ADATSZERKEZETEK ÉS ALGORITMUSOK

Python nyelven

Hasító tábla

Hasító táblázat

Tömb adatszerkezet általánosítása

Jól alkalmazható **adatszerkezet**, ha Beszúr(), Keres() és Töröl() műveleteket megvalósító **dinamikus** halmazra van szükség.

Beszúr(), Keres(), Töröl()- szótárműveletek O(1) időbonyolultsággal

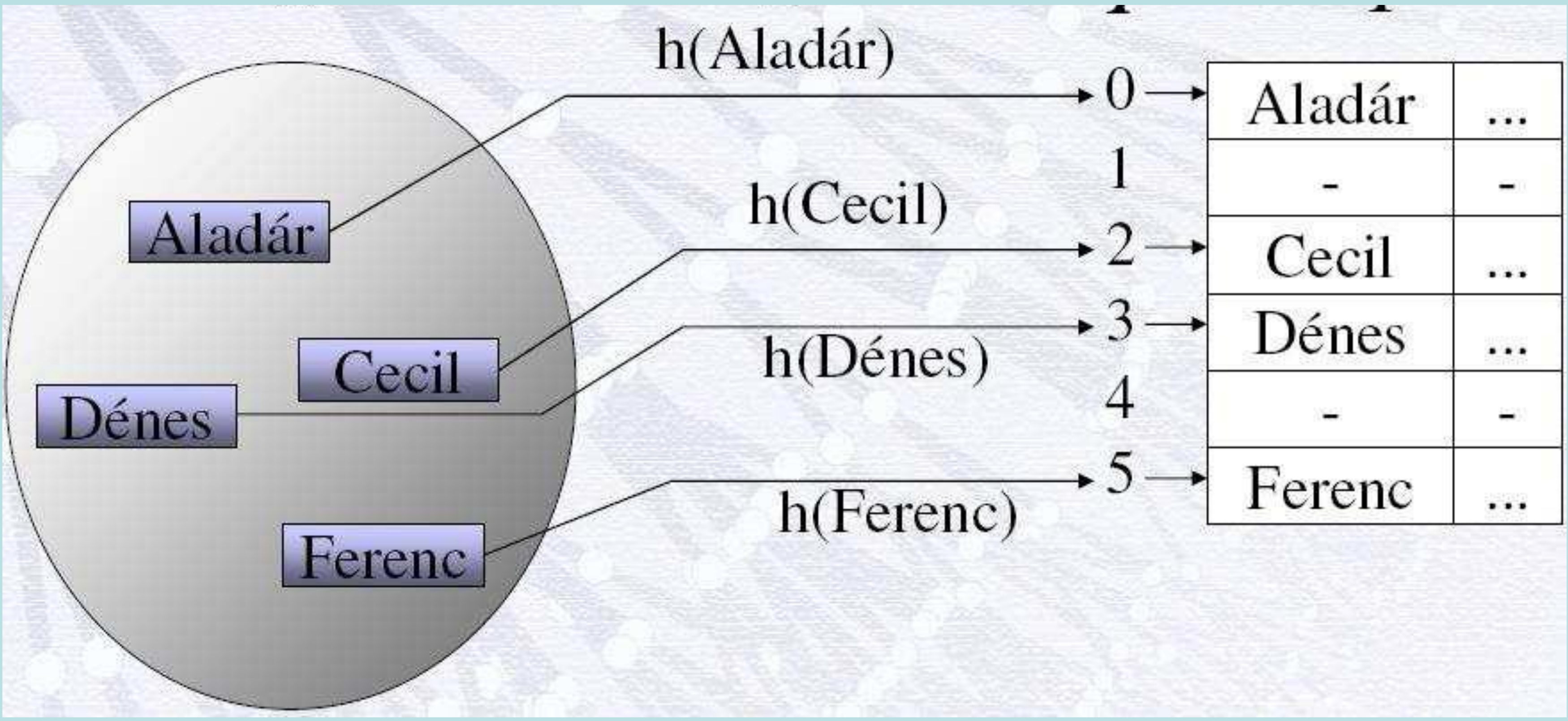
Hasítótáblák felhasználása

Adatbázisokban

Fordítóprogramok szimbólumtábláiban

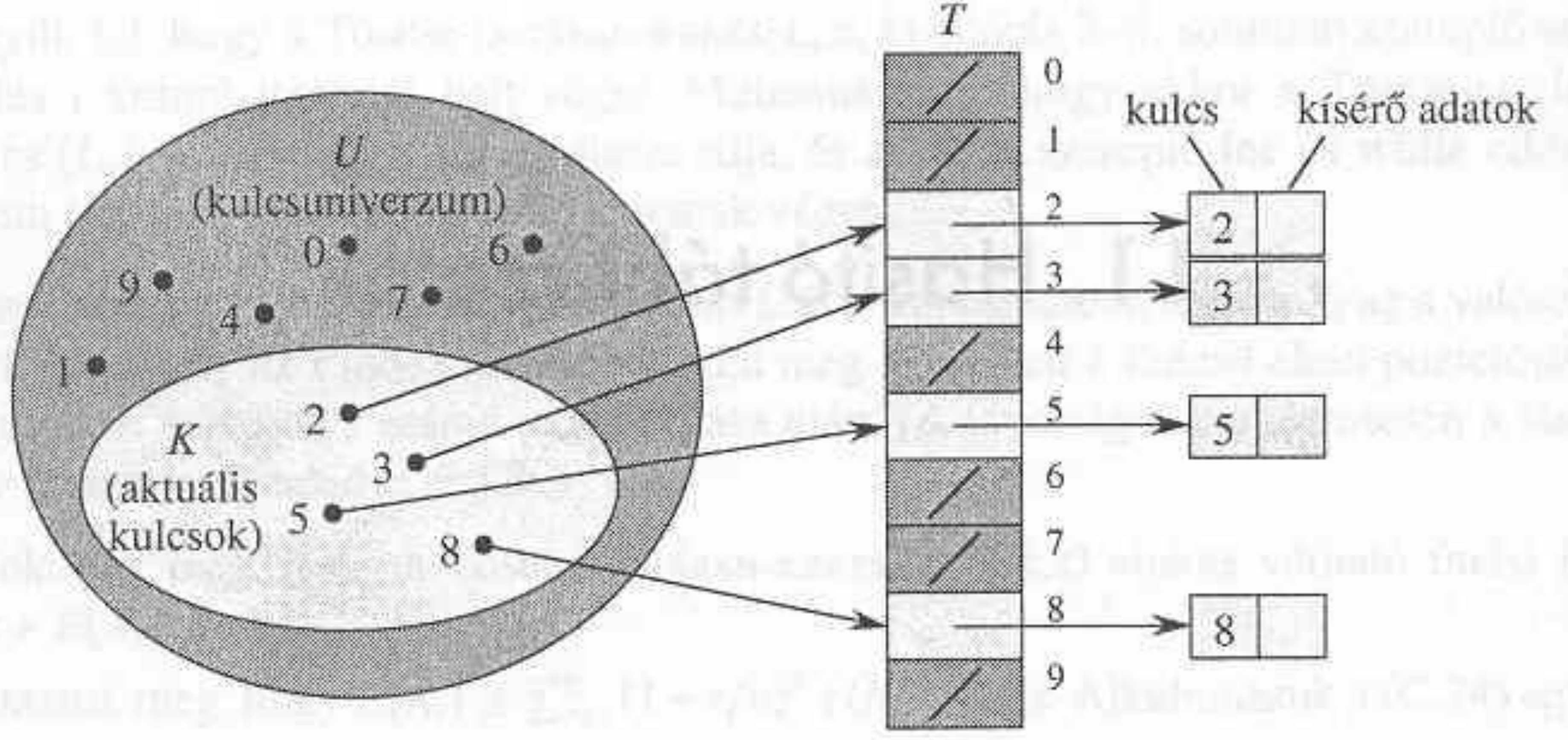
A hasítótáblák alapelve

- Minden adatrekordhoz egy a kulcsból számítható címet rendelünk, és ide fogjuk eltárolni
- A számítást hasítófüggvény végzi



HASÍTÓTÁBLÁK MEGVALÓSÍTÁSÁNAK LEHETŐSÉGEI

Halmaz megvalósítása közvetlen címzésű táblázattal



Feltételek:

Kisméretű kulcsuniverzum

Nincs két egyforma kulcsú elem

Adatelemek tárolása

Magában a közvetlen címzésű táblázatban, gyakran a kulcsmező tárolása sem szükséges	A réshez mutatóval kapcsolt külső objektumban
• Visszakeresés index alapján • Ha nem tároljuk a kulcsmezőt, el kell tudni dönteni, hogy a rés üres-e	

