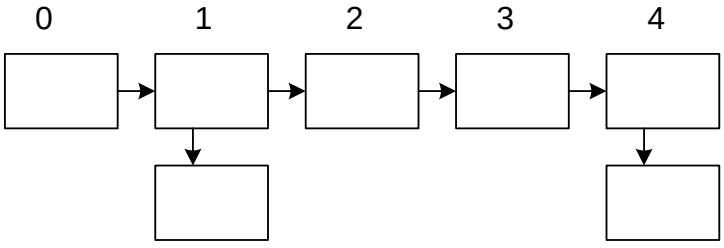
Lineárne hešovanie (linear hashing)

Obmedzenie doteraz známeho (statického) hešovania:

- **Vyhovuje iba pre súbory pevnej (vopred známej) veľkosti, lebo s touto veľkosťou je spätá hešovacia funkcia. Pri dynamickom raste efektívnosť degraduje, rekonštrukcia je ťažká až často aj nemožná.**
- **Pevná veľkosť súboru bez ohľadu na reálne vložené dáta.**

Myšlienka lineárneho hešovania:

- **pri raste a zmenšovaní súboru sa uloženie dát automaticky reorganizuje – veľkosť súboru sa mení podľa skutočne vložených dát**
- **zachováva okamžitý prístup k dátam v súbore (rovnako ako statické hešovanie)**
- **dovoľuje expanziu bez reorganizácie (ako dynamické a rozšíriteľné hešovanie)**
- **priestor zväčšuje postupne (nie skokom)**
- **zväčšenie adresovacieho priestoru <u>zmenou hešovacej funkcie</u>**
- **nevyžaduje priestor na index (adresár)**
- **používa preplňovacie bloky => hešovacia funkcia mapuje kľúče do čísel skupín blokov (bloky v skupine sú zreťazené)**



- **skupiny sa rozdeľujú (split) alebo spájajú pri splnení voliteľnej podmienky: najčastejšie je to prekročenie dolnej resp. hornej hranice hustoty súboru *d = n/N* (počet obsadených miest k počtu alokovaných miest)**

- **skupiny sa rozdeľujú (spájajú) vždy v pevnom ***<u>lineárnom</u>*** poradí bez ohľadu na situáciu v nich (odtiaľ názov) - nemusí sa rozdeliť práve preplnený blok**

