Algoritmusok és adatszerkezetek II.

Augmentált keresőfák

Szegedi Tudományegyetem

Bináris keresőfák műveletei

- h magas és n csúcsból álló bináris keresőfa esetén a Keres, Beszúr, Töröl műveletek $O(h)$ időben végrehajthatók
- x csúcs rangja: a fában tárolt kulcsok között x hanyadik helyen áll $<$ szerint
- Hogy keresnénk meg egy bináris keresőfa r rangú elemét?
- Hogyan határoznánk meg egy bináris keresőfa x csúcsának rangját?

Új műveletek – adott rangú kulcs meghatározása

- Cél: $r \leq n$ ranggal rendelkező elem megtalálása a fában
- Naiv elgondolás: elkezdem bejárni a fát $<$ szerinti sorrendben, és megállok az r-ediknek érintett csúcsnál

Új műveletek – adott rangú kulcs meghatározása

- Cél: $r \leq n$ ranggal rendelkező elem megtalálása a fában
- Naiv elgondolás: elkezdem bejárni a fát $<$ szerinti sorrendben, és megállok az r-ediknek érintett csúcsnál
- $O(h)$ helyett $\Theta(r)$ idejű algoritmus
  - Kiegyensúlyozott fa és kellően nagy r estében pedig $r \gg h$

Új műveletek – adott kulcs rangjának meghatározása

- Naiv elgondolás: elkezdem bejárni a fát $<$ szerinti sorrendben, és megállok, ha x kulcsot érintem
- A válasz a bejárás során érintett kulcsok száma lesz
- $O(h)$ helyett $O(n)$ idejű algoritmus

Új műveletek – adott kulcs rangjának meghatározása

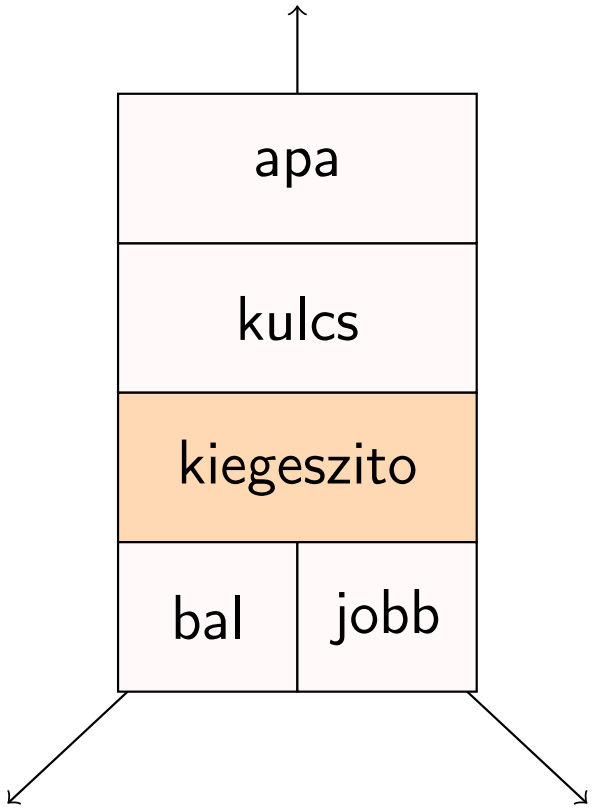
- Naiv elgondolás: elkezdem bejárni a fát $<$ szerinti sorrendben, és megállok, ha x kulcsot érintem
- A válasz a bejárás során érintett kulcsok száma lesz
- $O(h)$ helyett $O(n)$ idejű algoritmus

A megoldás

Minden csúcs tároljon el kiegészítő információt magáról!

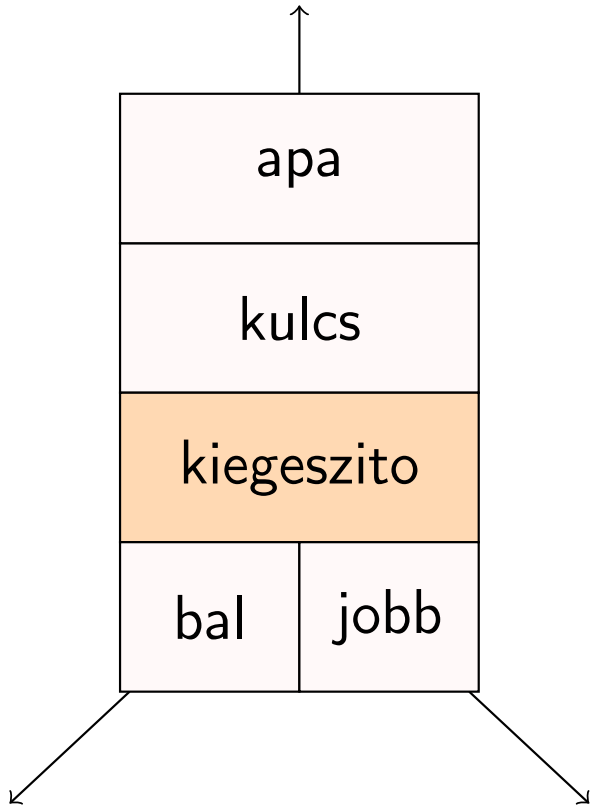
Rendezett-minta fa implementációja

```
class Node {
    Object kulcs;
    int kiegeszito;
    Node* apa;
    Node* bal;
    Node* jobb;
}
```



Rendezett-minta fa implementációja

```
class Node {
    Object kulcs;
    int kiegeszito;
    Node* apa;
    Node* bal;
    Node* jobb;
}
```



Megjegyzés

Az extra adattag az új műveletek hatékony (azaz $O(h)$ idejű) implementációját segítik

