Algoritmuselmélet

2-3 fák, B-fa, hash

Katona Gyula Y.

Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

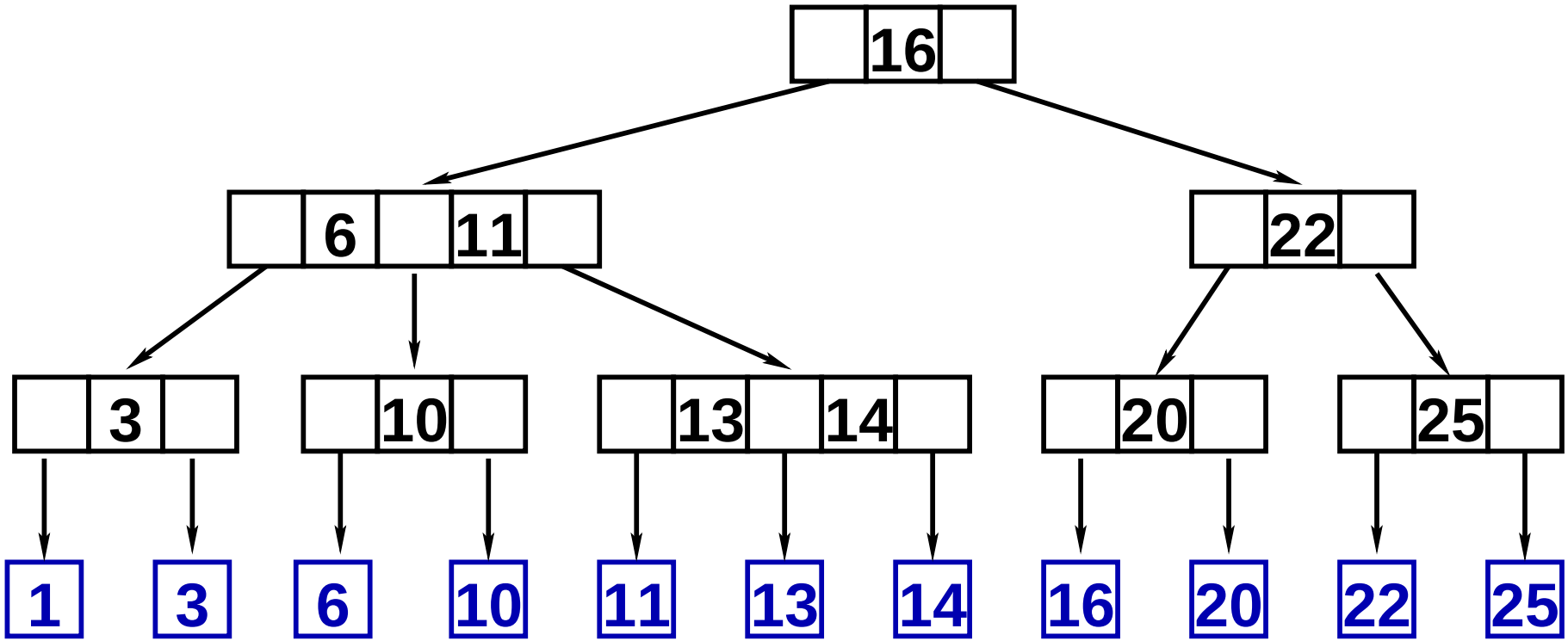
2-3-fák

Ez is fa, de a binárisnál bonyolultabb: egy nem-levél csúcsnak 2 vagy 3 gyereke 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.
- A fa levelei a gyökértől egyforma távolságra vannak.
- Minden belső (azaz nem levél) csúcsból 2 vagy 3 él megy lefelé; a belső csúcsok egy vagy két $s \in U$ útjelzőt tartalmaznak, ami a tárolt elemek egyike.
  - | 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.
  - A másik típusú csúcsoknál az utolsó két mező hiányzik:
    | m_1 | s_1 | m_2 |

Példa 2-3-fára



2-3-fa tulajdonságai

Tétel

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

Bizonyítás.

