Hešovanie v operačnej pamäti (Hashing, Hash Tables)

- Priemerná zložitosť operácií Find, Insert, Delete je **O(1)**
- Transformácia hodnoty kľúča **K** na adresu **A** pomocou **hešovacej funkcie h**

$$h(K) = A$$

- Hešovacia funkcia mapuje hodnotu kľúča veľkého rozsahu na adresu v tabuľke menšieho rozsahu **(m)**=> vznik kolízií
- Vhodnosť použitia ak je použitá len malá časť možných hodnôt kľúča

Problémy pri implementácii:

- Ako vytvoriť hešovaciu funkciu?
- Aká veľká má byť tabuľka?
- Akým spôsobom vyriešiť možné kolízie?

Hešovacia funkcia

Vlastnosti kvalitnej hešovacej funkcie:

- **Rýchly výpočet**
- **Rovnomerná distribúcia kľúčov do celej tabuľky – P(h(K) = i) = 1 / m**
- Výsledok funkcie je určený len hodnotou kľúča
- Súčasťou výpočtu sú všetky časti kľúča
- Pre podobné hodnoty kľúča generuje značne odlišné adresy

1. **Priame hešovanie** **h(K) = K**

2. **Modulo delenie** **h(K) = K mod m**

K	4	8	12	16	20
K mod 8	4	0	4	0	4
K mod 7	4	1	5	2	6

Pri m=8 je h(K) určená len poslednými 3 bitmi

m by malo byť **prvočíslo!**

3. **Mid-square**

   a) 1234*1234 = 1522756 => h(1234) = 227

   b) h(12345678) = 45 * 45 = 2025

4. **Skladanie (Folding)**

   a) 123|456|789 = 123 + 456 +789 => h(123456789) = 1368

   b) 123|456|789 = 321 + 456 + 987 => h(123456789) = 1764

5. **Výber číslic** **h(123456789) = 359**

6. **Polynomiálne adresovanie** **h(456) = X^2 * 4 + X * 5 + 6**

```
h := K[0];
for i:=1 to Length(K)-1 do
  h := (h * X + K[i]);
```

Nenumerické kľúče:

1. K = „25X3Z“
   ASCII: 0110010 0110101 1011000 0110011 1011010 = 13,534,370,266

2. K = „25X3Z“
   Suma ASCII hodnot znakov, 50 + 53 + 88 + 51 + 90 = 332

V praxi sa často využívajú **kombinácie uvedených metód**, aby sa odstránili nevýhody jednotlivých prístupov.

Pred nasadením hešovacej funkcie je vhodné vykonať **testy na vzorke reálnych dát**.

Riešenie kolízií

Kolízia nastane v prípade ak dva rôzne kľúče K1 a K2 majú rovnakú adresu – **h(K1) = h(K2).**

Kolízie môžu byť častejšie ako sa zdá:
Ak je v miestnosti 23 ľudí, je viac ako 50% šanca, že majú dvaja narodeniny v ten istý deň. (Hešovacia tabuľka s veľkosťou 365, je zaplnená len na 6,3%!).

Definujme α ako obsadenosť (load factor) hešovacej tabuľky, teda pomer medzi počtom obsadených pozícií **n** k veľkosti tabuľky **m**. $\alpha = n / m$

1. Otvorené adresovanie (Open Addressing, Closed hashing, Probing)

Riešenie kolízií s využitím miesta v tabuľke. Ak nastane kolízia, zistí sa nové voľné (otvorené) miesto pre umiestnenie prvku. Adresu prvku teda zistíme, posunutím základnej adresy o offset f(i).

$$H(i) = (h(K) + f(i)) \bmod m, \text{ pre } i=1..m$$

a) Linear probing $f(i) = i$

Problémom je vznik primárneho zhlukovania – zhluk prvkov okolo hešovacej adresy.
Majme hešovaciu tabuľku s veľkosťou 10 a obsadenými pozíciami 1,2,3. Potom je 40 % šanca, že ďalší prvok bude vložený na pozíciu 4!

b) Quadratic probing $f(i) = i * i$

Lepšie ako linear probing, ale vzniká sekundárne zhlukovanie – zhluk prvkov okolo jednotlivých pozícií offsetov.

c) Random probing $f(i) = i * Random$

Náhodné čísla musia byť reprodukovateľné (Napr. kľúč môže byť násadou generátora).

d) Dvojité hešovanie $f(i) = i * h2(K)$
Rôzne offsety pre rôzne kľúče. Odstraňuje problémy so zhlukovaním.
Hešovacia funkcia h2 by mala byť odlišná od primárnej hešovacej funkcie.
(Ak je **h(K1) = h(K2)**, potom by malo platiť **h2(K1)<>h2(K2)** a naopak)

2. Zreťazenie (Chaining, Open hashing)

a) Separate chaining

Riešenie kolízii zreťazením elementov, ktoré hešujú na tú istú pozíciu v tabuľke (vznikne teda m zreťazených zoznamov).

- Dobrá stratégia, ak nie je priveľa kolízií.
- Počet uložených prvkov môže byť väčší ako je veľkosť hešovacej tabuľky.

b) Overflow area

- Vyhradenie oblasti pre kolidujúce prvky.
- Pri vzniku kolízie sa vyhľadá voľné miesto v tejto oblasti.
- Zreťazenie

3. Použitie „košíkov“ (Buckets)

- Každá pozícia v tabuľke je schopná uchovať niekoľko prvkov.
- Pri pretečení využiť napr. zreťazenie.

Náročnosť operácií pre jednotlivé spôsoby riešenia kolízií

- závisí od obsadenosti tabuľky
- nezávislá od jej veľkosti

	Linear probing	Quadratic probing	Dvojité hešovanie	Zreťazenie
Vkladanie, Neúspešné vyhľadanie	$(1+(1/(1-\alpha))^2)/2$	$1/(1-\alpha)$	$1/(1-\alpha)$	α
Úspešné vyhľadanie	$(1 + 1/(1-\alpha))/2$	$(1/\alpha) * \log(1/(1-\alpha))$	$\ln (1/(1-\alpha))/ \alpha$	$1 + \alpha/2$

Teoretická náročnosť operácií:

Load Factors		0.10	0.25	0.50	0.75	0.90	0.99
		Successful Search					
Separate Chaining		1.05	1.12	1.25	1.37	1.45	1.49
Linear Probing	1.06	1.17	1.50	2.50	5.50	50.5	
Double Hashing	1.05	1.15	1.39	1.85	2.56	4.65	
Quadratic Probing		1.04		1.50		2.70	5.20
		Unsuccessful Search					
Separate Chaining		0.10	0.25	0.50	0.75	0.90	0.99
Linear Probing	1.12	1.39	2.50	8.50	50.5	5000	
Double Hashing	1.11	1.33	2.00	4.00	10.0	100.0	
Quadratic Probing		1.13	2.70			59.8	430

Napríklad pri 90% zaplnení tabuľky pri použití linear probing musíme vykonať priemerne 50 operácií!

Vymazávanie pri hešovacích tabuľkách

Pri použití otvoreného adresovania je pri vymazaní nutné nahradiť zmazaný prvok špeciálnou hodnotou, ktorá označuje zmazaný prvok, nie hodnotou **NULL**.

Rehešovanie (Rehashing)

- v prípade, že je tabuľka naplnená, prípadne je vhodné udržiavať ju v nenaplnenom stave
- zväčšenie veľkosti tabuľky a opätovné hešovanie všetkých prvkov (okrem vymazaných)