



Algoritmuselmélet 2. témakör

Pusztai Pál
pusztai@sze.hu

Tartalom

- Dinamikus halmazok
 - Vermek és sorok megvalósítása tömbökkel
 - Láncolt listák
- Mutatók és objektumok
 - Megvalósítás tömbökkel
- Gyökeres fák, bináris fák
- Hasító táblázatok
 - Közvetlen címzés
 - Hasító függvények
 - Ütközésfeloldás láncolással
 - Nyílt címzés
 - Lineáris és négyzetes kipróbálás, dupla hasítás



Dinamikus halmazok

A számítástechnikában használatos, időben változó, véges halmazokat **dinamikus halmazoknak** nevezzük.

A dinamikus halmazok objektumaiban szerepelhet egy (gyakran azonosítóként is használt) **kulcsmező** ill. az objektumoknak lehetnek **kísérő adatai** is.

Dinamikus halmazokon értelmezett **műveletek**:

- $KERES(H, k)$ Egy k kulcsú elem megkeresése a H halmazban. Az eredmény egy olyan elem x mutatója, amelyre $kulcs[x]=k$, ill. NIL ha nincs ilyen elem.
- $BESZÚR(H, x)$ A H halmaz bővítése az x által mutatott elemmel.
- $TÖRÖL(H, x)$ Törli az x által mutatott elemet a H halmazból.
- $MINIMUM(H)$ A H halmaz legkisebb kulcsértékű elemének a mutatóját adja.
- $MAXIMUM(H)$ A H halmaz legnagyobb kulcsértékű elemének a mutatóját adja.
- $KÖVETKEZŐ(H, x)$ Annak az elemnek a mutatója, amelynek kulcsértéke közvetlenül az x elem kulcsértéke után következik a teljes rendezés szerint, ill. NIL, ha x a legnagyobb kulcsú elem H -ban.
- $ELŐZŐ(H, x)$ Annak az elemnek a mutatója, amelynek kulcsértéke közvetlenül megelőzi az x elem kulcsértékét a teljes rendezés szerint, ill. NIL, ha x a legkisebb kulcsú elem.

Megjegyzés:

- Az első három műveletet megvalósító dinamikus halmazokat **szótáraknak** nevezzük.
- A többi művelet feltételezi, hogy a kulcsok egy **teljesen rendezett** halmazból valók (ahol teljesül a **trichotómia**, azaz a és b elemek esetén $a < b$, $a = b$, $a > b$ esetek közül pontosan egy teljesül).



Vermek

ÜRES-VEREM(V)

```

1  if  $tet\ddot{o}[V] = 0$ 
2      return IGAZ
3  else
4      return HAMIS
    
```

VEREMBE(V, x)

```

1   $tet\ddot{o}[V] \leftarrow tet\ddot{o}[V] + 1$ 
2   $V[tet\ddot{o}[V]] \leftarrow x$ 
    
```

VEREMBÖL(V)

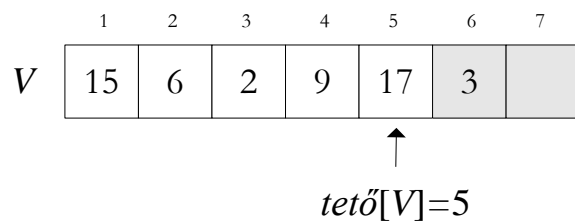
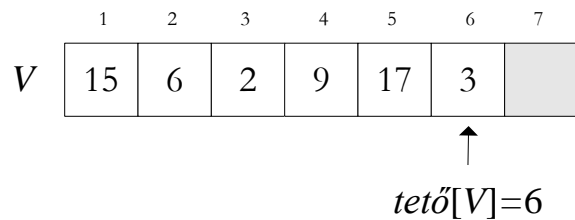
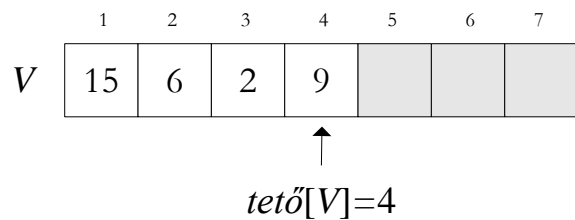
```

1  if ÜRES-VEREM( $V$ )
2      error „alulcsordulás”
3  else
4       $tet\ddot{o}[V] \leftarrow tet\ddot{o}[V] - 1$ 
5      return  $V[tet\ddot{o}[V] + 1]$ 
    
```

Megjegyzés: A vermeken értelmezett BESZÚR művelet neve VEREMBE, a TÖRÖL művelet neve pedig VEREMBÖL.

Hatékonyság: Mindhárom művelet $O(1)$ idejű.

Vermek



A V verem tömbös ábrázolása

Feladatok

- Milyen állapotú vermet kapunk a VEREMBE(V , 4), VEREMBE(V , 1), VEREMBE(V , 3), VEREMBŐL(V), VEREMBŐL(V) és VEREMBE(V , 8) hívások után, ha a vermet az $V[1..6]$ tömbben tároljuk, és a verem kezdetben üres? Mekkora a *tető*[V] értéke?



