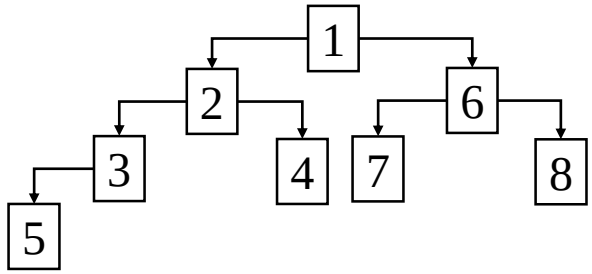
Algoritmusok és adatszerkezetek I.
3. előadás

Kupac

A kupac olyan ***véges*** elemsokaság, amely rendelkezik az alábbi tulajdonságokkal:

1. Minden elemnek ***legfeljebb két rákövetkezője*** (leszármazottja) lehet. Azaz ***bináris fá***nak tekinthető.
2. Minden ***eleme kisebb*** (vagy egyenlő) a közvetlen leszármazottainál.
3. A kupac ***balról folytonos***, azaz ha nem teljes a fa, akkor csak a legutolsó szintből hiányozhatnak elemek, de azok is csak a szint jobb széléről.

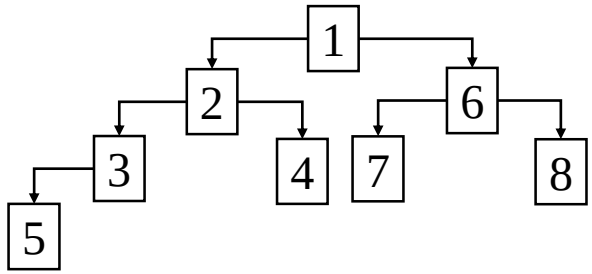


Kupac

A kupac ***ábrázolható folytonosan***, tömbben:

Az elhelyezésre a következő szabályok érvényesek:

- Az `i.` elem baloldali rákövetkezője a `2*i.` elem.
- Az `i.` elem jobboldali rákövetkezője a `2*i+1.` elem.
- Az `i.` elem előzője az `i div 2.` elem.



Kupac

Műveletei: Üres, üres?, első, Felcsúsztat, Lecsúsztat, Kupacba, Kupacból

Üres: Kupac

üres?(Kupac): Logikai

első(Kupac): Elem ∪{NemDef}

Kupacba(Kupac,Elem): Kupac ∪{NemDef}

Kupacból(Kupac,Elem): (Kupac ×Elem) ∪{NemDef}

Felcsúsztat(Kupac,Index): Kupac

Lecsúsztat(Kupac,Index): Kupac

Kupac

A műveletek specifikációja

Üres(K): ef: —, uf: K=()

üres?(K): ef: —, uf: üres?=K=()

Kupacba(K,e): ef: K=W és kupac(K), uf: K=(W←e) és kupac(K)

Kupacból(K,e): ef: K=(f,W) és kupac(K), uf: K=W és e=f és kupac(K)

első(K): ef: K=(f,W), uf: K=(f,W) és első=f

Kupac

Kupac ábrázolása:

```
Rekord(N: Egész;
       t: Tömb(1..MaxN:Elemtípus),
       hiba: Logikai)
```

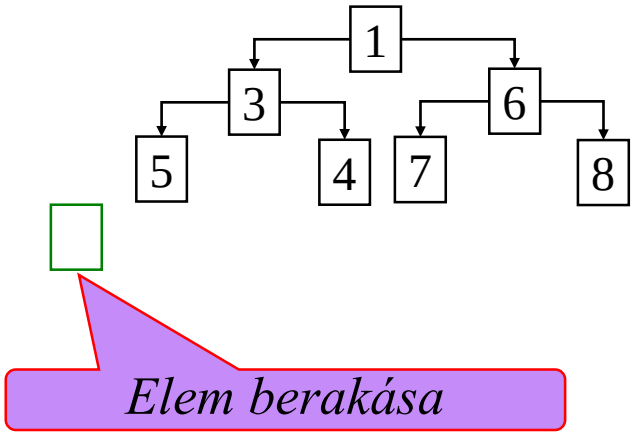
Kupac

Műveletek megvalósítása:

