16. Absztrakt adattípusok megvalósításai

Az adatkezelés szintjei:

1. Probléma szintje.
2. Modell szintje.
3. Absztrakt adattípus szintje.
4. Absztrakt adatszerkezet szintje.
5. Adatszerkezet szintje.
6. Gépi szint.

Absztrakt adattípus: $A = (E, M)$

1. E: értékhalmaz,
2. M: műveletek halmaza.

"Absztrakt" jelző jelentése:

i. Nem ismert az adatokat tároló adatszerkezet.

ii. Nem ismertek a műveleteket megvalósító algoritmusok, a műveletek specifikációjukkal definiáltak.

Alapvető absztrakt adattípusok

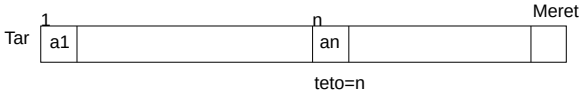
16.1. Verem

Értékhalmaz: $Verem = \{\langle a_1, \ldots, a_n \rangle : a_i \in E\}$
Műveletek:

$$V : Verem, x : E$$

$$\begin{array}{rcl}
\{Igaz\} & Letesit(V) & \{V = \langle\rangle\} \\
\{V = V\} & Megszuntet(V) & \{Hamis\} \\
\{V = V\} & Uresit(V) & \{V = \langle\rangle\} \\
\{V = \langle a_1, \ldots, a_n \rangle\} & VeremBe(V, x) & \{V = \langle a_1, \ldots, a_n, x \rangle\} \\
\{V = \langle a_1, \ldots, a_n \rangle \wedge n > 0\} & VeremBol(V, x) & \{V = \langle a_1, \ldots, a_{n-1} \rangle \wedge x = a_n\} \\
\{V = V\} & Urese(V) & \{Urese = Pre(V) = \langle\rangle\} \\
\{V = \langle a_1, \ldots, a_n \rangle\} & Tete je(V, x) & \{x = a_n \wedge V = Pre(V)\} \\
\{V = \langle a_1, \ldots, a_n \rangle \wedge n > 0\} & Torol(V) & \{V = \langle a_1, \ldots, a_{n-1} \rangle\}
\end{array}$$

1. Tömbös megvalósítás.
Hátránya: előre rögzített méretű memóriát kell foglalni.



1. ábra. Verem tömbös megvalósítása.

2. Megvalósítás lánccal.

2. ábra. Verem megvalósítása lánccal.

Dinamikus memória allokálás. m byte allokálása ténylegesen

$$\lceil (m+4)/16 \rceil * 16$$

byte-ot foglal a halomból! Tehát a tényleges tárigény, ha az adatelemek mérete m byte:

$$\lceil (m+4+4)/16 \rceil * 16$$

3. Kombinált megvalósítás.
K adatelemet tartalmazó tömbszeletek láncolása.
Memória igény:

3. ábra. Verem kombinált megvalósítása.

$$\lceil (\lceil n/K \rceil Sizeof(Elemtip)+4+4)/16 \rceil * 16$$

A müveletek (a MEGSZUNTET és URESIT kivételével) futási ideje n-elemű veremre: $T_{lr}(n) = \Theta(1)$

```
Unit VeremP;
  Interface
    Type
      Elemtip = ???                        ;(* a generikus paraméter *)
      Tipus = Pointer                 ;(* a Verem adattipus fő típusa *)
      Verem = VeremP.Tipus;
    { a Verem adattipus muveletei: }
      Procedure Letesit(Var V : Tipus);
      Procedure Megszuntet(Var V : Tipus);
      Procedure Uresit(Var V : Tipus);
      Procedure VeremBe(Var V : Tipus;
                        X : Elemtip);
      Procedure VeremBol(Var V : Tipus;
                     Var X : Elemtip);
      Function  ElemSzam(V : Tipus) : Word;
      Procedure Teteje(    V : Tipus;
                        Var X : Elemtip);
      Procedure Torol(Var V : Tipus);

  Implementation
{ reprezentáció pointerlánc használatával }
  Type
    Lanc = ^Cella;
    Cella = Record
```

```
                adat : Elemtip;
                csat : Lanc
              End;
    RepTip = Record
               Fej: Lanc;
               ESzam: Word
             End;

{ A műveletek megvalósítása }
  Procedure Letesit(Var V : Tipus);
    Var Vr : ^RepTip Absolute V;
    Begin
      New(Vr);
      Vr^.Fej := Nil;
      Vr^.Eszam:=0;
    End (* Letesit *) ;

  Procedure Uresit(Var V : Tipus);
    Var Vr : ^RepTip Absolute V;
      P, Q : Lanc;
    Begin
      P := Vr^.Fej;
      While P <> Nil Do Begin
        Q := p^.csat; Dispose(P);
        P := Q
      End;
      Vr^.Fej := Nil; Vr^.ESzam:= 0;
    End (* Uresit *) ;

  Procedure Megszuntet(Var V : Tipus);
    Var Vr : ^RepTip Absolute V;
    Begin
      Uresit(V);
      Dispose(Vr);
    End (* Megszuntet *) ;

  Procedure VeremBe(Var V : Tipus; X : Elemtip);
    Var Vr : ^RepTip Absolute V;
        Uj : Lanc;
    Begin
      New(Uj);
      Uj^.adat := X;
      Uj^.csat := Vr^.Fej;
      Vr^.Fej := Uj;
      Inc(Vr^.ESzam);
    End (* VeremBe *) ;

  Procedure VeremBol(Var V : Tipus; Var X : Elemtip);
    Var Vr : ^RepTip Absolute V; P : Lanc;
    Begin
      P := Vr^.Fej;
      If P <> Nil Then  Begin
        X := P^.adat; Vr^.Fej := P^.csat;
        Dec(Vr^.ESzam); Dispose(P)
      End
    End (* VeremBol *) ;
```

```
  Function ElemSzam(V : Tipus) : Word;
    Var Vr : ^RepTip Absolute V;
    Begin
      ElemSzam := Vr^.ESzam
    End (* Elemszam *) ;

  Procedure Teteje(V : Tipus; Var X : Elemtip);
    Var Vr : ^RepTip Absolute V;
    Begin
      If Vr^.Fej <> Nil Then
        X := Vr^.Fej^.adat
    End (* Teteje *) ;

  Procedure Torol(Var V : Tipus);
    Var Vr : ^RepTip Absolute V; P : Lanc;
    Begin
      P:= Vr^.Fej;
      If P <> Nil Then Begin
        Vr^.Fej := P^.csat;
        Dispose(P); Dec(Vr^.ESzam);
      End
    End (* Torol *) ;
End (* VeremP *).
```