Sorok

SORBA(S, x)

```
1   $S[vége[S]] \leftarrow x$ 
2  if  $vége[S] = hossz[S]$ 
3       $vége[S] \leftarrow 1$ 
4  else
5       $vége[S] \leftarrow vége[S] + 1$ 
```

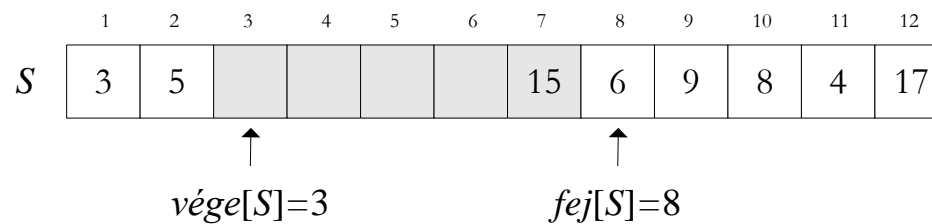
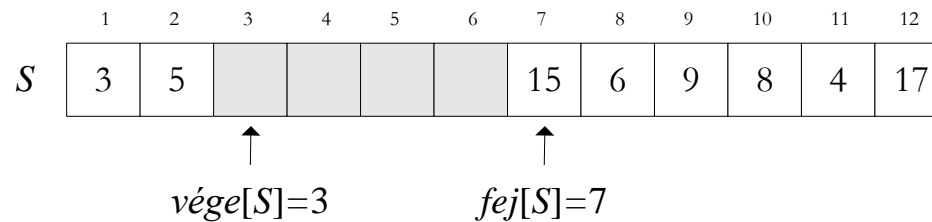
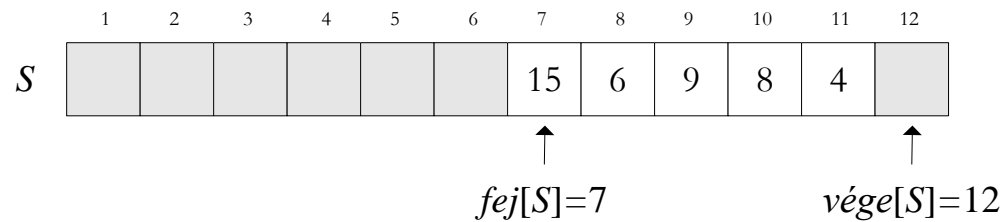
SORBÓL(S)

```
1   $x \leftarrow S[fej[S]]$ 
2  if  $fej[S] = hossz[S]$ 
3       $fej[S] \leftarrow 1$ 
4  else
5       $fej[S] \leftarrow fej[S] + 1$ 
6  return  $x$ 
```

Megjegyzés: A sorokon értelmezett BESZÚR művelet neve SORBA, a TÖRÖL művelet neve SORBÓL. A $fej[S]$ mutatja a sorban lévő első elem helyét, a $vége[S]$ pedig az első szabad helyet. Kezdetben $fej[S]=vége[S]=1$. Az alul- és túlszordulást a hívó kezeli.

Hatékonyság: Mindkét művelet $O(1)$ idejű.

Sorok



Egy sor megvalósítása az $S[1..12]$ tömbbel

Feladatok

- Milyen állapotú sort kapunk a $SORBA(S, 4)$, $SORBA(S, 1)$, $SORBA(S, 3)$, $SORBÓL(S)$, $SORBÓL(S)$ és $SORBA(S, 8)$ hívások után, ha a sort az $S[1..6]$ tömbben tároljuk, és a sor kezdetben üres? Mekkora a $fej[S]$ és a $vége[S]$ értéke, ha kezdetben $fej[S]=vége[S]=5$?
- Ha egy sort az $S[1..n]$ tömbbel valósítunk meg, akkor a sor miért csak $n-1$ elemet tartalmazhat n elem helyett?

Láncolt listák

LISTÁBAN-KERES(L, k)

```
1   $x \leftarrow fej[L]$   
2  while  $x \neq \text{NIL}$  és  $kulcs[x] \neq k$   
3       $x \leftarrow köv[x]$   
4  return  $x$ 
```

LISTÁBA-BESZÚR(L, x)

```
1   $köv[x] \leftarrow fej[L]$   
2  if  $fej[L] \neq \text{NIL}$   
3       $előző[fej[L]] \leftarrow x$   
4   $fej[L] \leftarrow x$   
5   $előző[x] \leftarrow \text{NIL}$ 
```

Hatékonyság: Egy n elemű lista esetén a keresés legrosszabb esetben $\Theta(n)$, a beszúrás $O(1)$ idejű.

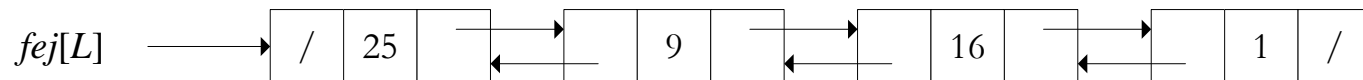
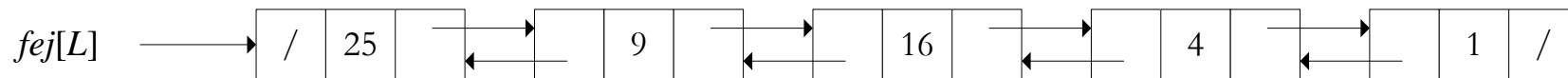
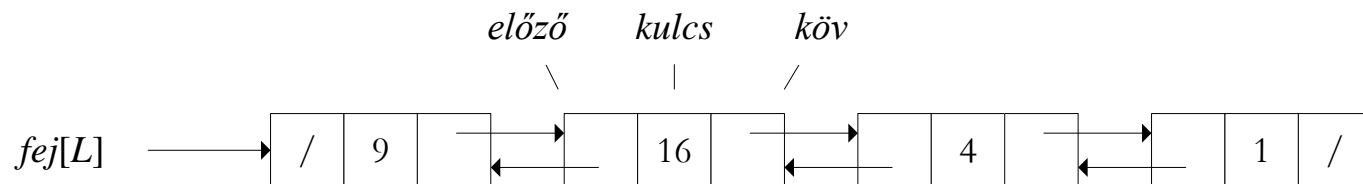
Láncolt listák