Minden belső csúcsnak legalább 2 gyereke van $\Longrightarrow$
az i-edik szinten legalább 2^{i-1} csúcs van ($1 \leq i \leq \ell$). $\Longrightarrow$
$2^{\ell-1} \leq n \Longrightarrow \ell - 1 \leq \log_2 n$. $\surd$ $\square$

2-3-fa tulajdonságai

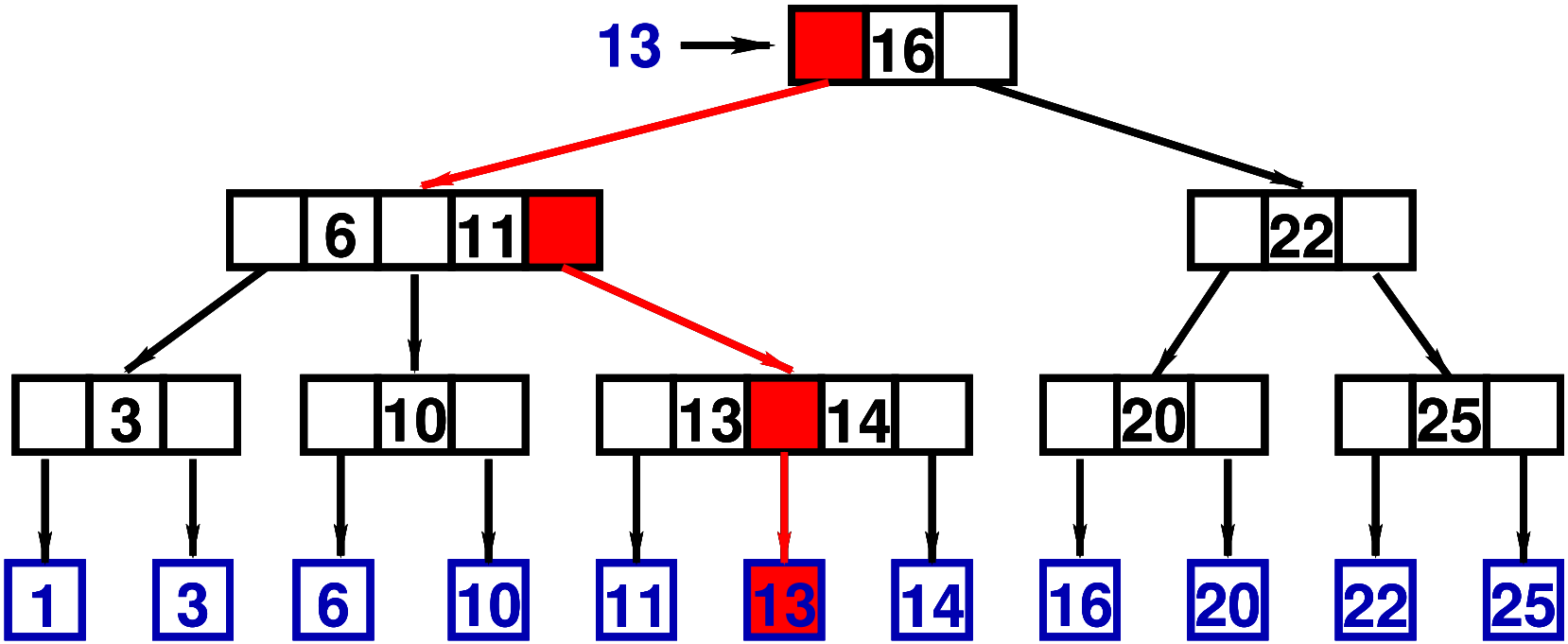
Tétel

Ha a fának ℓ szintje van, akkor a levelek száma legfeljebb $3^{\ell-1}$.
Megfordítva, $\ell \geq \log_3 n + 1$.

Bizonyítás.

