Seminár z algoritmizácie a programovania 1

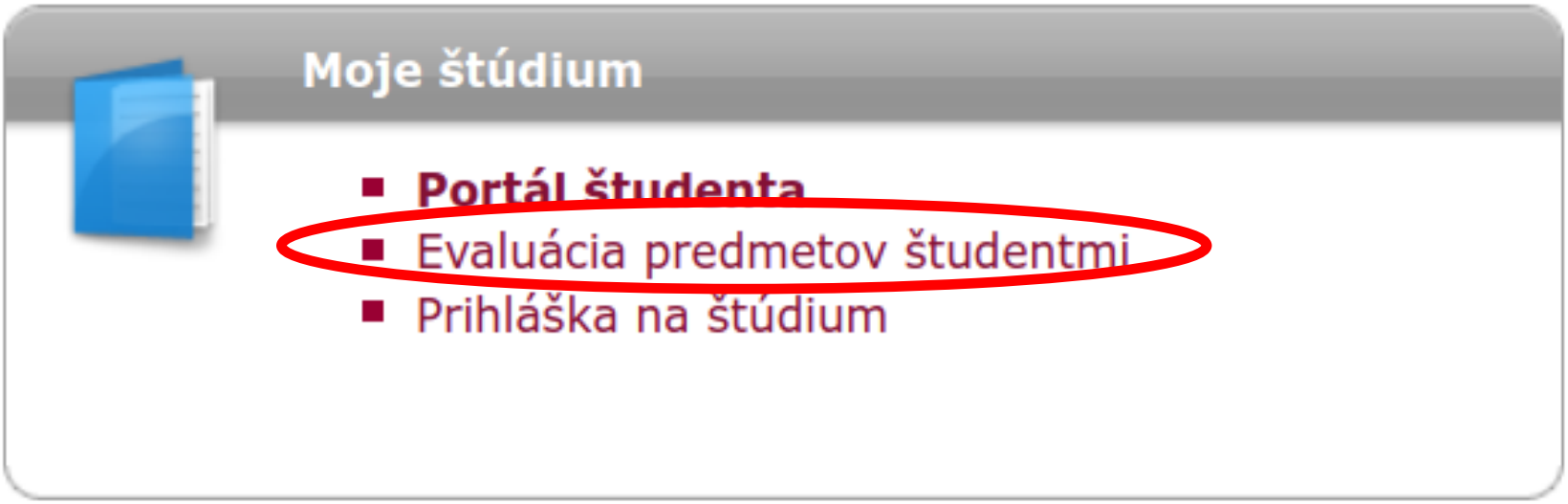
Martin Bobák
Ústav informatiky
Slovenská akadémia vied

Obsah prednášky

Hešovanie

Spätná väzba:
https://forms.gle/iKbuLdF6xDtNSEDP8

Oficiálna anketa k predmetu



Ďakujem vám za poznámky a námety na zlepšenie!

Motivácia

- mať (slovníkovú) dátovú štruktúru, ktorá bude pracovať s tabuľkou v O(1) (v priemere)
  - prvkom je dvojica (kľúč, objekt) <=> (k,o), my si to zjednodušíme a budeme pracovať s kľúčom k
- vyžadujeme, aby vyhľadávanie, vkladanie a mazanie v tejto dátovej štruktúre bolo v priemere konštantné
- vedeli by sme použiť pole alebo spájaný zoznam
  - vkladanie a mazanie majú vysokú časovú zložitosť
- najrýchlejšie vieme tieto operácie spraviť pomocou stromových dátových štruktúr (vyvážené vyhľadávacie stromy)
  - časová zložitosť: O(log n)

Motivácia

Tabuľka s priamym prístupom:

- obrovské pole, kľúč je index
- ak prvok neexistuje, špeciálna hodnota (napr. NULL)
- všetky operácie majú konštantnú časovú zložitosť

Nevýhody tohto riešenia:

- vysoká pamäťová zložitosť – vysoký počet neosadených prvkov
  - počet indexov/kľúčov je výrazne menší ako počet objektov, ktoré chceme uložiť (napr. pri telefónnom zozname niektoré čísla sa nepoužijú)