16.2. Sor

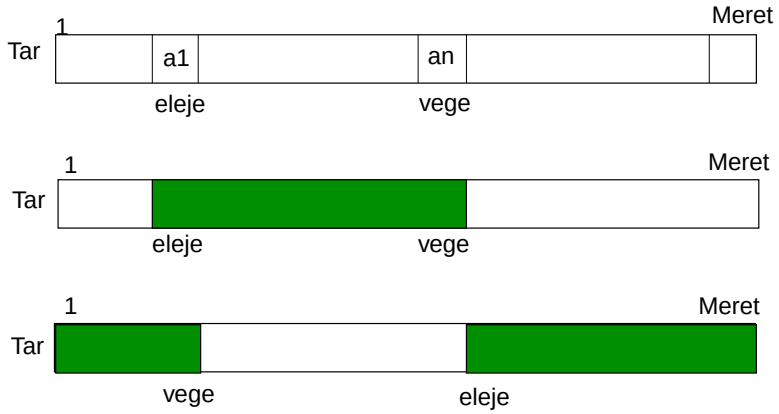
Értékhalmaz: $Sor = \{\langle a_1,\ldots,a_n\rangle : a_i \in E\}$
Műveletek:

$$S : Sor, x : E$$

$\{Igaz\}$	$Letesit(S)$	$\{S = \langle\rangle\}$
$\{S = S\}$	$Megszuntet(S)$	$\{Hamis\}$
$\{S = S\}$	$Uresit(S)$	$\{S = \langle\rangle\}$
$\{S = \langle a_1,\ldots,a_n\rangle\}$	$SorBa(S,x)$	$\{S = \langle a_1,\ldots,a_n,x\rangle\}$
$\{S = \langle a_1,\ldots,a_n\rangle \wedge n > 0\}$	$SorBol(S,x)$	$\{x = a_1 \wedge S = \langle a_2,\ldots,a_n\rangle\}$
$\{S = \langle a_1,\ldots,a_n\rangle\}$	$Elemszam(S)$	$\{Elemszam = n\}$
$\{S = \langle a_1,\ldots,a_n\rangle \wedge n > 0\}$	$Elso(S,x)$	$\{x = a_1 \wedge S = Pre(S)\}$
$\{S = \langle a_1,\ldots,a_n\rangle \wedge n > 0\}$	$Torol(S)$	$\{S = \langle a_2,\ldots,a_n\rangle\}$

1. Megvalósítás cirkuláris tömbbel.

```
Unit SorT ;
  Interface
    Type
      Elemtip = ???    ;(* generikus paraméter    *)
      Tipus  = Pointer ;(* a Sor adattipus típusa *)
      Sor = SorT.Tipus;
    { A Sor adattipus muveletei : }
    Procedure Letesit(Var S : Tipus);
    Procedure Uresit(Var S : Tipus);
```



4. ábra. Sor megvalósítása cirkuláris többel.

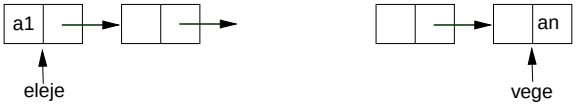
```
    Function  Elemszam(S : Tipus) :Word;
    Procedure SorBa(Var S : Tipus;
                        X : Elemtip);
    Procedure SorBol(Var S : Tipus;
                     Var X : Elemtip);
    Procedure Elso(    S : Tipus;
                   Var X : Elemtip);
    Procedure Torol(Var S : Tipus);

  Implementation
(* Reprezentáció cirkuláris tömbbel  *)
  Const
    Meret = ???      ;(* a cirkuláris tömb mérete, impl. paraméter  *)
  Type
    RepTip =  Record
                tar : Array[1..Meret] Of Elemtip;
                eleje, vege : 0..Meret
              End;
(* A műveletek megvalósítása  *)
  Procedure Letesit(Var S : Tipus);
    Var Sr : ^RepTip Absolute S;
    Begin
      New(Sr);
      Sr^.eleje := 0; Sr^.vege:=0;
    End (* Letesit *);

  Procedure SorBa(Var S : Tipus;
                      X : Elemtip);
    Var Sr : ^RepTip Absolute S;
    Begin
      With Sr^ Do
      If vege <> eleje Then Begin
        vege := (vege Mod Meret)+1;
        tar[vege] := X
      End;
    End (* SorBa *) ;

  Procedure SorBol(Var S : Tipus;
                   Var X : Elemtip);
    Var Sr : ^RepTip Absolute S;
```

```
    Begin
      With Sr^ Do
        If vege <> 0 Then  Begin
          eleje := eleje Mod Meret+1;
          X:= tar[eleje];
          If eleje = vege Then Begin
            eleje := 0; vege := 0
          End
        End
    End (* SorBol *) ;

  Function ElemSzam(S : Tipus) :Word;
    Var Sr : ^RepTip Absolute S; E: Integer;
    Begin
      With Sr^ Do Begin
        E := vege - eleje;
        If vege = 0 Then
          ElemSzam:=0
        Else If E > 0 Then
          ElemSzam:= E
        Else
          ElemSzam:= E + Meret
      End;
    End (* ElemSzam *) ;
  Procedure Uresit(Var S : Tipus);
    Var Sr : ^RepTip Absolute S;
    Begin
      Sr^.eleje := 0; Sr^.vege := 0
    End (* Uresit *) ;
  Procedure Elso(    S : Tipus;
                 Var X : Elemtip);
    Var Sr : ^RepTip Absolute S;
    Begin
      X := Sr^.tar[Sr^.eleje Mod Meret+1]
    End (* Elso *) ;

  Procedure Torol(Var S : Tipus);
    Var Sr : ^RepTip Absolute S;
    Begin
      With Sr^ Do
        If vege <> 0 Then Begin
          eleje := eleje Mod Meret+1;
          If eleje = vege Then
            Begin
              eleje := 0; vege := 0
            End
        End
    End (* Torol *) ;
End (* Sor *) .
```

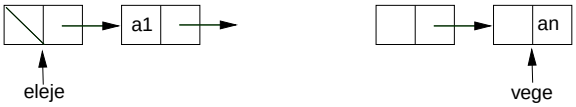
2. Sor megvalósítás lánccal.

3. Sor kombinált megvalósítása.

A müveletek (a MEGSZUNTET és URESIT kivételével) futási ideje n-elemű sorra: $T_{lr}(n) = \Theta(1)$



5. ábra. Sor megvalósítása lánccal.



6. ábra. Sor megvalósítása lánccal, fiktív első cellával.

16.3. Prioritási Sor

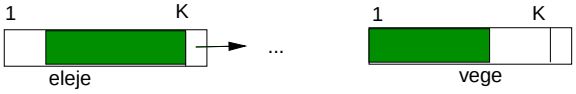
Értékhalmaz: $PriSor = \{\ S = \{a_1, \ldots, a_n\} : S \subseteq E\}$, E-n értelmezett a $\leq$ lineáris rendezési reláció.
Műveletek:

$$S : PriSor, x : E$$

$\{Igaz\}$	$Letesit(S, \leq)$	$\{S = \emptyset\}$
$\{S = S\}$	$Megszuntet(S)$	$\{Hamis\}$
$\{S = S\}$	$Uresit(S)$	$\{S = \emptyset\}$
$\{S = S\}$	$SorBa(S, x)$	$\{S = Pre(S) \cup \{x\}\}$
$\{S \neq \emptyset\}$	$SorBol(S, x)$	$\{x = \min(Pre(S)) \wedge Pre(S) = S \cup \{x\}\}$
$\{S = \{a_1, \ldots, a_n\}\}$	$Elemszam(S)$	$\{Elemszam = n\}$
$\{S \neq \emptyset\}$	$Elso(S, x)$	$\{x = \min(Pre(S)) \wedge Pre(S) = S\}$
$\{S \neq \emptyset\}$	$Torol(S)$	$\{S = Pre(S) - \{\min(Pre(S))\}\}$

Megvalósítás kupaccal.

```
Unit PriSor ;
  { Minimumos prioritási sor adattipus megvalósítása kupaccal }
Interface
  Type
    Elemtip = ???;
    RendRelTip = Function (Var X,Y: Elemtip): Boolean;
    Tipus = Pointer ;(* az adattípus fő típusa *)
    PriSorT=PriSor.Tipus;
{ A prioritási sor adattipus műveletei: }
    Procedure Letesit(Var S : Tipus; R: RendRelTip);
    Procedure Megszuntet(Var S : Tipus);
    Procedure Uresit(Var S : Tipus);
    Function  Elemszam(S : Tipus) : Word;
```