LISTÁBÓL-TÖRÖL(L, x)

```
1  if  $előző[x] \neq \text{NIL}$ 
2       $köv[előző[x]] \leftarrow köv[x]$ 
3  else
4       $fej[L] \leftarrow köv[x]$ 
5  if  $köv[x] \neq \text{NIL}$ 
6       $előző[köv[x]] \leftarrow előző[x]$ 
```

Hatékonyság: Egy n elemű lista esetén a törlés $O(1)$ idejű, ha azonban egy adott kulcsú elemet szeretnénk törölni, akkor ehhez legrosszabb esetben $\Theta(n)$ idő szükséges.

Láncolt listák



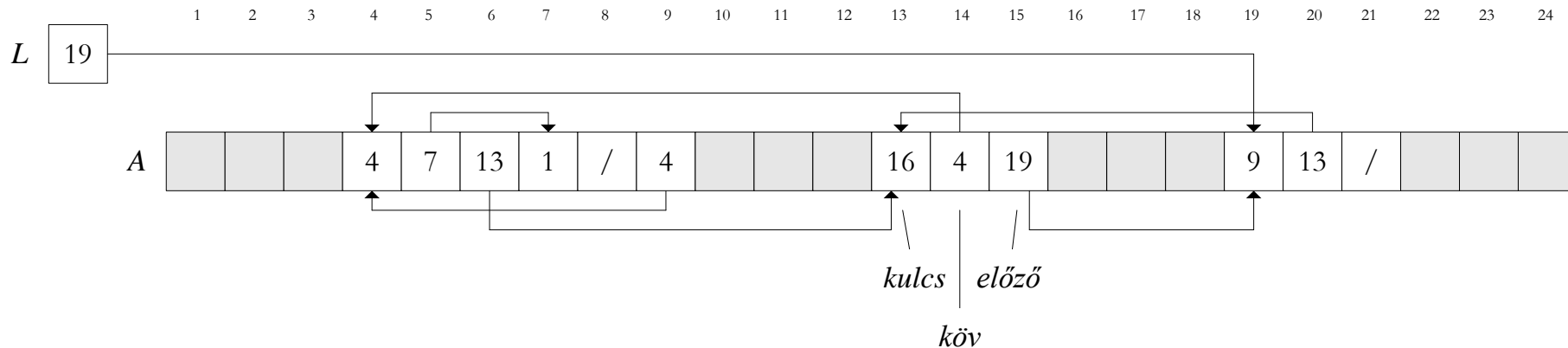
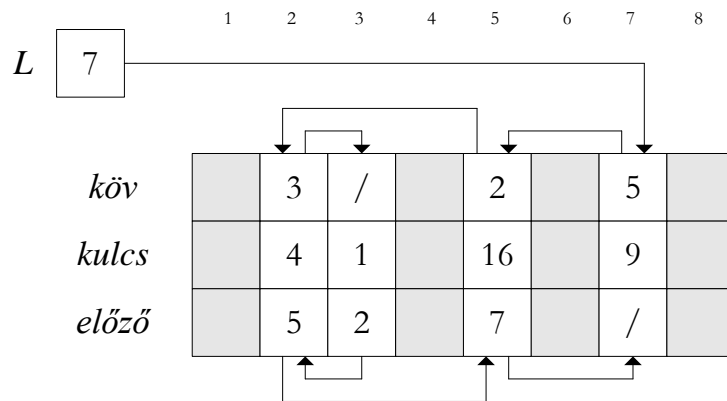
Dinamikus halmaz ábrázolása egy kétszeresen láncolt L listával

Feladatok

- Megvalósítható-e a dinamikus halmazokra értelmezett BESZÚR művelet egyszeresen láncolt listákra $O(1)$ időben? Mi a helyzet a TÖRÖL művelettel?
- Adjunk olyan nemrekurzív eljárást, amely $\Theta(n)$ időben megfordít egy n elemű, egyszeresen láncolt listát! Az eljárás konstans méretű segédmemóriát használhat.



Mutatók és objektumok ábrázolása



Objektumok ábrázolása tömbökkel

Feladatok

- Adjuk meg az alábbi kulcsokat tartalmazó kétszeresen láncolt lista több-tömbös ábrázolását úgy, hogy a tömbök 10 eleműek és az elemek rendre a páratlan tömbpozíciókba kerülnek!
 - $\langle 5, 4, 8, 2, 1 \rangle$
- Adjuk meg az alábbi kulcsokat tartalmazó kétszeresen láncolt lista egy-tömbös ábrázolását úgy, hogy a tömb 15 elemű és az elemek a tömböt előlről kezdve folyamatosan töltik fel!
 - $\langle 6, 2, 5, 3 \rangle$

Mutatók és objektumok ábrázolása

OBJEKTUMOT-LEFOGLAL()