Motivácia

- pole A[0...n-1]
- k prvkom nepristupujeme cez index, ale toto pole je "indexované" pomocou hešovacej funkcie h(k):

  A[k] <=> A[h(k)]
- hešovacia funkcia nemusí byť injektívna
  - môžu nastať kolízie t.j. dva rôzne kľúče k_1 a k_2 sa zahešujú rovnako – $h(k_1) = h(k_2)$

Hešovanie

- U je univerzum (množina) všetkých kľúčov
- K je množina využitých/dosiahnuteľných kľúčov,
- T je hešovacia tabuľka
- h(k) je hešovacia funkcia h: U -> $\mathbb{N}$
  - zobrazenie z univerza kľúčov na prvky (dáta)

Priame adresovanie

- každý prvok (dáta) má priradený kľúč z univerza U = {0, …, m-1}
  - m prvkov chcem uložiť do tabuľky veľkosti m (alebo väčšej)
- **neexistujú dva prvky s rovnakým kľúčom**

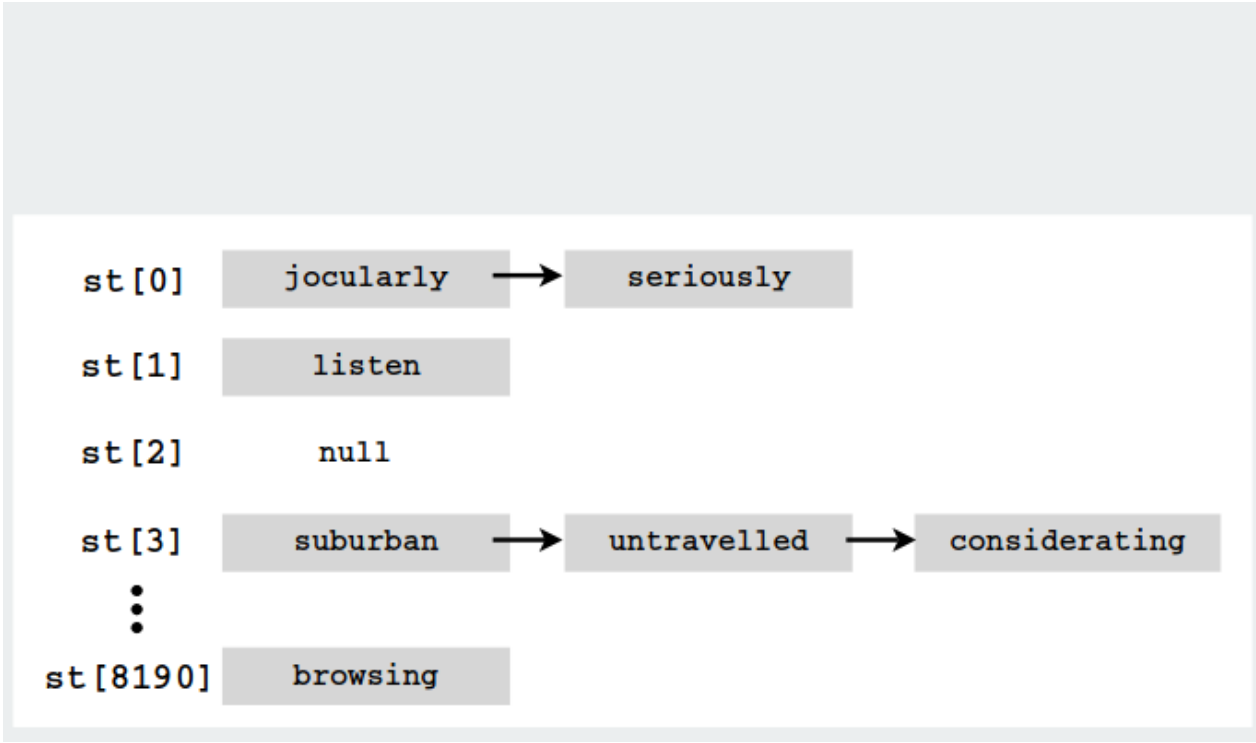
- hešovacia tabuľka je reprezentovaná ako pole veľkosti m
  - indexy poľa sú jednotlivé kľúče z univerza
  - prvky poľa sú ukazovatele na dáta
  - hešovacia funkcia je h(k) = k