7. ábra. Sor kombinalt megvalósítása.

```
    Procedure SorBa(Var S : Tipus;
                        X : Elemtip);
    Procedure SorBol(Var S : Tipus;
                     Var X : Elemtip);
    Procedure Elso(    S : Tipus;
                   Var X : Elemtip);
    Procedure Torol(Var S: Tipus);

Implementation
{ repezentacio tombos kupaccal}                  }
  Const
    Meret = ???    ;(* a reprezentáló tömb mérete, impl. paraméter *)
  Type
    Fa = Array[1..Meret] Of Elemtip;
    RepTip = Record
               F : Fa;
               Eszam : 0..Meret;
               Kis:RendReltip
             End;
{ Segéd műveletek: }

 Procedure Emel(Var F:Fa; Var Kis:RendReltip;
                                    i:Word);
 {Input: Ha j<>i akkor F[Apa(j)]<=F[j] }
 {Output: Minden pontra teljesul a kupac tulajdonsag}
  Var
  apa,fiu:Word; X:Elemtip;
Begin{Emel}
  X:=F[i];
  fiu:=i;
  apa:=fiu Shr 1;
  While (apa>0) And Kis(X, F[apa]) Do Begin
    F[fiu]:=F[apa];
    fiu:=apa; apa:=fiu Shr 1;
  End{while};
  F[fiu]:=X;
End{Emel};

Procedure Sullyeszt(Var F:Fa; Var Kis:RendReltip; Eszam, i:Word);
 {Input: Ha j<>i akkor F[j]<=F[Bfiu(j)] F[j]<=F[Jfiu(j)] }
 {Output: Minden pontra teljesul a kupac tulajdonsag}
Var
  apa,fiu:Word; X:Elemtip;
Begin{Sullyeszt}
  X:=F[i]; apa:=i; fiu:=apa Shl 1;
  While (fiu<=Eszam) Do Begin
    If Kis(F[fiu+1], F[fiu]) Then Inc(fiu);
    If Not Kis(F[fiu], X) Then Break;
    F[apa]:=F[fiu];
    apa:=fiu; fiu:=apa Shl 1;
  End{while};
  F[apa]:=X;
End{Sullyeszt};

{ a muveletek megvalositasa         }
  Procedure Letesit(Var S : Tipus;R:RendRelTip);
```

```
  Var Sr : ^ RepTip Absolute S;
  Begin
    New(Sr);
    Sr^.Eszam :=0;
    Sr^.Kis:=R;
  End (* Letesit *);
Procedure Megszuntet(Var S : Tipus);
  Var Sr : ^ RepTip Absolute S;
  Begin
    Dispose(Sr);
  End (* Letesit *);
Procedure Uresit(Var S : Tipus);
  Var Sr : ^ RepTip Absolute S;
  Begin
    Sr^.Eszam := 0
  End (* Uresit *) ;
Function ElemSzam(S : Tipus) : Word;
  Var Sr : ^ RepTip Absolute S;
  Begin
    ElemSzam := Sr^.Eszam
  End (* ElemSzam *) ;

Procedure SorBa(Var S : Tipus;
                    X : Elemtip);
  Var Sr : ^ RepTip Absolute S;
    apa, fiu : Integer;
  Begin
    With Sr^ Do
    If Eszam < Meret Then Begin
      Inc(Eszam);
      F[Eszam]:=X;
      Emel(F, Kis, Eszam);
    End{if}
  End (* SorBa *) ;

Procedure SorBol(Var S : Tipus;
                 Var X : Elemtip);
  Var Sr : ^ RepTip Absolute S;
    apa, fiu : integer; E : Elemtip;
  Begin
    With Sr^ Do
    If Eszam <> 0 Then Begin
      X:=F[1];
      F[1]:=F[Eszam];
      Dec(Eszam);
      Sullyeszt(F, Kis, Eszam, 1);
    End{if};
  End (* SorBol *) ;

Procedure Elso(S : Tipus; Var X : Elemtip);
  Var Sr : ^ RepTip Absolute S;
  Begin
    X := Sr^.F[1]
  End (* Elso *) ;

Procedure Torol(Var S : Tipus);
```

```
    Var E : Elemtip;
    Begin
      SorBol(S, E);
    End (* Torol *) ;

End (* PriSor *) .
```

A műveletek futási ideje.
SORBA és SORBOL futási ideje: $T_{lr}(n) = O(\lg n)$

16.4. Módosítható prioritási Sor

Műveletek: Prioritási sor műveletek + MODOSIT
Feltesszük, hogy az adatelemeket az 1..n számokkal azonosítjuk, továbbá az elemek típusa:

```
Type
  Elemtip=Record Kulcs:Kulcstip; Adat:Adattip End;
```

és a rendezés a kulcsmező alapján történik.
MODOSIT(S,I,K) művelet K-ra módosítja az i. elemet kulcsát és az új kulcs alapján módosítja az elem helyét az adatszerkezetben.
Szükség van olyan Hol függvényre, amely minden i-re megmondja, hogy hol van az F kupacban az i-edik elem:

$$F[Hol[i]] = i \wedge Hol[F[j]] = j$$

```
Unit ModPSor ;
Interface
  Type
    Kulcstip=???;
    AdatTip =???;
    Elemtip = Record Kulcs:Kulcstip; Adat:AdatTip End;
    Tipus = Pointer ;(* az adattipus fő tipusa *)
    MPriSor=ModPSor.Tipus;
  { A módosítható prioritási sor adattipus műveletei: }
    Procedure Letesit(Var S : Tipus);
    Procedure Megszuntet(Var S : Tipus);
    Procedure Uresit(Var S : Tipus);
    Function  Elemszam(S : Tipus) : Word;
    Procedure SorBa(Var S : Tipus;
                        X : Elemtip);
    Procedure SorBol(Var S : Tipus;
                     Var X : Elemtip);
    Procedure Elso(    S : Tipus; Var X : Elemtip);

    Procedure Torol(Var S : Tipus; i : Word);
    Procedure Modosit(Var S : Tipus;
                          i : Word; K:Kulcstip);

Implementation
 { repezentáció tömbös kupaccal  }
  Const
    Meret = ???    ;(* a reprezentalo tomb merete *)
  Type
    Fa = Array[1..Meret] Of 0..Meret;
    AdatSor=Array[1..Meret] of Elemtip;
    SorTip = Record
               A : AdatSor;
```

```
              F : Fa;
              Hol:Array[1..Meret] of 0..Meret;
              Eszam : 0..Meret;
            End;
    RepTip=SorTip;
{ Segéd műveletek: }

Procedure Emel(Var S:SorTip; i:Word);
 {Input: Ha j<>i akkor A[F[Apa(j)]]<=A[F[j]] }
 {Output: Minden pontra teljesul a kupac tulajdonsag}
Var
  apa,fiu,x:Word;
Begin{Emel}
  With S Do Begin
    x:=F[i];
    fiu:=i;
    apa:=fiu Shr 1;
    While (apa>0) And (A[x].kulcs < A[ F[apa] ].kulcs) Do Begin
      F[fiu]:=F[apa];
      Hol[F[apa]]:=fiu;
      fiu:=apa; apa:=fiu Shr 1;
    End{while};
    F[fiu]:=x;
    Hol[x]:=fiu
  End{with};
End{Emel};

Procedure Sullyeszt(Var S:SorTip; Eszam, i:Word);
 {Input: Ha j<>i akkor A[F[j]]<=A[F[Bal(j)]] A[F[j]]<=A[F[Jobb(j)]] }
 {Output: Minden pontra teljesul a kupac tulajdonsag}
 Var
  apa,fiu,x:Word;
Begin{Sullyeszt}
  With S Do Begin
    x:=F[i];
    apa:=i;
    fiu:=apa Shl 1;
    While (fiu<=Eszam) Do Begin
      If A[F[fiu+1]].kulcs < A[F[fiu]].kulcs Then Inc(fiu);
      If A[x].kulcs <= A[F[fiu]].kulcs Then Break;
      F[apa]:=F[fiu];
      Hol[F[fiu]]:=apa;
      apa:=fiu; fiu:=apa Shl 1;
    End{while};
    F[apa]:=x;
    Hol[x]:=apa
  End{with};
End{Sullyeszt};

Procedure Modosit(Var S : Tipus;
                      i : Word; K:Kulcstip);
  Var Sr : ^RepTip Absolute S;
    apa:Word;
  Begin
    apa:=Sr^.Hol[i] Shr 1;
    If K < Sr^.A[Sr^.F[apa]].kulcs Then Begin
```

```
    Sr^.A[i].kulcs:=K;
    Emel(Sr^, Sr^.Hol[i]);
  End Else Begin
    Sr^.A[i].kulcs:=K;
    Sullyeszt(Sr^, Sr^.Eszam, Sr^.Hol[i]);
  End;
End;
```

