Algoritmusok és adatszerkezetek II.

Kupacok

Szegedi Tudományegyetem

Fapacok (Treaps)

Emlékeztető

n kulcsból álló **véletlen** építésű bináris keresőfa h magasságának **várható értéke** $\log n$

- Adverzaliális műveleti sorrend mellett azonban n magas is lehet

Ötlet

A keresőfa,-és kupactulajdonságot egyidejűleg követeljük meg

1. Keresőfa tulajdonság biztosítja a kulcsok $O(h)$ kereshetőségét
2. Kupactulajdonság miatt h **várható értékben** $\log n$

Fapacok (Treaps)

Emlékeztető

n kulcsból álló **véletlen** építésű bináris keresőfa h magasságának **várható értéke** $\log n$

- Adverzaliális műveleti sorrend mellett azonban n magas is lehet

Ötlet

A keresőfa,-és kupactulajdonságot egyidejűleg követeljük meg

1. Keresőfa tulajdonság biztosítja a kulcsok $O(h)$ kereshetőségét
2. Kupactulajdonság miatt h **várható értékben** $\log n$
   - A kupactulajdonság ne az eltárolt kulcsokra, hanem egy **véletlenszerűen** generált kiegészítőinformációra teljesüljön!

Kupacok

Felhasználásuk

1. Prioritási sor megvalósításánál fontos, hogy a minimális/maximális kulcsot hatékonyan tudjuk visszaadni
2. Szintén fontos művelet egy adott kulcs értékének módosítása

Kupacok

Felhasználásuk

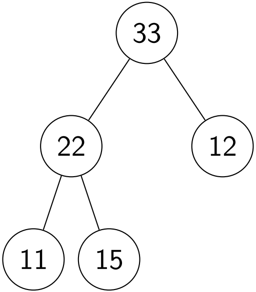
1. Prioritási sor megvalósításánál fontos, hogy a minimális/maximális kulcsot hatékonyan tudjuk visszaadni
2. Szintén fontos művelet egy adott kulcs értékének módosítása

Kupactulajdonság

Azt mondjuk, hogy egy fa rendelkezik a minimum (maximum) kupactulajdonsággal, ha minden p csúcsának minden q fiára

- $q = Nil$ vagy
- $p.kulcs < q.kulcs$ $(p.kulcs > q.kulcs)$

Példa maximum bináris kupacra



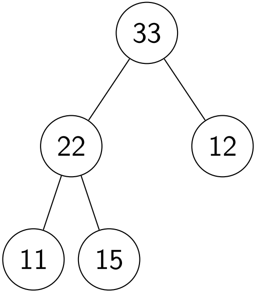
Beszúr(40)

0	1	2	3	4	5
–	33	22	12	11	15

Bináris kupac

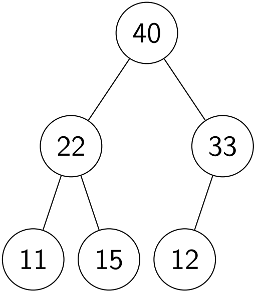
Teljes bináris fa, melyre teljesül a kupactulajdonság.
$\Rightarrow$ mivel legfeljebb egy belső pontnak lehet 2-nél kevesebb fia, így egyszerűen egy tömbbel implementálhatjuk

Példa maximum bináris kupacra



0	1	2	3	4	5
–	33	22	12	11	15

Beszúr(40)

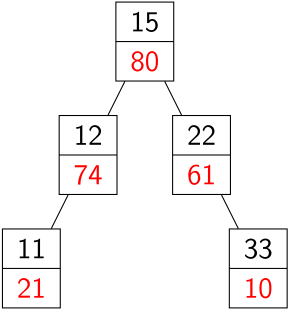


0	1	2	3	4	5	6
–	40	22	33	11	15	12

Bináris kupac

Teljes bináris fa, melyre teljesül a kupactulajdonság.
$\Rightarrow$ mivel legfeljebb egy belső pontnak lehet 2-nél kevesebb fia, így egyszerűen egy tömbbel implementálhatjuk