Közvetlen címzésű táblázat

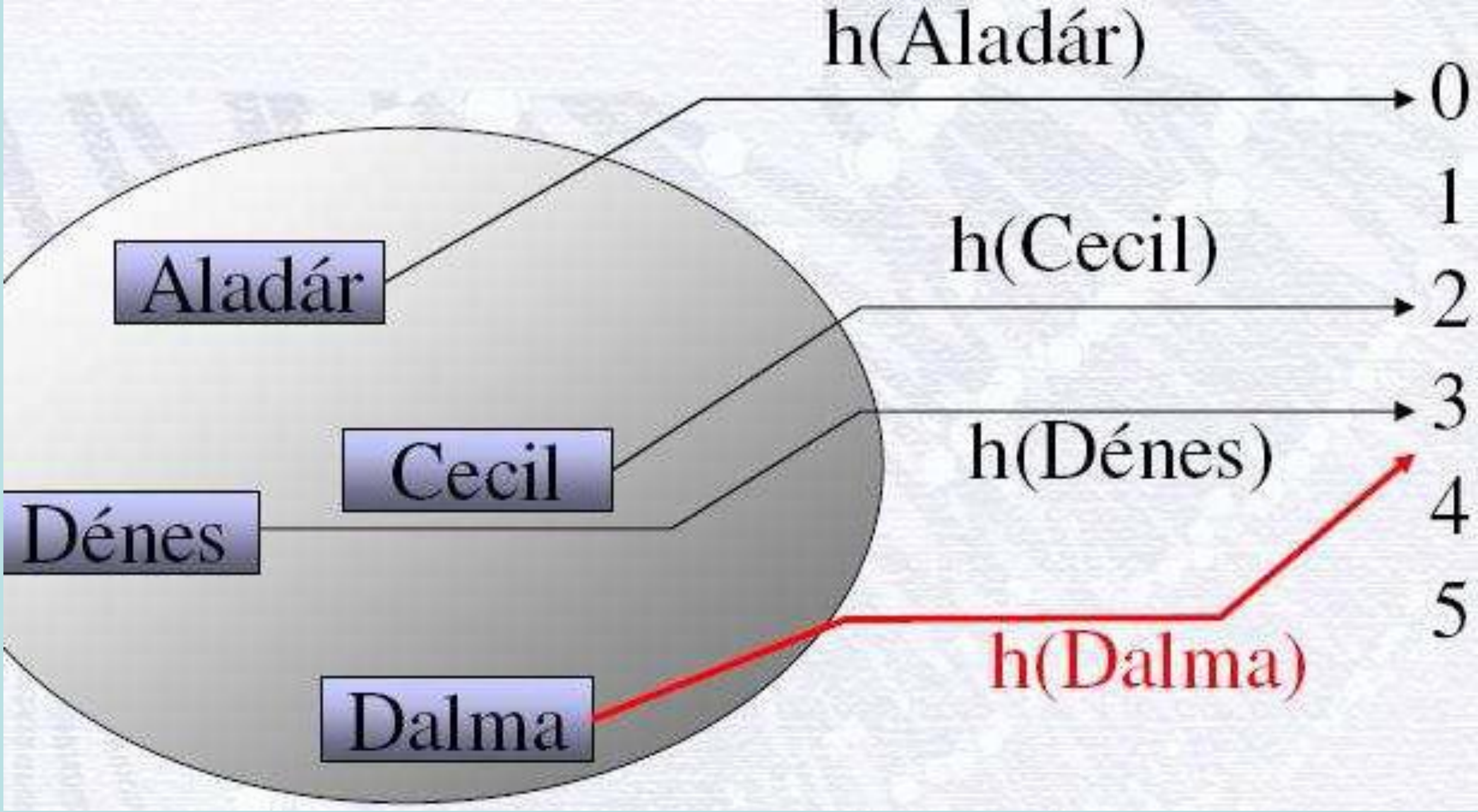
```
Keres(T,k)
return T[k]

Beszúrás(T,x
/*adat*/)
T[kulcs[x]] ←x

Törlés(T,x/*adat*/)
T[kulcs[x]]=null
```

- T[0…m-1] tömb segítségével
- Táblázat helyei – rés – megfelelnek a kulcsoknak
- K. rés a halmaz k kulcsú elemére mutat.
- Ha nincs k kulcsú elem akkot T[k]=null
- Műveletek – Közvetlen címzéssel – végrehajtási idő O(1)

