# Algoritmuselmélet 5. előadás

Katona Gyula Y.
Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi Tsz.
I. B. 137/b
kiskat@cs.bme.hu

2002 Február 26.

# 2-3-fák

Hatékony keresőfa-konstrukció
Ez is fa, de a binárisnál bonyolultabb: egy nem-levél csúcsnak 2 vagy 3 fia lehet.

A *2-3-fa* egy (lefelé) irányított gyökeres fa, melyre:

- A rekordok a fa leveleiben helyezkednek el, a kulcs értéke szerint balról jobbra növekvő sorrendben. Egy levél egy rekordot tartalmaz.

- Minden belső (azaz nem levél) csúcsból 2 vagy 3 él megy lefelé; ennek megfelelően a belső csúcsok egy illetve két $s \in U$ kulcsot tartalmaznak. A belső csúcsok szerkezete tehát kétféle lehet. Az egyik típus így ábrázolható:

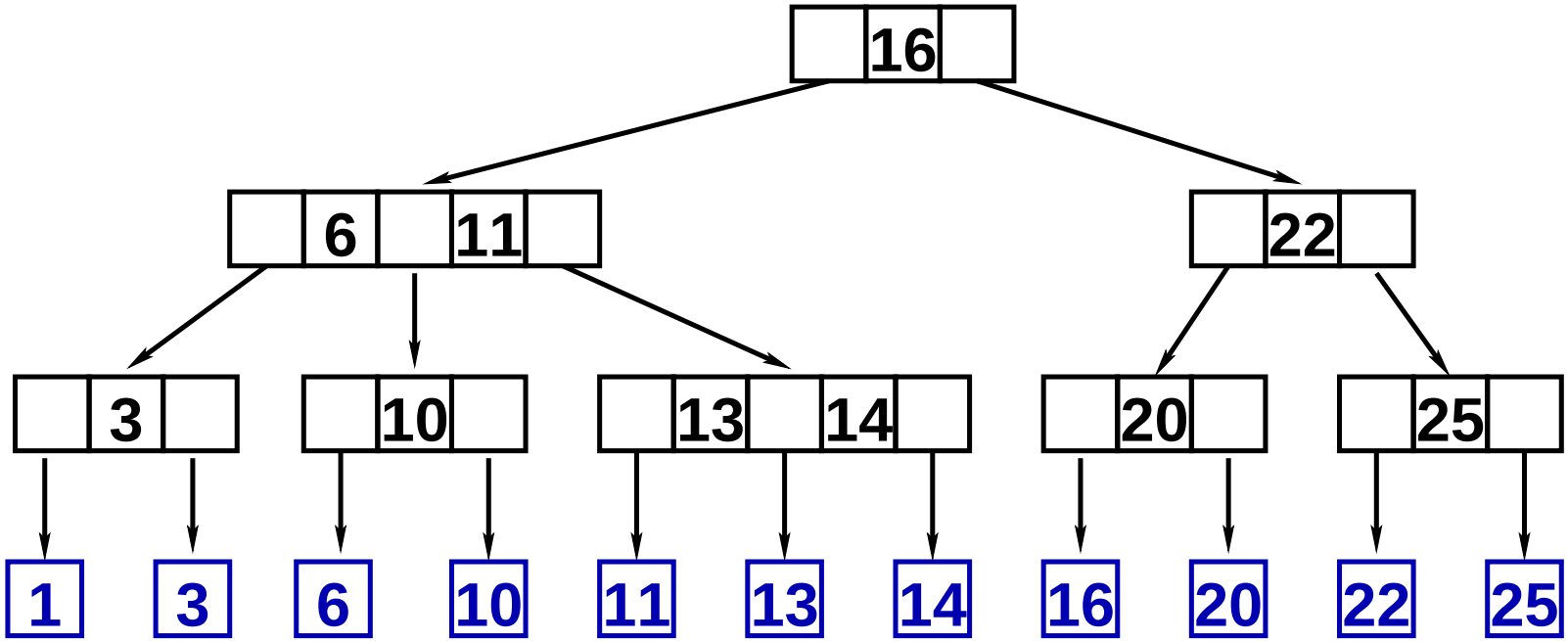
| $m_1$ | $s_1$ | $m_2$ | $s_2$ | $m_3$ |
|---|---|---|---|---|