Fapac példa



- A kulcsok keresőfa tulajdonság szerint helyezkednek el
- A véletlen felépítést az extra adattag eredményezi
- A kiegyensúlyozott fáknál megszokott módon állítjuk helyre a megkövetelt tulajdonságokat (pl. (Beszúr(27, 100)))

Vissza a kupacokhoz

n elemű kupacban hogy keresnénk meg a maximális elemet?
És egy adott kulcs rákövetkezőjét?
Egy n és egy m kulcsból álló kupacot miként egyesítenénk?

Vissza a kupacokhoz

n elemű kupacban hogy keresnénk meg a maximális elemet?	O(1)
És egy adott kulcs rákövetkezőjét?	O(m+n)
Egy n és egy m kulcsból álló kupacot miként egyesítenénk?	O(m+n)

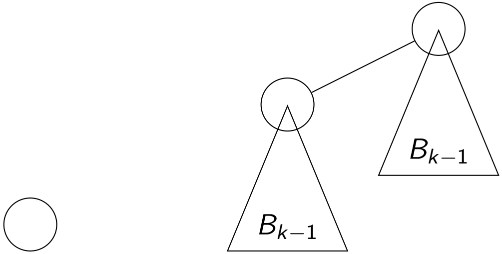
Kérdés

Lehetne hatékonyabban is?

Binomiális fa

Definíció

B_k binomiális fa egy rekurzív rendezett fa, amely két összekapcsolt B_{k-1} binomiális fából áll; az egyik fa gyökércsúcsa a másik fa gyökércsúcsának legbaloldalibb gyereke



(a) B_0 (b) $B_k (k \geq 1)$

Binomiális fákkal kapcsolatos állítások

Lemma

Ha B_k binomiális fa, akkor az alábbi állítások teljesülnek:

1. 2^k *csúcsa van*
2. i*-edik mélységében pontosan* $\binom{k}{i}$ *csúcs van* $(i = 0, 1, \ldots, k)$
3. *a gyökércsúcs fokszáma* k*, melynek gyerekeit balról jobbra megszámozva* $k-1, k-2, \ldots, 0$*-al,* i*-dik gyereke egy* B_i *részfa gyökércsúcsa.*

Binomiális fákkal kapcsolatos állítások

Lemma

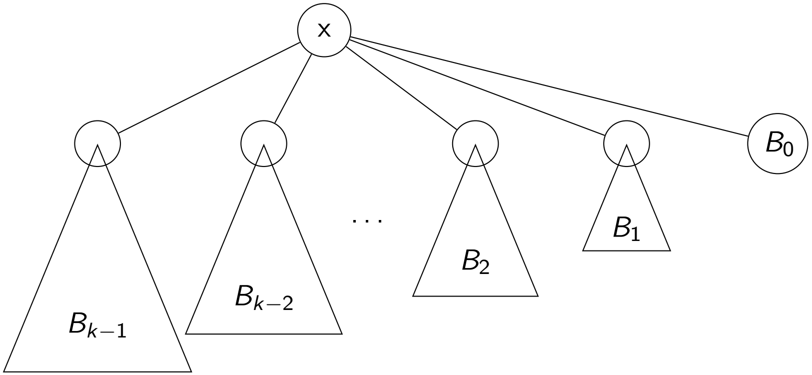
Ha B_k binomiális fa, akkor az alábbi állítások teljesülnek:

1. 2^k *csúcsa van*
2. i*-edik mélységében pontosan* $\binom{k}{i}$ *csúcs van* $(i = 0, 1, \ldots, k)$
3. *a gyökércsúcs fokszáma* k*, melynek gyerekeit balról jobbra megszámozva* $k-1, k-2, \ldots, 0$*-al,* i*-dik gyereke egy* B_i *részfa gyökércsúcsa.*

Következmény

n csúcsú binomiális fa minden csúcsának fokszáma legfeljebb $\log n$

Binomiális fák struktúrája



Binomiális kupac

Definíció

Egy H binomiális kupac binomiális fák olyan halmaza, amely

1. H minden binomiális fájára rendelkezik a minimumkupac (vagy maximumkupac) tulajdonsággal
2. H-ban nincsenek azonos fokszámmal rendelkező binomiális fák.