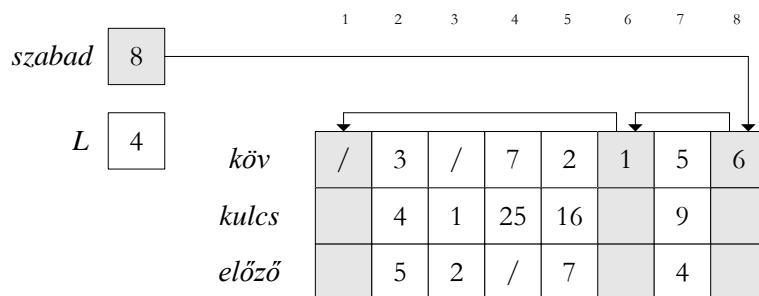
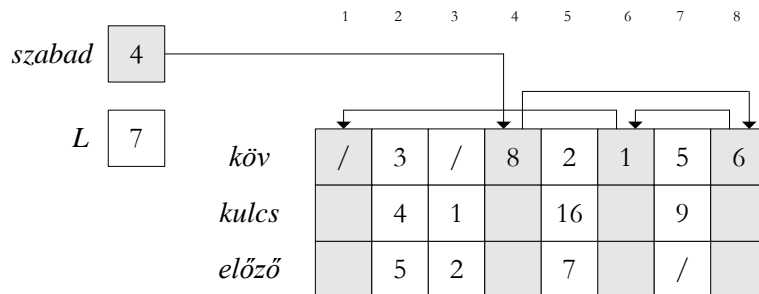
```
1  if szabad = NIL
2      error „nincs szabadhely”
3  else
4       $x \leftarrow \textit{szabad}$ 
5       $\textit{szabad} \leftarrow \textit{köv}[x]$ 
6      return  $x$ 
```

OBJEKTUMOT-FELSZABADÍT(x)

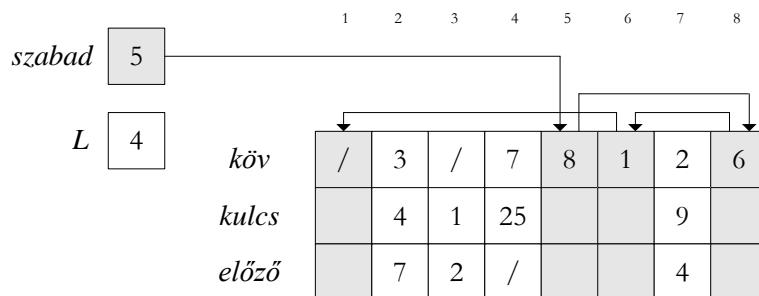
```
1   $\textit{köv}[x] \leftarrow \textit{szabad}$ 
2   $\textit{szabad} \leftarrow x$ 
```



Mutatók és objektumok ábrázolása



Az OBJEKTUMOT-LEFOGLAL() hívás
(ami 4-et ad vissza),
kulcs[4] beállítása 25-re, és a
LISTÁBA-BESZÚR(*L*, 4) végrehajtása utáni állapot



A LISTÁBÓL-TÖRÖL(*L*, 5) és az
OBJEKTUMOT-FELSZABADÍT(5) végrehajtása
utáni állapot

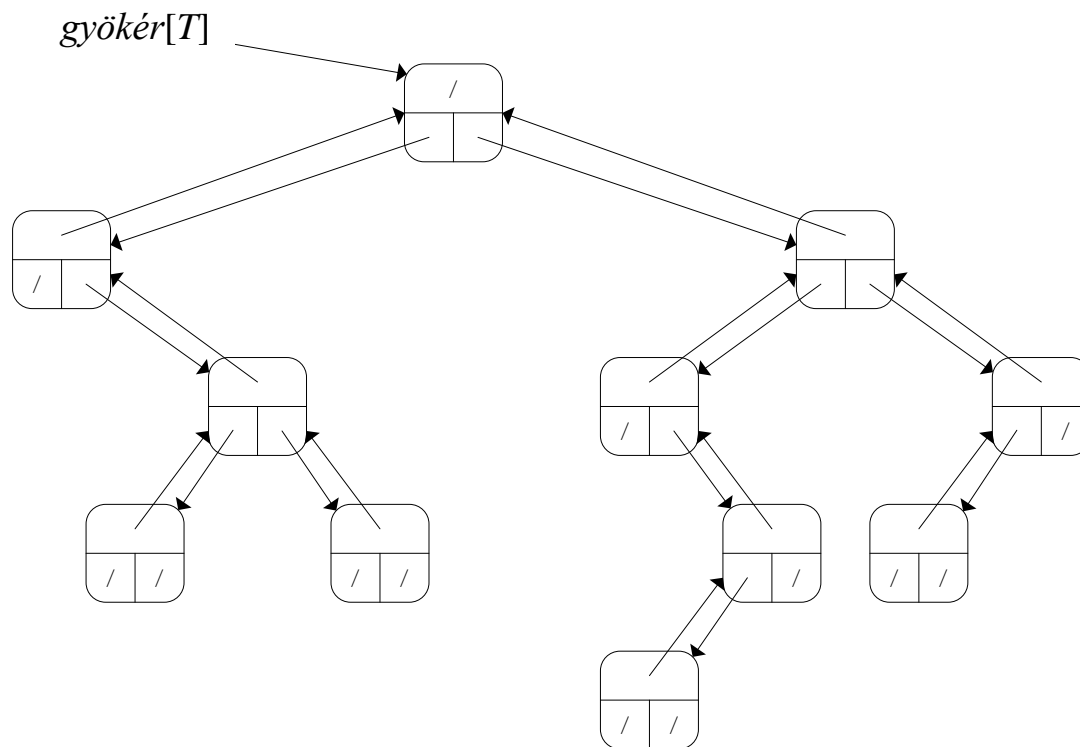
Objektumok helyfoglalása és felszabadítása

Feladatok

- Vegyük fel az előbbi L listába a 12-es kulcsú és 8 kulcsú elemet, majd töröljük a 3-as indexű elemet! Milyen állapotot kapunk?

