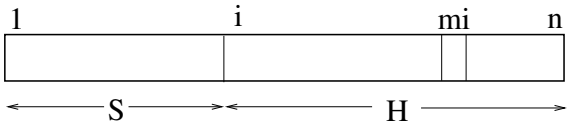
13. Rendezési algoritmusok

Rendezési probléma
Bemenet: Azonos típusú adatok $H = \{a_1, \ldots, a_n\}$ halmaza, amelyeken értelmezett egy $\leq$ lineáris rendezési reláció. (A $\leq$ rendezési reláció maga is lehet bemeneti paraméter.)
Kimenet: A H halmaz elemeinek egy $\leq$ rendezéstartó felsorolása, tehát olyan $S = \langle b_1, \ldots, b_n \rangle$ sorozat, amelyre $b_1 \leq b_2 \leq \ldots \leq b_n$, és $H = \{b_1, \ldots, b_n\}$.
Belső rendezés. H és S tárolása a főtárban történik.
Külső rendezés. Vagy H vagy S tárolása külső (lemezes állományban) tárolón történik.
Helyben rendezés. Ha a meneneti H halmazt és a kimeneti S sorzatot ugyanaz az adatszerkezet tárolja.

13.1. Kiválasztó rendezés

Elvi algoritmus:

$S := \langle\rangle$;
While $H <> \emptyset$ **Do Begin**
  $x := H$ minimális eleme;
  Tegyük x-et az S sorozat végére;
  Töröljük x-et H-ból;
End



1. ábra. A kiválasztó rendezés megvalósítása helyben

```
Procedure Kivalasztorend(Var T:Tomb);
Var
  i,j,mi : Integer; E : Elemtip;
Begin
  For i := 1 To N-1 Do Begin
    {S=T[1..i-1], H=T[i..N]}
    mi := i;
    For j := i+1 To Tmeret Do
      If T[j].kulcs < T[mi].kulcs Then
        mi := j;
    {T[mi]=Min(H)}
    E := T[i]; T[i] := T[mi]; T[mi] := E;
    {T[mi] S végére; T[mi] törlése H-ból}
  End
End (* Kivalasztorend *);
```

A KIVALASZTOREND *futási idejének elemzése*
Jelölje $T(n)$ a végrehajtott elemi művelet számát, ha $|H|$=n

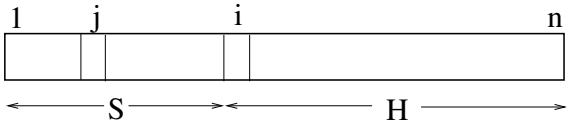
$$\begin{aligned}\sum_{i=1}^{n-1}(5+n-i) &\leq T(n) \leq \sum_{i=1}^{n-1}(5+2(n-i))\\ 5(n-1)+\sum_{i=1}^{n-1} i &\leq T(n) \leq 5(n-1)+2\sum_{i=1}^{n-1} i\\ 5(n-1)+\frac{n(n-1)}{2} &\leq T(n) \leq (n-1)(5+n)\end{aligned}$$

$$T_{lj}(n) = T_a(n) = T_{lr}(n) = \Theta(n^2)$$

13.2. Beszúró rendezés

Elvi algoritmus:

$S := \langle\rangle$; {üres output sorozat létesítés}
While $H <> \emptyset$ **Do Begin**
  $x := H$ egy tetszőleges eleme;
  Szúrjuk be x-et az S sorozatba;
  Töröljük x-et H-ból;
End



2. ábra. A beszúró rendezés megvalósítása helyben

```
Procedure Beszurorend (Var T:Tomb;K:RendRelTip);
Var
  i,j : Integer; E : Elemtip;
Begin
  For i := 2 To N Do Begin
    {S=T[1..i-1], H=[i..N]}
    E := T[i]; j := i-1;
    While(j>0)And K(E,T[j]) Do Begin
      T[j+1] := T[j]; Dec(j)
    End{while};
    {S[1..j] ≤ E < S[j+1..i-1] }
    T[j+1] := E;
  End{for}
End (* BeszuroRend *);
```