Binomiális kupac

Definíció

Egy H binomiális kupac binomiális fák olyan halmaza, amely

1. H minden binomiális fájára rendelkezik a minimumkupac (vagy maximumkupac) tulajdonsággal
2. H-ban nincsenek azonos fokszámmal rendelkező binomiális fák.

Következmény (előző lemma+2. tulajdonság)

n csúcsú binomiális kupac legfeljebb $\lfloor \log n \rfloor + 1$ binomiális fából áll

Binomiális kupac

Definíció

Egy H binomiális kupac binomiális fák olyan halmaza, amely

1. H minden binomiális fájára rendelkezik a minimumkupac (vagy maximumkupac) tulajdonsággal
2. H-ban nincsenek azonos fokszámmal rendelkező binomiális fák.

Következmény (előző lemma+2. tulajdonság)

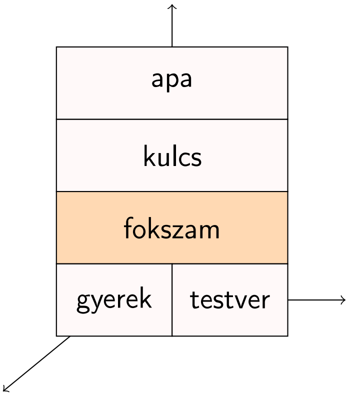
n csúcsú binomiális kupac legfeljebb $\lfloor \log n \rfloor + 1$ binomiális fából áll

Az előző következmény másképp

A gyökércsúcsok fokszámainak halmaza a $\subseteq \{0, 1, \ldots, \lfloor \log n \rfloor\}$

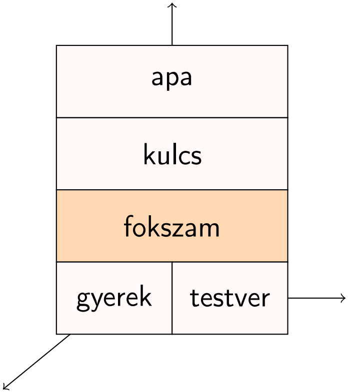
Binomiális kupacok implementációja

```
class Node {
    Object kulcs;
    Node *apa;
    int fokszam;
    Node *gyerek;
    Node *testver;
}
```



Binomiális kupacok implementációja

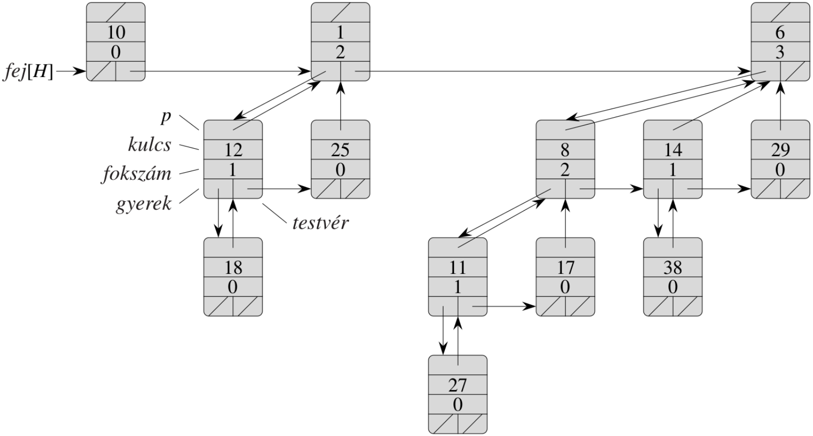
```
class Node {
    Object kulcs;
    Node *apa;
    int fokszam;
    Node *gyerek;
    Node *testver;
}
```