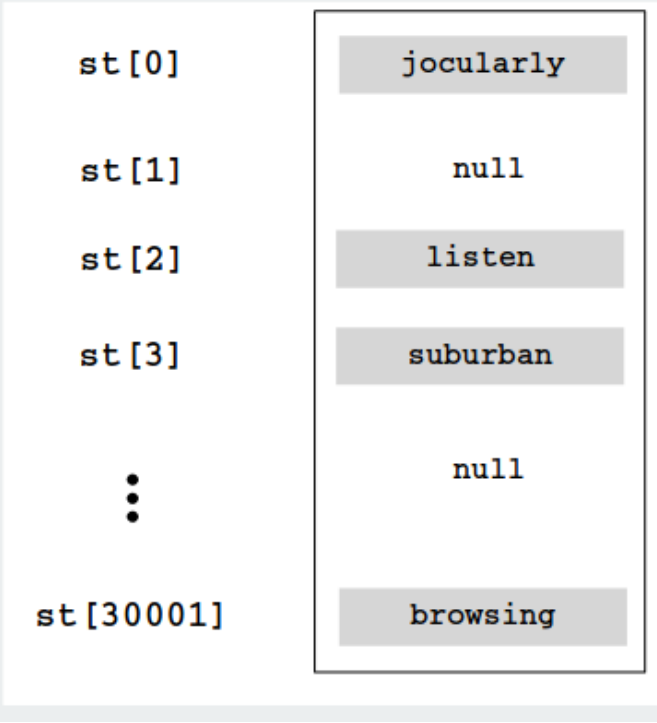
Hešovanie

- každý prvok (dáta) má priradený kľúč z univerza U = {0, …, n-1}
  - m prvkov chcem uložiť do tabuľky veľkosti n (m≥n)
- **nastávajú kolízie**

- Riešenie kolízií
  - zreťazené hešovanie
  - kukučie hešovanie, otvorené adresovanie