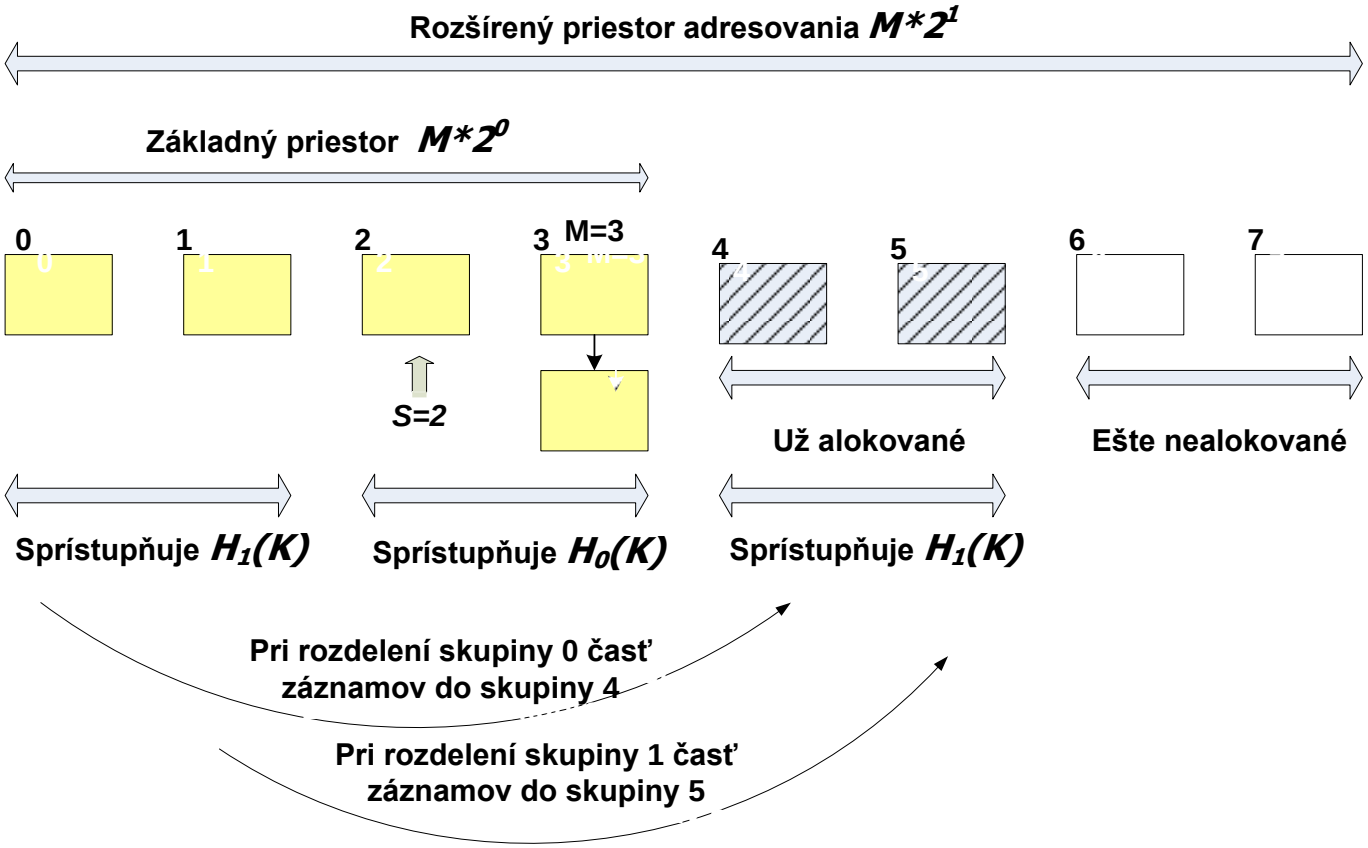
- **počet skupín blokov na začiatku sa zvolí *M,* kde *M* je zvyčajne násobok druhej mocniny (napr. *M = 4*, čiže *M = 4*2^0*), vtedy je súbor na úrovni *u = 0*. Adresy blokov: 0,1,2,3.**

- **na každej úrovni *u* súboru sa používajú 2 hešovacie funkcie:**
  $H_u(K) = K \bmod M*2^u$ (K mod 4 ... 2 posledné bity)
  $H_{u+1}(K) = K \bmod M*2^{u+1}$ (K mod 8 ... 3 posledné bity)

  $H_u(K)$ adresuje základný adresový priestor skupín blokov;
  $H_{u+1}(K)$ adresuje rozšírený (dvojnásobný) priestor, ktorý sa však alokuje postupne podľa potreby.

Keď sa alokuje všetkých $M*2^{u+1}$ blokov, nastane tzv. „úplná expanzia“: vtedy sa zväčší úroveň (*u = u + 1*) a začne sa používať ďalšia dvojica hešovacích funkcií. Analogicky pri zmenšovaní súboru.

- **tzv. split pointer *S* označuje skupinu blokov, ktorá sa má rozdeliť pri splnení zvolených podmienok; na začiatku sa nastaví *S = 0,* po rozdelení sa prejde na *S = S + 1*, po úplnej expanzii sa znovu nastaví *S = 0.***



Algoritmus nájdi záznam s kľúčom K:

V každom okamihu sú použité len dve hešovacie funkcie!

Najskôr sa na získanie indexu použije funkcia $H_u(K)$. Ak je výsledok menší ako S, tak použije funkcia $H_{u+1}(K)$. Na príslušnom idexe sa hľadá záznam.

Algoritmus vlož:

Predpoklad:

- **pevná hodnota *M* (počet skupín na začiatku)**

1. **Urči skupinu *i*, do ktorej sa záznam mapuje s použitím $i = H_u(K)$.**
2. **Ak $i < S$ (skupina *i* už bola rozdelená), tak i = $H_{u+1}(K)$.**
3. **Vlož záznam do skupiny *i* .**
4. **Ak teraz hustota *d* prekročila stanovenú hranicu, tak:**
   a) Vytvor nový blok s adresou $a = S + M*2^u$
   b) Záznamy v skupine *S*, pre ktoré $H_{u+1}(K) <> S$ prelož do skupiny *a*
   c) Aktualizuj parametre: $S = S + 1$;
   Ak $S >= M*2^u$ (došlo k úplnej expanzii), tak:
   $S = 0,\ u = u + 1$

V prípade, že dôjde ku kolízii, je možné ju riešiť pomocou preplňujúceho súboru.