  Itt $m_1, m_2, m_3$ mutatók a csúcs részfáira, $s_1$, $s_2$ pedig $U$-beli kulcsok, melyekre $s_1 < s_2$. Az $m_1$ által mutatott részfa minden kulcsa kisebb, mint $s_1$, az $m_2$ részfájában $s_1$ a legkisebb kulcs, és minden kulcs kisebb, mint $s_2$. Végül $m_3$ részfájában $s_2$ a legkisebb kulcs.Előfordulhat, hogy egy csúcsból az utolsó két mező hiányzik:

| $m_1$ | $s_1$ | $m_2$ |
|---|---|---|

- A fa levelei a gyökértől egyforma távolságra vannak.

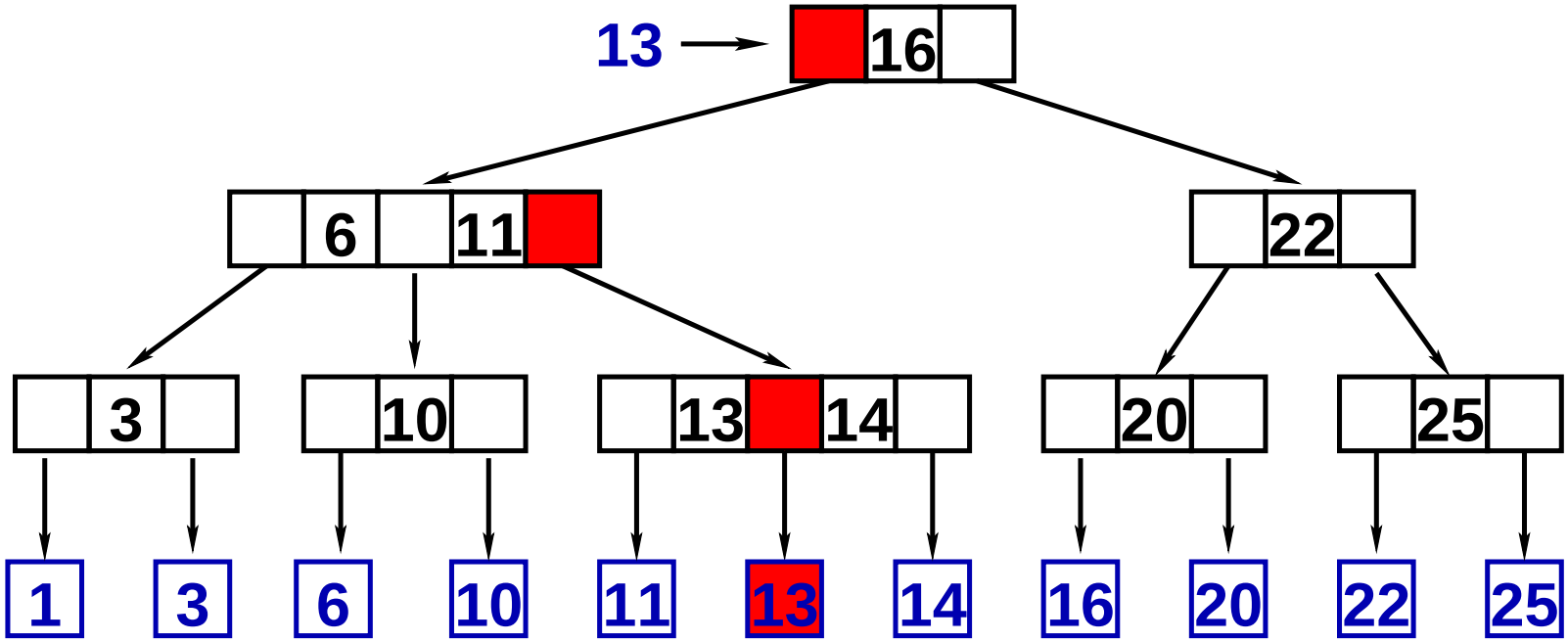
# Példa 2-3-fára




**Tétel.** *Ha a fának $m$ szintje van, akkor a levelek száma legalább $2^{m-1}$. Megfordítva, ha $|S| = n$ (itt $S \subseteq U$ a fában tárolt kulcsok halmaza; $|S|$ megegyezik a tárolt rekordok számával), akkor $m \leq \log_2 n + 1$.*