Minden belső csúcsnak legfeljebb 3 gyereke van $\Longrightarrow$
az i-edik szinten legfeljebb 3^{i-1} csúcs van ($1 \leq i \leq \ell$). $\Longrightarrow$
$n \leq 3^{\ell-1} \Longrightarrow \ell - 1 \geq \log_3 n$. $\surd$ $\square$

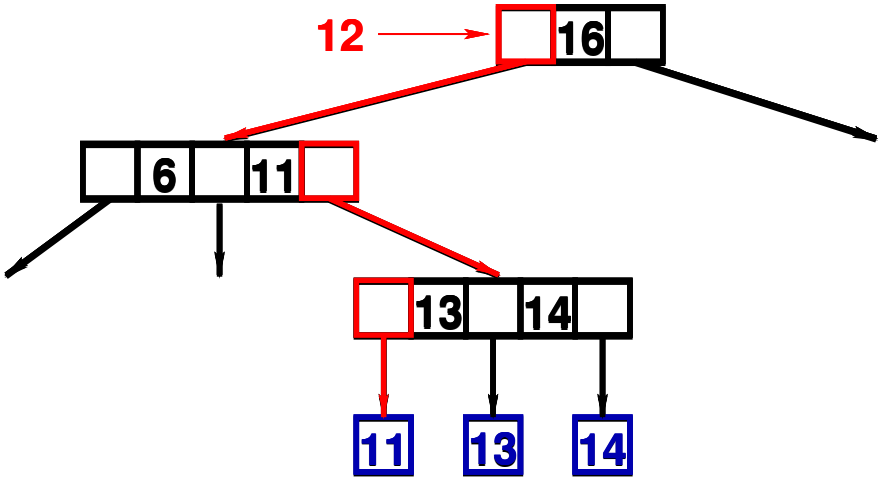
Keresés 2-3-fában

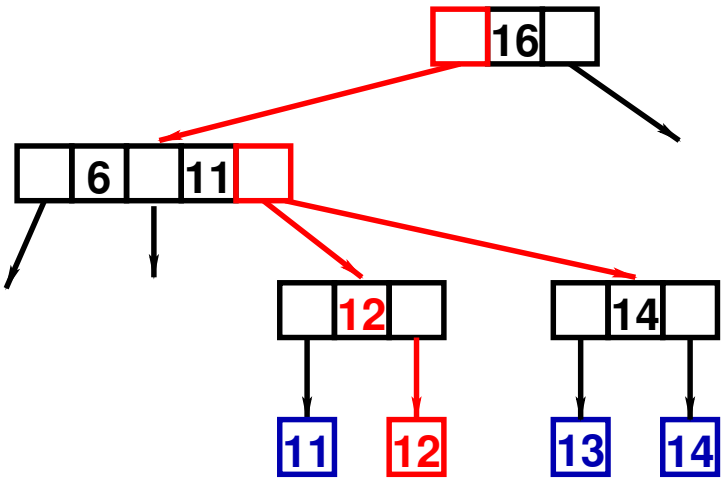


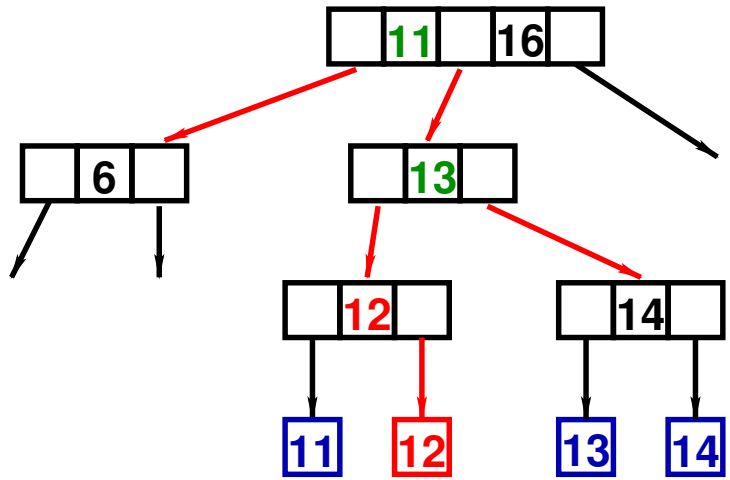
Hasonló, mint a bináris keresőfában.