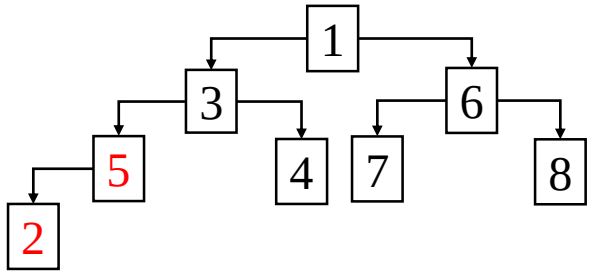
```
Üres(K):
  K.N:=0
Eljárás vége.
üres?(K):
  üres?:=(K.N=0)
Függvény vége.
első(K):
  első:=K.t(1)
Függvény vége.
```

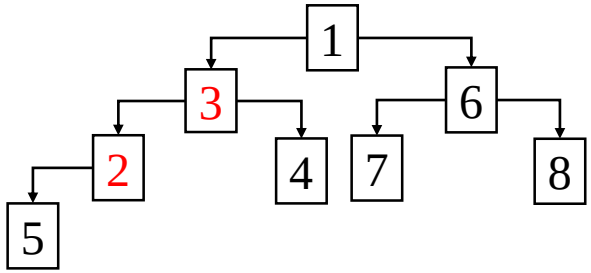
Kupac

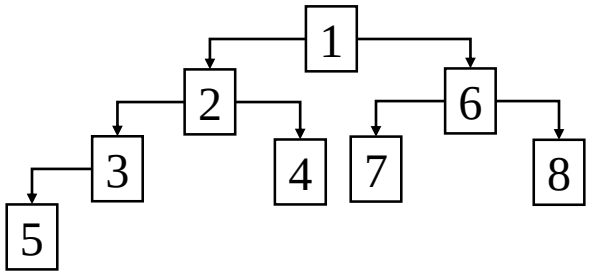
Elem berakása: Kupacba([1,3,6,5,4,7,8],2) ⇒ [1,2,6,3,4,7,8,5]



Felcsúsztatás:







Kupac

```
Kupacba(K,e):
  K.N:=K.N+1; K.t(K.N):=e; Felcsúsztat(K,K.N)
Eljárás vége.

Felcsúsztat(K,i):
  Ha i>1 akkor Ha K.t(i)<K.t(i div 2)
                 akkor Csere(K.t(i),K.t(i div 2))
                       Felcsúsztat(k,i div 2)
Eljárás vége.
```

Kupac

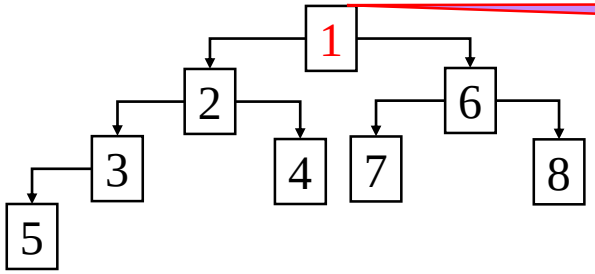
```
Felcsúsztat(K,i):
  Ciklus amíg i>1 és K.t(i)<K.t(i div 2)
    Csere(K.t(i),K.t(i div 2))
    i:=i div 2
  Ciklus vége
Eljárás vége.
```

Hatékonyabb lehet, ha csere helyett csak a végén tesszük a helyére.

```
Kupacból(K,e):
  e:=K.t(1); K.t(1):=K.t(K.N); K.N:=K.N-1
  Lecsúsztat(K,1)
Eljárás vége.
```

