Hashelés

A hashelés/hasítás célja egy olyan adattárolási eljárás megvalósítása, amelyben a keresés, beszúrás és törlés (szótár)műveletek várhatóan nagyon hatékonyak. Feltesszük, hogy a tárolandó rekordoknak van egy egyedi kulcs adattagja. A kulcsok egy U halmaz elemei (U a kulcsok univerzuma), amiről ezúttal nem szükséges feltételezni, hogy rendezhető.

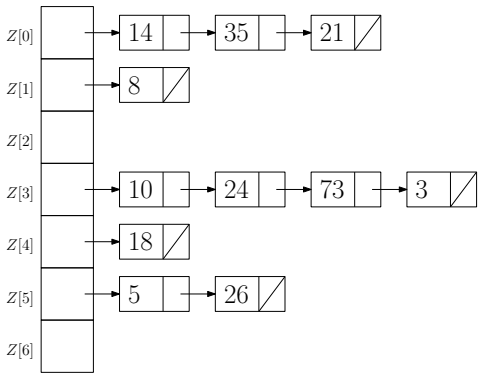
A legegyszerűbb megvalósítás az ún. direkt címzés. Ezt akkor használhatjuk, ha a kulcshalmaz nagyon kicsi. Ha $U = 0..m-1$, akkor egy m méretű tömbben/hasítótáblában tárolhatjuk a rekordokra mutató pointereket úgy, hogy a tömb k indexű tagja a k kulcsú rekordra mutat. Amennyiben nincs k kulcsú rekord, a megfelelő pointer értéke NULL.

Amennyiben a kulcshalmaz nagy, a direkt címzés nem alkalmazható. Ilyenkor felveszünk egy m részre osztott $Z[0..(m-1)]$ hasítótáblát, és bevezetünk egy $h : U \to 0..m-1$ hasító függvényt (tipikusan $|U| >> m$). A k kulcsú rekordot a hasítótábla $Z[h(k)]$ részében próbáljuk meg eltárolni. Ha két adat k_1 és k_2 kulcsára $h(k_1) = h(k_2)$, akkor kulcsütközésről beszélünk. A különféle hasítási eljárások a h függvény megválasztásában és a kulcsütközések kezelésében különböznek egymástól.

1. Láncolt/vödrös hashelés

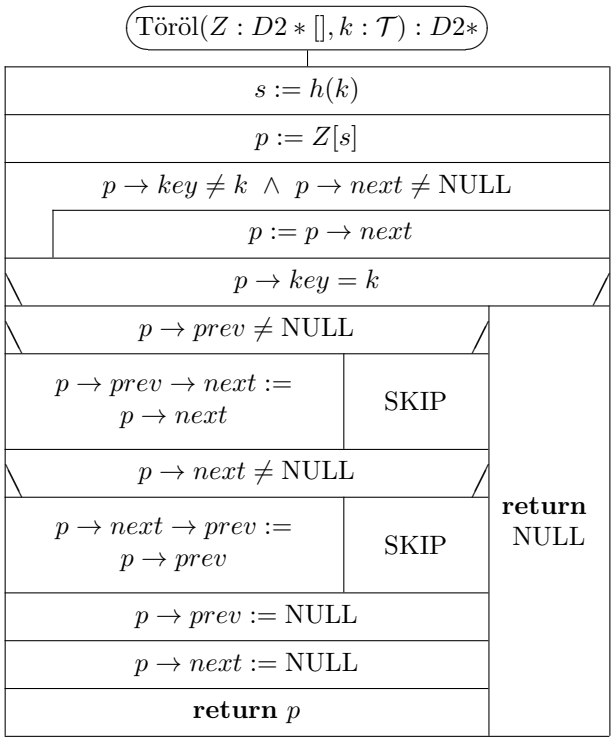
Az adatokat m darab S1L listában tároljuk. A hashtábla $Z[i]$ eleme az i. láncolt lista első elemére mutató pointer. Ez a lista azokat a k kulcsú elemeket tartalmazza, amikre $h(k) = i$. Beszúráskor a megfelelő lista első elejére szúrjuk be az új elemet azért, hogy a beszúrás konstans műveletigényű legyen.

Példa: $m = 7$, $h : \mathbb{N} \to 0..6$, $h(k) = k \bmod 7$. A táblába beszúrt elemek sorban: 3, 21, 73, 8, 24, 35, 26, 14, 5, 10, 18.



