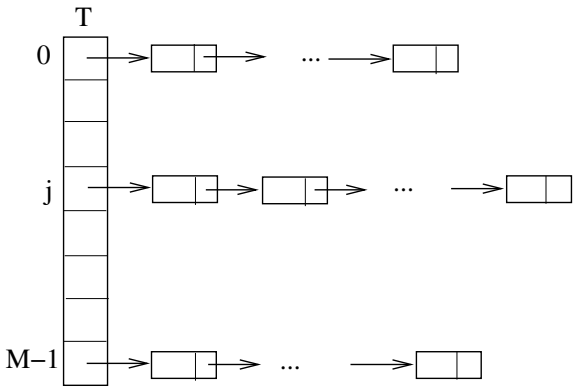
17. Tördelőtáblák (Hasítótáblák)

Legyen U az Elemtip, univerzum,
$H = \{a_1, \cdots, a_n\} \subseteq U$
Vegyünk egy T:Array[0..M-1] of Elemtip tömbböt, amelyben a H halmazt tárolni akarjuk.
Válasszunk egy
$h : U \to \{0, \cdots, M-1\}$
függvényt, amely minden a_i halmazelemre megadja azt a táblázat indexet, ahol az elemet tárolni akrajuk, $T[h(a_i)] := a_i$.
Ha $a_i \neq a_j$ és $h(a_i) = h(a_j)$ **ütközés**.

17.1. Ütközésfeloldás láncolással

Legyen T:Array[0..M-1] of Lanc, és legyen $T[j]$ a $\{x : x \in H \wedge h(x) = j\}$ elemek lánca.



1. ábra. Adatszerkezet ütközésfelodás láncolással módszerhez.

```
 Const
   Meret  = ???           ;(* a tördelőtábla mérete           *)
Type
  Kulcstip= ???           ;(* a halmaz elemeinek kulcstípusa  *)
  Adattip = ???           ;(* a halmaz elemeinek adattípusa   *)
  Elemtip = Record
              kulcs : Kulcstip;
              adat  : Adattip
            End;
  Index = 0 .. Meret;
  Lanc = ^Cella;
  Cella = Record
            adat : Elemtip;
            csat : Lanc
          End;
  Tabla = Array[Index] Of Lanc;

Function  H(K:Kulcstip):Index; { a tördelőfüggvény }
 Begin ??? End;

Procedure Letesit(Var T:Tabla); Var
  i : Index;
Begin{Letesit}
  For i := 0 To Meret Do
    H.T[i]:=Nil;
```

```
End{Letesit};

Function Keres(T:Tabla; K:Kulcstip):Lanc; Var
  i:Index; P:Lanc;
Begin
  P:=T[H[K]];
  While (P<>Nil) And (P^.adat.kulcs<>K) Do
    P:=P^.csat;
  Keres := P;
End{Keres};

Procedure Bovit(Var T:Tabla;
                    X:Elemtip);
Var
  P : Lanc;
Begin{Bovit}
  i:=H(X.kulcs);
  New(P);
  P^.adat:=X;
  P^.csat:=T[i];
  T[i]:=P;
End{Bovit};

Procedure Torol(Var T:Tabla;
                    K:Kulcstip);
Var
  i:Index; P,P0 : Lanc;
Begin
  i:=H(K);
  P:=T[i];
  P0:=P;
  While (P<>Nil) And (K<>P^.adat.kulcs) Do Begin
    P0:=P; P:=P^.csat;
  End;
  If P<>Nil Then Begin
    If P=P0 Then
     T[i]:=P^.csat
    Else
      P0^.csat := P^.csat;
    Dispose(P)
  End
End{Torol};
```

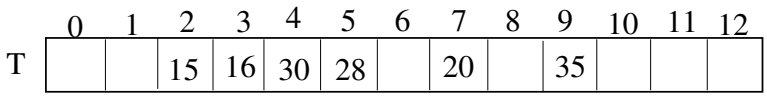
17.2. Ütközésfeloldás nyílt címzéssel

Pótvizsgálat (próbasorozat)
$h : U \times \{0,\cdots,m-1\} \to \{0,\cdots,m-1\}$
$\langle h(k,0),\cdots,h(k,m-1)\rangle$
$\langle 0,\cdots,m-1\rangle$ egy permutációja minden k kulcsra.
A k kulcsú elem helye a táblázatban az a $j = h(k,i)$, amelyre