Príklad:

Blokovací faktor:

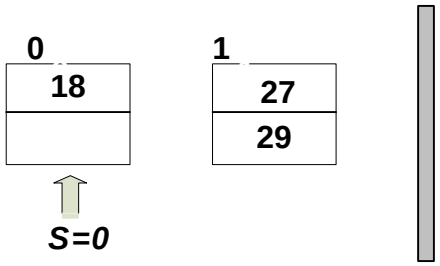
- **primárne bloky ... 2**
- **preplňujúce bloky ... 1**

Horná hranica hustoty dmax = 0,8.
Budeme vkladať záznamy s kľúčmi:
27 ... 11011

18 ... 10010
29 ... 11101
28 ... 11100
39 ... 100111
13 ... 01101
16 ... 10000
51 ..110011
19 ... 10011

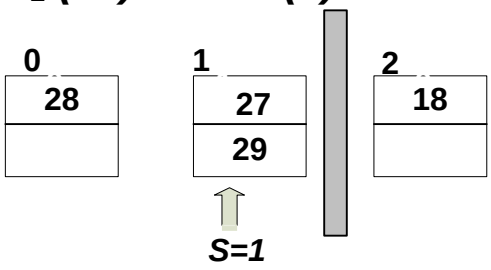
***M* = 2; $H_0(K) = K \bmod 2$; *S* = 0; *u = 0* ;**

Prvé 3 záznamy sa vložia priamo: párne do bloku 0, nepárne do bloku 1 ..... $H_0(27) = 1$; $H_0(18) = 0$; $H_0(29) = 1$;



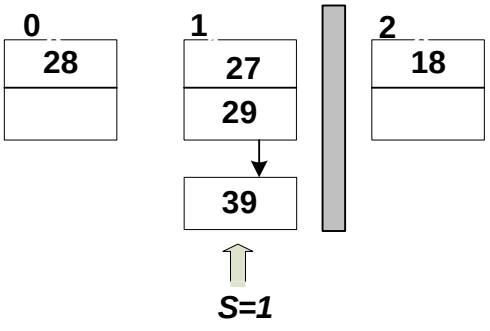
Záznam s kľúčom 28 vložíme do bloku 0, lebo $H_0(28) = 0$.
Po vložení hustota d = 1 > dmax => rozdelíme blok 0 (*S* = 0)
s použitím funkcie $H_1(K) = K \bmod 4$ a inkrementujeme *S*.

$H_1(28) = 00$ (0) $H_1(18) = 10$ (2)



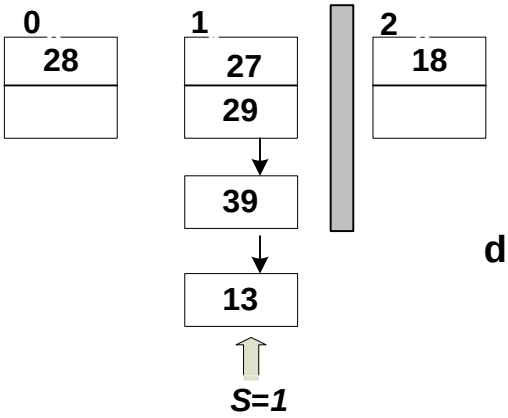
Vyhľadajme teraz záznam s kľúčom 18:
$H_0(18) = 0 < S$* , teda vyhľadávame v bloku *$H_1(18) = 2$.

Vložme 39. *$H_0(39) = 1$.* Skupina 1 nebola rozdelená, preto nepoužijeme funkciu *H_1*. Blok 1 je však plný, preto pripojíme preplňujúci blok.



