Algoritmuselmélet

Hashelés

Katona Gyula Y.

Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

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 $[0, 12 \cdot 10^6 - 1]$ intervallumból.
Jó lenne ha, $K \neq K'$ esetén $h(K) \neq h(K')$ teljesülne, de ez nem lehetséges. $\Longrightarrow$ ütközések elkerülhetetlenek

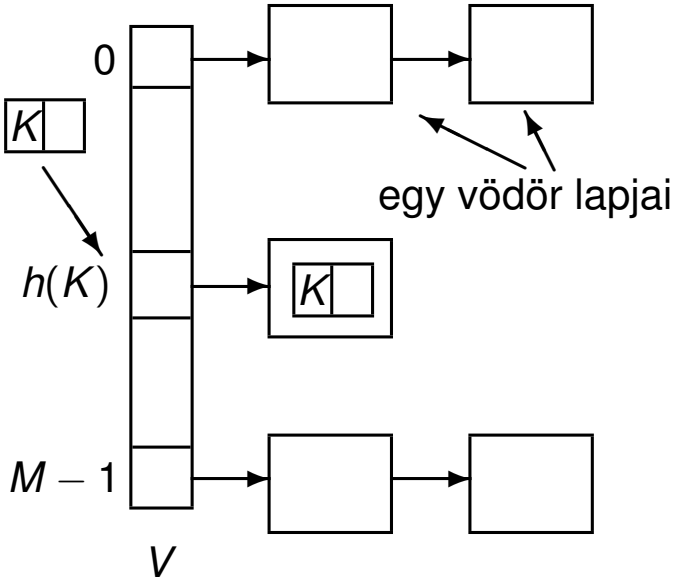
Hashelés alapvető ötelete

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.

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?

- 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, amíg megtaláljuk, vagy véget ér.

Törlés ugyanígy.

Hashelés 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: l/M
$\Longrightarrow$ Keresés átlagos lépésszáma: $1 + l/M$

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

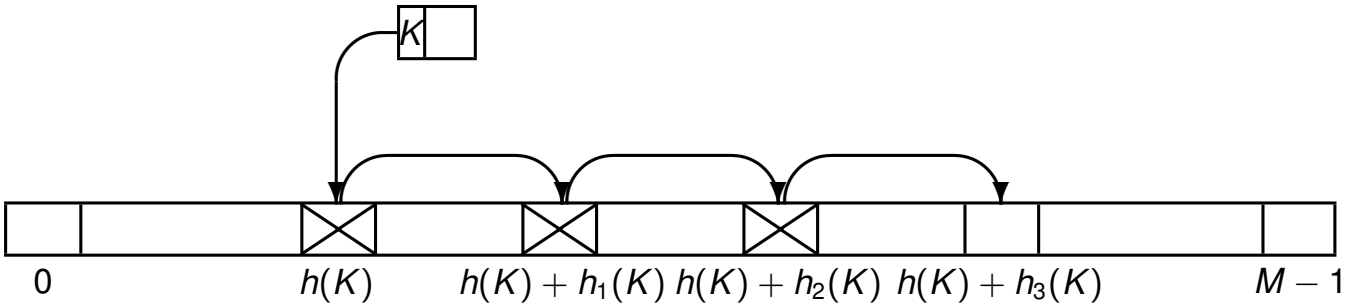
Csak belső memóriás módszerként hasznosak.

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.

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*
Java animáció: Hashelés lineáris próbával

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

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

Belátjuk, hogy ez tényleg permutáció.

Hashelés kvadratikus próbával

Lemma

Ha M egy $4k+3$ alakú prímszám, akkor nincs olyan n egész, melyre $n^2 \equiv -1 \pmod{M}$.

Bizonyítás.

Indirekt tegyük fel, hogy n egy egész szám és $n^2 \equiv -1 \pmod{M}$. $\Longrightarrow$

$$-1 = (-1)^{\frac{M-1}{2}} \equiv n^{2\frac{M-1}{2}} = n^{M-1} = n^{\varphi(M)} \equiv 1 \pmod{M}.$$

Az utolsó lépésnél az Euler-Fermat-tételt használtuk.  □

Hashelés kvadratikus próbával

Ha $0 \leq i < j \leq \frac{M-1}{2}$, akkor $i^2 \not\equiv j^2 \pmod{M}$. $\Longleftarrow j^2 - i^2 = (j-i)(j+i)$ felbontás egyik tényezője sem lehet osztható M-mel, tehát a szorzatuk sem. $\surd$

Ugyanígy $\Longrightarrow -i^2 \not\equiv -j^2 \pmod{M}$. $\surd$

A lemma miatt $(ij^{-1})^2 \not\equiv -1 \pmod{M}$, ahol j^{-1} a j elem inverze a $\pmod{M}$ maradékosztályok csoportjában a szorzásra nézve.