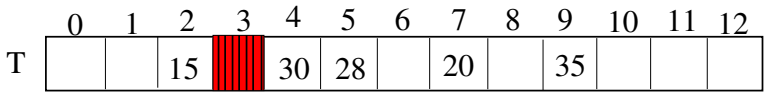
$$Min\{i : T[h(k,i)] = Szabad\}$$

Ekkor a próbák száma $i+1$.
Például, ha $U = Integer$, $h(x,i) = (x+i) mod m$. Ezt a tördelőfüggvényt alkalmazva bővítsük a kezdetben üres táblát egymás után a 15, 30, 16, 35, 20, és 28 elemekkel. Majd töröljük a táblából a 16-os elemet. Ha ezt követően a 28 elemet keressük a táblában,

akkor a $\langle 2,3,4,5,\ldots\rangle$ próbasorozat kell alkalmazni, de a keresés nem állhat meg a 3 indexű törölt táblaelemnél. Tehát a táblában a törölt helyeket nem tekinthetjük üresnek. Választanunk kell egy *Ures* és egy ettől különböző *Torolt* konstans kulcs értéket az üres, illetve a törölt táblaelemek jelölésére.



2. ábra. Tördelőtábla a 15, 30, 16, 35, 20, és 28 elemekkel való bővítés után.



3. ábra. Tördelőtábla a 16 elem törlése után.

```
{Tördelőtábla, ütközésfeloldás nyílt címzéssel}
Const
  M      = ??? ;{ a tördelötábla mérete}
  Ures   = ??? ;{ Kulcstípusú konstans, az üres hely jele }
  Torolt = ??? ;{ Kulcstípusú konstans, a törölt hely jele }
Type
  Kulcstip=??? ;{ a halmaz elemeinek kulcstípusa}
  Adattip =??? ;{ a halmaz elemeinek adattípusa }
  Elemtip = Record
              kulcs : Kulcstip;
              adat  : Adattip
            End;
  Index = 0 .. M-1;
  Tabla = Array[Index] Of Elemtip;

Function H(K:Kulcstip; i:Index) : Index; { a tördelőfüggvény }
Begin {???} End;

Function TKeres(Const T : Tabla;
                      K : Kulcstip) : Word;
Var i,     {a próbaszám}
  j : Word;{táblaindex}
Begin
  i:=0;
  Repeat
    j:=H(K,i);
    Inc(i);
  Until (i=M) Or (K=T[j].kulcs) Or
        (T[j].kulcs=Ures);

  If (K=T[j].kulcs) Then
    TKeres:=j
  Else
    TKeres:=M
End{TKeres};
```

```
Procedure TBovit(Var T:Tabla; X:Elemtip); Var
  i, j : Word;  K:Kulcstip;
Begin{Bovit}
  K:=X.kulcs;
  i := 0;
  Repeat
    j:=H(K,i);
    Inc(i);
  Until (i=M) Or (T[j].kulcs=Torolt) Or
                 (T[j].kulcs=Ures);

  If (T[j].kulcs<>Torolt) And
     (T[j].kulcs<>Ures) Then Begin
    Exit;       {nincs szabad hely a táblában}
  End;
  T[j].kulcs:=K;
  T[j].adat:=X.adat
End {Bovit };
```

```
Procedure TTorol(Var T : Tabla; K : Kulcstip); Var
  i, j : Word;
Begin {Torol} ;
  i := 0;
  Repeat
    j:=H(K,i);
    Inc(i);
  Until (i=M) Or (T[j].kulcs=K) Or
                 (T[j].kulcs=Ures);

  If (T[j].kulcs<>K) Then Begin
    Exit   {nincs K kulcsú elem a táblában}
  End;
  T[j].kulcs:=Torolt
End{Torol};
```

17.3. Pótvizsgálatok (próbasorozatok).

Lineáris pótvizsgálat
$h(k,i) = (h_0(k)+i) \bmod m$,
$h_0 : U \to \{0,\cdots,m-1\}$

Négyzetes pótvizsgálat
$h(k,i) = (h_0(k)+c_1 \cdot i + c_2 \cdot i^2) \bmod m$,
$h_0 : U \to \{0,\cdots,m-1\}$

Dupla tördelés
$h(k,i) = (h_1(k)+i \cdot h_2(k)) \bmod m$,
$h_1, h_2 : U \to \{0,\cdots,m-1\}$
A $h_2(k)$ értékeknek relatív prímnek kell lenni a táblázat m méretéhez. Pl. $h_1(k) = k \bmod m$ és $h_2(k) = 1 + (k \bmod m')$, ahol $m' = m-1$.