Lépésszám: $O(\ell)$, ahol $\log_3(n) + 1 \leq \ell \leq \log_2(n) + 1$, vagyis a lépésszám $\Theta(\log n)$

BESZÚR 2-3-fába







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

Lépésszám: $O(\ell)$, minden szinten legfeljebb 1 „vágás”.

TÖRÖL 2-3-fából

Legyen x a legalsó belső csúcs a kereső út mentén.

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

Ez is „felgyűrűzhet”. $\Longrightarrow$

Lépésszám: $O(\ell)$

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 $\lceil \frac{m}{2} \rceil$ gyereke van.

c) A levelek a gyökértől egyforma messze vannak.

d) Egy csúcsnak legfeljebb m gyereke lehet.

d) A tárolni kívánt rekordok itt is a fa leveleiben vannak; egy levélben a lapmérettől és a rekordhossztól függően több rekord is lehet. (Egy lapon belül a lapoknak nem kell rendezettnek lennie, de a balra levő lapon minden elem legyen kisebb a jobbra levőknél.)

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 2-3 fa ennek speciális esete, egy B_3 fa.

A B-fa szintszáma

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

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

Minden művelet lépésszáma: $\sim \frac{\log_2 n - 1}{\log_2 \lceil \frac{m}{2} \rceil} = \Theta\left(\frac{\log n}{\log m}\right)$, azaz a konstans szorzó kicsi, ha m nagy.
m viszont nem lehet túl nagy, hiszen a belső csúcsoknak egy lapon el kell férniük.

Például: Ha $m = 1MB \approx 2^{22}$bit, $100TB \approx 2^{50}$bit adat, $n = 2^{28}$ az alsó szint *lapjainak* száma, akkor $\ell \leq \frac{27}{21} + 2 < 4$. Egy rekord keresése tehát legfeljebb 4 lap elérését igényli.
2-3 fában 50 lépés is lehet.

Hashelés

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

Hashelés 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.

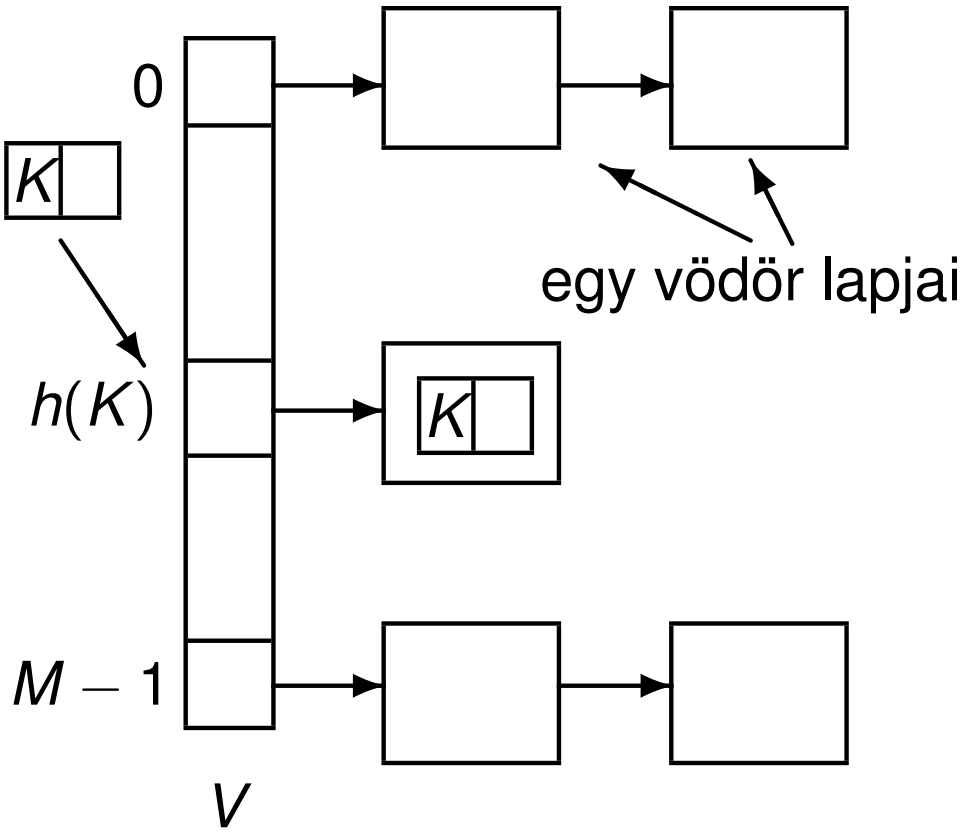
Fontos kérdés a megfelelő hash függvény kiválasztása is, pl.
$h(K) = konst.$ nyilván nem praktikus.
$h(K) = K$ pedig a ládarendezésnek felel meg, de akkor túl sok láda kell.