Fabejárások

- Inorder/preorder/posztorder bejárások
- Legegyszerűbb megvalósításuk rekurzióval történik
  - Jó azonban tudni, hogy rekurzió nélkül is megtehető mindez

```
void inorder(x){
  if(x==nil){return}
  inorder(x.bal)
  print(x.kulcs)
  inorder(x.jobb)
}
```

Fabejárások

- Inorder/preorder/posztorder bejárások
- Legegyszerűbb megvalósításuk rekurzióval történik
  - Jó azonban tudni, hogy rekurzió nélkül is megtehető mindez

```
void inorder(x){
  if(x==nil){return}
  inorder(x.bal)
  print(x.kulcs)
  inorder(x.jobb)
}
```

```
void preorder(x){
  if(x==nil){return}
  print(x.kulcs)
  preorder(x.bal)
  preorder(x.jobb)
}
```

Fabejárások

- Inorder/preorder/posztorder bejárások
- Legegyszerűbb megvalósításuk rekurzióval történik
  - Jó azonban tudni, hogy rekurzió nélkül is megtehető mindez

```
void inorder(x){
  if(x==nil){return}
  inorder(x.bal)
  print(x.kulcs)
  inorder(x.jobb)
}
```

```
void preorder(x){
  if(x==nil){return}
  print(x.kulcs)
  preorder(x.bal)
  preorder(x.jobb)
}
```

```
void postorder(x){
  if(x==nil){return}
  postorder(x.bal)
  postorder(x.jobb)
  print(x.kulcs)
}
```

Teljes bináris fa

- Olyan bináris fa, amelynek minden belső csúcsának 2 fia van

Fában lévő kulcsok száma

A fa i-edik szintjén 2^i csúcs található

Teljes bináris fa

- Olyan bináris fa, amelynek minden belső csúcsának 2 fia van

Fában lévő kulcsok száma

A fa i-edik szintjén 2^i csúcs található

$\Rightarrow$ a fában $n = \sum_{i=0}^{h} 2^i = 2^{h+1}$ csúcsot található[a]

Állítás: h magas fában $O(2^h)$ csúcs található

Teljes bináris fa

- Olyan bináris fa, amelynek minden belső csúcsának 2 fia van

Fában lévő kulcsok száma

A fa i-edik szintjén 2^i csúcs található

$\Rightarrow$ a fában $n = \sum_{i=0}^{h} 2^i = 2^{h+1}$ csúcsot található[a]

Állítás: h magas fában $O(2^h)$ csúcs található

Megfordítva: n csúcsból álló fa magassága $\Omega(\log n)$

[a] bizonyítás teljes indukcióval

Bináris keresőfa

Keresőfa tulajdonság

A fa minden x csúcsára teljesül, hogy

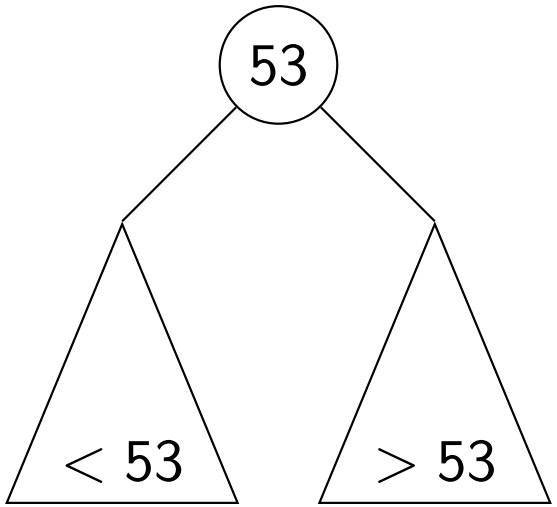
- $x.bal.kulcs < x.kulcs$ (amennyiben $x.bal != nil$)
- $x.kulcs < x.jobb.kulcs$ (amennyiben $x.jobb != nil$)

Keresőfa tulajdonság

A fa minden x csúcsára teljesül, hogy

- $x.bal.kulcs < x.kulcs$ (amennyiben $x.bal != nil$)
- $x.kulcs < x.jobb.kulcs$ (amennyiben $x.jobb != nil$)

$<$ rendezés tranzitivitásából adódóan



Kulcs keresése fában

```
FÁBANKERES(x, k) {
  if x == nil or k == x.kulcs
     return x

  if k < x.kulcs
      FÁBANKERES(x.bal, k)
  else
      FÁBANKERES(x.jobb, k)
}
```

Kulcs keresése fában

```
FÁBANKERES(x, k) {
  if x == nil or k == x.kulcs
     return x

  if k < x.kulcs
      FÁBANKERES(x.bal, k)
  else
      FÁBANKERES(x.jobb, k)
}
```

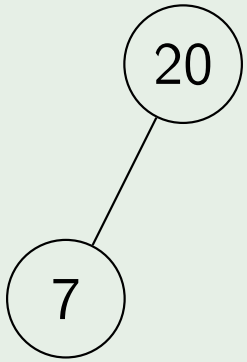
h magas fa esetén $O(h)$

Elem beszúrása bináris keresőfába

- A keresőfa tulajdonság fenntartása mellett levélként szúrunk be
- h magas fa esetén $O(h)$ idejű

Példa

Beszúr(7)



Elem törlése bináris keresőfából

Elem törlése bináris keresőfából

- 3 esetet különböztetünk meg x csúcs törlése kapcsán
  1. x-nek nincs gyereke
     - x apjának az x-re vonatkozó mutatóját *nil*-re állítjuk
  2. x-nek pontosan egy gyereke van
     - x apját "átkötjük" x egyedüli fiához
  3. x-nek 2 gyereke van
     - x-et megelőzőjével (bal oldali részfájának maximális elemével) helyettesítjük
- h magas fa esetén $O(h)$ idejű