12. tétel

Adatszerkezetek és algoritmusok vizsga

Frissült: 2013. január 28.

Kupac (heap)

Majdnem teljes fák

Teljes bináris fa

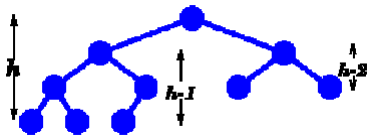
Egy bináris fa **teljes**, ha

- magassága h és
- $2^{h+1} - 1$ csomópontja van.

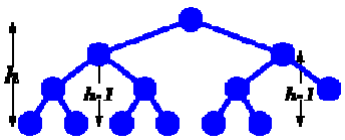
Majdnem teljes bináris fa

Egy h magasságú bináris fa akkor és csak akkor **majdnem teljes**, ha

- üres, vagy
- magassága h és
  - bal részfája $h-1$ magas és majdnem teljes
  - jobb részfája $h-2$ magas és teljes



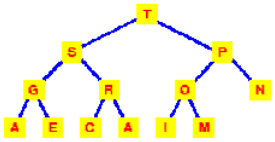
- magassága h és
  - bal részfája $h-1$ magas és teljes
  - jobb részfája $h-1$ magas és majdnem teljes



Definíció

Egy majdnem teljes fa *heap* tulajdonságú $\Longleftrightarrow$

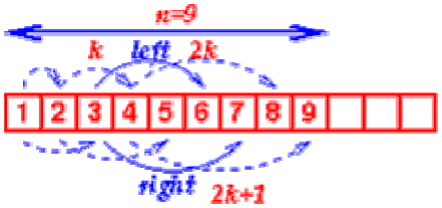
- üres, vagy
- vagy: a **gyökérben lévő kulcs nagyobb**, mint mindkét gyerekében, és mindkét részfája is *heap* tulajdonságú



Prioritásos sorok megvalósítására használjuk.

Reprezentáció

Dinamikusan allokált csomópontok és mutatók mint bármilyen más láncolt lista vagy fa.



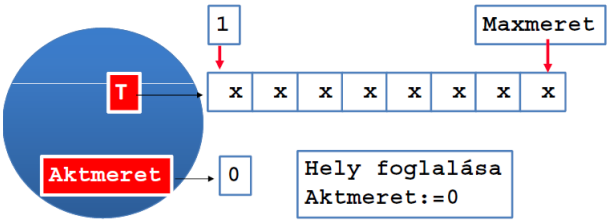
Használjunk egy tömböt és használjuk ki a "majdnem teljes" tulajdonságot.

- a k csomópont gyerekei a $2k$, $2k+1$-nél vannak
- a k szülője a $k/2$-nél van
- ha $k > n$, akkor a csomópont nem létezik

Műveletek

Létrehozás

Empty: $\rightarrow$ K - az üres kupac létrehozása.



Üres-e a kupac?

Isempty: $K \to L$

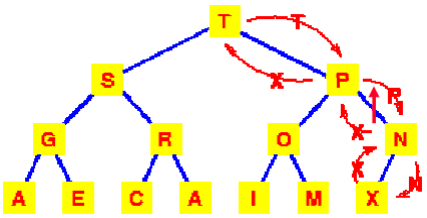
```
return (Aktmeret = 0)
```

Beszúrás

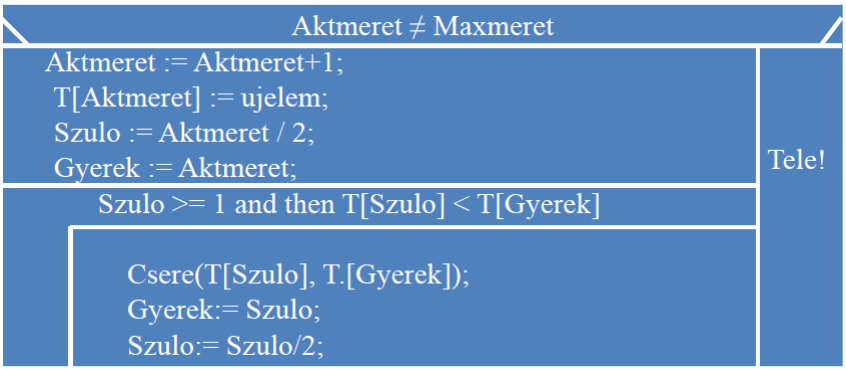
Insert: $K \times N \to K$ elem betétele a kupacba.

Menete: Adjunk egy elemet a kupachoz. Helyezzük a következő üres pozícióra a jobb szélén – ez kell legyen a következő kitöltendő hely.

Utána vigyük felfelé, amíg nagyobb, mint a szülei.