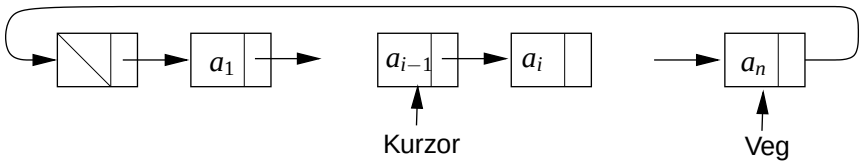
MODOSIT futási ideje: $T_{lr}(n) = O(\lg n)$.

16.5. Lista

Értékhalmaz: $Lista = \{\langle a_1,\ldots,a_{i-1}\rangle\langle a_i,\ldots,a_n\rangle : a_i \in E, i = 1,\ldots,n\}$
Műveletek:

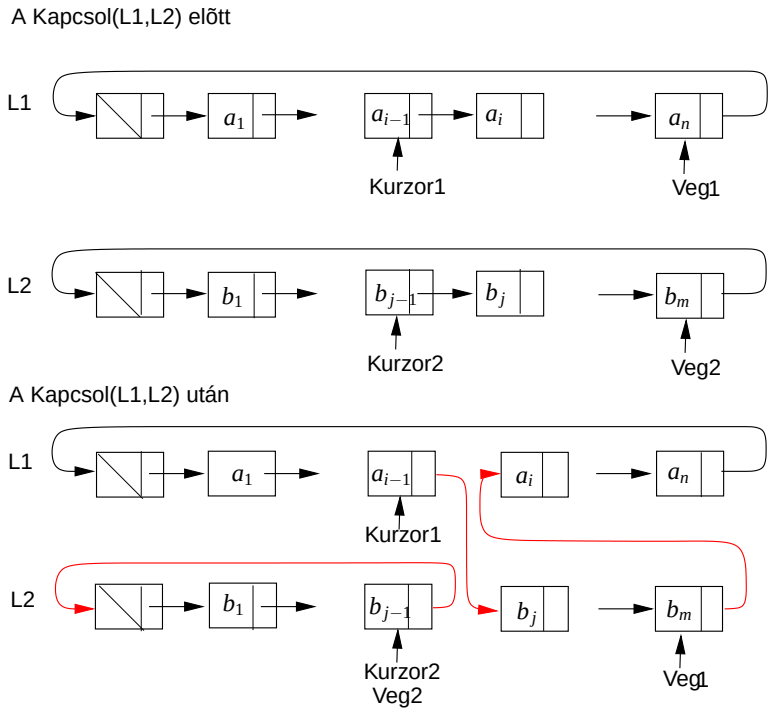
$$L, L1, L2 : Lista, x : E$$

$\{Igaz\}$	$Letesit(L)$	$\{L = \langle\rangle\langle\rangle\}$
$\{L = L\}$	$Megszuntet(L)$	$\{Hamis\}$
$\{L = L\}$	$Uresit(L)$	$\{L = \langle\rangle\langle\rangle\}$
$\{L = L\}$	$Urese(L)$	$\{Urese = (Pre(L) = \langle\rangle\langle\rangle\}$
$\{L = L\}$	$Elejen(L)$	$\{= Pre(L) = \langle\rangle\langle a_1,\ldots,a_n\rangle\}$
$\{L = L\}$	$Vegen(L)$	$\{Vegen = L = \langle a_1,\ldots,a_n\rangle\langle\rangle\}$
$\{L = \langle a_1,\ldots,a_{i-1}\rangle\langle a_i,\ldots,a_n\rangle\}$	$Elejere(L)$	$\{L = \langle\rangle\langle a_1,\ldots,a_n\rangle\}$
$\{L = \langle a_1,\ldots,a_{i-1}\rangle\langle a_i,\ldots,a_n\rangle\}$	$Vegere(L)$	$\{L = \langle a_1,\ldots,a_n\rangle\langle\rangle\}$
$\{L = \langle a_1,\ldots,a_{i-1}\rangle\langle a_i,\ldots,a_n\rangle \wedge i \le n\}$	$Tovabb(L)$	$\{L = \langle a_1,\ldots,a_{i-1},a_i\rangle \langle a_{i+1},\ldots,a_n\rangle\}$
$\{L = \langle a_1,\ldots,a_{i-1}\rangle\langle a_i,\ldots,a_n\rangle \wedge i \le n\}$	$Kiolvas(L,x)$	$\{x = a_i \wedge L = Pre(L)\}$
$\{L = \langle a_1,\ldots,a_{i-1}\rangle\langle a_i,a_{i+1}\ldots,a_n\rangle \wedge i \le n\}$	$Modosit(L,x)$	$\{L = \langle a_1,\ldots,a_{i-1}\rangle \langle x,a_{i+1},\ldots,a_n\rangle\}$
$\{L = \langle a_1,\ldots,a_{i-1}\rangle\langle a_i,\ldots,a_n\rangle\}$	$Bovit(L,x)$	$\{L = \langle a_1,\ldots,a_{i-1}\rangle \langle x,a_i,\ldots,a_n\rangle\}$
$\{L = \langle a_1,\ldots,a_{i-1}\rangle\langle a_i,a_{i+1},\ldots,a_n\rangle \wedge i \le n\}$	$Torol(L)$	$\{L = \langle a_1,\ldots,a_{i-1}\rangle \langle a_{i+1},\ldots,a_n\rangle\}$
$\{L1 = \langle\alpha_1\rangle\langle\beta_1\rangle, L2 = \langle\alpha_2\rangle\langle\beta_2\rangle\}$	$Kapcsol(L1,L2)$	$\{L1 = \langle\alpha_1\rangle\langle\beta_2\beta_1\rangle, L2 = \langle\alpha_2\rangle\langle\rangle\}$



8. ábra. Az $S = \langle a_1,\ldots,a_{i-1}\rangle\langle a_i,\ldots,a_n\rangle$ lista érték ábrázolása egyirányú körláccal.