Közvetlen címzés problémája

- Ha az U univerzum nagy és a tárolt elemek K halmaza U-hoz képes kicsi, akkor a T táblázat nagy helyet foglal, de nagy része üres, a kihasználtság kicsi
- Memóriaigény leszorítható egy alkalmasan választott függvény segítségével, amely a k kulcsú elemet egy k' résben tárolja ahol k'=h(k), vagyis **hasító függvényt** használunk a k' kiszámításához.
- $h:U \rightarrow \{0,1,\ldots m-1\}$
- Új probléma: kulcsütközés

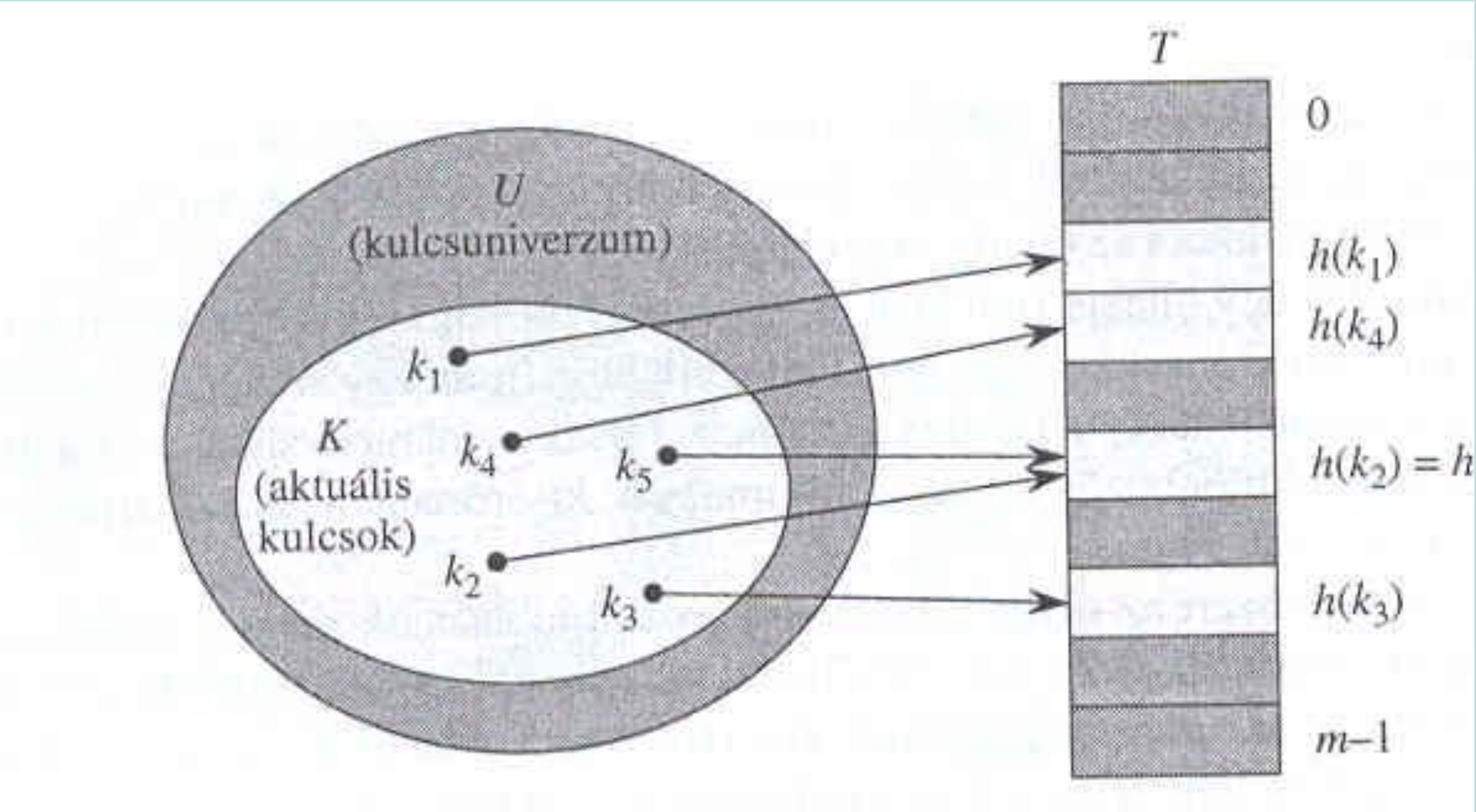
Kulcsütközés fogalma



Kulcsütközés

- Ha két kulcs ugyanarra a résre képződik le
- Kulcsütközés elvileg sem küszöbölhető ki, mert U>m, tehát mindenképp kell lennie legalább két kulcsnak, amelyek ugyanarra a résre képződnek le.