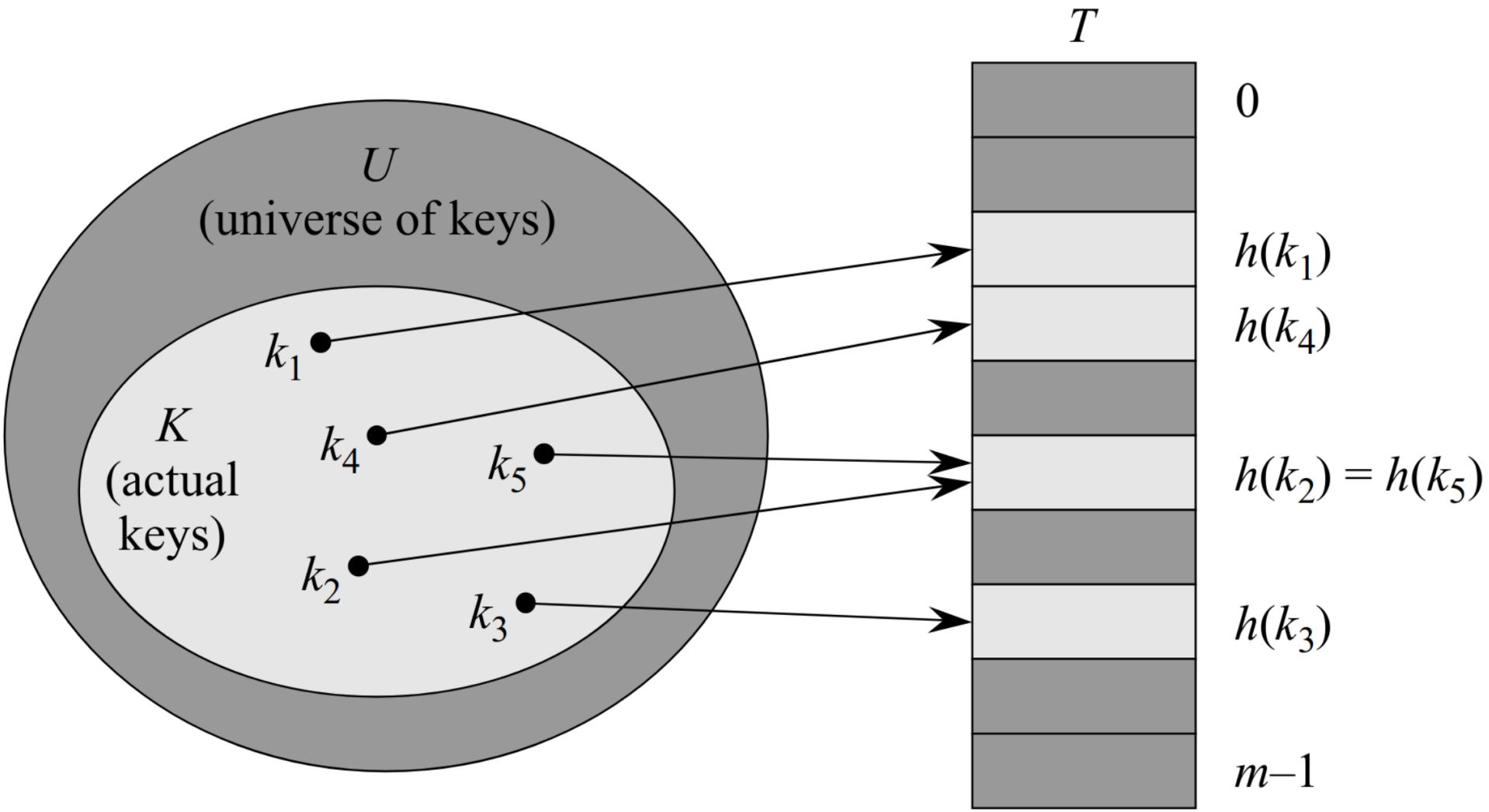
Riešenie kolízií





Zdroj: https://www.cs.princeton.edu/~rs/AlgsDS07/10Hashing.pdf

Hešovanie


T
U
(universe of keys)
K
(actual
keys)
k_1
k_4
k_5
k_2
k_3
0
$h(k_1)$
$h(k_4)$
$h(k_2) = h(k_5)$
$h(k_3)$
$m-1$

Hešovacie funkcie

Základné vlastnosti:

- **ľahko vypočítateľná** (aby sme vedel pristupovať k prvkom v konštantnom čase)
- **uniformná** (pravdepodobnosť zahešovania kľúča je na každú pozíciu hešovacej tabuľky rovnaká)
  Pr[h(k) = i] = 1/n
- **univerzálna** (pre každú dvojicu kľúčov je pravdepodobnosť kolízie je malá)
  Pr[h(k_1) = h(k_2)] ≤ 1/n pre každé $k_1 \neq k_2$