Minden művelet futási ideje legrosszabb esetben is $O(1)$.



9. ábra. Két lista kapcsolása.

16.6. Halmaz

Értékhalmaz: $Halmaz = \{ H = \{a_1, \ldots, a_n\} : H \subseteq E \}$
Műveletek:

$$H : Halmaz, x : E, I : Iterator$$

$$\begin{array}{rcl}
\{Igaz\} & Letesit(H) & \{H = \emptyset\} \\
\{H = H\} & Megszuntet(H) & \{Hamis\} \\
\{H = H\} & Uresit(H) & \{H = \emptyset\} \\
\{H = H\} & Eleme(H,x) & \{Eleme = x \in Pre(H)\} \\
\{H = \{a_1, \ldots, a_n\}\} & Elemszam(H) & \{Elemszam = n\} \\
\{H = H\} & Bovit(H,x) & \{H = Pre(H) \cup \{x\}\} \\
\{H = H\} & Torol(H,x) & \{H = Pre(H) - \{x\}\} \\
\{H = H\} & IterKezd(H,I) & \{\} \\
\{I = I\} & IterAd(I,x) & \{\} \\
\{I = I\} & IterVege(I) & \{\}
\end{array}$$

```
Forall x in H Do
  M(x);
```

$\equiv$

```
IterKezd(H,I);
While Not IterVege(I) Do Begin
  IterAd(I,x);
  M(x);
End;
```

16.7. Halmaz megvalósítása bitvektorral

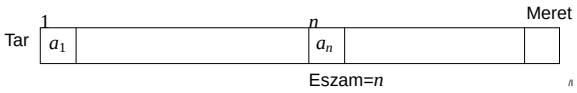
Feltétel: az Elemtipus felsorolás típus intervalluma lehet csak, Pl 1..MaxN.

```
Const
 MaxN=???;
Type
  Halmaz=Record
           T:Array[1..MaxN] of Boolean;
           Eszam:0..MaxN;
         End;
Var
  H:Halmaz;
```

$x \in H \Leftrightarrow$ `H.T[x]`
Az ELEME, BOVIT, TOROL futási ideje $\Theta(1)$.
Az iteráció futási ideje $O(MaxN)$. A megvalósítása:

```
 For x:=1 To MaxN Do
   If H.T[x] Then M(x)
```

16.8. Halmaz megvalósítása tömbbel



10. ábra. Halmaz ábrázolása tömbben.

A Műveletek futási ideje n-elemű halmaz esetén:
ELEME: $T_{lr}(n) = O(n)$, $T_a(n) = O(n)$.
BOVIT: $T_{lr}(n) = O(1)$, $T_a(n) = O(1)$.
TOROL: $T_{lr}(n) = O(n)$, $T_a(n) = O(n)$.
Az iteráció futási ideje $O(n)$.
Rendezett tömb esetén ELEME: $T_{lr}(n) = O(\lg n)$.

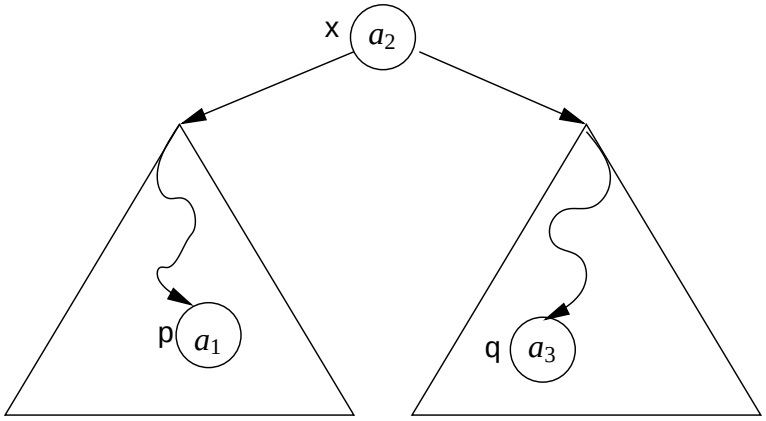
16.9. Halmaz megvalósítása láccal

11. ábra. Halmaz ábrázolása láncban.

A Műveletek futási ideje n-elemű halmaz esetén:
ELEME: $T_{lr}(n) = O(n)$, $T_a(n) = O(n)$.
BOVIT: $T_{lr}(n) = O(1)$, $T_a(n) = O(1)$.
TOROL: $T_{lr}(n) = O(n)$, $T_a(n) = O(n)$.
Az iteráció futási ideje $O(n)$.

16.10. Halmaz megvalósítása bináris keresőfával

A $F = (M, R, Adat)$ absztrakt adatszerkezetet *bináris keresőfának* nevezzük, ha
1. F bináris fa, $R = \{bal, jobb\}, bal, jobb : M \to M$
2. $Adat : M \to Elemtip$ és $Elemtip$-on értelmezett egy $\leq$ lineáris rendezési reláció,
3. $(\forall x \in M)(\forall p \in F_{bal(x)})(\forall q \in F_{jobb(x)})(Adat(p) \leq Adat(x) \leq Adat(q))$



12. ábra. Keresőfa tulajdonság: $Adat(p) = a_1 \leq Adat(x) = a_2 \leq a_3 = Adat(q)$

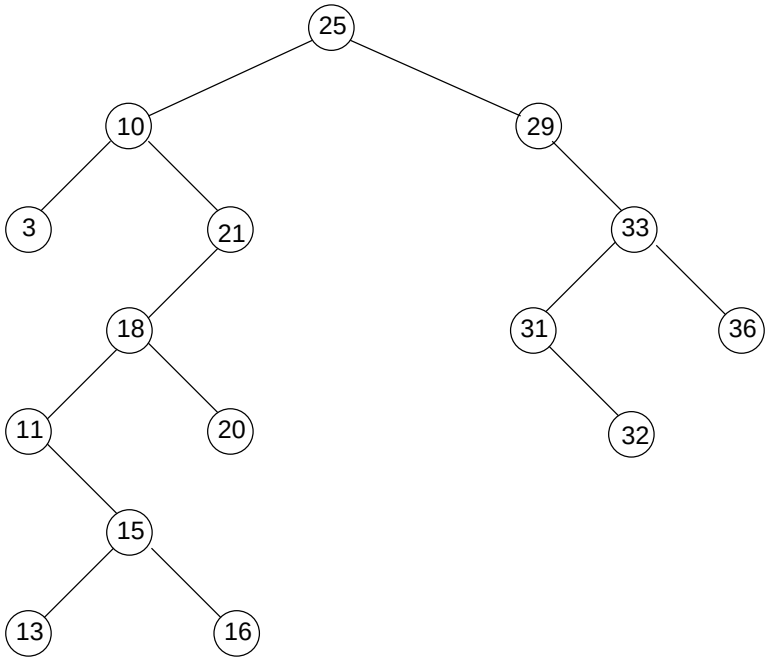
```
Procedure Inorder(F:BinFA; M:Muveltip);
 Begin
   If F<>Nil Then Begin
     Inorder(F^.bal);
     M(F^.adat);
     Inorder(F^.jobb);
  End
End{Inorder};
```

Állítás. F akkor és csak akkor bináris keresőfa, ha inorder bejárása rendezett sorozatot ad.
A BINKERFAKERES függvényeljárás egy nyilvánvaló megoldása a fában keresés feladatnak.