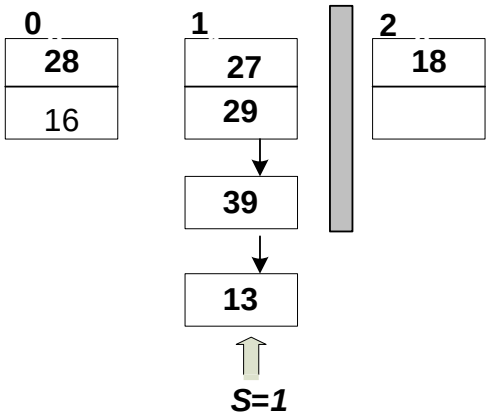
d = 0,71 < dmax **…. nerozdeľujeme**

13 vložíme do bloku 1, lebo *H_0 (13) =1*



d = 0,75 < dmax …. nerozdeľujeme

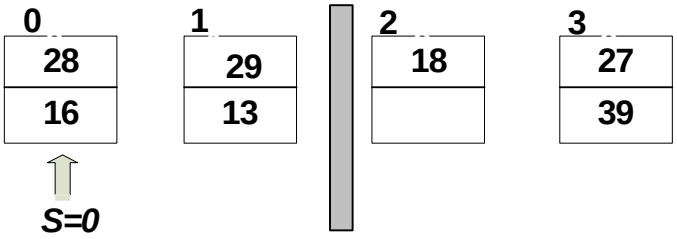
Vložíme 16.
Funkcia *H_0* mapuje 16 do bloku 0. Keďže ale 0< *S* , musíme aplikovať *H_1.* *H_1 (16) = 0,* vložíme teda do bloku 0.



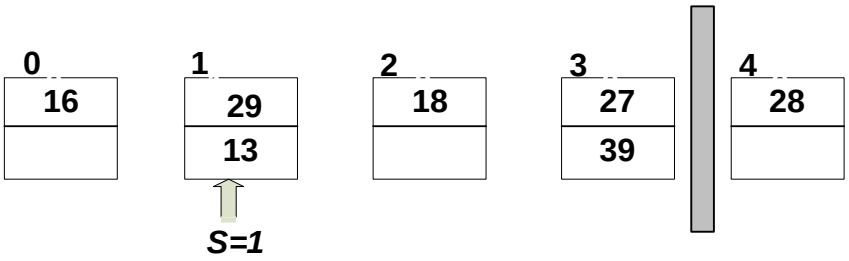
d = 0,87 >= dmax …. rozdeľujeme

Po vložení hustota d = 0,87 prevyšuje dmax, preto musíme rozdeliť skupinu 1 (lebo S = 1) použijúc funkciu *H_1 = K* mod *4.*
Inkrementujeme *S* ale keďže S + 1 >= M*2^u, došlo k úplnej expanzii (všetky bloky na úrovni *u* = 0 boli rozdelené).
Preto teraz položíme

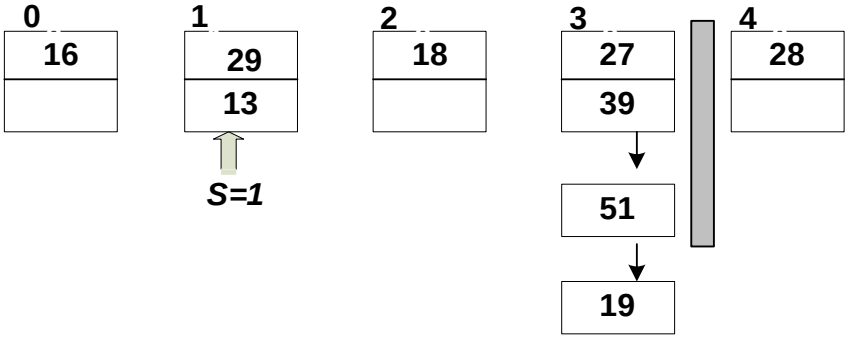
$S = 0,\ u = 1,\ \ H_2(K) = K \bmod 8$



Po rozdelení sa však hustota nezmenila, ostala d = 7/8 = 0,87. Musíme preto znovu rozdeľovať, a to blok 0 s použitím H_2 .