A BESZUROREND *futási idejének elemzése*

Legjobb eset: az input rendezett:
$T_{lj}(n) = \sum_{i=2}^{n} 5 = 5(n-1) = O(n)$
Legroszabb eset: az input fordítottan rendezett, ekkor a a **While** ciklus magja $i-1$-szer hajtódik végre, tehát

$$T_{lr}(n) = \sum_{i=2}^{n}(i-1) = \sum_{i=1}^{n-1} i = \quad \frac{n(n-1)}{2} = O(n^2)$$

Átlagos eset:
$rang(S,x) := |\{j : S[j] > x\}|$
$rang(S,T[i])$ lehetséges értékei:
$0,1,\cdots,i-1$ azonos $\frac{1}{i}$ valószínűséggel.
Tehát a while ciklusmag végrehajtási számának átlágos (várható) értéke:

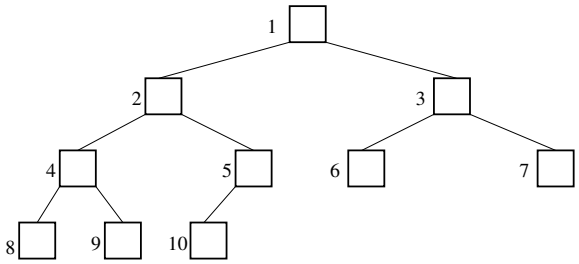
$$\frac{1}{i}(0+1+\cdots i-1) \quad = \quad \frac{1}{i}\frac{(i-1)i}{2} = \frac{1}{2}(i-1)$$

Tehát

$$T_a(n) = \sum_{i=2}^{n}\frac{1}{2}(i-1) = \frac{1}{2}\sum_{i=1}^{n-1} i = \frac{1}{2}\frac{n(n-1)}{2} = O(n^2)$$

13.3. Kupacrendezés. (Williams, Floyd 1964)

$S = (M,R,Adat)$ adatszerkezet, ahol
$M = 1..n$, $R = \{Apa : M \to M\}$
$Apa(i) = i$ **div** 2, ha $i > 1$
Az S (fa) adatszerkezet $Apa \to fiu$ kapcsolattal is megadható:



3. ábra. Kupac adatszerkezet

$Bal(i) := 2i$, ha $2i \leq n$
$Jobb(i) := 2i+1$, ha $2i+1 \geq n$
Def. S (maximumos) kupac, ha rendezett a $\leq$-ra nézve.
Általánosan, $T = k..l$. Ekkor több fából áll az adatszerkezet.
$Apa(i) = i$ **div** 2, ha $i \geq 2k$
Def. Az $\langle a_k,\cdots,a_l\rangle$ sorozat (maximumos) kupac, ha $\forall i \in k..l$ ha $r = i$ **div** $2 \geq k$ akkor $a_r \geq a_i$

```
{ Globális programelemek a KupacRend eljáráshoz :
  Const
    MaxN = ???           ;{ a maximális tömbméret }
  Type
    Kulcstip = ???       ;{ a rendezési mező típusa }
    Adattip  = ???       ;{ az adatmező típusa           }
    Elemtip  = Record
                 kulcs : Kulcstip;
                 adat  : Adattip
               End;
    Tomb = Array[1..MaxN] Of Elemtip;
}

Procedure KupacRend(Var T : Tomb; N:Longint);
  Var
    i : Longint; E : Elemtip;

  Procedure Sullyeszt(K,L : Longint );
  {Input : T[K+1..L] kupac,  Output: T[K..L] kupac   }
    Var  Apa,Fiu : Longint;
    Begin{Sullyeszt}
      E:=T[K]; Apa:=K; Fiu:=2*Apa;
      While (Fiu <= L) Do Begin
        If (Fiu < L) And (T[Fiu].kulcs < T[Fiu+1].kulcs) Then
          Fiu := Fiu+1;
        If E.kulcs >= T[Fiu].kulcs Then
          Break
        Else Begin
          T[Apa] := T[Fiu];
          Apa:=Fiu; Fiu:=2*Apa
        End
      End{while};
      T[Apa] := E
    End{Sullyeszt};
  Begin{KupacRend}
    For i := N Div 2 Downto 1 Do  Sullyeszt(i, N);{Kupacépít}
    For i := N Downto 2 Do Begin
      E:=T[i]; T[i]:=T[1]; T[1]:=E;
      Sullyeszt(1,i-1)
    End{for i};
  End{KupacRend};
```