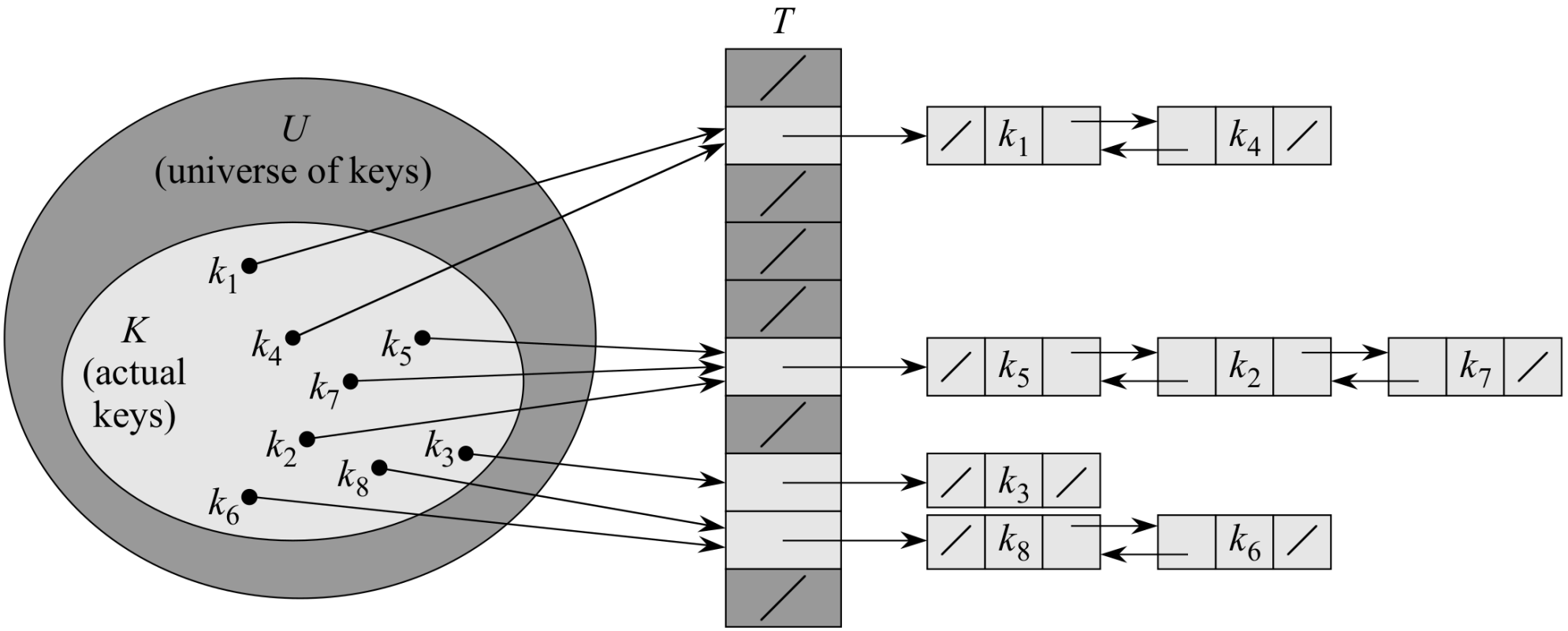
Zreťazené hešovanie

- každý prvok tabuľky T je ukazovateľ na zoznam prvkov, ktoré sa zahešovali na to isté miesto
  - nový prvok vložíme na začiatok zoznamu T[h(k)], obdobne ostatné operácie

Analýza:

- predpokladáme, že hešovacia funkcia je uniformná
  - na jedno miesto v T sa zahešuje m/n prvkov -> **faktor naplnenia**

Zreťazené hešovanie



Zreťazené hešovanie

Príklady hešovacích funkcií:

- $h(k) = k \bmod n$
- $h_A(k) = \lfloor n\ (k\ \ A \bmod 1)\ \rfloor$ *(A je konštanta)*
- $h_{ab}(k) = ((ak+b) \bmod p) \bmod n)$

Kukučie hešovanie

- dve tabuľky T_1 a T_2 veľkosti n
- dve hešovacie funkcie h_1 a h_2

Hľadanie:

- prvok x je v $T_1[h_1(x)]$ alebo $T_2[h_2(x)]$

Kukučie hešovanie

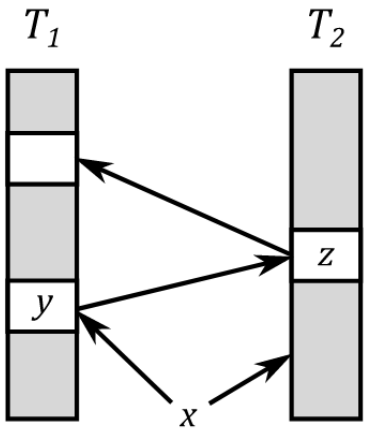
Vkladanie:

- skúsime $T_1[h_1(x)]$ alebo $T_2[h_2(x)]$, ak je miesto voľné, prvok naň uložíme
- ak sú obe miesta obsadené prvok y z T_1 odoberieme a pokúsime sa o $T_2[h_2(y)]$
- ak ak v $T_2[h_2(y)]$ je prvok z, odoberieme ho a skúsime sa o $T_1[h_1(z)]$
- Pr[prejdenia cesty dĺžky k] ≤ $½^k$