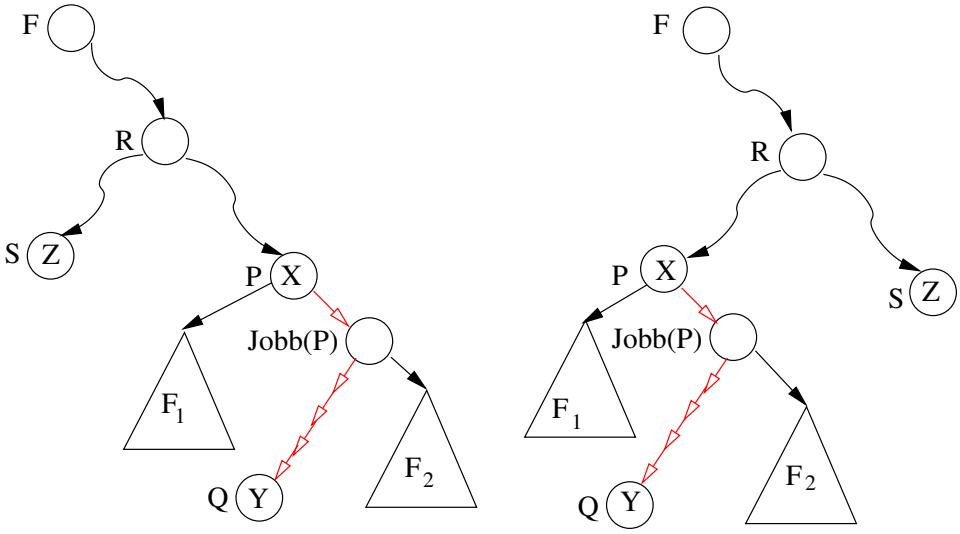
```
Function BinKerFaKeres(a:Adat; F:BinFA):BinFa;
 Begin
 While (F<>Nil) And (a<>F^.adat) Do
  If a<F^.adat Then
    F:=F^.bal
  Else
    F:=F^.jobb;
  BinKerFaKeres:=F;
End;
```

Az F bináris keresőfában a P pont követője az a Q pont, amelynek adata az $Adat(P)$-nél nagyobbak közül a legkisebb (ha van). A követő két esete:

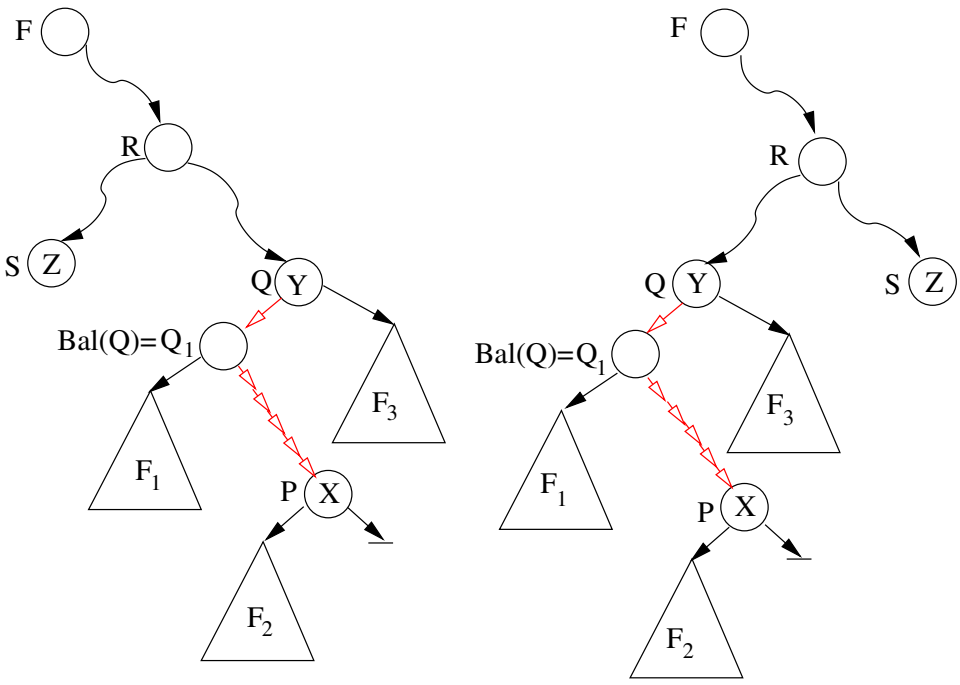
```
Unit BinFaM;
 Interface
  Type
    Kulcstip = ???        ;(* a rendezési mező típusa  *)
    Adattip  = ???        ;(* az adatmező típusa       *)
```



13. ábra. Példa bináris keresőfa



14. ábra. Pont követőjének 1. esete: $Jobb(P) \neq \perp$, $Kovet(P) = Min(Jobb(P))$



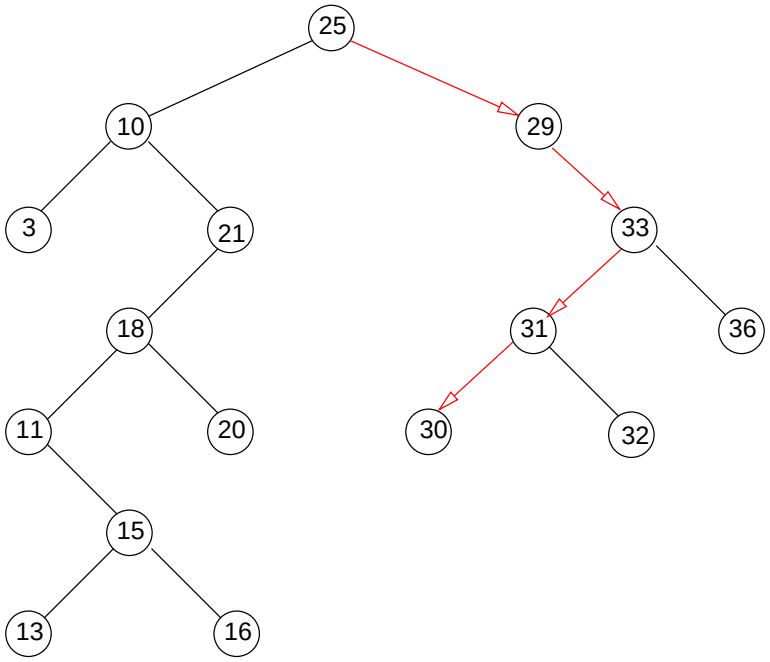
15. ábra. Pont követőjének 2. esete: $Jobb(P) = \perp, Kovet(P) = Q : Max(Bal(Q)) = P$

```
    Elemtip = Record
                kulcs: Kulcstip;
                adat : Adattip
              End;
    BinFa = ^BinFaPont;
    BinFaPont = Record
                  adat      : Elemtip;
                  bal, jobb : BinFa;
                  apa       : BinFa;
                End;
{ műveletek: }
    Function Mini(F : BinFa): BinFa;
    Function Maxi(F : BinFa): BinFa;
    Function Kovet(F : BinFa; p:BinFa): BinFa;
    Function Eloz(F : BinFa; p:BinFa): BinFa;

Implementation
  Function Mini(F : BinFa): BinFa;
    Var p:BinFA;
    Begin
      p:=F;
      While p^.bal <> Nil Do p:= p^.bal;
      Mini:=p;
    End;

  Function Maxi(F : BinFa): BinFa;
    Var p:BinFA;
    Begin
      p:=F;
      While p^.jobb <> Nil Do p:= p^.jobb;
      Maxi:=p;
```

```
  End;

Function Kovet(F : BinFa;p:BinFa): BinFa;
  Var q0,q:BinFA;
  Begin
    If p^.jobb <> Nil Then Begin
      q0:= p^.jobb;
      While q0^.bal <> Nil Do q0:= q0^.bal;
    End Else Begin
      q:=F; q0:=Nil;
      While q <> p Do
        If p^.adat.kulcs < q^.adat.kulcs Then Begin
          q0:= q; q:= q^.bal
        End Else
          q:= q^.jobb
    End;
    Kovet:= q0;
  End;

Function Eloz(F : BinFa;p:BinFa): BinFa;
  Var q0,q:BinFA;
  Begin
    If p^.bal <> Nil Then Begin
      q0:= p^.bal;
      While q0^.jobb <> Nil Do q0:= q0^.jobb;
    End Else Begin
      q:=F; q0:=Nil;
      While q <> p Do
        If p^.adat.kulcs > q^.adat.kulcs Then Begin
          q0:= q; q:= q^.jobb
        End Else
          q:= q^.bal
    End;
    Eloz:= q0;
  End;

Function Koveto(F : BinFa; p:BinFa): BinFa;
{Ha van apa pointer}
  Begin
    If p^.jobb <> Nil Then Begin
      p:= p^.jobb;
      While p^.bal <> Nil Do
        p:=p^.bal;
    End Else Begin
      While (p^.apa<>Nil) And (p^.apa^.bal<>p) Do
        p:=p^.apa;
      p:=p^.apa;
    End;
    Koveto:=p;
  End{Koveto};

Function Elozo(F : BinFa; p:BinFa): BinFa;
{Ha van apa pointer}
  Begin
    If p^.bal <> Nil Then Begin
      p:= p^.bal;
```

```
        While p^.jobb <> Nil Do
          p:=p^.jobb;
      End Else Begin
        While (p^.apa<>Nil) And (p^.apa^.jobb<>p) Do
          p:=p^.apa;
        p:=p^.apa;
      End;
      Elozo:=p;
    End{Elozo};

End (* BinFa *) .
```