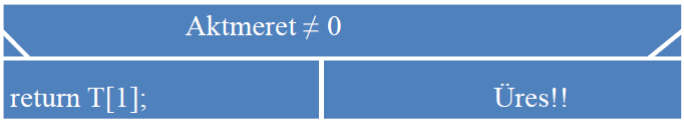


Struktogram



Maximuális elem lekérdezése

Max: $K \to E$

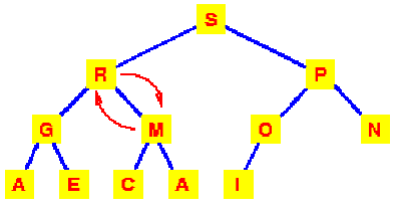


Maximális elem kivétele a kupacból

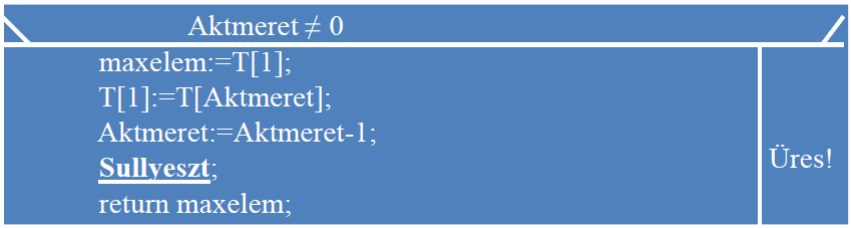
Delmax: $K \to K \times N$

Törlés menete. Törlés esetén mindig helyre kell állítani a heap tulajdonságot. A lépések:

- kiszedjük a gyökér elemet.
- a mostani utolsó elemet felvisszük a gyökérbe.
- Ekkor hibás lehet a heap tulajdonság. Ilyenkor a gyökér és gyerekeit vizsgáljuk, ha az egyik gyerek nagyobb mint a gyökér, csere a gyökérrel, és ezt rekurzívan, ameddig lehet és kell

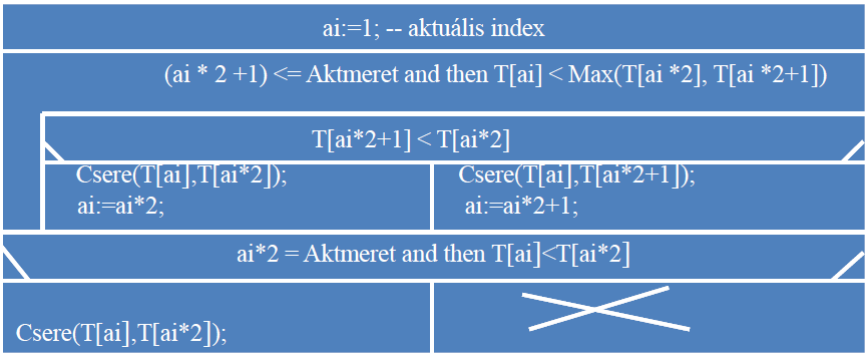


Struktogram.



A maximális elem értékét a *maxelem*-ben kapjuk.

Sullyeszt: feltételezzük, hogy a kupacban két jó részfa van, de a $T[1]$-t megfelelően le kell süllyeszteni

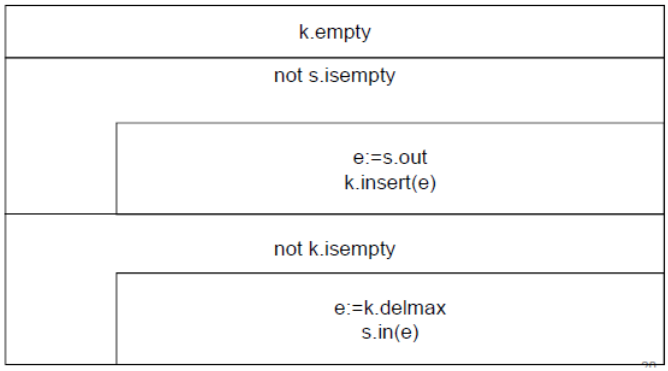


Kupac rendezés

Lépései:

- Szúrd be az elemeket egy kupacba!
- Amíg a sor ki nem ürül, vedd ki a kupacból a maximális elemet, és tedd az eredmény (rendezett) sorba!

Struktogramja:
Az s sorban lévő elemeket rendezzük a k kupac segítségével!



- Szúrjunk be minden elemet a kupacba.

  n elem, ezekhez kell $O(\log_2 n)$, össz. $O(n \log n)$
- sorrendben távolítsuk el az elemeket

  n elem, ezekhez kell $O(\log n)$, össz. $O(n \log n)$

Műveletigény összesen: $O(n \log n)$. Ez a sebesség garantált, nem ingadozik, mint a Quicksort-nál (az lehet gyorsabb is, de rosszabb esetben $O(n^2)$)! Emiatt használják valós idejű rendszereknél. Ott az idő szorít.