A kupacépítés futási ideje

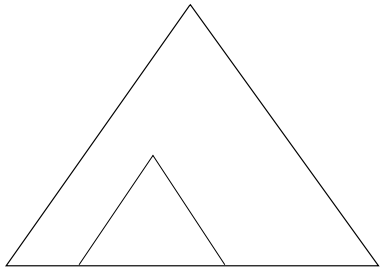
Def. h-magaságú telített bináris fa: F_h

$F_0 = \perp$, $F_h = \bigodot(F_{h-1}, F_{h-1})$ ha $h > 0$

F_h magassága $= h$

F_h pontjainak száma $\sum_0^{h-1} 2^k = 2^h - 1$

Legyen F n-pontú bináris kupac; $|F| = n$
Ekkor $h(F)$ az a legkisebb k, amelyre
$|F_k| \geq |F| \Leftrightarrow 2^k - 1 \geq n \Leftrightarrow k \geq \lg(n+1)$
$h = \lceil \lg(n+1) \rceil$
A kupacépítés során a SULLYESZT annyiszor hajtódik végre, mint ahány különböző nem levél részfája van a felépítendő n-elemű kupacnak.



4. ábra. Kupac részfái

Jelölje $R(n,k)$ az n-pontú kupac k magasságú részfáinak számát.
$R(n,k) \leq 2^{h-k} = \frac{2^h}{2^k} = \frac{2^{h-1}}{2^{k-1}} \leq \frac{n}{2^{k-1}}$
Mivel SULLYESZT futási ideje legrosszabb esetben azon fa magasságával arányos, amelybe a beszúrás történik, így a kupacépítés futási idejére a következőt kapjuk:

$$T_{lr}(n) \leq \sum_{k=1}^{h} R(n,k)O(k) \leq$$

$$\sum_{k=0}^{h-1} \frac{n}{2^k} O(k) \leq O(n \sum_{k=0}^{\infty} \frac{k}{2^k}) =$$

$$O(n \frac{\frac{1}{2}}{(1-\frac{1}{2})^2}) = O(n2) = O(n)$$

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \quad ha\, |x| < 1$$

"Mindkét oldalt deriválva":

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

A KUPACREND futási ideje

$$T_{lr}(n) \leq O(n) + \sum_{k=\lceil \frac{n}{2} \rceil}^{n} O(h) \leq$$

$$O(n) + O(\sum_{k=1}^{n} \lg n) =$$

$$O(n) + O(n \lg n) = O(n \lg n)$$

13.4. A gyorsrendezés (Hoare, 1962)

Elvi algoritmus:
Legyen FELOSZT(H,H_b,x,H_j) olyan művelet, amelyre teljesül a következő kimenetti feltétel:

$$x \in Pre(H) \wedge H_b = \{y : x \neq y \in Pre(H) \wedge y \leq x\} \wedge H_j = \{y : y \in Pre(H) \wedge x < y\}$$

Tehát $Pre(H) = H_b \cup \{x\} \cup H_j$
Ezért a következő oszd-meg-és-uralkodj elvű algoritmus a rendezési feladat helyes megoldását adja.

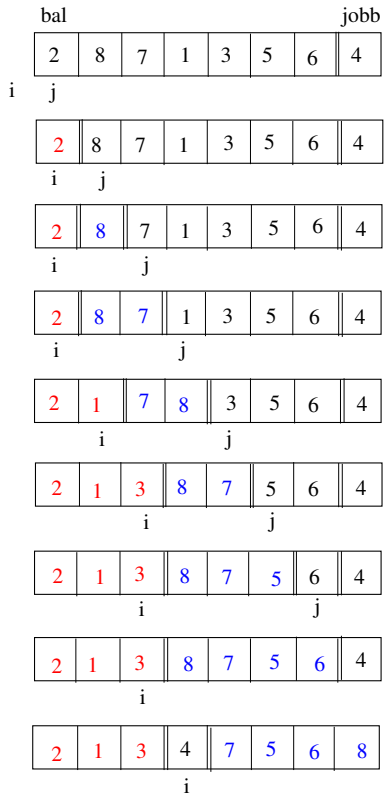
Rendez(H:Halmaz):Sorozat;
Var
$x : Elemtip$; S_b, S_j:Sorozat; H_b, H_j:Halmaz;
Begin{Rendez}
If $|H| \leq 1$ **Then Begin**
$S := H$
Return(S)
End;
$Feloszt(H,H_b,x,H_j)$; {megosztás}
$S_b := Rendez(H_b)$; {uralkodás}
$S_j := Rendez(H_j)$;
$S := S_b\langle x \rangle S_j$; {összerakás}
Return(S);
End{Rendez};