17.4. Tördelőfüggvények

Az osztás módszere: $h(k) = k \bmod m$
A szorzás módszere: $h(k) = \lfloor m(k \cdot A mod\, 1) \rfloor$
ahol $k \cdot A mod\, 1 = k \cdot A - \lfloor k \cdot A \rfloor$

$0 < A < 1$, pl. $A = (\sqrt{5}-1)/2 = 0.6180\cdots$

17.5. Tördelőtáblák hatékonysági elemzése

Legyen $h : U \to \{0,\ldots,m-1\}$ a tördelőfüggvény.
$\alpha = \frac{n}{m}$ a tábla *telítettségi tényezője*.
Egyenletességi feltétel:

$$\sum_{k:h(k)=j} Pr(k) = \frac{1}{m} \quad (j = 0,\cdots,m-1)$$

Feltesszük, hogy $h(k)$ olcsó, azaz $\Theta(1)$ időben kiszámítható.

Ütközésfeloldás láncolással
A BOVIT futási ideje legrossabb esetben is $O(1)$.
A KERES és TOROL futási ideje legrossabb esetben $O(n)$.
(Minden elem egyetlen láncban van és a keresés szekvenciális.)
Ha a $T[j]$ lánc hossza n_j, akkor

$$n = n_0 + n_1 + \cdots + n_{m-1}$$

és n_j várható értéke $n/m = \alpha$. Így egy k kulcsú elem keresésének ideje lineárisan függ a $T[h(k)]$ lánc $n_{h(k)}$ hosszától. A $T[h(k)]$ lánc azon elemeinek számát tekintsük, amelyekkel a keresés során összehasonlítódik a keresett k kulcs. Két esetet kell tekintenünk, az első az, amikor a keresés sikeres, a másik pedig a sikertelen keresés.

17.1. tétel. *Ha teljesül az egyenletességi feltétel, akkor láncolással történő ütközésfeloldás esetén a sikertelen keresés átlagos ideje* $\Theta(1+\alpha)$.

Bizonyítás. Az egyenletességi feltétel miatt bármely k kulcsra, ami még nincs a táblánan, egyforma valószínűséggel adódik bármely $j = h(k)$ táblaindex. Ezért egy k kulcs sikertelen keresésének átlagos ideje megegyezik annak átlagos idejével, hogy a $T[h(k)]$ láncot végigkeressük. Ennek a láncnak az átlagos hossza α, tehát $h(k)$ kiszámításával együtt az átlagos futási idő $\Theta(1+\alpha)$. ■

Sikertelen keresés esetén annak valószínűsége, hogy egy adott láncban keresünk, azonos a lánc hossszával.

17.2. tétel. *Ha teljesül az egyenletességi feltétel, akkor láncolással történő ütközésfeloldás esetén a sikeres keresés átlagos ideje* $\Theta(1+\alpha)$.

Bizonyítás. Az x elem sikeres keresése esetén megvizsgált elemek várható száma eggyel nagyobb, mint azoknak az elemeknek a várható száma, amelyek megelőzik x-et az x-et tartalmazó láncban. Az x-et megelőző elemeket x beszúrása után szúrtuk be, mivel új elemet mindig a lánc elejéhez illesztünk. Tehát átlagolni kell a táblázat n elemére az 1+azon elemek várható száma értékeket, amelyeket x után adtunk x láncához. Legyen x_i a táblázatba i-edikként beszúrt elem és legyen $k_i = x_i.kulcs$. Az egyenletességi feltétel miatt $Pr\{h(k_i) = h(k_j)\} = 1/m$. Ezért a sikeres keresés során vizsgált elemek számának várható értéke

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}\frac{1}{m}\right) &= 1+\frac{1}{nm}\sum_{i=1}^{n}(n-i) \\
&= 1+\frac{1}{nm}\left(\sum_{i=1}^{n}n-\sum_{i=1}^{n}i\right) \\
&= 1+\frac{1}{nm}\left(n^2-\frac{n(n+1)}{2}\right) \\
&= 1+\frac{n-1}{2m} \\
&= 1+\frac{\alpha}{2}-\frac{\alpha}{2n}\,.
\end{aligned}$$