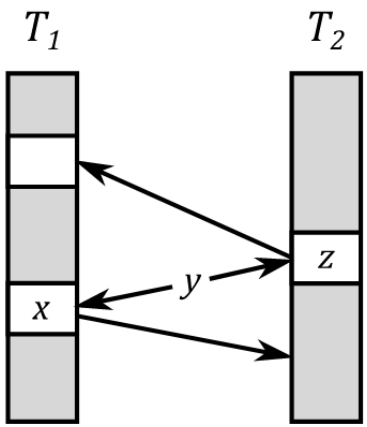
Kolízia:

- zacyklili sme sa -> vyberieme novú hešovaciu funkciu a celú tabuľku prehešujeme
- pri "slušných" hešovacích funkciách sa to nedeje často Pr[kolízie] = O(1/n)

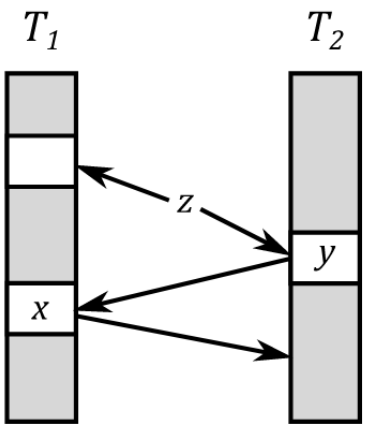
Kukučie hešovanie



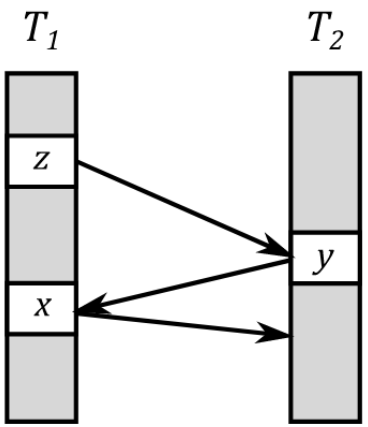
Step 1



Step 2



Step 3



Step 4

Otvorené adresovanie

Vkladanie:

- prechádzame hešovacou tabuľkou, kým nenarazíme na voľné miesto
  - po n neúspešných pokusoch vieme, že tabuľka je plná

Hľadanie:

- obdoba vkladania. Skúšame kým prvok nenájdeme, alebo narazíme na prázdne miesto (prvok sa v tabuľke nenachádza)

Mazanie:

- nemôžeme zmazať prvok. Danú pozíciu si označíme ako zmazanú (potrebujeme zabezpečiť kontinuitu tabuľky)

Stratégie otvoreného hešovania

Poznáme tri základné stratégie:

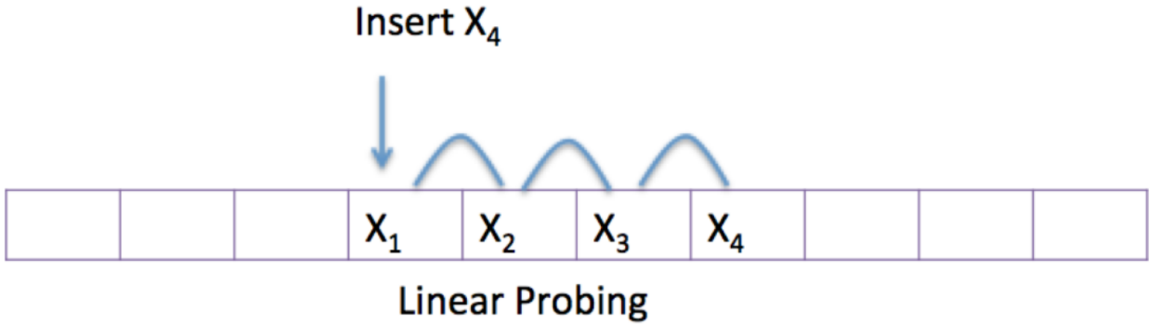
1. lineárne hešovanie
2. kvadratické hešovanie
3. dvojité hešovanie