Feladat: Készítsük el a láncolt hashelés kétirányú listás változatát, és rajzoljuk fel a törlés művelet struktogramját! A kétirányú lista nem ciklikus.

A hashelés láncolt megvalósításánál a korábban tanult E_1, E_2 adattípusok helyett a D_1 és D_2 adattípusokat használjuk. Ezek azonban mindenben megegyeznek a korábbi változatukkal.



2. Nyílt címzéses hashelés

Ebben az esetben nem engedjük meg azt, hogy a hashtábla egy eleméhez egynél több rekord is tartozzon. Ekkor a rekordokat tárolhatjuk közvetlenül a hashtáblában, azonban foglalkoznunk kell a kulcsütközésekkel. Ennek érdekében a kulcsokhoz most már nem egyetlen hash értéket vezetünk be, hanem egy m-tagú sorozatot, ahol m a hashtábla mérete. Legyen $h : U \times 0..(m-1) \to 0..m-1$ egy olyan függvény, amelyre teljesül, hogy tetszőleges k kulcsra a $h(k,0), h(k,1), ..., h(k,m-1)$ sorozat a $0..m-1$ számok egy permutációja. h-t hasító/hash függvénynek nevezzük, az előbbi sorozatot pedig k próbasorozatának hívjuk. Általában olyan próbasorozatot választunk, amely az első eleméből könnyen rekonstruálható. Az első elemet $h_1(k)$-val is jelölhetjük. Mi az órán az ún. lineáris, kvadratikus, és kettős hasítás nevű módszerekre nézünk példát.

Beszúrás: Ha a k kulcsú elemet akarjuk beszúrni a táblába, akkor először megnézzük, hogy szabad-e a $h(k,0)$ indexű pozíció. Amennyiben igen, beszúrjuk, ha nem, akkor megnézzük szabad-e a $h(k,1)$ pozíció. És így tovább, egészen addig, amíg nem találunk neki egy szabad helyet.

Keresés: Amennyiben meg szeretnénk keresni a k kulcsú elemet, úgy elkezdjük sorban végignézni a $h(k,0)$, $h(k,1)$,..., $h(k,m-1)$ indexű pozíciókat egészen addig, míg nem nem találjuk a keresett elemet, vagy üres helyre nem akadunk. Ha üres helyet találtunk, akkor nincs értelme tovább folytatni a keresést.

Törlés: Egy elem törlése során először megkeressük az előző eljárással a helyét. Fontos, hogy a törlésnél nem változtathatjuk a megtalált pozíciót üresre, hiszen ez gondot okozhatna a későbbi kereséseknél. Épp ezért a törölt elemek helyét valójában nem szabadítjuk fel, hanem az addig ott tárolt kulcsot lecseréljük egy U halmazban nem szereplő D (deleted) szimbólumra.

2.1. Lineáris próba

Lineáris próba esetén a próbasorozat egy számtani sorozatot alkot, ennek differenciája a leggyakrabban 1. Tehát $h(k,n) = h(k,0) + nd \bmod m$.

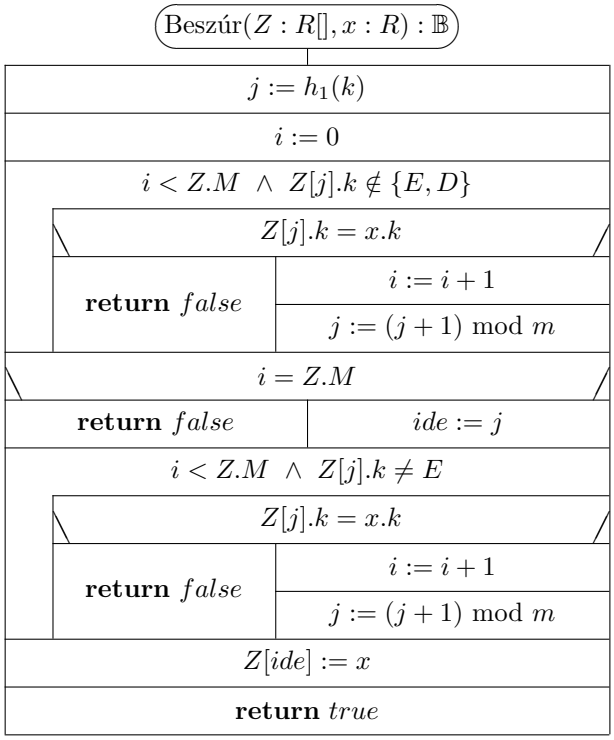
Példa: $m = 11$, $h : \mathbb{N} \to 0..10$, $h(k,n) = (k+n) \bmod 11$.

