Algoritmusok és adatszerkezetek II.

Bináris keresőfák és műveleteik

Szegedi Tudományegyetem

Gyakorlati követelmények

- 2 db ZH (márc. 9. és ápr. 27.): 30-30 pont (min. 10 pont/ZH)
- Kvízek: 4 darab 5-5 pontos Coospace kvíz (min. 2 pont/kvíz)
- Pluszpontok: ha a gyakorlati jegy legalább elégséges, akkor pluszpontottal max. 1 jegyet lehet javítani
- Javítás: a két ZH-ból az egyik (alapértelmezés szerint a rosszabbik) javítható

Gyakorlati követelmények

- 2 db ZH (márc. 9. és ápr. 27.): 30-30 pont (min. 10 pont/ZH)
- Kvízek: 4 darab 5-5 pontos Coospace kvíz (min. 2 pont/kvíz)
- Pluszpontok: ha a gyakorlati jegy legalább elégséges, akkor pluszpontottal max. 1 jegyet lehet javítani
- Javítás: a két ZH-ból az egyik (alapértelmezés szerint a rosszabbik) javítható

Amennyiben a részteljesítésenkénti minimumok megvannak, úgy a gyakorlati jegyek a következők szerint alakulnak

Gyakorlati jegy	
[0–40) pont	elégtelen (1)
[40–50) pont	elégséges (2)
[50–60) pont	közepes (3)
[60–70) pont	jó (4)
[70–∞) pont	jeles (5)

Gyakorlati követelmények – Coospace kvízek

- Coospace-en az **előadás** színterében lesznek közzétéve (de a gyakorlati teljesítés részét képezik)
- Az első két kvíz 2 kvíz márc. 8. 23:55-ig, a 3. és 4. kvízek ápr. 26. 23:55-ig tölthetők ki
- 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)
- Az első két kvíz 2 kvíz márc. 8. 23:55-ig, a 3. és 4. kvízek ápr. 26. 23:55-ig tölthetők ki
- 3 kitöltés/kvíz, amelyek közül a legjobbat vesszük figyelembe
- A teszteket a kiélesedésüket követő nap végéig hibátlatul kitöltők pluszpontot kapnak

Követelmények – kollokvium

- A gyakorlat sikeres teljesítése esetén kollokvium tehető
- 8 db 5 pontos elméleti és gyakorlati kiskérdéssel
- Pluszpontok (minimumba nem számítanak bele)
- Amennyiben a gyakorlati jegy >=4, elővizsga tehető az utolsó előadáson (vizsgaalkalomnak számít)

Követelmények – kollokvium

- A gyakorlat sikeres teljesítése esetén kollokvium tehető
- 8 db 5 pontos elméleti és gyakorlati kiskérdéssel
- Pluszpontok (minimumba nem számítanak bele)
- Amennyiben a gyakorlati jegy >=4, elővizsga tehető az utolsó előadáson (vizsgaalkalomnak számít)

Kollokviumi jegy	
[0–20) pont	elégtelen (1)
[20–25) pont	elégséges (2)
[25–30) pont	közepes (3)
[30–35) pont	jó (4)
[35–∞) pont	jeles (5)

A vizsga értékelése (Forrás: London András honlapja)

- Negatívumok
  - Nyelvileg értelmetlen mondatok.
  - Bőbeszédűség: Ha egy oldal átbogarászása után egy sornyi (vagy esetleg 0 bitnyi) információt találok annak nem fogok örülni.
  - Kritikus helyen olvashatatlanná váló leírás, kritikus helyen zavarossá váló leírás: Az írásos vizsga jegye a leírtakért jár; elhiszem, hogy "a hallgató jól tudja, de akkor úgy is kell leírni".

A vizsga értékelése (Forrás: London András honlapja)

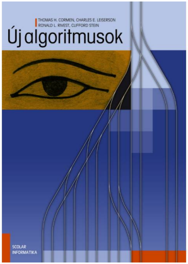
- Negatívumok
  - Nyelvileg értelmetlen mondatok.
  - Bőbeszédűség: Ha egy oldal átbogarászása után egy sornyi (vagy esetleg 0 bitnyi) információt találok annak nem fogok örülni.
  - Kritikus helyen olvashatatlanná váló leírás, kritikus helyen zavarossá váló leírás: Az írásos vizsga jegye a leírtakért jár; elhiszem, hogy "a hallgató jól tudja, de akkor úgy is kell leírni".