Bizonyítás: Minde belső csúcsnak legalább $2$ fia van $\Longrightarrow$
az $i$-edik szinten legalább $2^{i-1}$ csúcs van ($1 \leq i \leq m$). $\Longrightarrow$
$2^{m-1} \leq n$, $\Longrightarrow m - 1 \leq \log_2 n$. ✓

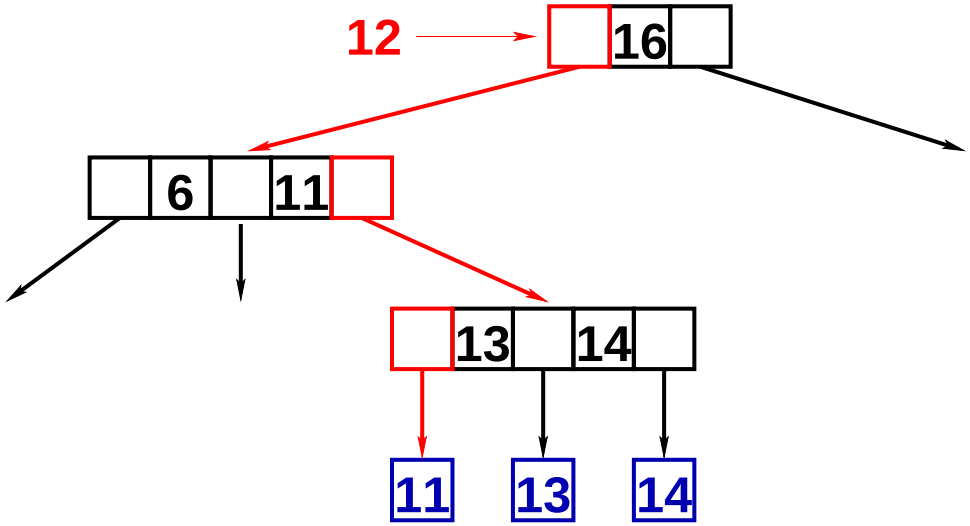
# Keresés 2-3-fában

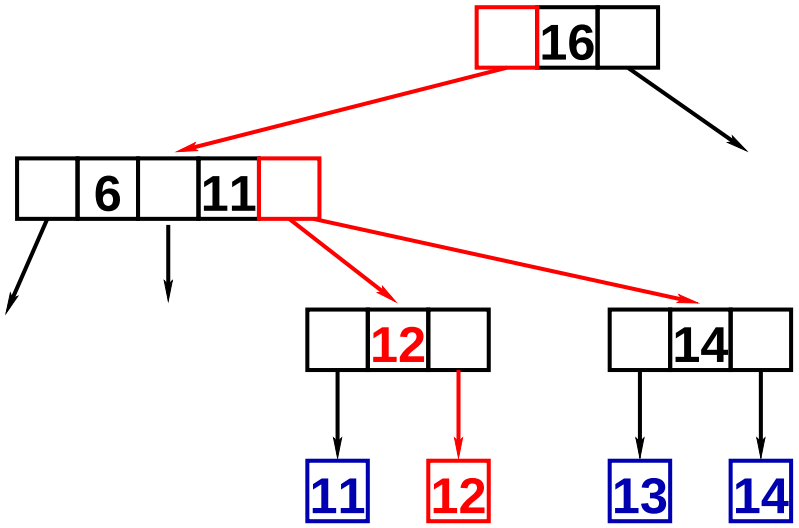


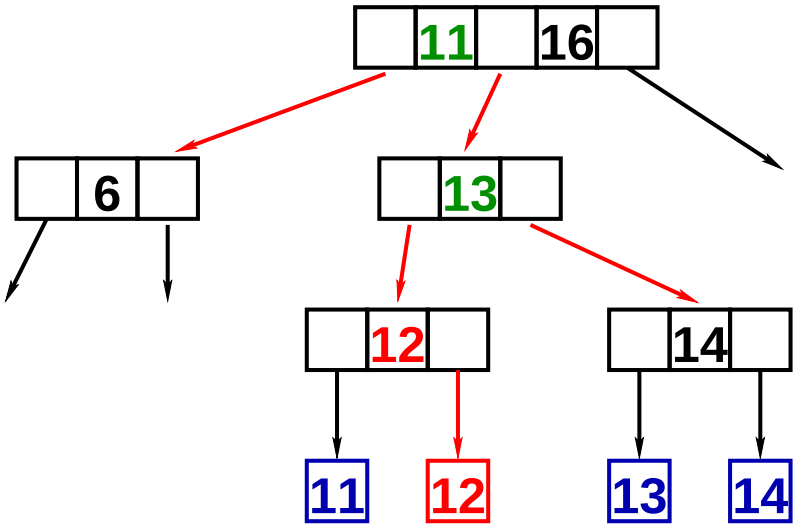

Hasonló, mint a bináris kereső fában.