Művelet	k	$h_1(k)$	Próbasorozat	0	1	2	3	4	5	6	7	8	9	10
Beszúr	24	2	2_E ✓			24								
Beszúr	16	5	5_E ✓			24			16					
Beszúr	57	2	$2, 3_E$ ✓			24	57		16					
Beszúr	32	10	10_E ✓			24	57		16					32
Beszúr	15	4	4_E ✓			24	57	15	16					32
Töröl	57	2	$2, 3_{57}$ ✓			24	D	15	16					32
Beszúr	21	10	$10, 0_E$ ✓	21		24	D	15	16					32
Keres	2	2	$2, 3_D, 4, 5, 6_E$ ✗	21		24	D	15	16					32
Beszúr	2	2	$2, 3_D, 4, 5, 6_E$ ✓	21		24	2	15	16					32
Beszúr	2	2	$2, 3_2$ ✗	21		24	2	15	16					32
Keres	21	10	$10, 0_{21}$ ✓	21		24	2	15	16					32

A lépések lejátszása után határozzuk meg a hashtábla kitöltöttségi arányszámát: $\alpha = \frac{6}{11}$

Esetleg meg lehet kérdezni, hogy a fenti műveletek végrehajtása után mennyi egy sikeres keresés várható lépésszáma, de ehhez számológép szükséges. Az előadáson tanult képlet szerint ez $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$. Ennek értéke jelenleg körülbelül 1,73.

Feladat: Készítsük el a beszúrás struktogramját! A hasítótáblába illesztett adatok típusa R (rekord), ami rendelkezik egy k nevű kulcs adattaggal. Az algoritmus visszatérési értéke egy logikai érték, ami megmondja, hogy sikeres volt-e a beszúrás.



2.2. Négyzetes próba

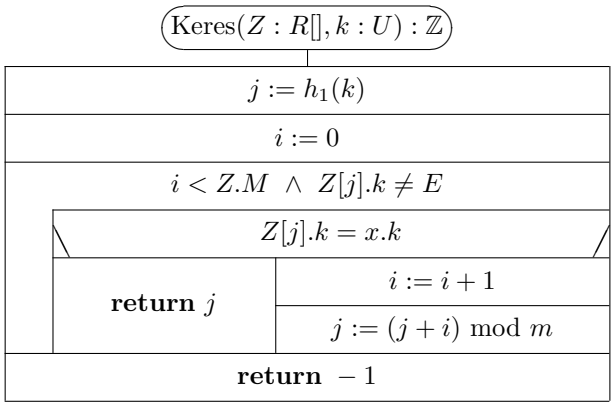
Négyzetes próba esetén a próbasorozat egy másodfokú függvény segítségével írható le, vagyis $h(k,n) = (h(k,0) + c_1 \cdot n + c_2 \cdot n^2) \bmod m$ valamilyen c_1 és c_2 konstansokra. (Ezeket a konstansokat úgy kell megválasztani, hogy a próbasorozat a teljes táblát kiadja. Például kettőhatvány m esetén a $c_1 = c_2 = \frac{1}{2}$ egy jó választás, ha pedig m egy $4k+3$ alakú prím, akkor bármilyen c_2 jó lesz $c_1 = 0$ esetén.) $c_1 = c_2 = \frac{1}{2}$ esetén $h(k,n) = h(k,n-1) + n$.

Példa: $m = 8$, $h : \mathbb{N} \to 0..7$, $h(k,n) = (k + \frac{n+n^2}{2}) \bmod 8$.

Művelet	k	$h_1(k)$	Próbasorozat	0	1	2	3	4	5	6	7
Beszúr	13	5	5_E ✓						13		
Beszúr	20	4	4_E ✓					20	13		
Beszúr	31	7	7_E ✓					20	13		31
Beszúr	87	7	$7, 0_E$ ✓	87				20	13		31
Beszúr	12	4	$4, 5, 7, 2_E$ ✓	87		12		20	13		31
Töröl	31	7	7_{31} ✓	87		12		20	13		D
Keres	12	4	$4, 5, 7_D, 2_{12}$ ✓	87		12		20	13		D
Keres	15	7	$7_D, 0, 3_E$ ✗	87		12		20	13		D
Beszúr	4	4	$4, 5, 7_D, 2, 6_E$ ✓	87		12		20	13		4
Töröl	10	2	$2, 3_E$ ✗	87		12		20	13		4
Beszúr	35	3	3_E ✓	87		12	35	20	13		4
Beszúr	10	2	$2, 3, 5, 0, 4, 1_E$ ✓	87	10	12	35	20	13		4

$\alpha = \frac{7}{8}$

Feladat: Készítsük el a keresés struktogramját! Az algoritmus visszatérési értéke egy egész szám, ami megadja a paraméterül kapott kulcsú rekord indexét. Sikertelen keresés esetén a visszatérési érték -1.