Lineárne hešovanie

Hešovacia funkcia:

$$h'(k, i) = (h(k)+ i) \bmod n$$

- ak je h(k) obsadená, skúsime h(k)+1, h(k)+2,…
- problém je primárne zhlukovanie, čím sa výrazne zvýši zložitosť jednotlivých operácií



Zdroj: http://courses.csail.mit.edu/6.851/fall17/scribe/lec10.pdf

Kvadratické hešovanie

Hešovacia funkcia:

$$h'(k, i) = (h(k) + c_1 i + c_2 i^2) \bmod n$$

- problém je vhodná voľba c_1 a c_2
- ak sa dva kľúče k1 a k2 zahešujú na rovnaké miesto, potom majú rovnakú celú postupnosť adries – sekundárne zhlukovanie

Dvojité hešovanie

Hešovacia funkcia:

$$h(k, i) = (h_1(k) + ih_2(k)) \bmod n$$

- ak je $h_1(k)$ obsadená, skúsime $h_1(k)+h_2(k)$, $h_1(x)+2h_2(k)$,…
  - ak chceme prejsť celú tabuľku, $h_2(k)$ nesmie deliť n
- problém je primárne zhlukovanie, čím sa výrazne zvýši zložitosť jednotlivých operácií

Dokonalé hešovanie

- v najhoršom prípade máme konštantnú časovú zložitosť
- množina kľúčov dopredu poznáme a nemení sa
  - kľúčové slová v programovacom jazyku
- dve úrovne hešovania

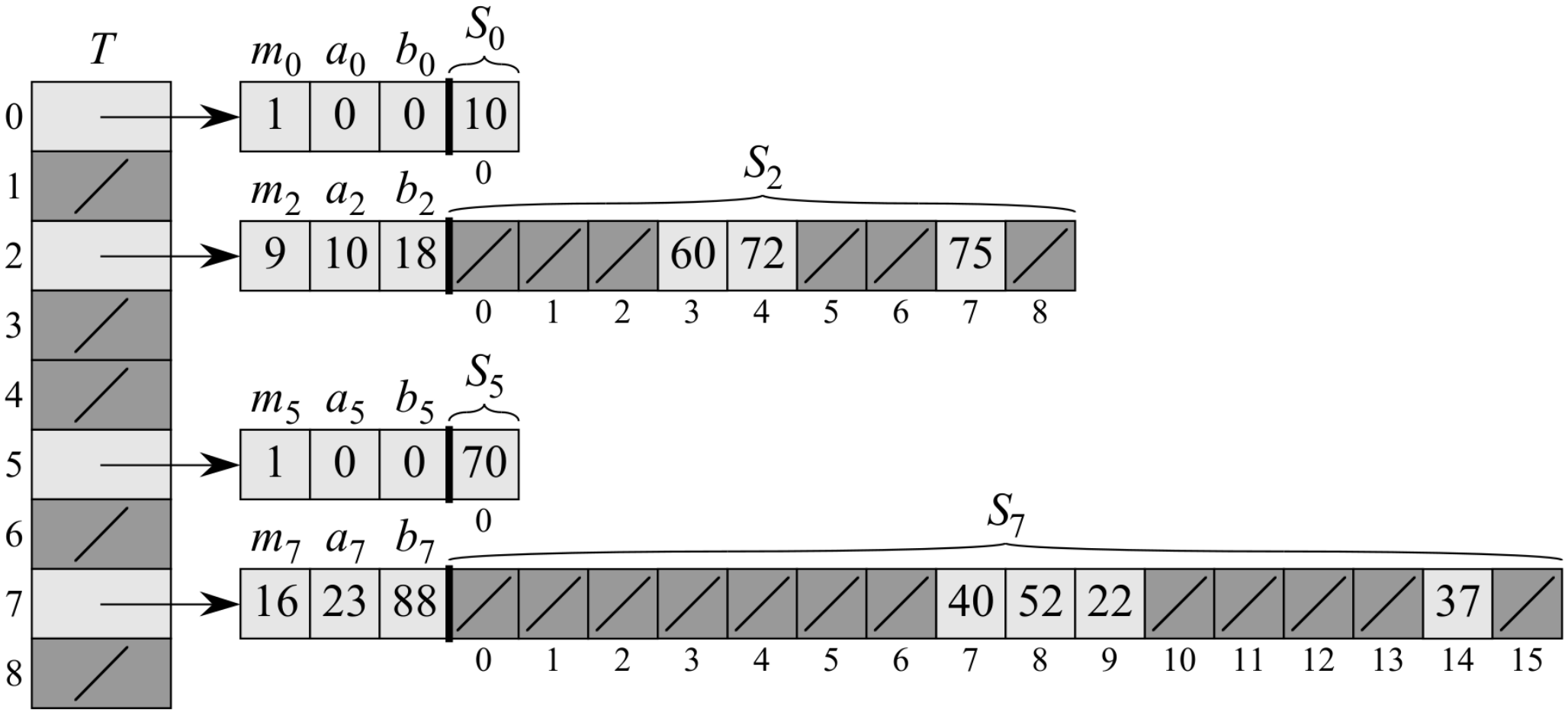
Prvá úroveň:

