7. tétel

Adatszerkezetek és algoritmusok vizsga

Frissült: 2013. január 28.

Kupac (heap)

Majdnem teljes fák

Teljes bináris fa

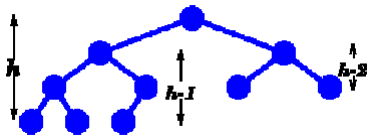
Egy bináris fa **teljes**, ha

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

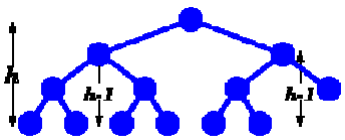
Majdnem teljes bináris fa

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



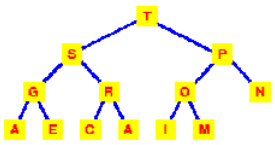
- magassága h és
  - bal részfája $h-1$ magas és teljes
  - jobb részfája $h-1$ magas és majdnem teljes



Definíció !!

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

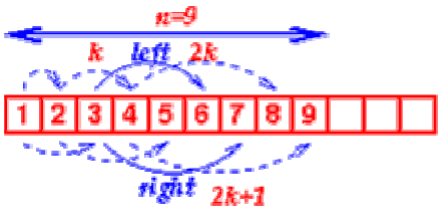
- üres, vagy
- a **gyökérben lévő kulcs nagyobb**, mint mindkét gyerekében, és mindkét részfája is *heap* tulajdonságú



Prioritásos sorok megvalósítására használjuk.

Reprezentáció

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



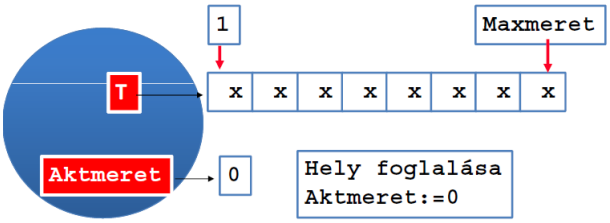
Használjunk egy tömböt és használjuk ki a "majdnem teljes" tulajdonságot.

- 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: $K \to L$

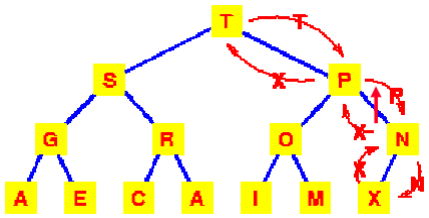
```
return (Aktmeret = 0)
```

Beszúrás !!

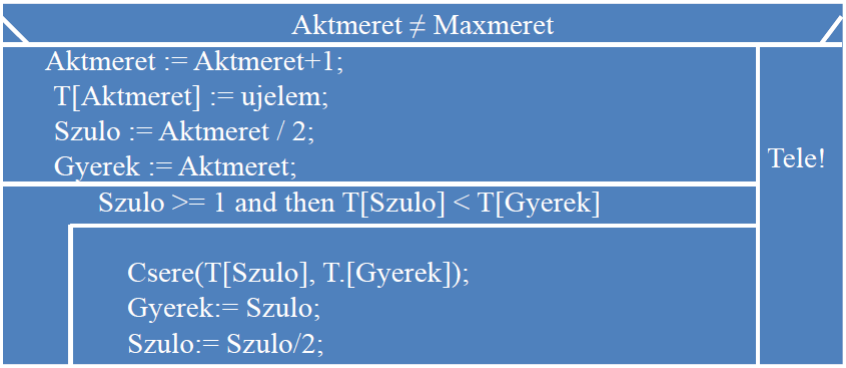
Insert: $K \times N \to K$ elem betétele a kupacba.

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.

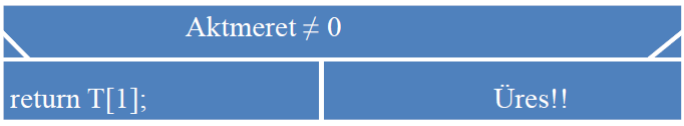


Struktogram: !!