Bináris keresőfa bővítése: az új pont beillesztése a keresőút végére.
A Q pont a K kulcshoz tartozó keresőút (K-keresőút) vége, ha $K < Adat(Q)$ esetben $Bal(Q) = \perp$, illetve $K \geq Adat(Q)$ esetben

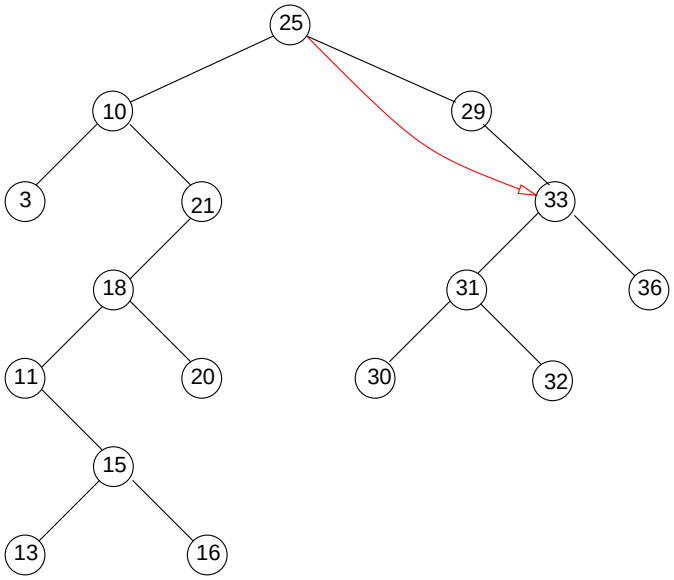


16. ábra. A példa bináris keresőfa bővítése a 30 kulccsal.

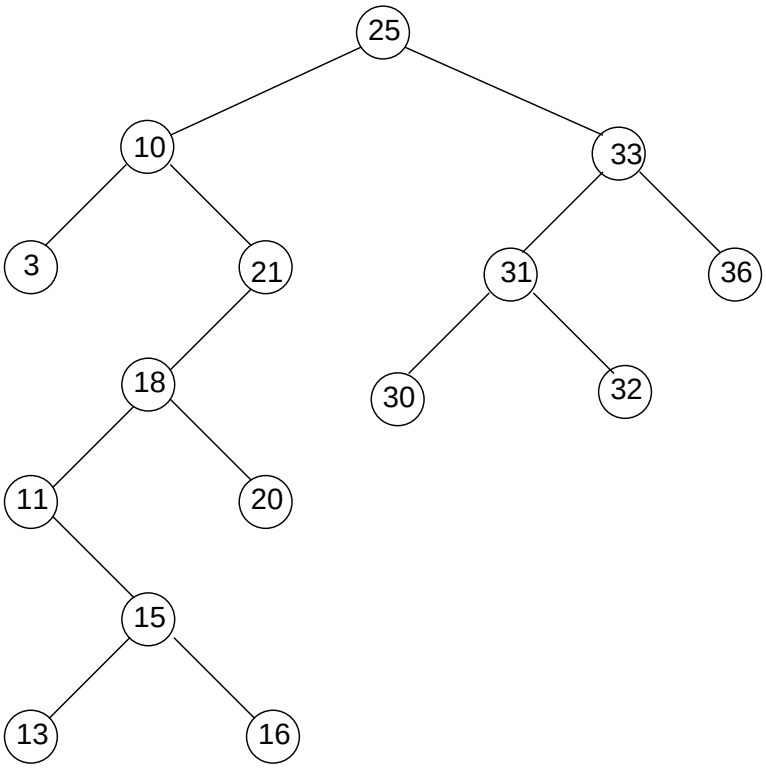
$Jobb(Q) = \perp$, továbbá az F gyökértől Q-ig vezető úton minden R pontra teljesül, hogy $Q \in F_{Bal(R)}$, ha $K < Adat(R)$ és $Q \in F_{Jobb(R)}$, ha $K \geq Adat(R)$.

Törlés bináris keresőfából.

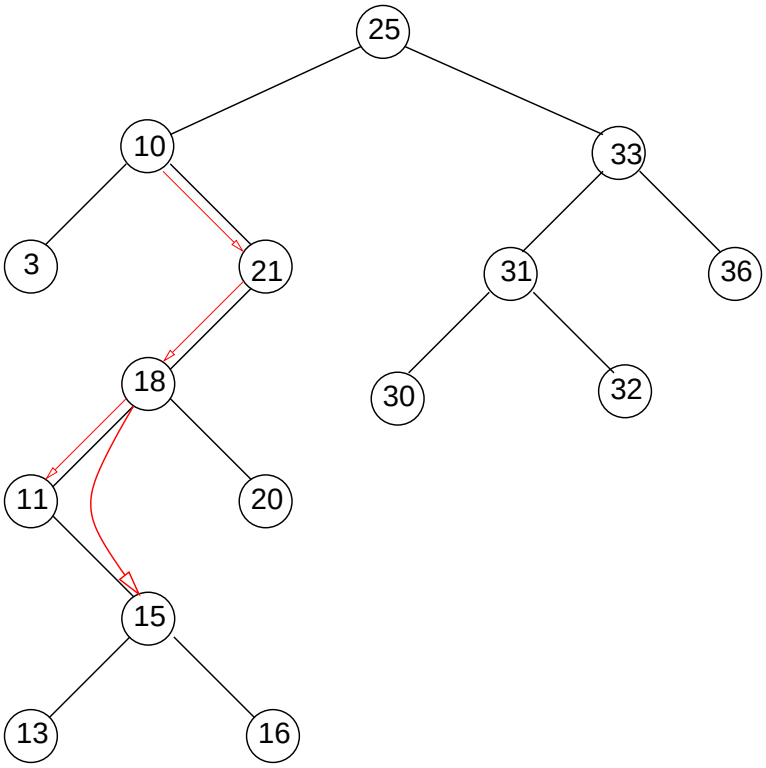
```
Unit BinKerFa ; { Binaris keresofa }
 Interface
  Type
    Kulcstip = ???         ;(* a rendezési mező tíusa  *)
    Adattip  = ???         ;(* az adatmező típusa       *)
    Elemtip = Record
                kulcs: Kulcstip;
                adat : Adattip
              End;
    BinFa = ^BinFaPont;
    BinFaPont = Record
                  adat : Elemtip;
                  bal, jobb : BinFa
                End;
```