Posledné 2 kľúče 51 a 19 mapuje funkcia H_1 do bloku 3 . Blok 3 je však plný, preto sa do tejto skupiny vložia 2 preplňujúce bloky, pre každý záznam jeden.
Aj po ich vložení je d= 0,75 preto sa nebude rozdeľovať a výsledná štruktúra bude mať tvar:



Algoritmus vymaž:

1. **Urči skupinu i, do ktorej sa záznam mapuje s použitím i := $H_u(K)$.**
2. **Ak i < S (skupina i už bola rozdelená), tak i := $H_{u+1}(K)$.**
3. **Vyber záznam zo skupiny i.**
4. **Ak teraz hustota d klesla pod stanovenú hranicu, tak:**
    a. **Ak S > 0, tak záznamy uložené v poslednej skupine a**

(a = S + M*2^u - 1) presuň do skupiny na pozícii b = S-1 a poslednú skupinu zruš. Nastav S := b;

b. Ak S = 0 a u > 0, tak záznamy uložené v poslednej skupine (a = M*2^u - 1) presuň do skupiny na pozícii b = M*2^{u-1} - 1 a poslednú skupinu zruš. Nastav S := b; Nastav u := u – 1;

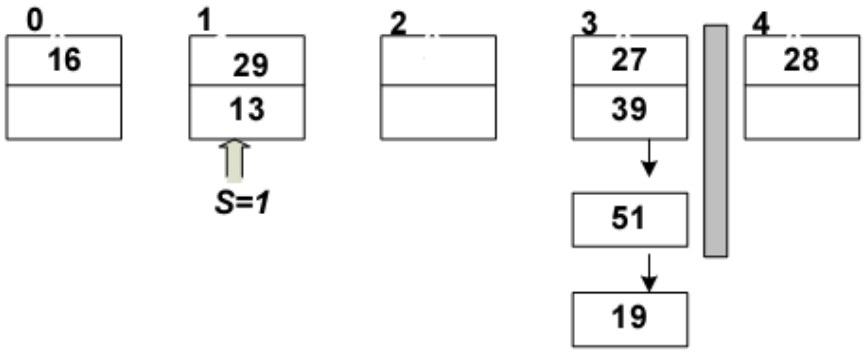
Príklad: (vychádzame zo stavu na poslednom obrázku)

Dolná hranica hustoty súboru nech je dmin = 0.64.

Máme u = 1, S = 1, M = 2
H_1 = K mod 4
H_2 = K mod 8

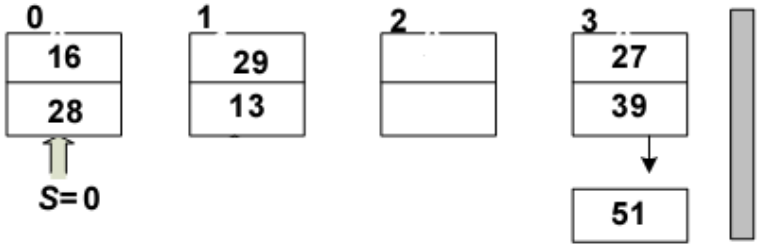
Budeme postupne odoberať hodnoty 18, 19, 13, 28.

Záznam s kľúčom 18 sa mapuje H_1 do skupiny 2. Keďže S < 2 nachádza sa záznam v skupine 2. Po odobratí kľúča 18 bude hustota d = 0.66. d > dmin …. nespájame skupiny.
S = 1, u = 1

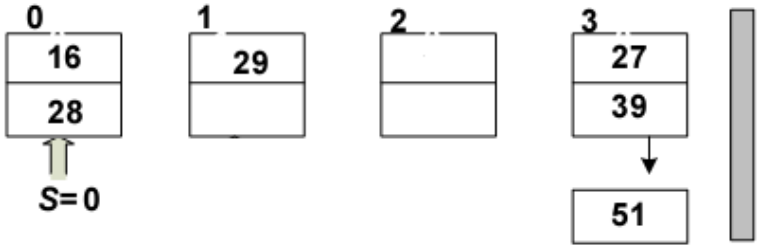


Funkcia H_1 mapuje kľúč 19 do skupiny 3. Po odobratí kľúča 19 zo skupiny 3 bude hustota d = 0.63. Pretože d < dmin uskutoční sa spájanie skupín a zmenšeniu veľkosti súboru. Záznamy zo skupiny 4 (*S + M*2^u - 1*) sa presunú do skupiny 0 (*S-1*).