Hasítófüggvények



Kulcsütközés feloldása I.

- Túlcsordulási terület használatával
- Csak akkor alkalmazható ha nagyon kevés a kulcsütközés
- Beszúrás:

  a, T[h(k)] =null → beszúrás T[h(k)]-ba

  b, T[h(k)] ≠ null → beszúrás túlcsordulási területre
- Keresés:

  a, T[h(k)] = keresett → megvan

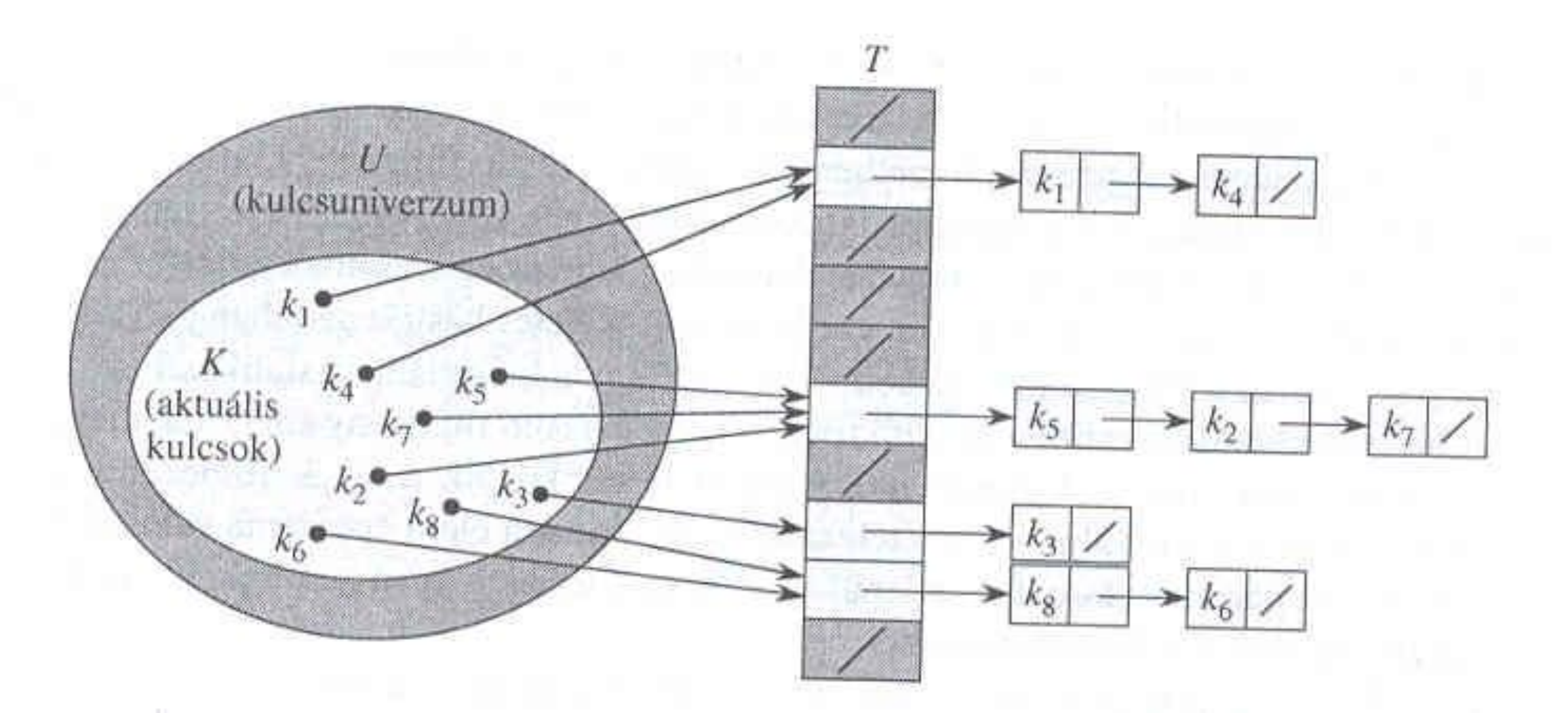
  b, T[h(k)] = Ø → nincs

  c, T[h(k)] ≠ keresett

  keresett túlcsordulási terület → megvan

  keresett túlcsordulási terület → nincs

Kulcsütközés feloldása II. Láncolással



Példa

- Kövessük végig láncolásos ütközésfeloldás esetén az 5,28,19,15,20,33,12,17,10 kulcsok hasító táblázatba való beszúrását. Legyen a rések száma 9 és a hasító függvény h(k)=k mod 9

Ütközésfeloldás láncolással II.

- Ugyanarra a résre leképződő elemeket összefogjuk egy láncolt listába

Láncolt-hasító-beszúrás(T,x)

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

Láncolt-hasító-keresés(T,k)

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

Láncolt-hasító-törlés

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

Beszúrás és törlés műveletigénye: O(1)

Rendezés?

Keresés időigénye

- Legrosszabb esetben a keresés időigénye O(n)
- Jó esetben:
  - Legyen α=n/m (α a kitöltési tényező, vagyis a láncba fűzött elemek átlagos száma)
  - Egyenletes hasítás esetén O(1+α), vagyis a keresés átlagos ideje állandó

Hasító függvények

- Hogyan működik egy jó hasító függvény?
  - –Egyenletességi feltétel: egyenletesen ossza szét a kulcsokat
    - Ehhez ismerni kellene a kulcsok eloszlás függvényét
  - –A hasonló kulcsokat véletlenszerűen ossza szét
  - –Kevés kulcsütközést produkáljon
  - –Egyszerűen kiszámítható legyen
  - –A hasítófüggvény kötelezően determinisztikus

A kulcsok természetes számokkal való megjelenítése

A kulcsokat természetes számként kezeljük,