Tehát a sikeres keresés futási ideje, beszámítva $h(k)$ kiszámítását is

$$T_a(n) = \Theta(2+\frac{\alpha}{2}-\frac{\alpha}{2n}) = \Theta(1+\alpha)$$

A nyílt címzés hatékonyságának elemzése

A legjobb eset

KERES, BÖVIT, TOROL : $O(1)$

A legrosszabb eset

BÖVIT: $O(n)$

KERES, TOROL : $O(n)$ **Átlagos eset**

17.3. tétel. *Nyílt címzés esetén a sikertelen keresés során a próbák számának várható értéke* $\leq \frac{1}{1-\alpha}$.

Bizonyítás. $p_i = Pr\{$pontosan i táblaelem foglalt$\}$ azaz, i a legkisebb index, amelyre $T[h(k,i)] = Szabad\}$
Ekkor a próbák számának várható értéke:

$$1 + \sum_{i=0}^{n} i\, p_i$$

Jelölje q_i annak valószínűségét, hogy legfeljebb i próbát kell végezni. Tehát

$$1 + \sum_{i=0}^{n} i\, p_i = \sum_{i=0}^{n} i\,(q_i - q_{i-1}) = \sum_{i=0}^{n} q_i$$

$q_1 = \frac{n}{m}$
$q_2 = \frac{n}{m}\frac{n-1}{m-1}$

$q_i = \frac{n}{m}\frac{n-1}{m-1}\cdots\frac{n+1-i}{m+1-i} \leq (\frac{n}{m})^i = \alpha^i$. Tehát

$$1 + \sum_{i=0}^{n} i\, p_i \leq \sum_{i=0}^{n} \alpha^i \leq \sum_{i=0}^{\infty} \alpha^i \leq \frac{1}{1-\alpha}$$

A BOVIT algoritmus átlagos futási ideje : $O(\frac{1}{1-\alpha})$. ■

17.4. tétel. *Nyílt címzés esetén a sikeres keresés során a próbák számának várható értéke* $\leq \frac{1}{\alpha}\ln\frac{1}{(1-\alpha)}$.

Bizonyítás. Sikeres keresés ugyanazon próbasorozatot hajtja végre, mint amikor az elemet beraktuk a táblázatba.
Ha a k kulcsú elem i-edikként került a táblázatba, akkor a próbák száma $\frac{m}{m-i}$. Tehát a próbák számának várható értéke:

$$\frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i} = \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} = \frac{1}{\alpha}(H_m - H_{m-n}),$$

ahol $H_i = \sum_{j=1}^{i}\frac{1}{j}$ az i. harmonikus szám. $1/x$ csökkenő, így

$$\begin{aligned}\frac{1}{\alpha}(H_m - H_{m-n}) &= \frac{1}{\alpha}\sum_{k=m-n+1}^{m} 1/k \leq \\ \frac{1}{\alpha}\int_{m-n}^{m}(1/x)dx &= \frac{1}{\alpha}\ln\frac{m}{m-n} = \frac{1}{\alpha}\ln\frac{1}{1-\alpha}\end{aligned}$$

■

Tehát a Torol algoritmus átlagos futási ideje (= sikeres kerersés) $O(\frac{1}{\alpha}ln\frac{1}{1-\alpha})$

Pl. ha $\alpha = 0.5$ akkor 1.387, ha $\alpha = 0.9$ akkor 2.559 próba kell átlagosan a törléshez.

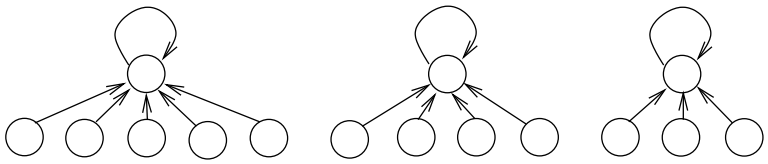
17.6. Az UnioHolvan adattípus megvalósítása

Az UnioHolvan absztrakt adattípus.
Értékhalmaz: $UnioHolvan = \{\{H_1,\ldots,H_k\} : H_i \subseteq E, i = 1,\ldots,k, i \neq j \Rightarrow H_i \cap H_j = \emptyset\}$
Műveletek:

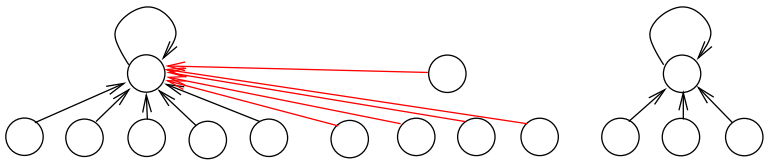
$$S : UnioHolvan, X : Elemtip, N, N1, N2 : NevTip$$

$\{Igaz\}$	$Letesit(S)$	$\{S = \emptyset\}$
$\{S = S\}$	$Megszuntet(S)$	$\{Igaz\}$
$\{S = S\}$	$Uresit(S)$	$\{S = \emptyset\}$
$\{S = \{H_1,\ldots,H_k\} \wedge X \in \cup H_i\}$	$Holvan(S,X,N)$	$\{N = Y \wedge (X \in H_i \wedge Y \in H_i)\}$
$\{S = \{H_1,\ldots,H_k\} \wedge X \notin \cup H_i\}$	$Holvan(S,X,N)$	$\{N = X \wedge S = Pre(S) \cup \{\{X\}\}$
$\{S = \{H_1,\ldots,H_k\} \wedge N1 \in H_i \wedge N2 \in H_j\}$	$Unio(S,N1,N2)$	$\{S = Pre(S) - H_i - H_j \cup \{H_i \cup H_j\}\}$
$\{S = \{H_1,\ldots,H_k\}\}$	$Elemszam(S)$	$\{= k \wedge S = Pre(S)\}$
$\{S = \{H_1,\ldots,H_k\}\}$	$ReszElemszam(S,N)$	$\{= \lvert H_i\rvert \wedge N \in H_i \wedge S = Pre(S)\}$
$\{S = S\}$	$IterKezd(S,I)$	$\{\}$
$\{I = I\}$	$IterAd(I,x)$	$\{\}$
$\{I = I\}$	$IterVege(I)$	$\{\}$
$\{S = \{H_1,\ldots,H_k\} \wedge (\exists i)(N \in H_i)\}$	$ReszIterKezd(S,N,I)$	$\{\}$
$\{I = I\}$	$ReszIterAd(I,x)$	$\{\}$
$\{I = I\}$	$ReszIterVege(I)$	$\{\}$

Adatszerkezet választása.
A diszjunkt részhalmazok ábrázolhatók olyan $Rep : \{1,\ldots,n\} \to \{1,\ldots,n\}$ függvénnyel, hogy bármely $x,y \in \{1,\ldots,n\}$-re $Rep(x) = Rep(y)$ akkor és csak akkor, ha x és y ugyanazon részhalmazhoz tartozik. Ezt szemlélteti a 4. ábra. Ekkor a HOLVAN művelet konstans időben megvalósítható, az UNIO azonban az egyik részhalmaz elemszámával arányos idejű lesz. Egy másik lehet-

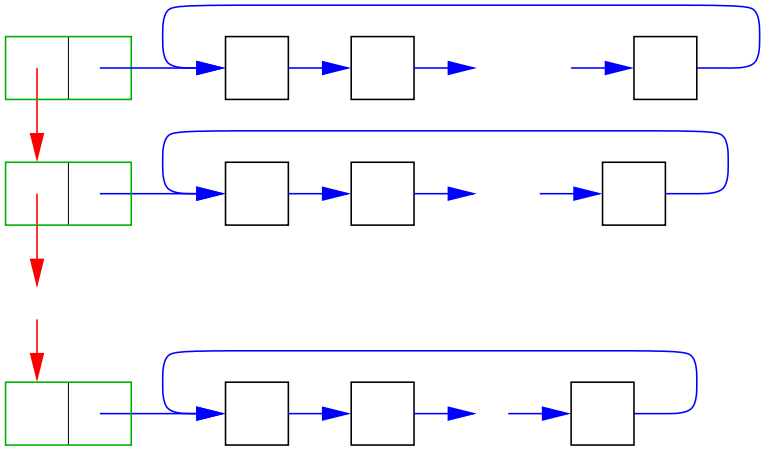


4. ábra. Egyszerű adatszerkezet az UnioHolvan adattípushoz.