Lépésszám: $O(m)$, ahol $\log_3(n) + 1 \leq m \leq \log_2(n) + 1$

# BESZÚR 2-3-fába




Ha a gyökeret is „vágni” kell $\Longrightarrow$ új gyökér, nő a fa magassága.

Lépésszám: $O(m)$, (minden szinten legfeljebb 1 „vágás”)

# TÖRÖL 2-3-fából

Legyen $x$ a legalsó belső csúcs a kereső út mentén.

- Ha $x$-nek három fia van $\Longrightarrow$ $\surd$
- ha $x$-nek csak két fia van
  - ⋆ ha $x$ (valamelyik) szomszédos testvérének 3 fia van $\Longrightarrow$ egyet átteszünk $x$ alá
  - ⋆ ha $x$ egyik szomszédos testvérének sincs három fia $\Longrightarrow$ „összevonunk” két kettes csúcsot

Ez is „felgyűrűzhet” $\Longrightarrow$

Lépésszám: $O(m)$

# $B$-fák

R. Bayer, E. McCreight, 1972
A 2-3-fa általánosítása.
Nagy méretű adatbázisok, külső táron levő adatok feldolgozására használják.
Több szabvány tartalmazza valamilyen változatát.

Probléma: Nem az összhasonlítás időigényes, hanem az adatok kiolvasása, de sokszor egy adat kiolvasásához amúgy is kiolvasunk több más adatot, egy lapot.

$\Longrightarrow$ A fa csúcsai legyenek lapok, a költség a lapelérések száma.

# $B$-fa definíciója

Egy *$m$-edrendű $B$-fa*, röviden $B_m$-fa egy gyökeres, (lefelé) irányított fa, melyre érvényesek az alábbiaknak:

a) A gyökér foka legalább 2, kivéve esetleg, ha a fa legfeljebb kétszintes.

b) Minden más belső csúcsnak legalább $\lceil \frac{m}{2} \rceil$ fia van.

c) A levelek a gyökértől egyforma messze vannak.

d) Egy csúcsnak legfeljebb $m$ fia lehet.

d) A tárolni kívánt rekordok itt is a fa leveleiben vannak; egy levélben a lapmérettől és a rekordhossztól függően több rekord is lehet, növekvő, láncolt listában.

A belső csúcsok hasonlítanak a 2-3-fák belső csúcsaira. Egy belső csúcs így néz ki:

| $m_0$ | $s_1$ | $m_1$ | $s_2$ | $m_2$ | $\ldots$ | $s_i$ | $m_i$ |
|---|---|---|---|---|---|---|---|

# A $B$-fa szintszáma

Tegyük fel, hogy egy $B$-fának $n$ levele és $k$ szintje van, és keressünk összefüggést e két paraméter között.
A kicsi fáktól eltekintve a gyökérnek legalább két fia van, a többi belső csúcsnak pedig legalább $\lceil \frac{m}{2} \rceil$.
$\Longrightarrow n \geq 2\lceil \frac{m}{2} \rceil^{k-2}$, $\Longrightarrow \log_{\lceil \frac{m}{2} \rceil} \frac{n}{2} + 2 \geq k$