A Feloszt algoritmus:
Feloszt(H:Halmaz;**Var** H_b, H_j:Halmaz; x:Elemtip);

Begin{Feloszt}
x felosztó elem választás;
$H_b := \emptyset$; $H_j := \emptyset$;
For $y \in H$ **Do**
{Invariáns: $Max(H_b) \leq x < Min(H_j)$}
If $y \leq x \wedge y \neq x$ **Then**
$H_b := H_b + \{y\}$
Else
$H_j := H_j + \{y\}$
$\{H = H_b \cup H_j\}$
End{Feloszt};

Megjegyzés: lehet $H_b = \emptyset$ vagy $H_j = \emptyset$

Megvalósítás: helyben, tömbbel
Feloszt Lomuto-féle megvalósítása



5. ábra. A FELOSZT eljárás működése.

Function Feloszt(bal, jobb : Longint):Longint ;
$\{H = T[bal..jobb]\}$
Var x,E : Elemtip;
  i,j : Longint;
Begin{Feloszt}
  x:= T[jobb];
  i:=bal-1;
  For j:=bal **To** jobb-1 **Do**
    {Invariáns:$H_b = T[bal..i], H_j = T[i+1..j-1]\}$
    If $T[j] \leq x$ **Then Begin**
      i:=i+1;
      E:=T[i]; T[i]:=T[j]; T[j]:=E;
    End;
  i:=i+1;
  E:=T[i]; T[i]:=T[jobb]; T[jobb]:=E;
  Feloszt:=i;
$\{H_b = T[bal..i-1], H_j = T[i+1..jobb]\}$
End{Feloszt};

Procedure GYORSREND(Var T:Tomb);
  Procedure Rendez(bal,bobb : Longint);
  Var f : Longint;

```
  Begin{Rendez}
    f:= Feloszt(bal, jobb);
    If bal<f-1 Then
      Rendez(bal, f-1);
    If f+1<jobb Then
      Rendez(f+1, jobb)
  End{Rendez};
Begin
  Rendez(1, N)
End; {GyorsRend}
```

13.4.1. A gyorsrendezés hatékonysága

Legrosszabb eset

$$T_{lr}(n) = \max_{0 \le q \le n-1} (T_{lr}(q) + T_{lr}(n-q-1)) + \Theta(n)$$

$$\begin{aligned} T_{lr}(n) &\le \max_{0 \le q \le n-1} (cq^2 + c(n-q-1)^2) + \Theta(n) \\ &= c \cdot \max_{0 \le q \le n-1} (q^2 + (n-q-1)^2) + \Theta(n) \end{aligned}$$

A $q^2 + (n-q-1)^2$ kifejezés maximumát a $0 \le q \le n-1$ intervallum valamelyik végpontjában veszi fel, ezért $\max_{0 \le q \le n-1}(q^2 + (n-q-1)^2) \le (n-1)^2 = n^2 - 2n + 1$. Tehát

$$\begin{aligned} T_{lr}(n) &\le cn^2 - c(2n-1) + \Theta(n) \\ &\le cn^2 \end{aligned}$$

Tehát $T_{lr}(n) = O(n^2)$.