Megjegyzés

Balgyerek, jobbtestvér ábrázolást használunk

Binomiális kupacok szerveződése



Forrás: CLRS: Új algoritmusok 19.3 ábrája

Minimális kulcs keresése

```
BinomiálisKupacbanMin(H) {
  y = Nil
  x = H.fej //gyökérlista kezdőeleme
  min = Inf

  while (x != Nil) {
     if (x.kulcs < min) {
        min = x.kulcs
        y = x
     }
     x = x.testver
  }
 return y
}
```

Minimális kulcs keresése

```
BinomiálisKupacbanMin(H) {
  y = Nil
  x = H.fej //gyökérlista kezdőeleme
  min = Inf

  while (x != Nil) {
     if (x.kulcs < min) {
        min = x.kulcs
        y = x
     }
     x = x.testver
  }
 return y
}
```

n kulcs esetén $O(\log n)$

H_1 és H_2 kupacok egyesítése H kupaccá

- Bináris kupacok esetében nem volt hatékonyan megoldható, binomiális kupacra azonban tudunk adni $O(\log n)$ algoritmust
- Alapötlet
  - Fokszám szerint nemcsökkenő sorrendben fűzzük össze a H_1-ben és H_2-ben található binomiális fákat
    - Legfeljebb $\log n_1 + 1 + \log n_2 + 1 \leq 2 \log n + 1 = O(\log n)$ hosszú listát kapunk
  - 2 darab f fokszámú binomiális fából egy darab $f+1$ fokszámú binomiális fa hozható létre (ennek ideje $O(1)$)
  - Számoljuk föl az összefűzött gyökérlistában a megegyező fokszámú binomiális fákat

H_1 és H_2 kupacok egyesítése H kupaccá

- Bináris kupacok esetében nem volt hatékonyan megoldható, binomiális kupacra azonban tudunk adni $O(\log n)$ algoritmust
- Alapötlet
  - Fokszám szerint nemcsökkenő sorrendben fűzzük össze a H_1-ben és H_2-ben található binomiális fákat
    - Legfeljebb $\log n_1 + 1 + \log n_2 + 1 \leq 2 \log n + 1 = O(\log n)$ hosszú listát kapunk
  - 2 darab f fokszámú binomiális fából egy darab $f+1$ fokszámú binomiális fa hozható létre (ennek ideje $O(1)$)
  - Számoljuk föl az összefűzött gyökérlistában a megegyező fokszámú binomiális fákat
- Tulajdonképp a beszúrás is egy egyesítés

H_1 és H_2 kupacok egyesítése H kupaccá

- Bináris kupacok esetében nem volt hatékonyan megoldható, binomiális kupacra azonban tudunk adni $O(\log n)$ algoritmust
- Alapötlet
  - Fokszám szerint nemcsökkenő sorrendben fűzzük össze a H_1-ben és H_2-ben található binomiális fákat
    - Legfeljebb $\log n_1 + 1 + \log n_2 + 1 \leq 2 \log n + 1 = O(\log n)$ hosszú listát kapunk
  - 2 darab f fokszámú binomiális fából egy darab $f + 1$ fokszámú binomiális fa hozható létre (ennek ideje $O(1)$)
  - Számoljuk föl az összefűzött gyökérlistában a megegyező fokszámú binomiális fákat
- Tulajdonképp a beszúrás is egy egyesítés

Példa

$B_0 - B_2 - B_3 + B_0 - B_1 \Rightarrow B_0 - B_0 - B_1 - B_2 - B_3 \Rightarrow$
$B_1 - B_1 - B_2 - B_3 \Rightarrow B_2 - B_2 - B_3 \Rightarrow B_4$

Minimális kulcsú csúcs kivágása