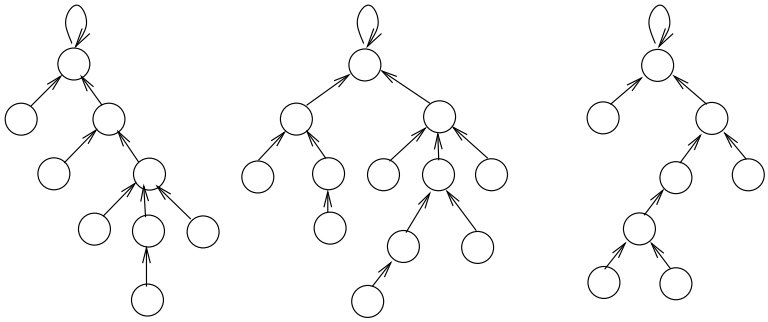


5. ábra. Az UNIO művelet megvalósítása egyszerű adatszerkezet esetén.

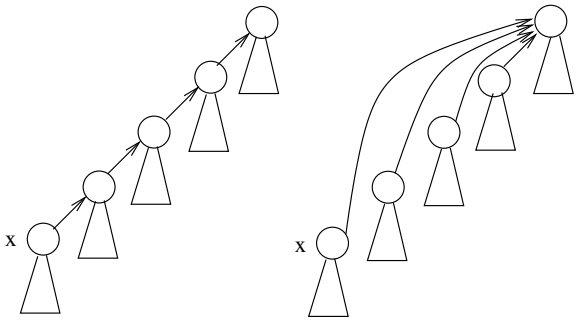
séges adatszerkezet a megvalósításhoz a kétdimenziós lánc, amit a a 6. ábra mutat. Ekkor az UNIO művelet konstans időben megvalósítható, a HOLVAN művelet azonban csak $O(n)$ időben.
Tehát egymásnak ellentmodó követelményeket kellene kielégíteni, amelyik adatszerkezet jó a HOLVAN művelethez, az nem jó az UNIO-hoz. Kevesebbet követeljünk a HOLVAN hatékony, azaz konstans idejű megvalósításánál. Ábrázoljuk a részhalmazokat fával, mint azt a 7. ábra mutatja. Ekkor az UNIO konstans időben negvalósítható. A HOLVAN futási idejének felső korlátja a fa magassága, tehát arra kell törekedni, hogy fa magassága ne legyen nagy.
Két heurisztokát használunk ennek elérésére:



6. ábra. Kétdimenziós lánc az UNIOHOLVAN típus megvalósításához.

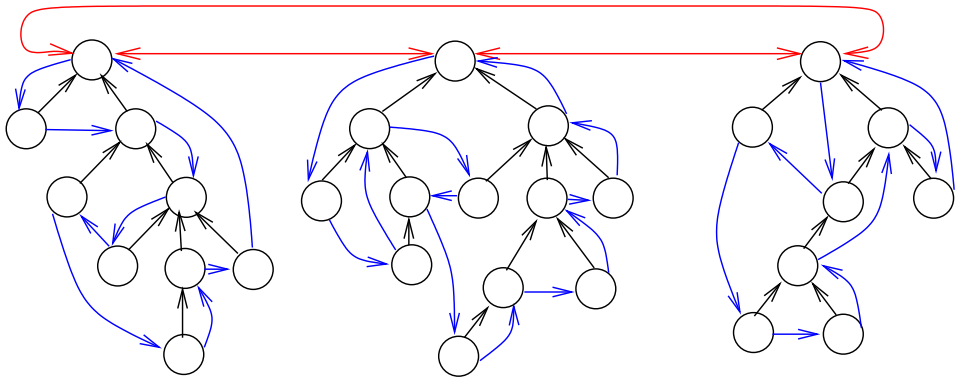


7. ábra. Halmazerdő adatszerkezet az UNIOHOLVAN adattípus megvalósításához.

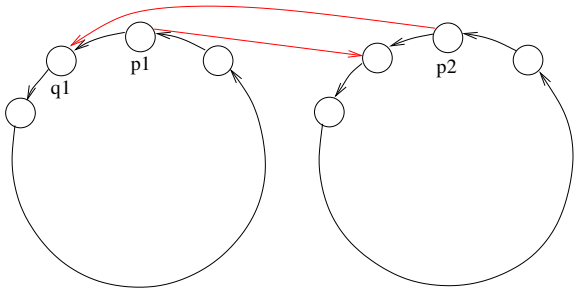


8. ábra. Úttömörítés HOLVAN művelet során.