Legjobb eset

Ha minden FELOSZT felezi az intervallumot (a felosztandó halmazt). Ekkor minden szinten cn a FELOSZT futási ideje, és $\lceil \lg n \rceil$ szint léven, a teljes futási idő:

$$T_{lj}(n) = O(n \lg n)$$

Átlagos eset

13.1. lemma. *Legyen X a* GYORSREND *eljárás végrehajtása során a* FELOSZT *által végrehajtott összehasonlítások száma n elemű bemenetre. Ekkor* GYORSREND *teljes futási ideje $O(n+X)$.*

Célunk X átlagos (várható) értékének kiszámítása.

Legyen a T bemeneti tömb elemei rendezetten $z_1, z_2, \ldots, z_n$, tehát z_i az i-edik legkisebb eleme a bemenetnek. Definiáljuk a $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$ halmazokat, tehát a rendezésben z_i és z_j közötti elemek halmaza.

Az algoritmus mikor hasonlítja össze z_i és z_j elemeket?
Vegyük észre, hogy bármely két elem legfeljebb egyszer hasonlítódik össze,mert a felsztó elem nem szerepel a felbontásban keletkező H_b és H_j halmazokban, amelyere rekurzív hívás történik. Jelölje ezen összehasonlítások számát $X_{i,j}$.

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \ .$$

Az összehasonlítások $E(X)$ átlagos számára kapjuk:

$$\begin{aligned} E(X) &= E(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(X_{ij}) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr\{z_i \text{ összehasonlít } z_j\} \end{aligned} \tag{1}$$

z_i és z_j összehasonlítására akkor és csak akkor lkerül sor, ha az első felosztó elem a Z_{ij} halmazból vagy z_i, vagy z_j. Mivel a Z_{ij} halmaznak $j-i+1$ eleme van, és minden elem egyforma valószínűséggel lehet a felosztó elem, ezért

$$\begin{aligned} Pr\{z_i \text{ összehasonlít } z_j\} &= Pr\{z_i \text{ vagy } z_j \text{ első felosztó elem } Z_{ij}\text{-ből}\} \\ &= Pr\{z_i \text{ első felosztó elem } Z_{ij}\text{-ből}\} \\ &\quad + Pr\{z_j \text{ első felosztó elem } Z_{ij}\text{-ből}\} \\ &= \frac{1}{j-i+1} + \frac{1}{j-i+1} \\ &= \frac{2}{j-i+1} \end{aligned} \tag{2}$$

$$\begin{aligned} E(X) &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\ &= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\ &< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} \\ &= \sum_{i=1}^{n-1} O(\lg n) \\ &= O(n \lg n) \end{aligned} \tag{3}$$

Tehát GYORSREND átklagos futási ideje

$$T_a(n) = O(n \lg n)$$

A gyorsrendezés véletlenített változata.

```
Function VeletlenFeloszt( bal, jobb : Longint):Longint ;
Var E : Elemtip;
  i : Longint;
Begin{VeletlenFeloszt}
  i:=Random(jobb-bal+1)+1;
  E:=T[i]; T[i]:=T[jobb]; T[jobb]:=E;
  VeletlenFeloszt:=Feloszt(bal,jobb);
End{VeletlenFeloszt};
```

