n)$

BESZÚR

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



Egy cserénél a szülőt kisebbre cseréljük, így a másik gyermeknél továbbra is kisebb marad. \implies

A kupac tulajdonság csak felfelé sérülhet.

Költség: $O(\ell) = O(\log_2 n)$

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: $O(n) + O(n \log_2 n) = O(n \log_2 n)$

Megjegyzés: Legjobb ismert rendező algoritmus.

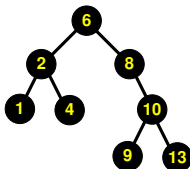
Pontos implementációval:

$2n \lfloor \log_2 n \rfloor + 3n$ (összehasonlítások száma) és $n \lfloor \log_2 n \rfloor + 2,5n$ (cserék száma).

Tároljuk az összehasonlítható elemeket tartalmazó U halmaz egy részhalmazát, hogy **BESZÚR**, **TÖRÖL**, **KERES**, **MIN**, **MAX** hatékonyak legyenek.

Definíció (Keresőfa-tulajdonság)

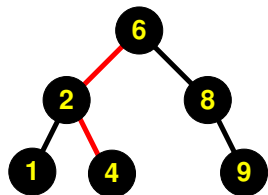
Tetszőleges x csúcsra és az x baloldali részfájában levő bármely y csúcsra igaz, hogy $elem(y) \leq elem(x)$. Hasonlóan, ha z egy csúcs az x jobb részfájából, akkor $elem(x) \leq elem(z)$.



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.

Naiv algoritmusok



KERES(4, S)

KERES(s,S): Összehasonlítjuk s -et S gyökerében tárolt s' elemmel.

- 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: $O(\ell)$, ahol ℓ a fa mélysége

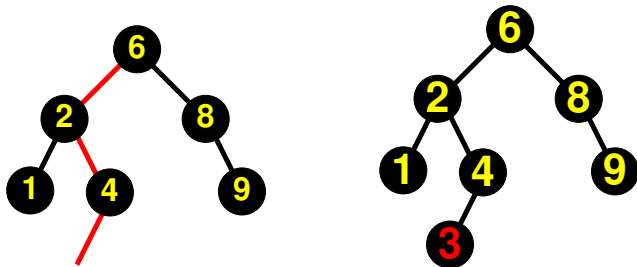
MIN: mindig balra lépünk, amíg lehet

MAX: mindig jobbra lépünk, amíg lehet

Lépésszám: $O(\ell)$

Naiv BESZÚR

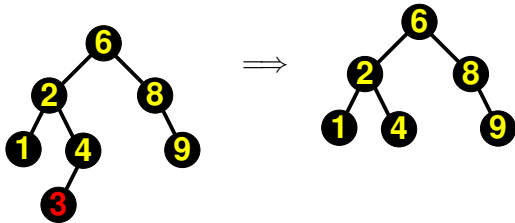
BESZÚR(s, S): KERES(s, S)-sel megkeressük, hova kerülne, és új levelet adunk hozzá, pl. **BESZÚR**(3, S):



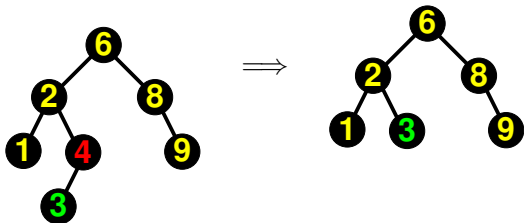
Lépésszám: $O(\ell)$

Naiv TÖRÖL

- $TÖRÖL(s, S)$: Ha s levél, akkor triviális, pl. $TÖRÖL(3, S)$:

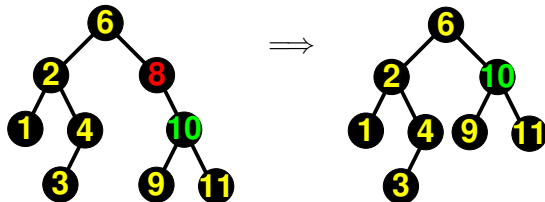


- $TÖRÖL(s, S)$: Ha s -nek egy fia van, akkor: $s \leftarrow \text{gyerek}(s)$, pl. $TÖRÖL(4, S)$:



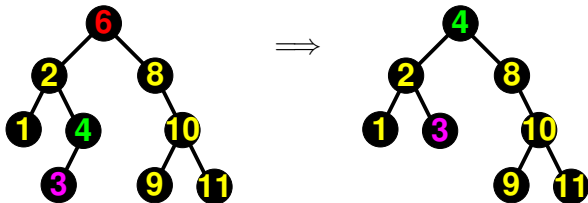
Naiv TÖRÖL

- Vagy pl. TÖRÖL(8, S')



- TÖRÖL(s, S): Ha s-nek két fia van, akkor visszavezetjük az előző esetre. s helyére tegyük $y := \text{MAX}(\text{bal}(s))$ -t és töröljük y-t. Pl.

TÖRÖL(6, S')



Naiv TÖRÖL

Állítás

$y := \text{MAX}(\text{bal}(s))$ csúcsnak nem lehet két fia.

Bizonyítás.

Ha lenne két fia, akkor lenne egy y' jobb fia is. De ekkor $y' > y$. ⚡

Lépésszám:

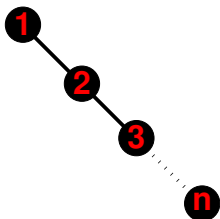
nincs gyerek (levél): keresés + egy mutató átállítása $\implies O(\ell)$

egy gyerek: keresés + két mutató átállítása $\implies O(\ell)$

két gyerek: keresés + max. keresés + csere + egy vagy két mutató átállítása $\implies O(\ell)$

Faépítés naiv beszúrásokkal

Ha pl. az $1, 2, \dots, n$ sorozatból építünk fát így, akkor ezt kapjuk:



Az építés költsége: $2 + 3 + \dots + (n - 1) = O(n^2)$

Tétel

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

Animáció: Bejárások, bináris keresőfa