Vödrös hashelés

Főleg külső táron tárolt, nagy állományok kezelésére.

Minden elemet, amelyre $h(K) = i$ beteszünk $V(i)$-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? Kétféle módszer is lehetséges:

- Az első szabad helyre tesszük, ha kell, új lappal bővítünk (az elején), vagy
- 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, amíg megtaláljuk, vagy véget ér.

Törlés ugyanígy, a láncolt listából a törlés konstans lépésben végrehajtható.

Hashelés költsége

Külső táras szerkezet $\Longrightarrow$ lapelérések száma.
M vödör van, és ℓ-lapnyi rekordot tárolunk
$\Longrightarrow$ egy vödörbe átlagosan $\approx \ell/M$ lap kerül
$\Longrightarrow$ átlagos lánchossz: ℓ/M
$\Longrightarrow$ Keresés átlagos lépésszáma: $1 + \ell/M$

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 $\ell = 1\,000\,000/5 = 200\,000 \Longrightarrow M \approx 220\,000 - 240\,000$
$\Longrightarrow$ keresés átlagos költsége valamivel 2 lapelérés alatt marad.

Hashelés nyitott címzéssel

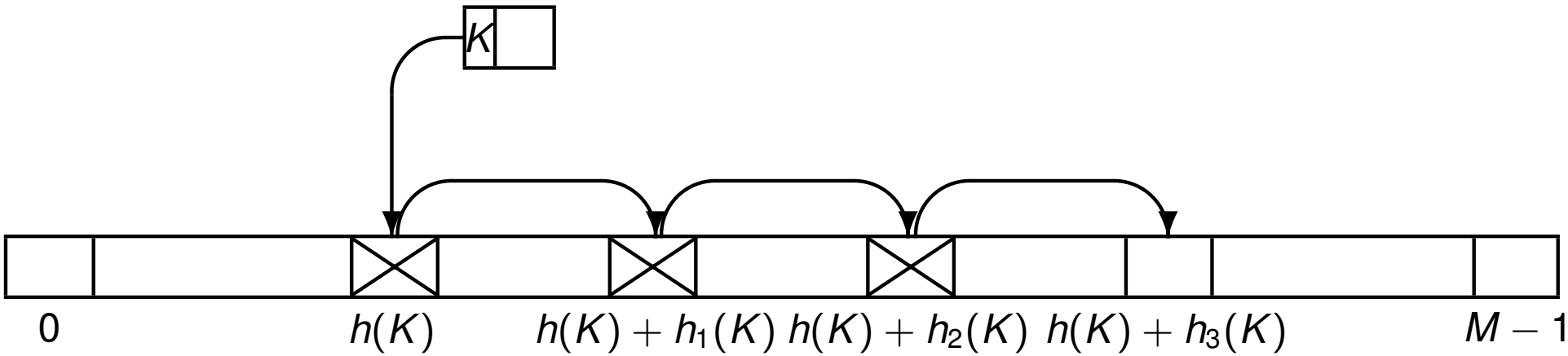
Csak belső memóriás módszerként hasznosak.