Kupac

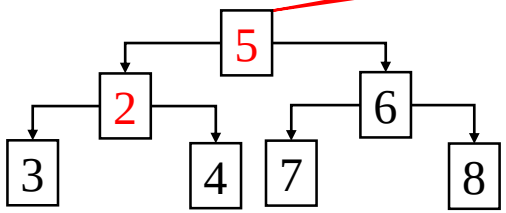
Elemkivétel: Kupacból([1,2,6,3,4,7,8,5]) ⇒ (1,[2,3,6,5,4,7,8])

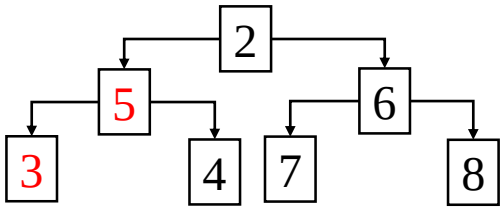


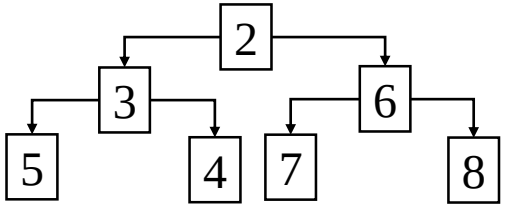
Elemkiolvasás

„Utolsó előre fuss!”

Előrehozás + lecsúsztatás:

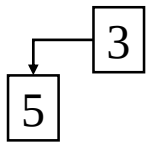


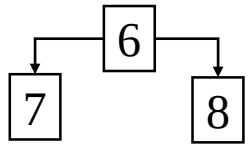




Kupac

```
Lecsúsztat(K,i):
  Ha i≤K.N div 2
     akkor j:=kisebb(2*i,2*i+1)
           Ha K.t(i)>K.t(j)
              akkor Csere(K.t(i),K.t(j))
                    Lecsúsztat(K,j)
Eljárás vége.
kisebb(j,k):
  Ha k>K.N vagy K.t(j)≤K.t(k) akkor kisebb:=j
                               különben kisebb:=k
Eljárás vége.
```





Kupac

```
Lecsúsztat(K,i):
  j:=kisebb(2*i,2*i+1)
  Ciklus amíg i≤K.N div 2 és K.t(i)>K.t(j)
    Csere(K.t(i),K.t(j))
    i:=j; j:=kisebb(2*i,2*i+1)
  Ciklus vége
Eljárás vége.
```

Hatékonyabb lehet, ha csere helyett csak a végén tesszük a helyére.

Kupac

Műveletigény:

Lecsúsztat = Kupacból: $O(\log_2(N))$

Felcsúsztat = Kupacba: $O(\log_2(N))$

Használjuk a kupacot rendezésre!

Rendezési idő: $O(N*\log_2(N))$

Kupacrendezés

```
Rendez(X):
  Üres(K)
  Ciklus i=1-től N-ig
    Kupacba(K,X(i))
  Ciklus vége
  Ciklus i=1-től N-ig
    Kupacból(K,X(i))
  Ciklus vége
Eljárás vége.
```

Prioritási sor

Prioritási sor: speciális sor

Mindig a legfontosabb (minimális vagy maximális) lép ki belőle.
Műveletei: Üres, üres?, PrSorba, PrSorból, első.

Emlékeztető (ciklikus tömb megvalósítás esetén):

- időrendbeli tárolás (kb.)

  Sorba – végére – O(1), Sorból – minimumkiválasztás, majd az utolsó a minimum helyére – O(N)

- nagyság szerinti tárolás

  Sorba – beillesztés a helyére – O(N), Sorból – elejéről – O(1)

Prioritási sor

Prioritási sor megvalósítása kupaccal

Ötletek:

- A prioritási sor speciális kupac, ahol elemek **sorszámát** tároljuk, egy prioritás tömbben pedig a prioritásukat. (Ha egyéb jellemzőjük lenne, azt is külön tárolnánk.)
- A prioritási sor speciális kupac, ahol az elemek rekordok, és az egyik mezőjük a prioritás.