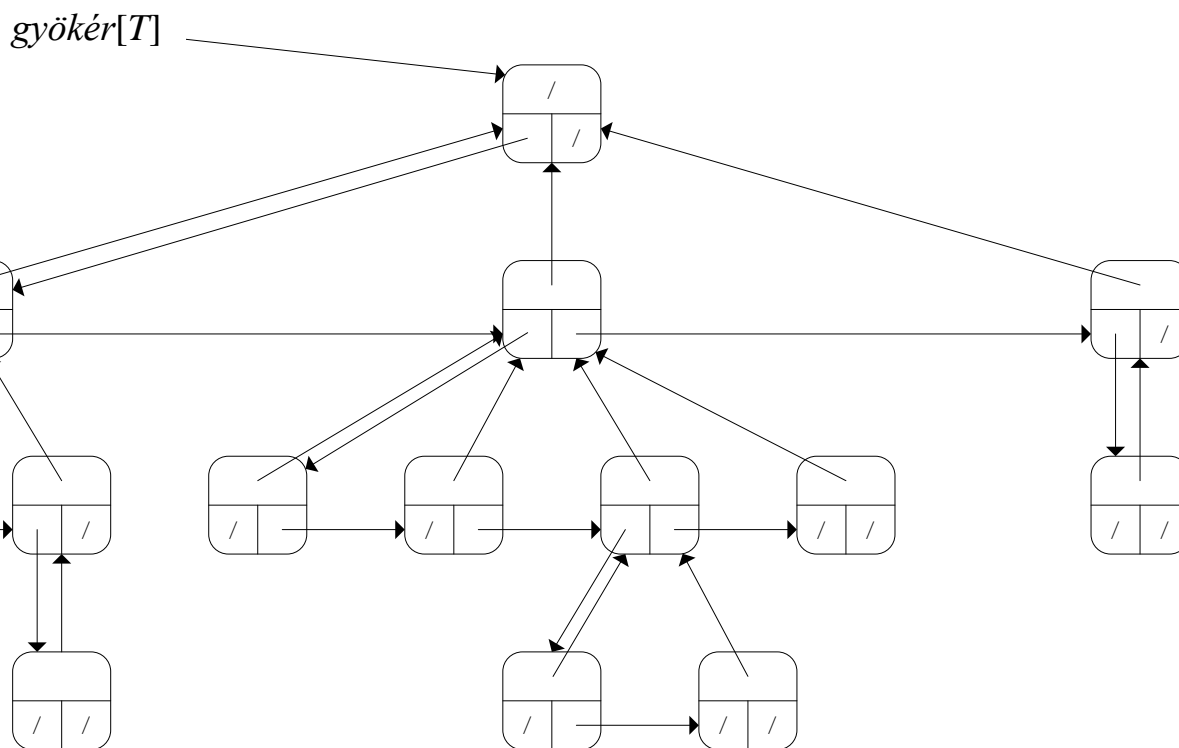


Gyökeres fák ábrázolása



Bináris fa ábrázolása

Gyökeres fák ábrázolása



Bal-gyermek, jobb-testvér reprezentáció

Hasító táblázatok

Feladat: Szótár megvalósítása tömb segítségével.

Azokat a dinamikus halmazokat, amelyek csak a BESZÚR, KERES és TÖRÖL műveleteket támogatják, **szótáraknak** nevezzük.

Közvetlen címzés: akkor alkalmazható, ha lefoglalhatunk akkora tömböt, amelyben minden lehetséges kulcsnak megfelel egy tömbelem.

Hatékonyság: Minden szótárművelet $O(1)$ idejű.

Hasító táblázatok: akkor alkalmazzuk, ha a tárolt kulcsok száma a lehetséges kulcsokhoz képest viszonylag kicsi.

Az adatok tárolására a tárolt kulcsok számával arányos méretű tömböt használunk.