$$k \leq \frac{\log_2 n - 1}{\log_2 \lceil \frac{m}{2} \rceil} + 2.$$

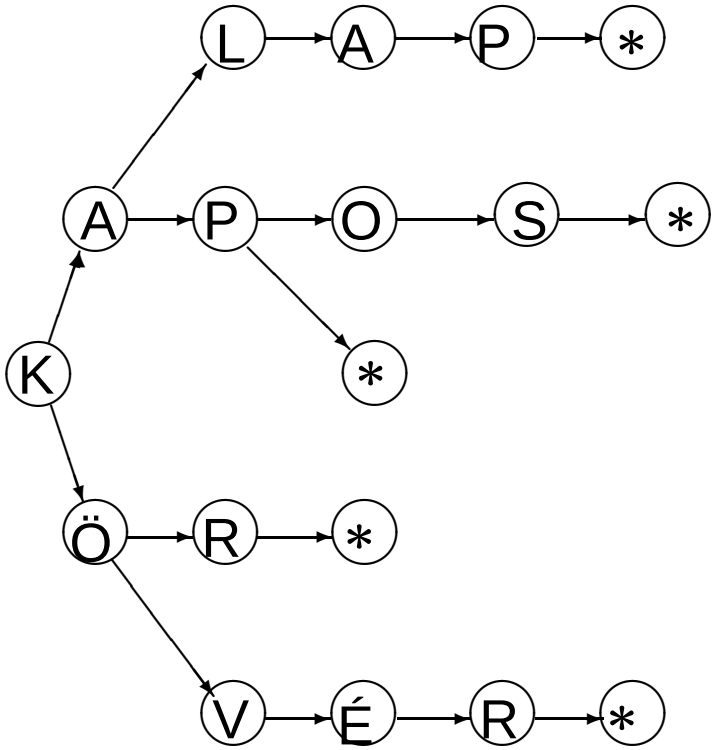
Minden művelet lépésszáma: $\sim \frac{\log_2 n - 1}{\log_2 \lceil \frac{m}{2} \rceil} = O(\log n)$, de a konstans kicsi, ha $m$ nagy.

Például: Például, ha $m = 32$, $n = 2^{20}$ (itt $n$ az alsó szint *lapjainak* száma), akkor $k \leq \frac{19}{4} + 2 < 7$. Egy rekord keresése tehát legfeljebb 6 lap elérését igényli.
Java animáció: $B$-fák

# Szófák

Legyen $\Sigma$ egy véges halmaz, $\Sigma^*$ a $\Sigma$-beli elemekből alkotott véges hosszú sorozatok.
$\Sigma^*$ egy részhalmazát (szavak egy halmazát) szeretnénk tárolni.

KÖR, KÖVÉR, KAPOS, KAP, KALAP $\Longrightarrow$




$*$ azt jelenti, hogy itt a szó vége
A szófa adatszerkezet érzéketlen a beszúrások sorrendjére, a fa alakja csak a tárolt szavak összességétől függ.

A tömbbel megvalósított szófában való keresés: a szóbeli betűk száma

# Hash-elés

# Hash-elés

Nem tételezzük fel a lehetséges kulcsok összességének (az $U$ univerzumnak) a rendezettségét.
Olyan módszercsalád, amely a keresés, beszúrás, törlés és módosítás gyors és egyszerű megvalósítását teszi lehetővé.

Nincs rendezés $\Longrightarrow$ nincs MIN, MAX, ...

Cél $\Longrightarrow$ $S \subseteq U$ kulcshalmazzal azonosított állomány megszervezése úgy, hogy a fenti műveletek átlagos értelemben hatékonyak legyenek.

Példa: Magyar állampolgárok személyi nyilvántartása
$\Longrightarrow$ kulcs= 11 jegyű személyi szám
lehetséges személyi számok $\Longrightarrow$ $4 \cdot 10^2 \cdot 12 \cdot 31 \cdot 10^3 \approx 148$ millió darab
elég lefoglalni 11 millió rekordnak helyet
Olyan $h$ függvény kell, ami minden személyi számhoz rendel egy egészet a $[0, 12 \cdot 10^6 - 1]$ intervallumból.
Jó lenne ha, $K \neq K'$ esetén $h(K) \neq h(K')$ teljesülne, de ez nem lehetséges. $\Longrightarrow$ ütközések elkerülhetetlenek