A félév során érintett főbb témakörök

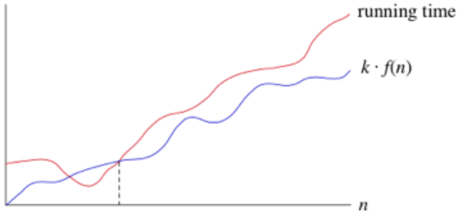
- Kiegyensúlyozott és augmentált keresőfák
- Binomiális és Fibonacci kupacok
- Geometriai algoritmusok
- Számelméleti algoritmusok
- Mintaillesztő algoritmusok

Hasznos források

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

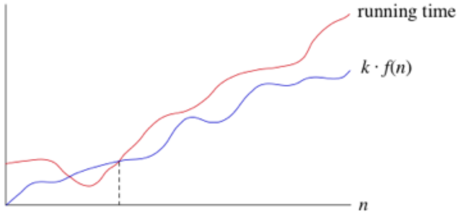


- Hackerrank versenyek
- Algoritmusok vizualizációja



Létezik egy olyan küszöbérték n_0, amihez meg tudunk adni egy $k > 0$ konstanst, hogy a futási idő minden $n > n_0$ esetén legalább $k \cdot f(n)$

Jelölés: $T(n) = \Omega(f(n))$ vagy másképp $T(n) \in \Omega(f(n))$

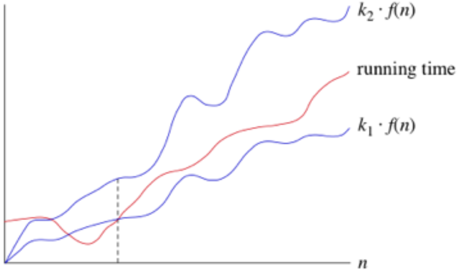


Létezik egy olyan küszöbérték n_0, amihez meg tudunk adni egy $k > 0$ konstanst, hogy a futási idő minden $n > n_0$ esetén legalább $k \cdot f(n)$

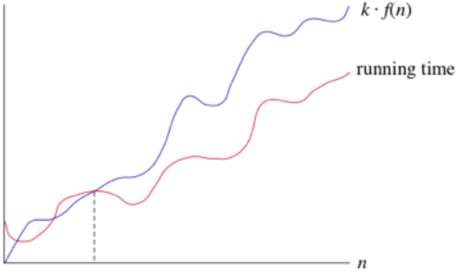
Jelölés: $T(n) = \Omega(f(n))$ vagy másképp $T(n) \in \Omega(f(n))$

Értelmezés: $T(n)$ "kellően nagy" inputra "érdemben" meghaladja $f(n)$-t

Ismétlés — Aszimptotikus jelölések: Θ



Ismétlés — Aszimptotikus jelölések: O



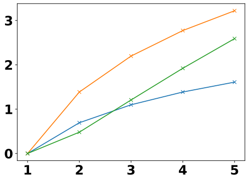
Kérdés

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

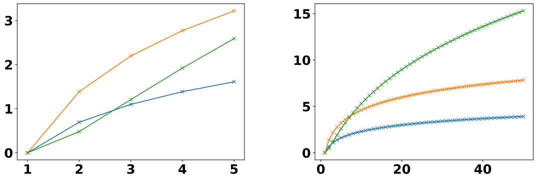
(beugratós) Kérdés

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 $\Omega(1)$?

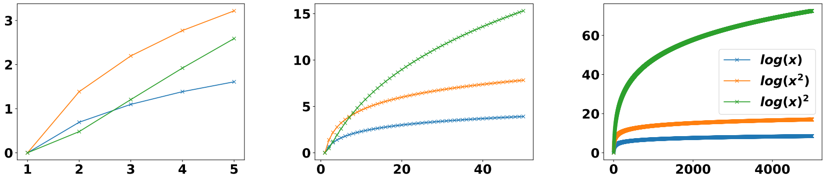
3 algoritmusunk van, amelyek az n méretű inputjuk függvényében $O(log(n^2))$, $O(log(n)^2)$ és $O(log(n))$ futási idővel rendelkeznek. Mi mondható el az algoritmusok hatékonyságáról?