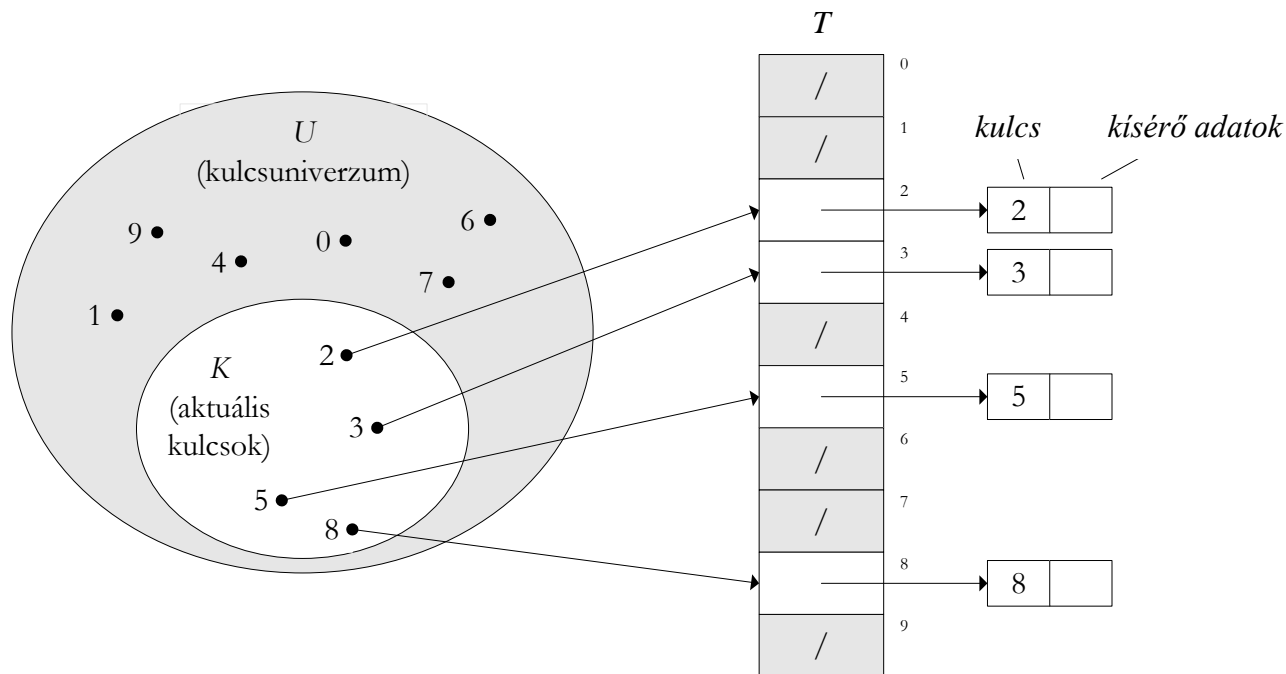
A tömb elemeinek közvetlen címzésére a kulcs helyett egy, a kulcsból **kiszámított értéket** használunk, amelyet **hasító függvénnyel** számolunk ki.

Hatékonyság: Minden szótárművelet **átlagosan** $O(1)$ idejű.

Hasító táblázatok

Feltétel: A tárolandó kulcsuniverzum $U = \{0, 1, 2, \dots, m-1\}$, ahol m nem túl nagy, és nincs két egyforma kulcsú elem.

Megvalósítás: Egy $T[0..m-1]$ tömb, amelyet **közvetlen címzésű táblázatnak** nevezünk. A táblázatban minden hely (más néven **rész**) megfelel az U univerzum egy kulcsának.



Dinamikus halmaz megvalósítása közvetlen címzésű táblázattal

Hasító táblázatok

KÖZVETLEN-CÍMZÉSŰ-KERESÉS(T, k)

1 **return** $T[k]$

KÖZVETLEN-CÍMZÉSŰ-BESZÚRÁS(T, x)

1 $T[kulcs[x]] \leftarrow x$

KÖZVETLEN-CÍMZÉSŰ-TÖRLÉS(T, x)

1 $T[kulcs[x]] \leftarrow \text{NIL}$

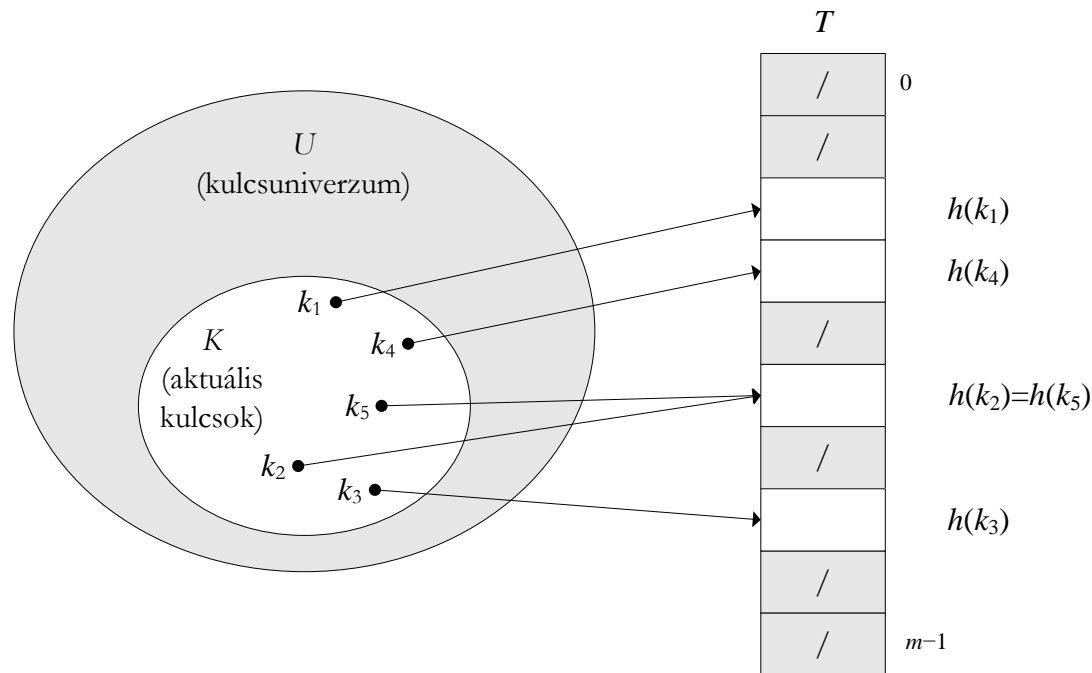
Hatékonyság: Mindhárom művelet $O(1)$ idejű.

Hasító táblázatok

A **hasító függvény**: $h : U \rightarrow \{0, 1, \dots, m-1\}$

Megvalósítás: Egy $T[0..m-1]$ tömb, amelyet **hasító táblázatnak** nevezünk. A k kulcsú elem a táblázat $h(k)$ részére **képződik le** (vagy más szavakkal $h(k)$ a k kulcs **hasított értéke**).