- zreťazené hešovanie

Druhá úroveň:

- namiesto spájaného zoznamu, prvky uložíme v hešovacej tabuľke S_j, ktorá má príslušnú hešovaciu funkciu h_j
- vhodný výber h_j zabezpečí, aby na druhej úrovni nenastali kolízie

Dokonalé hešovanie



Porovnanie

- extra miesto pre ukazovatele vs prázdne miesto v tabuľke
- malá tabuľka + ukazovatele vs veľké súvislé pole

		load factor α			
		50%	66%	75%	90%
linear probing	get	1.5	2.0	3.0	5.5
	put	2.5	5.0	8.5	55.5
double hashing	get	1.4	1.6	1.8	2.6
	put	1.5	2.0	3.0	5.5

Zdroj: https://www.cs.princeton.edu/~rs/AlgsDS07/10Hashing.pdf

Aplikácie

- rýchle porovnávanie reťazcov
  - odstránenie duplikátov
- efektívne indexovanie
  - vyhľadávanie reťazcov

Jednosmerné hešovacie funkcie

Je náročné:

- nájsť kľúč, ktorý sa zahešuje na hľadanú hodnotu
- nájsť dva rôzne kľúče s rovnakým hešom

Príklady:

- ~~MD4, MD5, SHA-0, SHA-1~~, SHA-2, WHIRLPOOL, RIPEMD-160.

Použitie:

- digitálne odtlačky, uchovanie hesiel

Zdroje

Thomas Cormen, Charles Leiseson, Ronald Rivest, Clifford Stein. Introduction to Algorithms.

Robert Sedgewick and Kevin Wayne. Algorithms.

https://algs4.cs.princeton.edu/34hash/

https://www.geeksforgeeks.org/hashing-data-structure/

http://en.wikipedia.org/wiki/Hash_table

http://en.wikipedia.org/wiki/Hash_function

https://cs.stanford.edu/~rishig/courses/ref/l13a.pdf

Hodnotenie študentov

Semestrálna práca: 25 bodov (aspoň 6b)

Záverečný test: 25 bodov (aspoň 10b)

Spolu: max. 50 bodov

Absolvovanie predmetu (podľa stupnice STU):

A 46 – 50 bodov	**D** 33 – 36 bodov
B 42 – 45 bodov	**E** 28 – 32 bodov
C 37 – 41 bodov	**FX** 27 – 0 bodov

Záverečný test

Termín

- piatok 14.5.2021 (posledný seminár)
- čas: 10:00 – 10:30
- ~~miestnosť~~: dištančne

Témy:

- semináre 1 až 11
- výber možností (teória aj prax -> doplnenie časti kódu)

Ďakujem vám za pozornosť!

Spätná väzba:
https://forms.gle/iKbuLdF6xDtNSEDP8