3 algoritmusunk van, amelyek az n méretű inputjuk függvényében $O(log(n^2))$, $O(log(n)^2)$ és $O(log(n))$ futási idővel rendelkeznek. Mi mondható el az algoritmusok hatékonyságáról?
(Hint: $log(a * b) = log(a) + log(b)$)



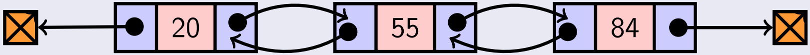
3 algoritmusunk van, amelyek az n méretű inputjuk függvényében $O(log(n^2))$, $O(log(n)^2)$ és $O(log(n))$ futási idővel rendelkeznek. Mi mondható el az algoritmusok hatékonyságáról? (Hint: $log(a * b) = log(a) + log(b)$)



Bináris fa implementációja

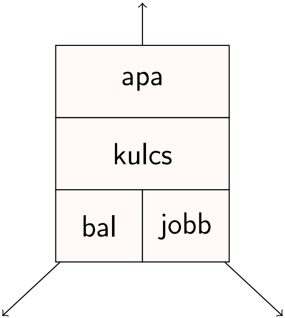
- Egy olyan "megengedő" kétszeresen láncolt lista, ahol az elemek (egy helyett) két elemhez is kapcsolódhatnak

Láncolt lista



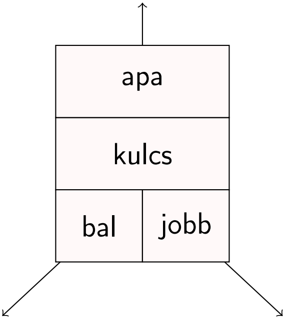
Bináris fa implementációja

```
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

- 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 $n = \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$ csúcs van[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 $n = \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$ csúcs van[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

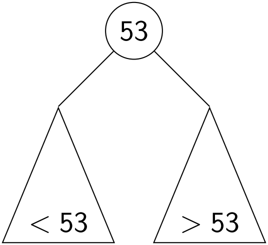
- $x.bal.kulcs < x.kulcs$ (amennyiben $x.bal != nil$)
- $x.kulcs < x.jobb.kulcs$ (amennyiben $x.jobb != nil$)

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

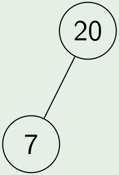
```
FÁBANKERES(x, k) {
  if x == nil or k == x.kulcs
     return x

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

h magas fa esetén $O(h)$

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

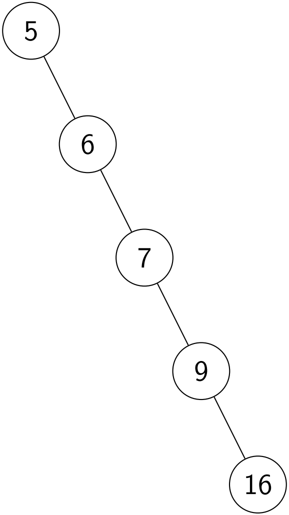
Példa

Beszúr(7)



Beszúrás – példa

Milyen fát eredményez az 5,6,7,9,16 kulcsok beszúrása?

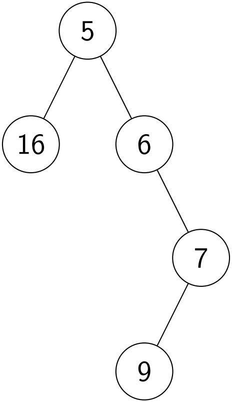


Beszúrás – példa

Tegyük föl, hogy $>$ szerint $7>9>6>5>16$. Most hogy néz ki a fa?

Beszúrás – példa

Tegyük föl, hogy > szerint 7>9>6>5>16. Most hogy néz ki a fa?



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 **megelőző**jével (bal oldali részfájának maximális elemével) helyettesítjük
- h magas fa esetén $O(h)$ idejű

Összefoglalás

- Az aktuális igényeinket hatékonyan kielégítő algoritmusok és adatszerkezetek megválasztása fontos
- Aszimptotikus jelöléssel a költségeink nagyságrendjének alakulása érzékeltethető (konstansokra való érzéketlenség)
- h magas bináris fában $n = O(2^h)$ csúcs

Összefoglalás

- Az aktuális igényeinket hatékonyan kielégítő algoritmusok és adatszerkezetek megválasztása fontos
- Aszimptotikus jelöléssel a költségeink nagyságrendjének alakulása érzékeltethető (konstansokra való érzéketlenség)
- h magas bináris fában $n = O(2^h)$ csúcs
  $\Rightarrow$ n csúcsú fa esetén $h = \Omega(\log n)$, továbbá $h = O(n)$
- A bináris keresőfák műveletei $O(h)$ idejűek