Probléma: Több kulcs is leképeződhet ugyanarra a részre, azaz **ütközés** történik.



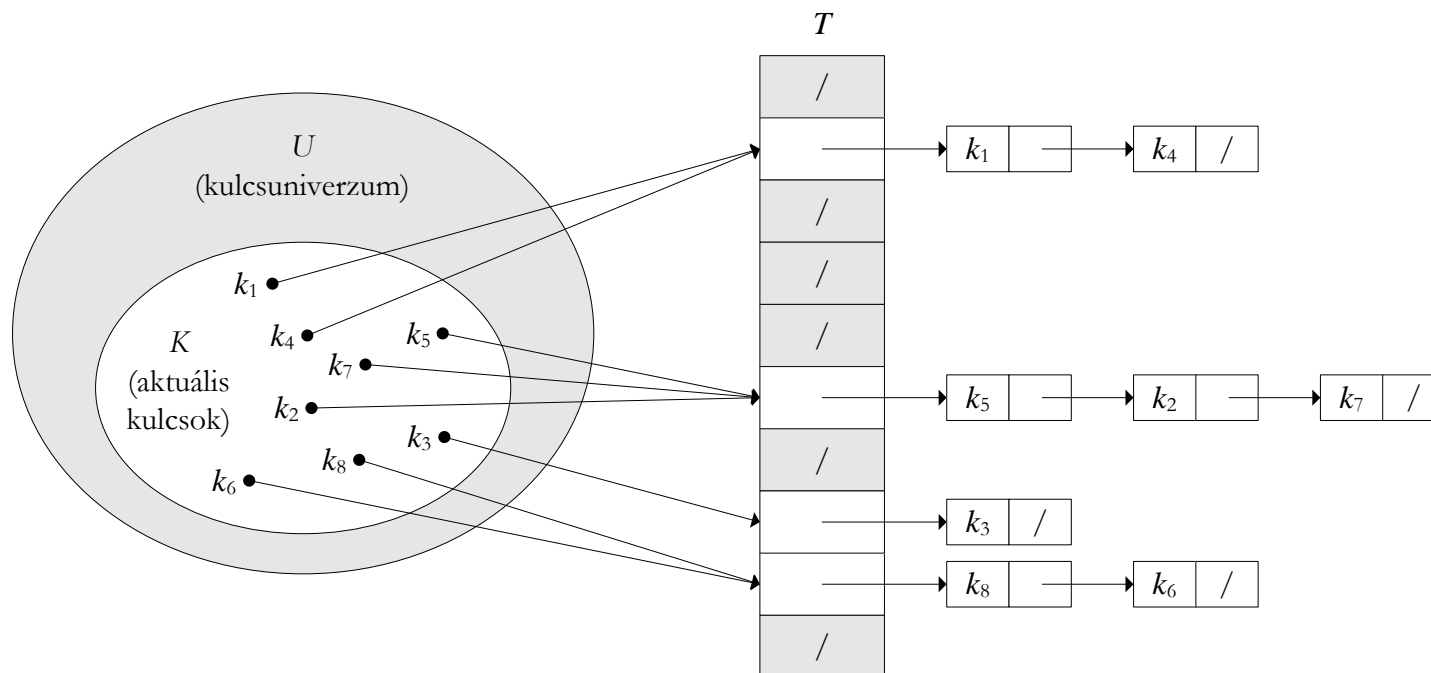
Hasító táblázat réseire való leképezés hasító függvénnyel

Hasító táblázatok

- A hasító függvények fajtái:
 - Osztásos módszer: $h(k) = k \bmod m$
 - Jó értékek m számára: 2 hatványokhoz nem túl közeli prímek.
 - Pl. $n=2000$ elemünk van, akkor az $m=701$ választással a sikertelen keresések átlagosan 3 elemet vizsgálnak meg.
 - Szorzásos módszer: $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$
 - A egy konstans a $(0, 1)$ intervallumból.
 - Az m értéke nem kritikus, általában valamilyen 2 hatvány.
 - Univerzális hasítási technika:
 - A hasító függvényt egy alkalmas függvényosztályból a futás során véletlenszerűen választjuk ki.

Hasító táblázatok

Ütközésfeloldás láncolással: Az ugyanarra a résre leképződő elemeket összefogjuk egy láncolt listába.



Ütközésfeloldás láncolással

Hasító táblázatok

LÁNCOLT-HASÍTÓ-KERESÉS(T, k)

1 a k kulcsú elem keresése a $T[h(k)]$ listában

LÁNCOLT-HASÍTÓ-BESZÚRÁS(T, x)

1 x beszúrása a $T[h(kulcs[x])]$ lista elejére

LÁNCOLT-HASÍTÓ-TÖRLÉS(T, x)

1 x törlése a $T[h(kulcs[x])]$ listából

Hatékonyság: A beszúrás ideje $O(1)$, a törlés kétszeresen láncolt listák esetén $O(1)$, a keresés és az egyszerűen láncolt listákból való törlés legrosszabb esetben a lista hosszával arányos.

Ha **egyszerű egyenletes hasítást** alkalmazunk (azaz minden elem egyforma valószínűséggel képződik le bármely résre, függetlenül attól, hogy a többiek hová kerültek), és a hasító táblázat réseinek száma **arányos** a táblázatbeli elemek számával, akkor a keresés **átlagos** ideje $O(1)$.

