Algoritmusok és adatszerkezetek II.

Bináris keresőfák és műveleteik

Szegedi Tudományegyetem

Gyakorlati követelmények

- 2 db ZH (ápr. 9. és máj. 7.): 40-40 pont (min. 18 pont/ZH)
- Kvízek: 4 darab 5-5 pontos Coospace kvíz (min. 0 pont)
- Pluszpontok (minimumba nem számítanak bele)
- Javítás: a nagy ZH-k bármelyike javítható az utolsó előadáson

Gyakorlati követelmények

- 2 db ZH (ápr. 9. és máj. 7.): 40-40 pont (min. 18 pont/ZH)
- Kvízek: 4 darab 5-5 pontos Coospace kvíz (min. 0 pont)
- Pluszpontok (minimumba nem számítanak bele)
- Javítás: a nagy ZH-k bármelyike javítható az utolsó előadáson

Gyakorlati jegy

[0–51) pont	elégtelen (1)
[51–66) pont	elégséges (2)
[66–76) pont	közepes (3)
[76–86) pont	jó (4)
[86–∞) pont	jeles (5)

Gyakorlati követelmények – Coospace kvízek

- Coospace-en az előadás szinterében lesznek közzétéve (de a gyakorlati teljesítés részét képezik)
- Közzététel: febr. 26., márc. 19, ápr. 16, ápr. 30
- Beadási határidő: ápr. 9, máj. 14 az előadás kezdetéig
- 3 kitöltés/kvíz, amelyek közül a legjobbat vesszük figyelembe

Gyakorlati követelmények – Coospace kvízek

- Coospace-en az előadás színterében lesznek közzétéve (de a gyakorlati teljesítés részét képezik)
- Közzététel: febr. 26., márc. 19, ápr. 16, ápr. 30
- Beadási határidő: ápr. 9, máj. 14 az előadás kezdetéig
- 3 kitöltés/kvíz, amelyek közül a legjobbat vesszük figyelembe
- Tesztenként a 10 leggyorsabb hibátlan kitöltőnek pluszpont jár

Követelmények – kollokvium

- A gyakorlat sikeres teljesítése esetén kollokvium tehető
- 10 db 5 pontos elméleti és gyakorlati kiskérdéssel
- Pluszpontok (minimumba nem számítanak bele)
- Elővizsga az utolsó előadáson (vizsgaalkalomnak számít)

Követelmények – kollokvium

- A gyakorlat sikeres teljesítése esetén kollokvium tehető
- 10 db 5 pontos elméleti és gyakorlati kiskérdéssel
- Pluszpontok (minimumba nem számítanak bele)
- Elővizsga az utolsó előadáson (vizsgaalkalomnak számít)

Kollokviumi jegy

[0–25) pont	elégtelen (1)
[26–32) pont	elégséges (2)
[32–38) pont	közepes (3)
[38–44) pont	jó (4)
[44–∞) pont	jeles (5)

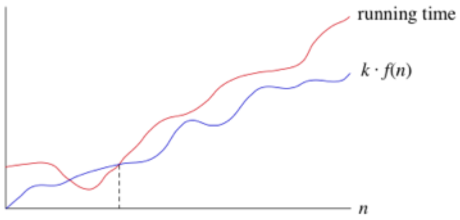
A félév során érintett főbb témakörök

- Kiegyensúlyozott és augmentált keresőfák
- Binomiális és Fibonacci kupacok
- Geometriai algoritmusok
- Számelméleti algoritmusok
- Mintaillesztő algoritmusok

Hasznos források

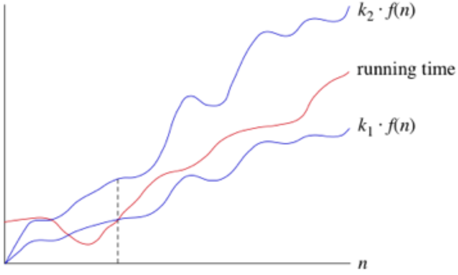
- Ajánlott irodalom
  - Thomas H. Cormen – Charles E. Leiserson – Ronald L. Rivest – Clifford Stein: **Új algoritmusok**. Kiadó: SCOLAR
- Hackerrank versenyek
- Algoritmusok vizualizációja

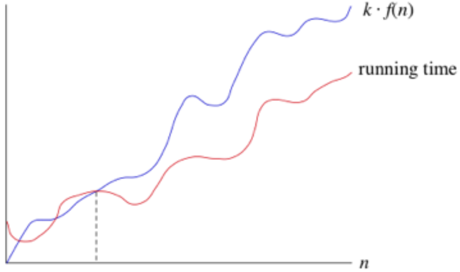
Ismétlés — Aszimptotikus jelölések: Ω



https://www.desmos.com/calculator/t8paytwq1w

Ismétlés — Aszimptotikus jelölések: Θ




$k \cdot f(n)$
running time
n

Kérdés

Hatékony-e az az algoritmus, amelyik futási ideje n méretű inputra *legalább* $O(n \log n)$?

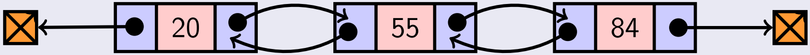
(beugratós) Kérdés

Hatékony-e az az algoritmus, amelyik futási ideje n méretű inputra *legalább* $O(n \log n)$?
Másképp szólva, mit tudunk arról az algoritmusról, amelynek futási ideje $\Omega(1)$?

Bináris fa implementációja

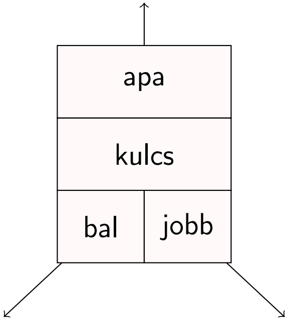
- Egy olyan ”megengedő” kétszeresen láncolt lista, ahol az elemek (egy helyett) két elemhez is kapcsolódhatnak

Láncolt lista



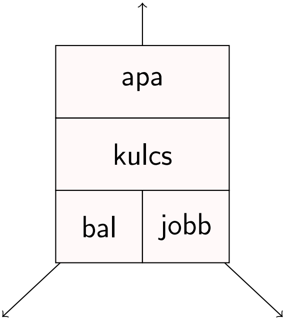
Bináris fa implementációja

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



Bináris fa implementációja

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



Megjegyzés

2 mutatóval és egy segédbittel (bal fiú-e az adott csúcs) is megvalósítható lenne

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 $x.jobb! = nil$)

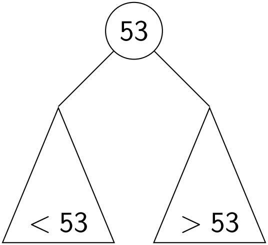
Bináris keresőfa

Keresőfa tulajdonság

A fa minden x csúcsára teljesül, hogy

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

$<$ 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)
}
```

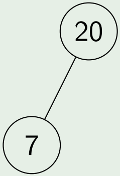
h magas fa esetén $O(h)$

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