Ha nem természetes szám akkor természetes számmá kell alakítani pl. ASCII kódok…

$h(k) = \Sigma \text{Karakterkód}(k_i)$
(k є Szöveg)

Osztásos módszer

h(k)=k mod m

Jellemzői

- Gyors hasítás
- Jó, ha m egy kettőhatványhoz nem közeli prim.
- Rések száma m
- Példa:2000 db adat van a kitöltési tényező lehet ~3, tervezzük meg a hasítófüggvényt!

Szorzásos módszer

$h(k)=Közepe^{M}(A*k)$

$Közepe^{M}(x)$ visszaadja az x középső M darab helyiértékét

Nevezetes módszer a négyzetközép módszer: $h(k)=Közepe^{M}(k*k)$

Az univerzális hasítás

- Véletlenszerűen kiválasztott hasítófüggvénnyel próbálja minimalizálni a kulcsütközést.

Kulcsütközés feloldása – Nyílt címzés III.

A duplikátumot egy előre definiált szisztematikus üres hely kereséssel helyezzük el!

- Beszúrás: Ha T[h(uj)] foglalt, egy/több lépés előre/hátra amíg nem találunk egy üres helyet, ide elmentjük
- Keresés: Ha T[h(keresett)] nem a keresendő elem, a beszúrásnál használt azonos lépések végrehajtása,amíg nem találjuk meg, vagy az első üres helyig. Ha az induló vagy bármelyik elem üres, akkor nincs a táblázatban
- Törlés:Keresés után az elem bejelölése töröltnek (nem üres!)

Kulcsütközés feloldása IV. – Dupla hasítás

Használjunk két hasító függvényt (h1, h2)!

Beszúrás: Ha a h1 függvény alkalmazásával kulcsütközés lépne fel,
próbáljuk egy másik h2 függvénnyel, ami máshova képez

Keresés: Ha a h1 által megadott helyen nem találjuk a keresett elemet, keressük a h2 szerint is

Törlés: A fenti kereséssel megtalált elemet töröljük

Köszönöm a figyelmet!

- Vizsga információ!

Példa

- Készítsünk 3 jegyű számok rendezésére alkalmas radix rendező algoritmust. A szétválogatáshoz foglaljunk egy int[10,n] T mátrixot, az egyes sorokban levő elemszám tárolására alkalmazzunk egy int[10] index vektort. Javasoljon más, hatékony adatszerkezetet a feladat megoldásához.