Feladatok

- Milyen hosszú lesz a leghosszabb lista és hány ilyen hosszúságú listát kapunk az alábbi kulcsok beszúrása után, ha a rések száma 9 és a hasító függvény $h(k) = k \bmod 9$?
 - $\langle 5, 28, 19, 15, 20, 33, 12, 17, 10 \rangle$



Hasító táblázatok

A **nyílt címzés** az ütközések kezelésének egy másik módszere.

A keresést és beszúrást a hasító táblázat réseinek egymás utáni **kipróbálásával** végezzük. Az elemeket magában a hasító táblázatban tároljuk.

A **kipróbálandó pozíciók** sorrendje a fix $0, 1, \dots, m-1$ helyett (ami $\Theta(n)$ keresési időre vezetne) a beszúrandó kulcs és a kipróbálási szám függvénye.

A **hasító függvény**: $h : U \times \{0, 1, \dots, m-1\} \rightarrow \{0, 1, \dots, m-1\}$

Feltétel: minden k kulcsra a $\langle h(k, 0), h(k, 1), \dots, h(k, m-1) \rangle$ **kipróbálási sorozat** a $\langle 0, 1, \dots, m-1 \rangle$ egy permutációja legyen.

HASÍTÓ-BESZÚR(T, k)

```

1   $i \leftarrow 0$ 
2  repeat
3       $j \leftarrow h(k, i)$ 
4      if  $T[j] = \text{NIL}$ 
5           $T[j] \leftarrow k$ 
6          return
7      else
8           $i \leftarrow i+1$ 
9  until  $i = m$ 
10 error „hasító táblázat túlsordulás”
```

Hasító táblázatok

HASÍTÓ-KERES(T, k)

```

1   $i \leftarrow 0$ 
2  repeat
3       $j \leftarrow h(k, i)$ 
4      if  $T[j] = k$ 
5          return  $j$ 
6       $i \leftarrow i+1$ 
7  until  $T[j] = \text{NIL}$  vagy  $i = m$ 
8  return NIL
    
```

Megjegyzés: Ha törölnünk is kell elemeket, akkor az ütközések feloldására legtöbbször nem is a nyílt címzést, hanem a láncolást használjuk.

Hatékonyság: Ha **egyenletes hasítást** alkalmazunk (azaz minden rögzített k kulcs esetén a $\langle h(k, 0), h(k, 1), \dots, h(k, m-1) \rangle$ kipróbálási sorozat egyforma valószínűséggel lehet a $\langle 0, 1, \dots, m-1 \rangle$ bármelyik permutációja), akkor egy $\alpha = n/m < 1$ kitöltöttségi aránnyal rendelkező nyílt címzéses hasító táblázatban a beszúrás és a sikertelen keresés várható próbaszáma legfeljebb $1/(1-\alpha)$, a sikeres keresésé pedig $(1/\alpha) \ln(1/(1-\alpha))$.

Pl. $\alpha = 1/2$ esetén 2, ill. 1.38, $\alpha = 9/10$ esetén 10, ill. 2.55.

Hasító táblázatok

Legyen $h': U \rightarrow \{0, 1, \dots, m-1\}$ egy közös hasító függvény és $i = 0, 1, \dots, m-1$.

Lineáris kipróbálás

$$h(k, i) = (h'(k) + i) \bmod m$$

Megjegyzés: A kezdeti próbapozíció meghatározza az egész kipróbálási sorozatot, ezért csak m kipróbálási sorozat fordul elő. Hosszú, elemek által teljesen kitöltött sorozatok jönnek létre.

Négyzetes kipróbálás

$$h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$$

Megjegyzés: c_1 és $c_2 \neq 0$ segédállandók. Ahhoz, hogy a táblázatot teljesen kihasználjuk megkötéseket kell tennünk a c_1 , c_2 és m értékekre. Itt is csak m különböző kipróbálási sorozat van használatban.

Dupla hasítás

$$h(k, i) = (h_1(k) + i h_2(k)) \bmod m$$

Megjegyzés: A $h_2(k)$ értéknek relatív prímnek kell lennie a táblázat m méretéhez képest. Itt már m^2 különböző kipróbálási sorozat van használatban. (Az ideális egyenletes hasításhoz $m!$ kéne).

Hasító táblázatok

0	
1	79
2	
3	
4	69
5	98
6	
7	72
8	
9	14
10	
11	50
12	

$$h(k, i) = (h_1(k) + ih_2(k)) \bmod m$$

$$m = 13$$

$$h_1(k) = k \bmod 13$$

$$h_2(k) = 1 + (k \bmod 11)$$

Beszúrás dupla hasítás esetén

Feladatok

- Milyen hasító táblázatokat kapunk az egyes nyílt címzési módszerek esetén, ha egy üres hasító táblázatba a következő kulcsokat szűrjük be rendre?
 - $\langle 10, 22, 31, 4, 15, 28, 17, 88, 59 \rangle$
- Lineáris kipróbálás ($m=11$)
 - $h(k, i) = (h'(k) + i) \bmod m$
 - $h'(k) = k \bmod m$
- Négyzetes kipróbálás ($m=9, c_1=1, c_2=3$)
 - $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$
 - $h'(k) = k \bmod m$
- Dupla hasítás ($m=11$)
 - $h(k, i) = (h_1(k) + i h_2(k)) \bmod m$
 - $h_1(k) = k \bmod m$
 - $h_2(k) = 1 + (k \bmod (m-1))$