Prioritási sor

Prioritási sor ábrázolása (kupac+prioritás tömb):

```
Rekord(N: egész,
       t: Tömb(1..MaxN: Egész),
       pr: Tömb(1..MaxN:Egész))
```

Műveletei: Üres, üres?, PrSorba, PrSorból, első

Megoldás: újraírjuk a kupac műveleteit.

Prioritási sor

Műveletek megvalósítása:

```
Üres(P):
  P.N:=0
Eljárás vége.
üres?(P):
  üres?:=(P.N=0)
Függvény vége.
első(P):
  első:=P.t(1)
Függvény vége.
```

Prioritási sor

```
PrSorba(P,e,pr):
  P.N:=P.N+1; P.t(P.N):=e; P.pr(e):=pr
  Felcsúsztat(P,P.N)
Eljárás vége.
PrSorból(P,e,pr):
  e:=P.t(1); pr:=P.pr(e); P.t(1):=P.t(P.N)
  P.N:=P.N-1; Lecsúsztat(P,1)
Eljárás vége.
```

Prioritási sor

```
Felcsúsztat(P,i):
  Ha i>1 akkor
     Ha P.pr(P.t(i))<P.pr(P.t(i div 2))
        akkor Csere(P.t(i),P.t(i div 2))
              Felcsúsztat(k,i div 2)
Eljárás vége.
```

Prioritási sor

```
Lecsúsztat(P,i):
  Ha i≤P.N div 2
     akkor j:=kisebb(2*i,2*i+1)
           Ha P.pr(P.t(i))>P.pr(P.t(j))
              akkor Csere(P.t(i),P.t(j))
                    Lecsúsztat(P,j)
Eljárás vége.
kisebb(j,k):
  Ha k>P.N vagy P.pr(P.t(j))≤P.pr(P.t(k))
     akkor kisebb:=j különben kisebb:=k
Eljárás vége.
```

Prioritási sor

Prioritási sor ábrázolása (kupac rekord elemekkel):

```
Rekord(N: egész,
       t: Tömb(1..MaxN: Elemtípus))
Elemtípus=Rekord(pr: Egész,
                 egyéb: Egyébtípus)
```

Műveletei: Üres, üres?, PrSorba, PrSorból, első

Megoldás: újraírjuk a kupac műveleteit.

Prioritási sor

Műveletek megvalósítása:

```
Üres(P):
  P.N:=0
Eljárás vége.
üres?(P):
  üres?:=(P.N=0)
Függvény vége.
első(P):
  első:=P.t(1)
Függvény vége.
```

Prioritási sor

```
PrSorba(P,e):
  P.N:=P.N+1; P.t(P.N):=e
  Felcsúsztat(P,P.N)
Eljárás vége.
PrSorból(P,e):
  e:=P.t(1); P.t(1):=P.t(P.N); P.N:=P.N-1
  Lecsúsztat(P,1)
Eljárás vége.
```

Prioritási sor

```
Felcsúsztat(P,i):
  Ha i>1 akkor
     Ha P.t(i).pr<P.t(i div 2).pr
        akkor Csere(P.t(i),P.t(i div 2))
              Felcsúsztat(P,i div 2)
Eljárás vége.
```

```
Felcsúsztat(P,i):
  Ha i>1 akkor
     Ha P.pr(P.t(i))<P.pr(P.t(i div 2))
        akkor Csere(P.t(i),P.t(i div 2))
              Felcsúsztat(k,i div 2)
Eljárás vége.
```

Prioritási sor

```
Lecsúsztat(P,i):
  Ha i≤P.N div 2
     akkor j:=kisebb(2*i,2*i+1)
           Ha P.t(i).pr>P.t(j).pr
              akkor Csere(P.t(i),P.t(j))
                    Lecsúsztat(P,j)
Eljárás vége.
kisebb(j,k):
  Ha k>P.N vagy P.t(j).pr≤P.t(k).pr
     akkor kisebb:=j különben kisebb:=k
Eljárás vége.
```

```
Lecsúsztat(P,i):
  Ha i≤P.N div 2
     akkor j:=kisebb(2*i,2*i+1)
           Ha P.pr(P.t(i))>P.pr(P.t(j))
              akkor Csere(P.t(i),P.t(j))
                    Lecsúsztat(P,j)
Eljárás vége.
```

Prioritási sor

Prioritási sor megvalósítása kupaccal – értékelés

A prioritási sor speciális kupac, ahol elemek sorszámát tároljuk, egy prioritás tömbben pedig a prioritásukat.