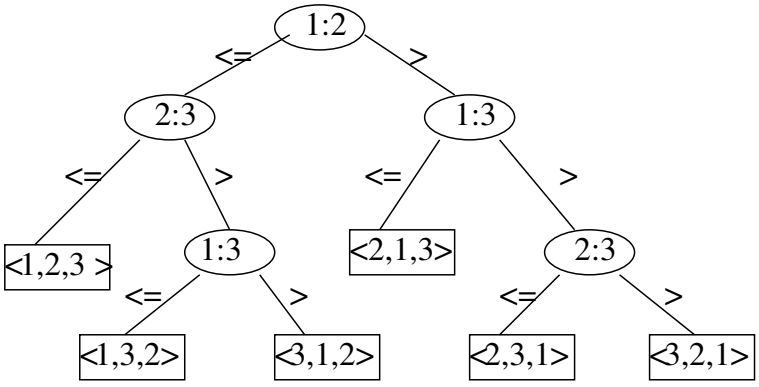
A Hoare-féle felsosztás.

```
{ Globalis objektumok a GyorsRend eljarashoz :
  Const
    MaxN = ???           ;(* a tömb indextipusa = 1..MaxN *)
  Type
    Kulcstip = ???       ;(* a rendezési mező típusa      *)
    Adattip  = ???       ;(* az adatmező típusa           *)
    Elemtip  = Record
                 kulcs : Kulcstip;
                 adat  : Adattip
               End;
    Tomb = Array[1..N] Of Elemtip;
  Procedure BeszuroRend(Var T : Tomb); {\$I...}
 }
Procedure GyorsRend(Var T : Tomb; N:Longint);

  Function HoareFeloszt( Bal,Jobb : Longint): Longint;
    Var
      Fe,E : Elemtip;
      i,j : Longint;
    Begin
      Fe := T[(Bal+Jobb) Div 2];
      i := Bal-1; j := Jobb+1;
      While True Do Begin
        Repeat
          Inc(i)
        Until (T[i].kulcs >= Fe.kulcs);
        Repeat
          Dec(j)
        Until (Fe.kulcs >= T[j.kulcs]);
        If i < j  Then Begin
          E := T[i]; T[i] := T[j]; T[j] := E;
        End Else Begin
          Feloszt:=j;
          Exit
```

```
      End;
    End{while};
  End (* HoareFeloszt *);

Procedure Rendez(bal,jobb : Longint);
  Var
    f : Longint;
  Begin
    f := HoareFeloszt(bal, jobb);
    If bal+10 < f Then
      Rendez(bal, f);
    If f+10 < jobb Then
      Rendez(f+1, jobb)
  End (* Rendez *);

Begin(* GyorsRend *)
  Rendez(1, N);
  Beszurorend(T)
End (* GyorsRend *);
```

13.5. Általános rendezési algoritmusok lr. esetének alsó korlátja

Döntési fa modell



6. ábra. Döntési fa

13.2. tétel. *Minden n elemet rendező döntési fa magassága* $\Omega(n \log n)$.

Bizonyítás. A fa leveleinek száma $\geq n!$, tehát ha a fa magassága h, akkor $2^{h-1} \geq n!$
$h-1 \geq lg(n!) \geq lg((\frac{n}{e})^n) = n \lg n - n \lg e = \Omega(n \lg n)$ ■

Minden általános rendezési algoritmusra:

$$T_{lr}(n) = \Omega(n \lg n)$$

13.6. Lineáris idejű rendezési algoritmusok

Számláló rendezés

```
{ Globális programelemek a SzamlaloRend eljáráshoz :
  Const
    N = ???                   ;(* a tomb indextipusa = 1..N *)
    M = ???                          ;(* a kulcstipus: 0..M *)
  Type
    Kulcstip = 0..M      ;(* a rendezesi mezo kulcstipusa   *)
    Adattip  = ???       ;(* az adatmezo tipusa             *)
    Elemtip  = Record
                 kulcs : Kulcstip;
                 adat  : Adattip
               End;
    Tomb = Array[1..N] Of Elemtip;
 }

Procedure SzamlaloRend(Var T,T1 : Tomb);
  Var
    i,j : Longint;
    S: Array[0..M] Of Longint;
  Begin
    For i := 0 To M Do S[i]:= 0;
    For i := 1 To N Do Inc(S[T[i].kulcs]);
    For i := 1 To M Do S[i]:= S[i-1]+S[i];
      (* S[i]=|{j| T[j] <= i}| *)
    For i := N DownTo 1 Do
      Begin
        j := T[i].kulcs;
        T1[ S[j] ]:= T[i];
        Dec(S[j]);
      End
  End (* SzamlaloRend *);
```

A SZAMLALOREND algoritmus futási ideje $\Theta(M+N)$

Stabilnak nevezzük az olyan rendezési algoritmust, amely megőrzi az azonos kulcsú elemek sorrendjét.

Állítás. A SZAMLALOREND algoritmus stabil rendezés.