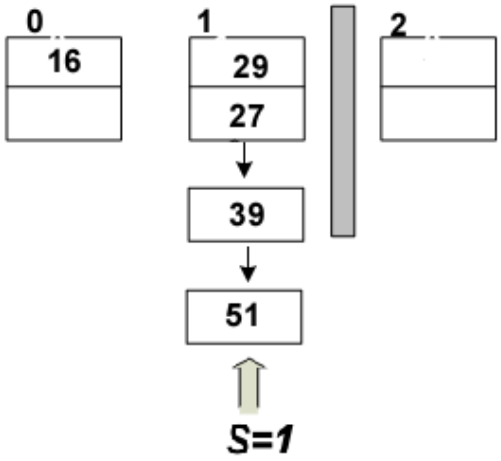
Dekrementuje sa S. Hustota d = 0,77. Všetky záznamy sú mapované pomocou H_1 a v prípade potreby sa bude rozdeľovať skupina 0.
S = 0, u = 1



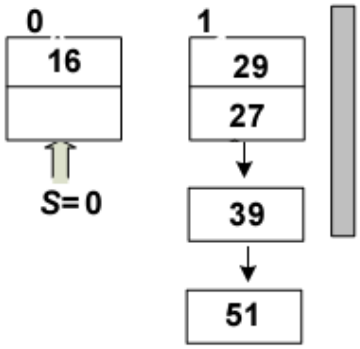
Záznam s kľúčom 13 s mapuje H_1 do skupiny 1. Keďže S < 1 nachádza sa záznam v skupine 1. Po odobratí kľúča 13 bude hustota d = 0.66. d > dmin …. nespájame skupiny.
S = 0, u = 1



Zo skupiny 0 vymažeme kľúč 28, ktorý tam mapuje funkcia H_1. Po odobratí kľúča 28 bude hustota d = 0.55, čo je menej ako dmin. Keďže S = 0 presunú sa záznamy zo skupiny 3 ($M*2^u - 1$) do skupiny 1 ($M*2^{u-1} - 1$). Skupina 3 je následne zrušená.
S = $M*2^{u-1} - 1$ = 1
u = u – 1 = 0
Aktuálne sa budú používať funkcie H_0 = K mod 2 a H_1 = K mod 4, najbližšie sa bude rozdeľovať skupina 1. Hustota d = 0.625.



Po skončení zlučovania je hustota súboru stále menšia ako dmin a preto sa uskutoční ďalšie zlučovanie. Všetky záznamy zo skupiny 2, by sa mali presunúť do skupiny 0, ale keďže skupina 2 žiadne záznamy neobsahuje môžeme ju hneď zmazať.
Hustota d = 0.83. d > dmin …. nespájame skupiny.
S = 0, u = 0



Tu sa ukazuje potreba vhodne zadefinovať hodnoty dmax a dmin. Po spojení skupín hustota súboru prekračuje dmax a malo by dôjsť k rozdeleniu skupín, čo by ale viedlo k zacykleniu celého systému. Je samozrejme možné zadefinovať pravidlo, že po rozdelení skupiny nemôže nasledovať spájanie a naopak.

V tu uvedenom príklade je pre názornosť použitý jednoduchý, ale nie úplne vhodný spôsob definovania podmienky pri ktorej dôjde k rozdeleniu/spojeniu bloku. Lepších možností existuje veľa (napr. sa môže hustota súboru počítať ako pomer celkového počtu záznamov v štruktúre k počtu dostupných miest pre záznamy v hlavnom súbore).

Pre preplňujúce bloky je vhodné zvoliť menší blokovací faktor ako pre bloky primárne.
Voľbou blokovacích faktorov, hodnoty dmax resp. aj dmin, hodnoty M je možné správanie súboru „vyladiť“.

Počet blokových prenosov pri vyhľadávaní zhruba od 1,1 po 1,4.

Niektoré namerané hodnoty pre počet blokových prenosov pri operácii vyhľadávania:

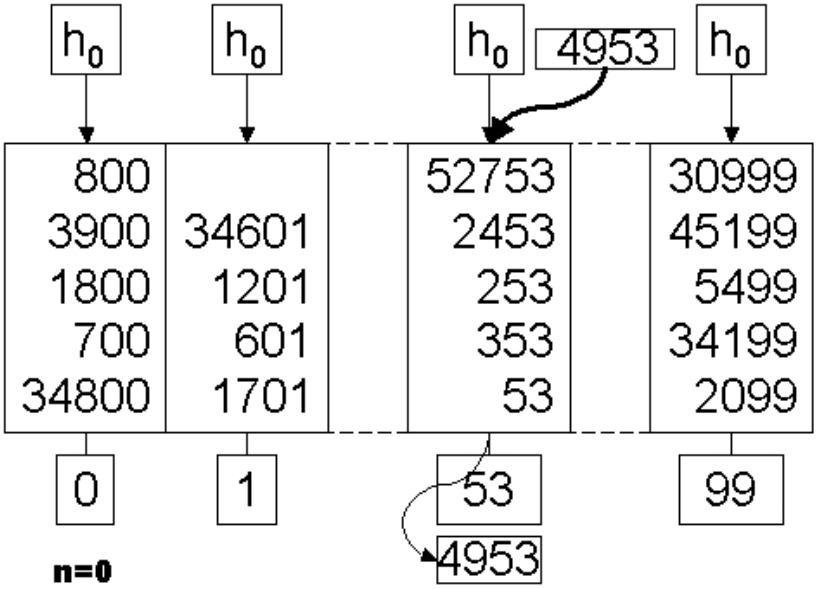
Faktor primárneho bloku	Faktor preplňujúceho bloku	dmax	Stredný počet prenosov
50	15	0,75	1,05
50	12	0,9	1,35
20	6	0,85	1,24
10	3	0,85	1,33

Iný príklad fungovania lineárneho hešovania:

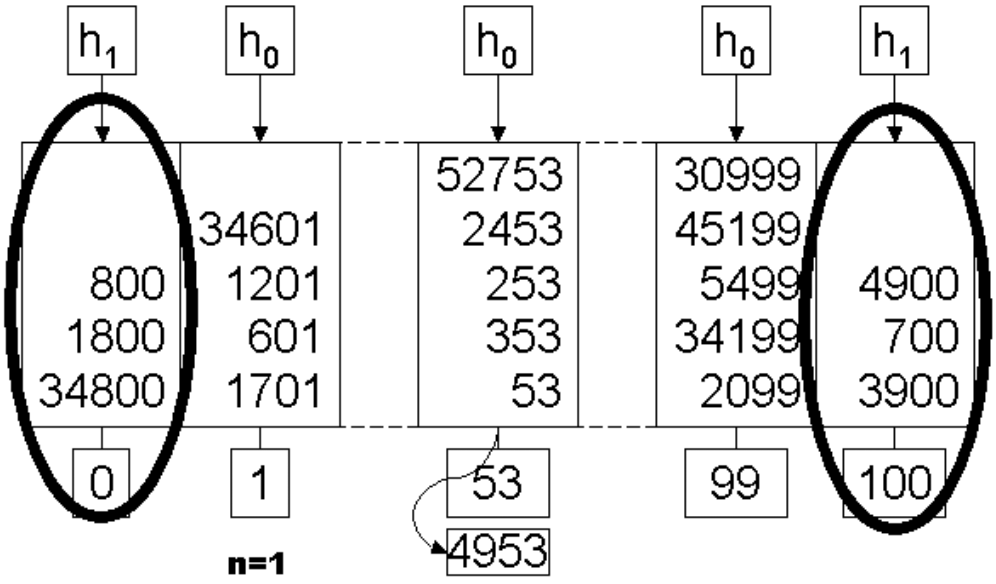
$h_i(k)=k \bmod (2^i*N)$; N = 100 **$h_0(k)=k \bmod 100$**
$h_1(k)=k \bmod 200$