$\Longrightarrow i^2 \not\equiv -j^2 \pmod{M}$ $\surd$

Hashelés kvadratikus próbával

Sikeres keresés költsége:

$$C_N \approx 1 - \log(1-\alpha) - \frac{\alpha}{2}$$

Sikertelen keresés költsége:

$$C'_N \approx \frac{1}{(1-\alpha)} - \alpha - \log(1-\alpha)$$

Ezek az összefüggések valamivel általánosabban érvényesek az olyan módszerekre, amelyekre $h_i(K) = f_i(h(K))$, vagyis ahol a $h(K)$ érték már az egész próbasorozatot meghatározza.

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 táblaméréthez.
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.

Két gyakran használt módszer hash-függvény előállítására az osztó- és a szorzómódszer.

Osztómódszer

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 tábla mérete sem teljesen közömbös.
Például ha M a 2 egy hatványa, akkor $h(K)$ csak a kulcs utolsó néhány bitjétől függ.
A jó M értékeket illetően van egy széles körben elfogadott recept:
D. E. Knuth javaslata: M-et prímnek választjuk, úgy, hogy M nem osztja $r^k \pm a$-t, ahol r a karakterkészlet elemszáma (pl. 128, vagy 256) és $a,\ k$ „kicsi" egészek.
M prím:

- Lényeges feltétel a kvadratikus maradék próbánál.
- Kettős hashelésnél könnyű hozzájuk relatív prím számot találni .

Szorzómódszer

β egy rögzített paraméter.

$$h(K) := \lfloor M \cdot \{\beta K\} \rfloor.$$

$\{x\}$ jelöli az x valós szám törtrészét.
Szemléletesen: $\{\beta K\}$ kiszámításával a K kulcsot „véletlenszerűen" belőjük a $[0, 1)$ intervallumba, majd az eredményt felskálázzuk a címtartományba.
Hatékonyan számítható speciális eset:
$M = 2^t$, $w = 2^{32}$, és legyen A egy a w-hez relatív prím egész.
Ekkor $\beta = \frac{A}{w}$ választás mellett $h(K)$ igen jól számolható.
A számok bináris ábrázolásával dolgozva lényegében egy szorzást és egy eltolást kell elvégezni.
A szorzómódszer jól viselkedik számtani sorozatokon.
pl. termék1, termék2, termék3, . . . esetében.
Megmutatható, hogy a $h(K), h(K + d), h(K + 2d) \ldots$ sorozat közelítőleg számtani sorozat lesz, azaz h jól „szétdobja" a kulcsok számtani sorozatait.

Szorzómódszer

Tétel (T. Sós Vera, 1957)

Legyen β irracionális szám, és nézzük a $0, \{\beta\}, \{2\beta\}, \ldots, \{n\beta\}$ pontok által meghatározott $n+1$ részintervallumot $[0,1)$-ben. Ezek hosszai legfeljebb 3 különböző értéket vehetnek fel, és $\{(n+1)\beta\}$ a leghosszabbak egyikét fogja két részre vágni.

Következmény: A szorzómódszer esetén mindig elég egyenletesen lesznek szétszórva a hashértékek (ha a bemenet egy számtani sorozat).

A $[0,1)$-beli számok közül a legegyenletesebb eloszlást a $\beta = \phi^{-1} = \frac{\sqrt{5}-1}{2} = 0.618033988\ldots$ és a $\beta = \phi^{-2} = 1 - \phi^{-1}$ értékek adják.
$\Longrightarrow$ Érdemes a szorzómódszernél az A-t úgy választani, hogy $\frac{A}{w}$ közel legyen ϕ^{-1}-hez. $\Longrightarrow$ *Fibonacci-hashelés*

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.
Java animáció: Hashelés