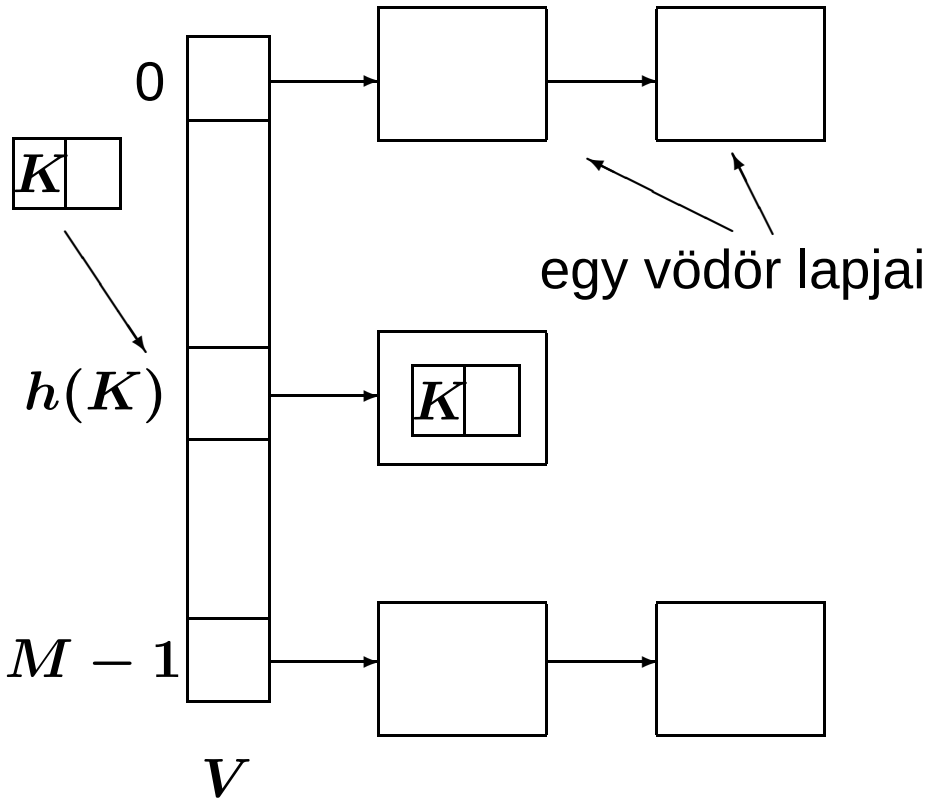
# Hash-elés alapvető ötelete

Veszünk egy alkalmas $h$ hash-függvényt, elsőnek a $K$ kulcsú elemet a $h(K)$ cellába próbáljuk illeszteni.
Ha később érkezik egy $K'$ elem, amire $h(K) = h(K')$, akkor ütközés van.
Az ütközések feloldására több módszer is van, próbálunk más helyet találni $K'$-nek.

Fontos kérdés a megfelelő hash függvény kiválasztása is, pl. $h(K) = konst.$ nyilván nem praktikus.

# Vödrös hash-elés

Főleg külső táron tárolt, nagy állományok kezelésére.
Minden elemet, amelyre $h(K) = i$ betesszük $V(i)$-be, ha több ilyen is van, láncolt listaként.
$V[0 : M - 1]$ vödörkatalógus, $V[i]$ mutató egy vödörbe, amiben az elemek listái (lapláncai) vannak. (A vödrök mérete általában kicsi.)



# Kulcsok a vödörben

Hogyan helyezzük az új kulcsot a vödörbe?

- Az első szabad helyre tesszük, ha kell új lappal bővítünk (az elején).
- Kulcs szerint rendezve vannak, beszúráskor a helyére tesszük.

## Keresés a hash-táblában

- Kiszámítjuk $h(K)$-t
- A $V[h(K)]$ vödörben keresünk szekvenciálisan, addig megyünk, amig megtaláljuk, vagy véget ér.

Törlés ugyanígy.

# Hash-elés költsége

Külső táras szerkezet $\Longrightarrow$ lapelérések száma.
$M$ vödör van, és $l$-lapnyi rekordot tárolunk
$\Longrightarrow$ egy vödörbe átlagosan $\approx l/M$ lap kerül
$\Longrightarrow$ átlagos lánchossz

$\Longrightarrow$ Keresés áltagos lépéshossza: $1 + l/M$

Hogyan válasszuk meg $M$-et?:

$l/M$ legyen kb. 1, de hagyjunk rá $20\%$-ot

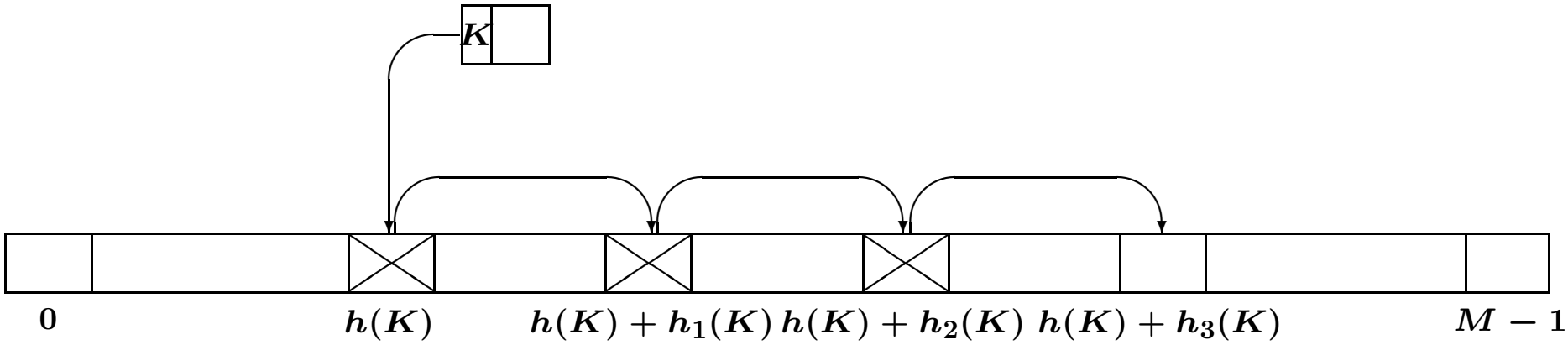
Példa: 1 000 000 rekordból álló állományt szeretnénk láncolásos módszerrel kezelni, egy lapon 5 rekord fér el.
Ekkor $l = 1\,000\,000/5 = 200\,000 \Longrightarrow M \approx 220\,000 - 240\,000$
$\Longrightarrow$ keresés átlagos költsége valamivel 2 lapelérés alatt marad.

# Hashelés nyitott címzéssel

Csak belső memóriás módszerként hasznosak.

Fő ötlet: ha $h(K)$ már foglalt, keresünk egy üreset valamilyen módszerrel.

Legyen $0, h_1(K), h_2(K), \ldots, h_{M-1}(K)$ a $0, 1, \ldots, M-1$ számok egy permutációja
$\Longrightarrow$ végigpróbálgatjuk a $h(K) + h_i(K) \pmod M$ sorszámú cellákat ($i = 0, 1, \ldots, M-1$) az első üres helyig, ahol a rekordot elhelyezzük.
$\Longrightarrow$ Ha nincs üres, a tábla betelt.



# Lineáris próbálás

$h_i(K) := -i$

Visszafelé lépkedünk egyesével $h(K)$-tól indulva az első üres helyig.

Sikeres keresés átlagos költsége:

$$C_N = \frac{1}{2}\left(1 + \frac{1}{1-\alpha}\right)$$

Sikertelen keresés átlagos költsége:

$$C'_N = \frac{1}{2}\left(1 + \left(\frac{1}{1-\alpha}\right)^2\right)$$

ahol $\alpha = N/M$ – a telítettségi (betöltöttségi) tényező, $N$ – a táblában levő rekordok száma, $M$ – a tábla celláinak száma

| $\alpha$ | $2/3$ | $0,8$ | $0,9$ |
|---|---|---|---|
| $C_N$ | $2$ | $3$ | $5,5$ |
| $C'_N$ | $5$ | $13$ | $50,5$ |

Példa: $M = 7, h(K) := K \pmod 7$, lineáris próba,
beillesztendő: $3, 11, 9, 4, 10$

| $0$ | $1$ | $2$ | $3$ | $4$ | $5$ | $6$ |
|---|---|---|---|---|---|---|
| $10$ | $4$ | $9$ | $3$ | $11$ | | |

Ha most töröljük a $9$-et, akkor később nem találnánk meg a $4$-et.
$\Longrightarrow$ $9$ helyére egy speciális TÖRÖLT jelet pl. $*$-ot teszünk. $\Longrightarrow$

| $0$ | $1$ | $2$ | $3$ | $4$ | $5$ | $6$ |
|---|---|---|---|---|---|---|
| $10$ | $4$ | $*$ | $3$ | $11$ | | |

Lineáris próba hátránya: Ha már sok cella tele van, kialakulnak egybefüggő csomók, megnő a keresési, beillesztési út $\Longrightarrow$ *elsődleges csomósodás*

Java animáció: Hash-elés lineáris próbával