Maximuális elem lekérdezése

Max: $K \to E$

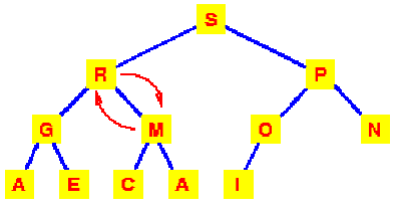


Maximális elem kivétele a kupacból

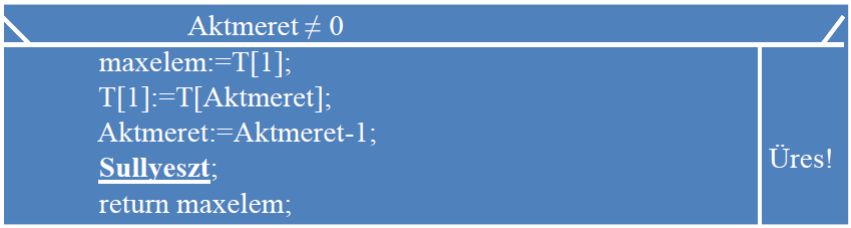
Delmax: $K \to K \times N$

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

- kiszedjük a gyökér elemet.
- a következő legnagyobb 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

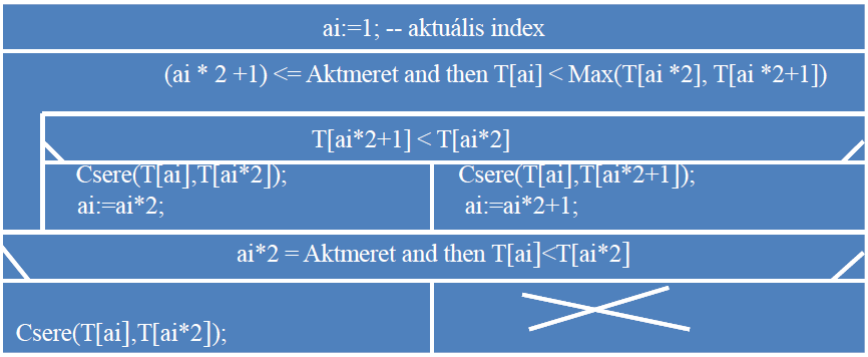


Struktogram.



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

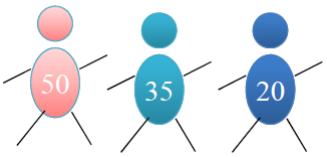
Sullyeszt: feltételezzük, hogy a kupacban két jó részfa van, de a $T[1]$-t megfelelően le kell süllyeszteni



Elsőbbségi / prioritásos sor

A sorban lévő elemeknek valamifajta rendezése is van. Ezek a prioritások. Úgy kell rendezzük az elemeket, hogy a legnagyobb (legkisebb) prioritású elemet töröljük először.

Fogalma. "sürgősségi osztály"



Vannak tennivalók, melyiket kell először elvégezni? Pl. operációs rendszer, mely prioritásos feladatokat (job) dolgoz fel. Itt különböző további algoritmusok, amelyek a prioritást meghatározzák egy-egy folyamat (job) számára. Vagy: telekommunikációban a csomagok továbbításánál is használják.

ADT Axiomatikus leírás !!

E alaptípus feletti P elsőbbségi sor típus jellemzése:
Egyszerűsítés: csak prioritásokat teszünk bele (N).
Műveletek:

- empty: $\rightarrow P$ (az üres prior. sor konst.- létrehozás)
- isempty: $P \rightarrow L$ (üres a prior.sor?)
- insert: $P \times N \rightarrow P$ (elem betétele a prioritásos sorba)
- delmax: $P \rightarrow P \times N$ (maximális elem kivétele a pr. sorból)
- max: $P \rightarrow N$ (maximális elem lekérdezése)

Megszorítások: *delMax* és max *értemezési* tartománya: $P \setminus \{\text{empty}\}$

Axiómák:

1. isempty(empty) vagy: p=empty→isempty(p)
2. isempty(p) →p = empty
3. ¬ isempty(insert(p,n))
4. insert(delmax(p))=p
5. max(p)=delmax(p)$_2$ (2: második komponens)
6. delmax(p)$_1$ ≠ empty → max(p) ≥ max(delmax(p)$_1$)
7. n ≥ max(p) → delmax(insert(p,n))$_1$=p ∧ max(insert(p,n))=n
8. n < max(p) → max(insert(p,n)) = max(p)
9. delmax(insert(empty,n)) = (empty,n)
10. max(insert(empty,n)) = n

(7. és 8.-nál feltettük, hogy nem üres a prioritásos sor)

ADS reprezentáció

1. rendezetlen tömbbel, a beérkezési idő szerint → max műveletigénye mindig egy *maxker*, vagyis $\theta(n)$
2. rendezett tömbbel → *insert* műveletigénye:
   - a hely megkeresése → logker $\theta(\log_2 n)$
   - tőle jobbra léptetés: $\theta(n)$
   - összesen: $\theta(n)$
3. heap (kupac) adatszerkezettel – fentebb

ADS műveletek megvalósítása

Fentebb, a kupac leírásánál.