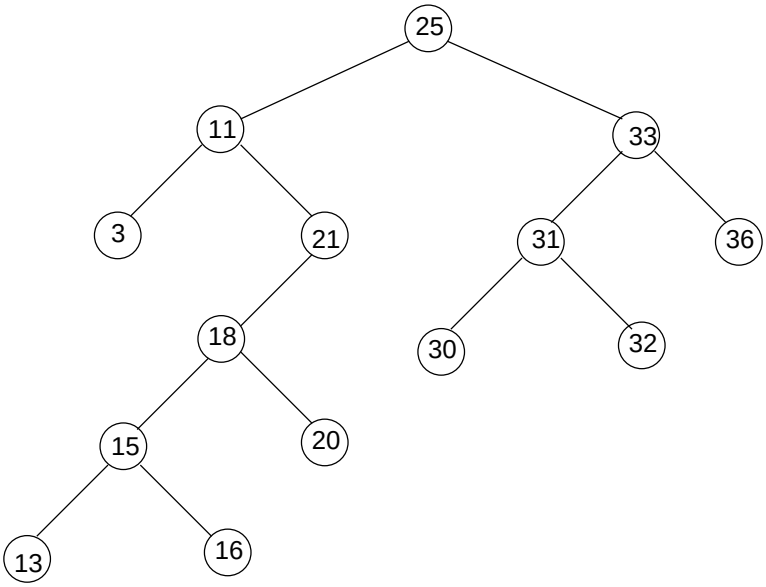
17. ábra. A 29 kulcs törlése: egyszerű eset, a törlendő pontnak nincs bal fia.



18. ábra.



19. ábra. A 10 kulcs törlése: a törlendő pontnak két fia van. A törlendő pont helyettesítése a követőjével és a követő tényleges törlése.



20. ábra.

```
  { műveletek: }
    Function Keres(F : BinFa;
                   K : Kulcstip) : BinFa;
    Procedure Bovit0(Var F : BinFa;       (* a bővítendő fa          *)
                         X : Elemtip);    (* a bővítendő elem        *)
    Procedure Bovit(Var F : BinFa;        (* a bővítendő fa          *)
                        X : Elemtip;      (* a bővítendő elem        *)
                    Var Tobb : Boolean ); (* lehet többszörös elem ? *)
    Procedure Torol(Var F : BinFa;
                        K : Kulcstip;     (* a törlendő pont kulcsa  *)
                    Var Volt : Boolean  );(* volt ilyen pont ?       *)

Implementation
  Function Keres(F : BinFa; K : Kulcstip) : BinFa;
    Begin
      While (F <> Nil) Do
        If K < F^.adat.kulcs Then
          F := F^.bal
        Else If K > F^.adat.kulcs Then
          F := F^.jobb
        Else
          Break;

      Keres:= F;
    End (* Keres *) ;

  Procedure Bovit0(Var F : BinFa;
                       X : Elemtip);
  {Rekurzív bővítés, többszörös elem megengedett és nem ellenőrzött}
    Begin
      If F = Nil Then Begin
        New(F);
        F^.adat:= X;
        F^.bal:= Nil; F^.jobb:= Nil;
      End Else Begin
        If X.kulcs < F^.adat.kulcs Then
          Bovit0(F^.bal, X)
        Else
          Bovit0(F^.jobb, X)
      End;
    End (* Bovit0 *) ;

  Procedure Bovit(Var F : BinFa;
                      X : Elemtip;
                  Var Tobb : Boolean );
    Var
      P, Apa, Ujp : BinFa;
      Nincs : Boolean;
    Begin
      New(Ujp); Nincs := True;
      With Ujp^ Do Begin
        adat := X ;
        bal := Nil; jobb := Nil
      End;
      If F = Nil Then
        F := Ujp
```

```
      Else Begin
        P := F;
        While (P <> Nil) Do Begin
          Apa := P;
          If X.kulcs < P^.adat.kulcs Then
            P := P^.bal
          Else
            If X.kulcs > P^.adat.kulcs Then
              P := P^.jobb
          Else Begin{P^.adat.kulcs=K}
            Nincs:= False;
            If Not Tobb Then Begin
              Tobb:= True; Dispose(Ujp); Exit
            End Else
              p:= p^.jobb
          End
        End{while};
        If X.kulcs < Apa^.adat.kulcs Then
          Apa^.bal := Ujp
        Else
          Apa^.jobb := Ujp
      End{else:F<>Nil};
      Tobb := Not Nincs
    End (* Bovit *) ;

Procedure Torol(Var F : BinFa; K : Kulcstip; Var Volt : Boolean);
Var
  P, Apa, T : BinFa;  Tovabb : Boolean;
Begin
      P := F; Apa := P; Tovabb := True;
      (* a K kulcsu pont keresese *)
      While (P <> Nil) And Tovabb Do
        If K < P^.adat.kulcs Then Begin
          Apa := P; P := P^.bal
        End Else If K > P^.adat.kulcs Then Begin
          Apa := P; P := P^.jobb
        End Else
          Tovabb := False;
      {end while}
      Volt := Not Tovabb;
      If Volt Then Begin
      (* P^.adat.kulcs=K, Apa a P pont apja, ha P=F akkor P=Apa *)
        T := P; (* a törlendő pont T *)
        If P^.bal = Nil Then      {P-nek nins bal fia }
          If P = Apa^.bal Then
            Apa^.bal := P^.jobb
          Else If P = Apa^.jobb Then
            Apa^.jobb := P^.jobb
          Else F := P^.jobb

        Else If P^.jobb = Nil Then{P-nek nics jobb fia }
          If P = Apa^.bal Then
            Apa^.bal := P^.bal
          Else
            If P = Apa^.jobb Then
              Apa^.jobb := P^.bal
```

```
          Else
            F := P^.bal
        Else Begin                  {P-nek ket fia van }
           T := P^.jobb; Apa := T;
           While T^.bal <> Nil Do Begin{legyen T a P kovetoje}
             Apa := T; T := T^.bal
           End;
           (* P helyebe T kerül *)
           P^.adat := T^.adat;
           If T = Apa Then
             P^.jobb := T^.jobb
           Else
             Apa^.bal := T^.jobb;
         End;
        Dispose(T)
      End
    End (* Torol *) ;
End (* BinKerFa *) .
```

A ELEME, BOVIT és TOROL műveletek futási ideje $O(h(F))$.
A Műveletek futási ideje n-elemű halmaz esetén:
ELEME: $T_{lr}(n) = O(n)$, $T_a(n) = O(\lg n)$.
BOVIT: $T_{lr}(n) = O(n)$, $T_a(n) = O(\lg n)$.
TOROL: $T_{lr}(n) = O(n)$, $T_a(n) = O(\lg n)$.