2.3. Kettős hasítás

Ebben az esetben a próbasorozat egy számtani sorozat, azonban a differencia kulcsonként eltérő. Az első elemet egy $h_1 : U \to 0..m-1$, a számtani sorozat differenciáját pedig egy $h_2 U \to 0..m-1$ függvény adja meg. Ezekből a h hasítófüggvényt a $h(k,n) = (h_1(k) + n \cdot h_2(k)) \bmod m$ képlet adja meg.

Példa: $m = 11$, $h_1(k) = k \bmod 11, h_2(k) = 1 + (k \bmod 10)$.

Művelet	k	$h_1(k)$	$h_2(k)$	Próbasorozat	0	1	2	3	4	5	6	7	8	9	10
Beszúr	23	1	4	1_E ✓		24									
Beszúr	42	9	3	9_E ✓		24								42	
Beszúr	31	9	2	$9, 0_E$ ✓	31	24								42	
Beszúr	110	0	1	$0, 1, 2_E$ ✓	31	24	110							42	
Beszúr	55	0	6	$0, 6_E$ ✓	31	24	110				55			42	
Töröl	55	0	6	$0, 6_{55}$ ✓	31	24	110				D			42	
Keres	72	6	3	$6_D, 9, 1, 4_E$ ✗	31	24	110				D			42	
Beszúr	14	3	5	3_E ✓	31	24	110	14			D			42	
Töröl	22	3	5	$0, 3, 6_D, 9, 1, 4_E$ ✗	31	24	110	14			D			42	
Töröl	110	0	1	$0, 1, 2_{110}$ ✓	31	24	D	14			D			42	
Beszúr	13	2	4	$2_D, 6_D, 10_E$ ✓	31	24	13	14			D			42	

3. A hasítófüggvény megválasztása

Ha marad idő a gyakorlat végén, akkor átismételhetjük, hogy milyen jó hasítófüggvényeket tanultak előadáson. Az alábbiakban a jegyzet megfelelő részét idézzük:

A $h : U \to 0..(m-1)$ függvény egyszerű egyenletes hasítás, ha a kulcsokat a rések között egyenletesen szórja szét, azaz hozzávetőleg ugyanannyi kulcsot képez le az m rés mindegyikére. Tetszőleges hasító függvénnyel kapcsolatos elvárás, hogy egyszerű egyenletes hasítás legyen.

3.1. Osztó módszer

Ha a kulcsok egész számok, gyakran választják a

$$h(k) = k \bmod m$$

hasító függvényt, ami gyorsan és egyszerűen számolható, és ha m olyan prím, amely nincs közel a kettő hatványokhoz, általában egyenletesen szórja szét a kulcsokat a 0..(m-1) intervallumon.

Ha pl. a kulcsütközést láncolással szeretnénk feloldani, és kb. 2000 rekordot szeretnénk tárolni $\alpha \approx 3$ kitöltöttségi aránnyal, akkor a 701 jó választás: A 701 ui. olyan prímszám, ami közel esik a 2000/3-hoz, de a szomszédos kettőhatványoktól, az 512-től és az 1024-től is elég távol van.

3.2. Kulcsok a $[0, 1)$ intervallumon:

Ha egyenletesen oszlanak el, a

$$h(k) = \lfloor k \cdot m \rfloor$$

függvény is kielégíti az egyszerű, egyenletes hasítás feltételét.

3.3. Szorzó módszer:

Ha a kulcsok valós számok, tetszőleges $0 < A < 1$ konstanssal alkalmazható a

$$h(k) = \lfloor \{k \cdot A\} \cdot m \rfloor$$

hasító függvény. ($\{x\}$ az x törtrésze.) Nem minden lehetséges konstanssal szór egyformán jól. Knuth az $A = \frac{\sqrt{5}-1}{2} \approx 0,618$ választást javasolja, mint ami a kulcsokat valószínűleg szépen egyenletesen fogja elosztani a rések között. Az osztó módszerrel szemben előnye, hogy nem érzékeny a hasító tábla méretére.