1.**Egyesítés a nagyobbikhoz:** Az UNIO művelet mindig a nagyobb elemszámú részhalmaz gyökeréhez kapcsolja a kisebb elemszámú részhalmaz gyökerét.
2. **Úttömörítés:** Minden HOLVAN művelet során a fának mindazon pontjait, amelyen a keresés áthalad a gyökérhez kapcsoljuk.
Az iterátor műveletek hatékony megvalósításához ábrázoljuk a részhalmazokat körláncban, továbbá kétirányú körláncban a részhalmazok gyökereit. Tehát az adatszerkezet: $(M, Adat, R)$; ahol az R szerkezeti kapcsolatok: $R = \{Apa, Csat, Elore, Vissza\}$, $Apa, Csat, Elore, Vissza : M \to M$. Az Apa kapcsolat fa, a $Csat$ kapcsolat körlánc, az $(Elore, Vissza)$ kapcsolatpár pedig kétirányú körlánc.
A részhalmazok elemszámát nem tároljuk, hanem $Apa(x) = -E$ legyen, ha x gyökér és az x-gyökerű részhalmaz elemszáma E. Két körlánc konstans időben egyesíthető, amit a 10. ábra mutat.



9. ábra. Adatszerkezet az UNIOHOLVAN adattípus megvalósításához.



10. ábra. Két körlánc egyesítése:$q1 := Csat(p1); Csat(p1) := Csat(p2); Csat(p2) := q1$.

```
Const
  MaxN=10000;
  Null=0;
Type
  Adattip = ???         ;(* a felhasználó definiálja *)
  Index=1..MaxN;
  Kulcstip = Index;
  Elemtip = Record
              kulcs : Kulcstip;
              adat  : Adattip
            End;
  Nevtip = Kulcstip;
  Tartip = Array[Index] Of
             Record
               adat:Adattip;
               apa:Integer;
               csat:Index;
```

```
              elore,vissza:Index;
             End;
   RepTip = Record
              Fa : Tartip;
              Fej: 0..MaxN;
              Eszam : 0..MaxN;
            End;

   UnioHol  = RepTip ;(* az UnioHolvan adattipus tipusa *)
   Iterator=^IterRep;
   IterRep = Record
              Hra: ^TarTip;
              kezd,pont: Integer
            End;
   ReszIterator=Record
                  Hra: ^TarTip;
                  kezd,pont: Integer
              End;

Procedure Letesit(Var S:UnioHol; N:Index);
  Var i:Index;
  Begin
    S.Eszam:=0;
    S.Fej:=0;
    For i:=1 To N Do S.Fa[i].apa:=0;
  End;

Procedure Uresit(Var S:UnioHol);
  Var i:Index;
  Begin
    S.Eszam:=0;
    S.Fej:=0;
    For i:=1 To MaxN Do S.Fa[i].apa:=0;
  End{Uresit};

Procedure Holvan(Var S : UnioHol; X : Elemtip; Var N : Nevtip);
  Var  j,k:Index;
  Begin
    j:=X.kulcs; N:=j;
    If S.Fa[j].apa=Null Then Begin
      Inc(S.Eszam);
      S.Fa[j].adat:=X.adat;
      S.Fa[j].apa:=-1;
      S.Fa[j].csat:=j;      {egyelemű körlánc}
      If S.Fej=0 Then Begin {az első részhalmaz}
        S.Fej:=N;
        S.Fa[N].elore:=N;
        S.Fa[N].vissza:=N;
      End Else Begin
        k:=S.Fa[S.Fej].elore; {az új részhalmaz bekötése a kettős körláncba}
        S.Fa[j].elore:=k;
        S.Fa[k].vissza:=j;
        S.Fa[j].vissza:=S.Fej;
        S.Fa[S.Fej].elore:=j;
      End
    End Else Begin
```

```
      While S.Fa[N].apa > 0 Do
        N:=S.Fa[N].apa;
      While j <> N Do Begin   { úttömörítés }
        k:=S.Fa[j].apa;
        S.Fa[j].apa:=N;
        j:=k
      End;
    End
  End{Holvan};

Procedure Unio(Var S : UnioHol;
                  N,M : Nevtip);
  Var
    k:Nevtip; p:Index;
  Begin
    If (N<>M) And (S.Fa[N].apa<0) And (S.Fa[M].apa<0) Then Begin
      Dec(S.Eszam);
      If S.Fa[N].apa > S.Fa[M].apa Then Begin  {egyesítés a nagyobbikhoz}
        k:=N; N:=M; M:=k
      End;
      S.Fa[N].apa:=S.Fa[N].apa + S.Fa[M].apa;
      S.Fa[M].apa:=N;
      p:=S.Fa[N].csat;            {a két körlánc egyesítése}
      S.Fa[N].csat:=S.Fa[M].csat;
      S.Fa[M].csat:=N;

      k:=S.Fa[M].vissza;
      S.Fa[M].vissza:=S.Fa[M].elore;
      S.Fa[S.Fa[M].elore].vissza:= K;
    End
  End{Unio};

Procedure Eleme(    S : UnioHol;
                    K : Kulcstip;
                Var Van : Boolean;
                Var N : Nevtip );
  Var
    i,j:Index;
  Begin
    j:=K;
    Van:=S.Fa[j].apa<>Null;
    If Van Then Begin
      N:=j;
      While S.Fa[N].apa > 0 Do
        N:=S.Fa[N].apa;
      While j <> N Do Begin   { úttömörítés }
        i:=S.Fa[j].apa;
        S.Fa[j].apa:=N;
        j:=i
      End;
    End
  End{Eleme};

Function Elemszam(S : UnioHol) : Integer;
  Begin
    Elemszam:=S.Eszam
```

```
  End;

Function ReszElemszam(S : UnioHol;
                      N : Nevtip) : Integer;
  Begin
    ReszElemszam:=-S.fa[N].apa
  End;

Procedure IterKezd(S : UnioHol; Var I: Iterator);
  Begin
    New(I); I^.kezd:= S.Fej;
    I^.Hra:= @S.Fa;
    I^.pont:= S.Fej;
  End;

Procedure IterAd(Var I:Iterator; Var X: NevTip);
  Begin
    With I^ Do Begin
      If pont = Null Then exit;
      X:= pont;
      pont:= Hra^[pont].elore;
      If pont=kezd Then
        pont:= 0
    End;
  End;

  Function  IterVege(I: Iterator): Boolean;
    Begin
      IterVege:= I^.pont =Null;
    End;

Procedure ReszIterKezd(S : UnioHol; N: Nevtip; Var I: Iterator);
  Begin
    New(I);
    I^.kezd:= N;
    I^.Hra:= @S.Fa;
    I^.pont:= N;
  End;

Procedure ReszIterAd(Var I: Iterator; Var X: Elemtip);
  Begin
    With I^ Do Begin
      If pont = Null Then exit;
      X.kulcs:= pont;
      X.adat:= Hra^[pont].adat;
      pont:= Hra^[pont].csat;
      If pont=kezd Then
        pont:= Null
    End;
  End;

Function ReszIterVege(I: Iterator): Boolean;
  Begin
    ReszIterVege:= I^.pont = Null;
  End;
```