Radix (számjegyes) rendezés

Példa:

```
329  720  720  329
457  436  329  436
657  457  436  457
839  657  839  657
```

```
436  329  457  720
720  839  657  839
```

Bemenet: H=$\{a_1,\ldots,a_n\}$, az elemek típusa

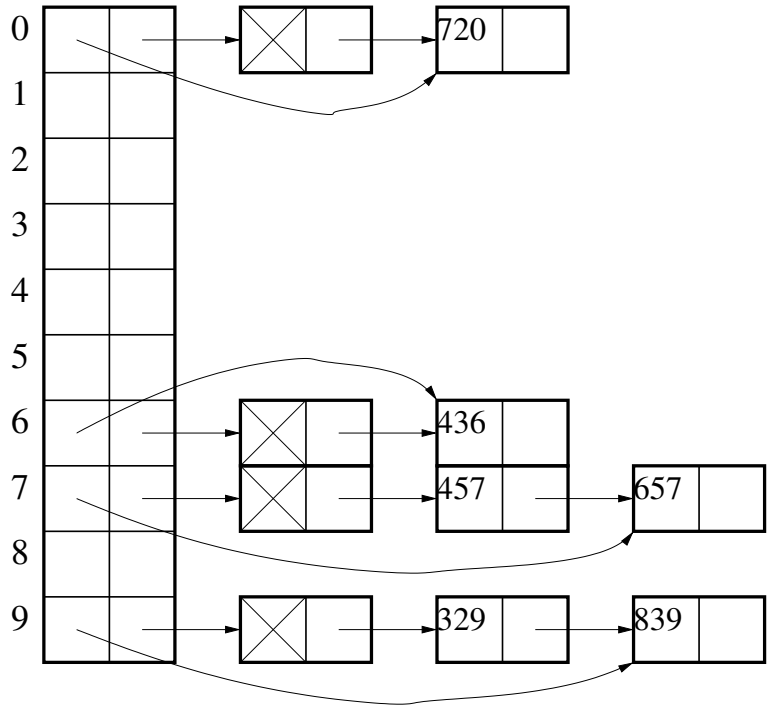
```
Type
 KulcsTip=Array[1..d] of Char {String};
 Adattip =???;
 Elemtip=Record Adat:Adattip; Kulcs:Kulcstip End;
```

A rendezési reláció a lexikografikus rendezés:
$X =< x_1,\ldots,x_d >, Y =< y_1,\ldots,y_d >$

Def. $X < Z$ (lexikografikusan), ha $(\exists i)(1 \leq i \leq d)((x_i < y_i) \wedge (\forall j < i)(x_j = y_j))$

Elv:

```
For i:=d DownTo 1 Do
 H Stabil rendezése a kulcs i-edik jegye szerint;
```



7. ábra. Adatszerkezet a radix rendezéshez.

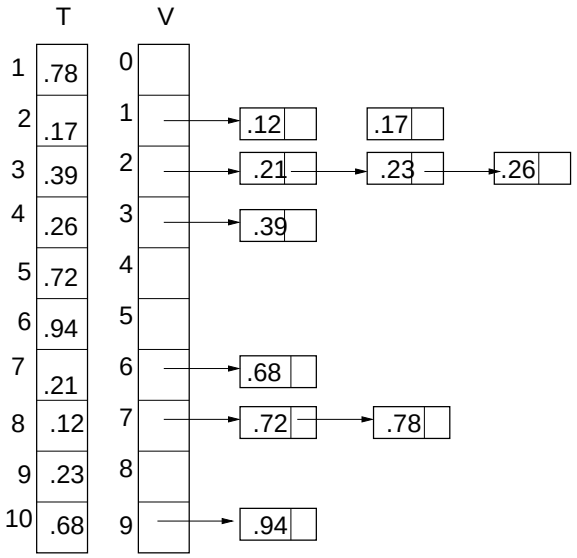
```
{ Globalis programelemek a RadixRen eljarashoz: Type
     Elemtip = Record  (* a rendezendo adatok tipusa *)
                 kulcs : String[???];
                 adat  : ???
```

```
              End;
     Lanc = ^Cella;
     Cella = Record
               Elem: Elemtip;
               Csat: Lanc
             End;

 }
Procedure RadixRend(Var L : Lanc);
  Var
    T : Array[Char] Of Record
                         Eleje,Vege:Lanc;
                       End;
    C : Char; E : Lanc;
    i, Maxhossz : Word;
Begin
  Maxhossz := 0; E:=L;(* a maximalis szohossz meghatarozasa *)
  While E <> Nil Do
    Begin
      If Length(E^.Elem.kulcs) > Maxhossz Then
        Maxhossz := Length(E^.Elem.kulcs);
      E:= E^.Csat
    End;
  For C := Chr(0) To Chr(255) Do        (* ures reszlistak letrehozasa *)
    Begin
      New(T[C].Vege); T[C].Eleje:= T[C].Vege;
    End;

  For i := Maxhossz Downto 1 Do
    Begin
      While L <> Nil Do (* szavak szetosztasa a reszlistakra, *)
        Begin                      (* az i-edik betu szereint            *)
          E:= L; L:= L^.Csat;
          If i <= Length(E^.Elem.kulcs) Then
            C := E^.Elem.kulcs[i]
          Else
            C := ' ';
          T[C].Vege^.Csat:= E;
          T[C].Vege:= E;
        End;
      L:= Nil;
      For C := Chr(255) DownTo Chr(0) Do
        Begin                           (* a reszlistak osszekapcsolasa *)
          T[C].Vege^.Csat:= L; L:= T[C].Eleje^.Csat;
          T[C].Vege:=T[C].Eleje;
        End
```

```
    End
End (* RadixRend *);
```