Insert 4953

1. 4593 do overflow bloku

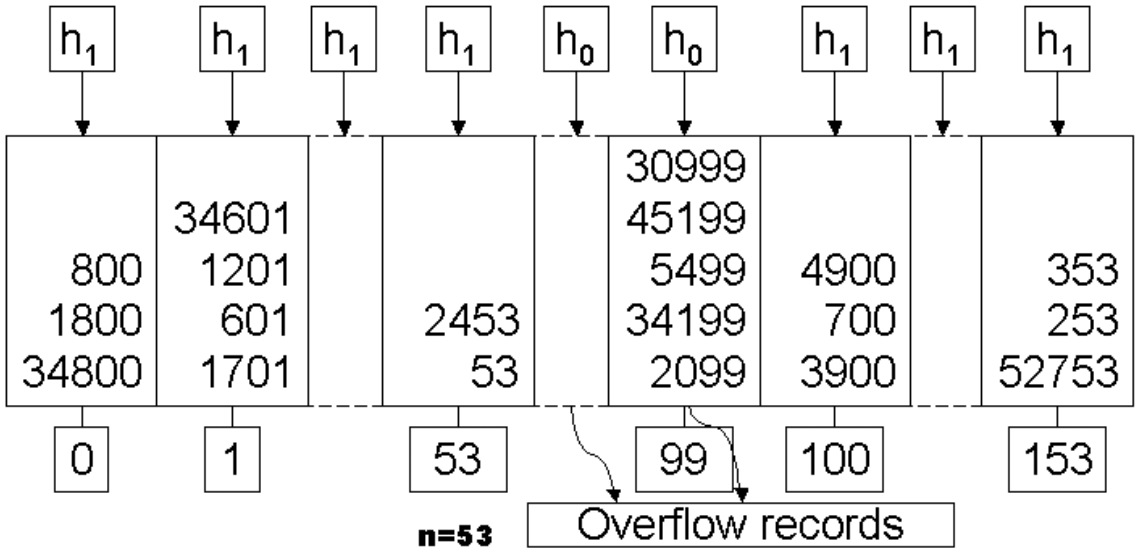


2. Rozdelenie (Split) bloku 0



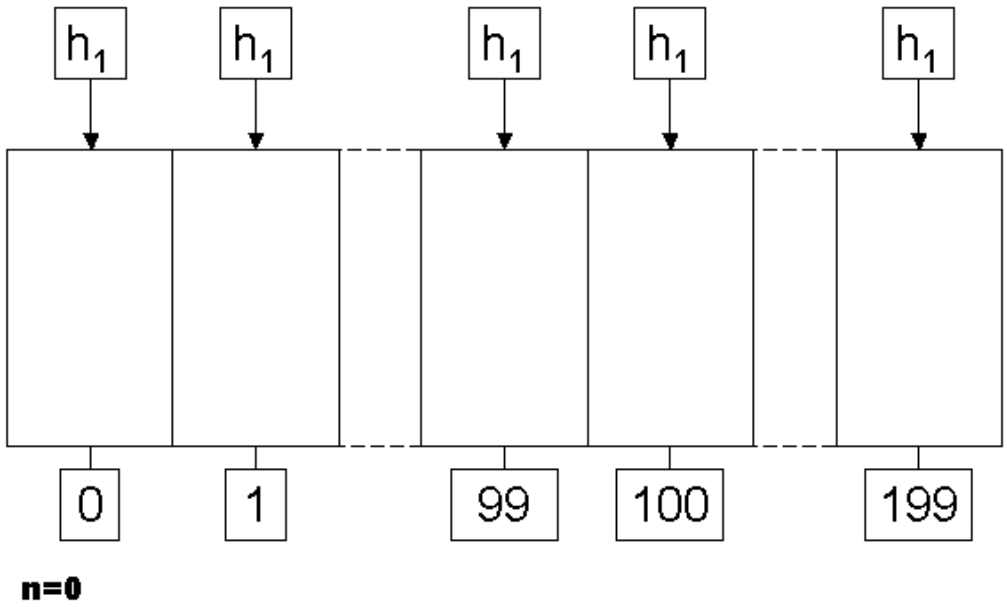
Insert

Blok 53 je rozdelený => zrušenie overflow bloku



Insert

Všetky bloky 0 až 99 sú rozdelené



Pri ďalšej kolízii sa postupuje v rozdeľovaní blokov opäť od začiatku (nasledujúci rozdelený bude blok 0) tentoraz až po blok 199. Pri rozdelení sa hešuje funkciou h_2.

	Rozšíriteľné a dynamické hešovanie	**Lineárne hešovanie**
+	Nevyžaduje úplnú reorganizáciu súboru	Nevyžaduje úplnú reorganizáciu súboru
	Efektívne využité miesto	Nepotrebuje index (adresár) blokov
	Problém s rovnakými kľúčmi riešiteľný ako kolízia prvkov	
-	Nepriamy prístup (cez adresár)	Overflow bloky sú nutné a často používané
	Adresár môže byť veľký	Preplnený blok môže dlhšie čakať na spracovanie
		Menej efektívne využitý priestor