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.



Törlésnél egy $*$ jelet teszünk a cellába. A későbbi beszúrásoknál ez üresnek számít, viszont keresésnél foglaltnak.

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.

α	$2/3$	$0,8$	$0,9$
C_N	2	3	$5,5$
C'_N	5	13	$50,5$

Lineáris próbálás

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*
Animáció: Hashelés

Hashelés álvéletlen próbával

A $0, h_1(K), h_2(K), \ldots, h_{M-1}(K)$ próbasorozat a $0, 1, \ldots, M-1$ számoknak egy a K kulcstól független álvéletlen permutációja.
A sorozatnak gyorsan és hatékonyan reprodukálhatónak kell lennie, ezért nem lehet „valódi” véletlent használni.
Ha $h(K) = h(L)$, akkor a K és L kulcsok teljes próbasorozata is megegyezik. $\Longrightarrow$ másodlagos csomósodás

Kvadratikus maradék próba

Legyen M egy $4k+3$ alakú prímszám, ahol k egy egész.
Ekkor a próbasorozat legyen

$$0, 1^2, -(1^2), 2^2, -(2^2), \ldots, \left(\frac{M-1}{2}\right)^2, -\left(\frac{M-1}{2}\right)^2.$$

Belátható, hogy ez tényleg permutáció.
A lépésszámok itt is a telítettségtől függenek, de kedvezőbbek a lineáris próba lépésszámainál.

Kettős hashelés

G. de Balbine, J. R. Bell, C. H. Kaman, 1970 körül.
Lényeg: h mellett egy másik h' hash-függvényt is használunk de azt csak a próbasorozat előállításához.
A $h'(K)$ értékek relatív prímek legyenek az M táblaméretehez.
A K kulcs próbasorozata: $h_i(K) := -i \cdot h'(K)$.
Ha M és $h'(K)$ relatív prímek
$\Longrightarrow 0, -h'(K), -2h'(K), \ldots, -(M-1)h'(K)$ sorozat elemei mind különbözők modulo M.
Fontos sajátossága: különböző K és K' kulcsok próbasorozatai jó eséllyel akkor is különbözők lesznek, ha $h(K) = h(K')$.

Kettős hashelés

A legjobb ismert implementációk időigénye (empirikus adatok alapján)

$$C_N \approx \frac{1}{\alpha} \log \frac{1}{(1-\alpha)} \quad \text{és} \quad C'_N \approx \frac{1}{1-\alpha}.$$

- A kettős hashelés kiküszöböli mindkétféle csomósodást.
- Sikertelen keresés esetén minden érdekes α-ra gyorsabb, mint a lineáris próbálás.
- Sikeres kereséskor csak az $\alpha \geq 0,8$ tartományban lesz gyorsabb a lineáris próbálásnál.

Hash-függvények

- Legyen könnyen (gyorsan) számítható,
- és minél kevesebb ütközést okozzon.

A második követelmény elég nehezen megfogható, mert a gyakorlatban előforduló kulcshalmazok egyáltalán nem véletlenszerűek.
Hasznos tanácsok: $h(K)$ értéke lehetőleg a K kulcs minden bitjétől függjön és a h értékkészlete a teljes $[0, M-1]$ címtartomány legyen.

Például: Legyen $h(K) := K \pmod M$,
ahol M a tábla vagy a vödörkatalógus mérete.
Feltesszük, hogy a kulcsok egész számok.
A $h(K)$ számítása gyors és egyszerű.

A kettős hashelés második függvénye

Olyan h' függvény kell, melynek értékei a $[0, M-1]$ intervallumba esnek, és relatív prímek az M-hez.
Ha M prím $\Longrightarrow$

$$h'(K) := K \pmod{M-1} + 1.$$

$\Longrightarrow$ $h'(K)$ és M relatív prímek.
Mivel $h'(K)$ minden értéket felvesz 1 és $M-1$ között, ezért elég sok különböző próbasorozatot ad.