Vödrös rendezés

Tegyük fel, hogy a rendezendő $H = \{a_1, \ldots, a_n\}$ halmaz elemeinek kulcsai a $[0,1)$ intervallumba eső valós számok (Real). Vegyünk m db vödröt, $V[0], \ldots, V[m-1]$ és osszuk szét a rendezendő halmaz elemeit a vödrökbe úgy, hogy az a_i elem a $\lfloor a_i.kulcs * m \rfloor$ sorszámú vödörbe kerüljön. Majd rendezzük az egy vödörbe került elemeket, és a vödrök sorszáma szerinti növekvő sorrenben füzzük össze a rendezett részsorozatokat.



8. ábra. Példa vödrös rendezésre

```
{ Globális programelemek a VodrosRend eljáráshoz :
  Const
    N = ???              ;(* a tömb indextipusa = 1..N *)
   Type
    Kulcstip = Real      ;(* a rendezési mező kulcstípusa   *)
    Adattip  = ???       ;(* az adatmező típusa             *)
    Elemtip  = Record
                 kulcs : Kulcstip;
                 adat  : Adattip   End;
    Tomb = Array[1..N] Of Elemtip; }
Procedure VodrosRend(Var T,T1 : Tomb);
  Const
    M=N; {a vödrök száma}
  Type
    Lanc=^Cella;
    Cella=Record
            index: Word;
            Csat: Lanc
          End;
```

```
Var
  E: Elemtip;
  V:Array[0..M-1] Of Lanc;
  i,j,k : Word; p,q,Uj: Lanc;

Begin{VodrosRend}
  For i := 0 To M-1 Do V[i]:= Nil;
  For i := 1 To N Do Begin {az elemek szétosztása vödrökbe}
    j:= Trunc(T[i].kulcs*M);
    New(Uj); Uj^.index:= i;
    Uj^.csat:= V[j]; V[j]:= Uj;
  End;
  i:= 1;                    {a vödrökben lévő elemek összefűzése és}
  For j := 0 To M-1 Do Begin{rendezése beszúro rendezéssel}
    p:= V[j];
    While p <> Nil Do Begin
      E:= T[p^.index];
       k:= i-1;
       While (k>0) And (T1[k].kulcs > E.kulcs) Do Begin
         T1[k+1]:= T1[k]; Dec(k);
       End;
       T1[k+1]:= E; q:= p;
       p:= p^.Csat; Dispose(q);
       Inc(i);
    End{while p};
 End{for i};
End;{VodrosRend}
```

A VODROSREND futási idejének elemzése.

Legrosszabb eset

Ez akkor következik be, ha minden elem egy vödörbe kerül, és mivel a vödröket beszúrú endezéssel rendezzük, aminem a legrosszabb esete $O(n^)$, így $T_{lr}(n) = O(n^2)$.

Legjobb eset eset

Ez akkor következik be, ha minden elem külön vödörbe kerül, így $T_{lj}(n) = O(n)$.

Átlagos eset eset

$T_a(n) = \Theta(n)$>