Az UNIOHOL adattípus megvalósításának hatékonysági elemzése.

Összesítéses elemzés. Egy műveletegyüttes, mint az absztrakt adattípusok hatékonysági elemzését megadhatjuk úgy is, hogy nem külön-külön vizsgáljuk az egyes műveletek futási idejét, hanem müveletsorok összesített futási idejét mérjük. Ha egy $P = P_1; \ldots; P_m$ műveltsor összesített futási ideje $T(P)$, akkor a $T(P)/m$ hányadost az egyes műveletek **amortizált** futási idejének nevezzük.

$$lg^{(i)}n = \begin{cases} n & \text{ha} \quad i = 0\,, \\ lg(lg^{(i-1)}n) & \text{ha } i > 0 \text{ és } lg^{(i-1)}n > 0, \\ \text{nemdef} & \text{egyébként} \end{cases}$$

$lg^*n = min\{i \geq 0 : lg^{(i)}n \leq 1\}$

$P = P_1; \cdots; P_m$ műveletsorozat, ahol
$P_i \in \{\text{UNIO}, \text{HOLVAN}\}$ és
n azon HOLVAN műveletek száma, amelyek új elemet adnak a halmazrendszerhez, azaz a részhalmazok elemszámának összege a műveletsor végén n, akkor a műveletsor futási ideje:

$T(P) = O(m\,lg^*n)$
lg^*n minden gyakorlatban előforduló n-re ≤ 5, mivel $lg^*(2^{65536}) = 5$.
Tehát az HOLVAN műveletek amortizált futási ideje praktikusan konstans.