- Csak akkor alkalmazható, ha az elemek sorszámozhatók.
- Gazdaságos az elemek kupacban való mozgatása miatt.

A prioritási sor speciális kupac, ahol az elemek rekordok, s az egyik mezőjük a prioritás.

- Egyszerűbb megvalósítás, nem sorszámozható elemekre is.
- Lassú lehet hosszú elemek mozgatása a kupacban.

Prioritási sor

Módosítható prioritási sor ábrázolása (kupac + volt tömb), ha csak felfelé mozgás van:

```
Rekord(N: egész,
       t: Tömb(1..MaxN: Elemtípus),
       volt: Tömb(1..MaxN: Logikai))
```

Műveletei: Üres, üres?, PrSorba, PrSorból, első

A már sorban levő elemeket prioritásuk megváltozása esetén újra felvesszük, a már kivetteket kivételkor lépjük át.

Prioritási sor

Ha egy elemet már kivettünk a sorból (a megnőtt prioritással), akkor újabb kivétel esetén eldobjuk.

```
PrSorból(P,e):
  Ciklus
    e:=P.t(1); P.t(1):=P.t(P.N); P.N:=P.N-1
    Lecsúsztat(P,1)
  amíg P.volt(e.index)
  Ciklus vége
  P.volt(e.index):=igaz
Eljárás vége.
```

Prioritási sor

Módosítható prioritási sor ábrázolása (kupac + prioritás tömb + index tömb):

```
Rekord(N: egész,
       t: Tömb(1..MaxN: Elemtípus),
       pr: Tömb(1..MaxN: Egész),
       index: Tömb(1..MaxN: Egész))
```

Műveletei: Üres, üres?, PrSorba, PrSorból, első, Fel, Le

A már sorban levő elemeket prioritásuk megváltozása esetén mozgatni kell, azaz tudnunk kell, hol voltak.

Prioritási sor

Műveletek megvalósítása:

```
Üres(P):
  P.N:=0
Eljárás vége.
üres?(P):
  üres?:=(P.N=0)
Függvény vége.
első(P):
  első:=P.t(1)
Függvény vége.
```

Prioritási sor

```
PrSorba(P,e,pr):
  P.N:=P.N+1; P.t(P.N):=e; P.pr(e):=pr
  P.index(e):=P.N; Felcsúsztat(P,P.N)
Eljárás vége.
PrSorból(P,e,pr):
  e:=P.t(1); pr:=P.pr(e); P.t(1):=P.t(P.N)
  P.N:=P.N-1; P.index(P.t(1)):=1; Lecsúsztat(P,1)
Eljárás vége.
Le(P,i):
  Lecsúsztat(P,P.index(i))
Eljárás vége.
```

Prioritási sor

```
Lecsúsztat(P,i):
  Ha i≤P.N div 2
     akkor j:=kisebb(2*i,2*i+1)
           Ha P.pr(P.t(i))>P.pr(P.t(j))
              akkor Csere(P.t(i),P.t(j))
                    P.index(P.t(i)):=i
                    P.index(P.t(j)):=j
                    Lecsúsztat(P,j)
Eljárás vége.
```

Prioritási sor

```
Fel(P,i):
  Felcsúsztat(P,P.index(i))
Eljárás vége.
Felcsúsztat(P,i):
  Ha i>1 akkor
     Ha P.pr(P.t(i))<P.pr(P.t(i div 2))
        akkor Csere(P.t(i),P.t(i div 2))
              P.index(P.t(i)):=i
              P.index(P.t(i div 2)):=i div 2
              Felcsúsztat(P,i div 2)
Eljárás vége.
```

Algoritmusok és adatszerkezetek I.
3. előadás vége