Datové struktury I

8. přednáška: Lineární přidávání a kukaččí hešování

Jirka Fink

https://ktiml.mff.cuni.cz/~fink/

Katedra teoretické informatiky a matematické logiky
Matematicko-fyzikální fakulta
Univerzita Karlova v Praze

Zimní semestr 2024/25

Kukačkové hešování (Pagh, Rodler, 2004)

Popis

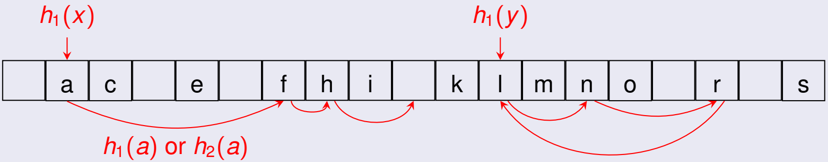
Pro dvě hešovací funkce h_1 a h_2 prvek x musí být uložen v přihrádce $h_1(x)$ nebo $h_2(x)$. V jedné přihrádce může být uložen nejvýše jeden prvek.

Operace Find a Delete

Triviální, složitost $\mathcal{O}(1)$ v nejhorším případě.

Příklad operace Insert

- Úspěšné vložení prvku x do přihrádky $h_1(x)$ po třech přesunech
- Prvek y není možné vložit do $h_1(y)$



Kukačkové hešování: Algoritmus pro operaci Insert

Vložení prvku x do tabulky T

```
pos ← h1(x)
for ⌈6 log m⌉ krát ① do
    if T[pos] je prázdná then
        T[pos] ← x
        return
    swap(x, T[pos])
    if pos == h1(x) ② then
        pos ← h2(x)
    else
        pos ← h1(x)
rehash()
insert(x)
```

Rehash

- Náhodně vygenerujeme nové hešovací funkce h_1 a h_2 z $\mathcal{H}$
- Můžeme zvětšit velikost tabulky
- Vložíme všechny prvky do nové tabulky ③

1. Po n pokusech jsme už určitě v cyklu. Lze ukázat, že v cyklu jsme s velkou pravděpodobností už po $\Omega(\log n)$ krocích.
2. Potřebuje najít druhou pozici, ve které prvek x může být uložen.
3. Při vkládání prvků do nové tabulky může dojít k Rehash, takže si při implementaci musíme dát pozor, abychom některé prvky neztratili.

Kukačkové hešování: Složitost (bez důkazu)

Věta

Předpoklady:

- Hešovací systém je $\lceil 6 \log n \rceil$-nezávislý
- Libovolné $c > 1$
- Počet prvků je n a velikost tabulky je $m \geq 2cn$

Pak:

- Očekávaná složitost operace Insert bez přehešování je $\mathcal{O}(1)$
- Očekávaný počet přehešování při vkládání n prvků do tabulky velikosti m je $\mathcal{O}(1)$
- Očekávaná amortizovaná složitost operace Insert je $\mathcal{O}(1)$

Hešovací systémy

- Tabulkové hešování garantuje stejnou složitost
- Existuje 6-nezávislý hešovací systém nedávající konstantní složitost

Lineární přidávání

Cíl

- Chtěli bychom ušetřit paměť, a tak prvky budeme ukládat přímo do tabulky
- V jedné přihrádce může být jen jeden prvek

Operace Insert

Nový prvek x vložíme do prázdné přihrádky $h(x) + i \mod m$ s nejmenším možným $i \geq 0$.

Operace Find

Iterujeme dokud nenajdeme prvek nebo prázdnou přihrádku.

Operace Delete

- Lína varianta: Přihrádku smazaného prvku označkujeme, aby následné operace Find pokračovali v hledání
- Varianta bez značkování: Zkontroluje a přesouvá prvky v celém řetězci

Lineární přidávání: Složitost operací Find, Insert a Delete

Předpoklady

- $m \geq (1 + \epsilon)n$
- Pokud přihrádky po smazaných prvcích jen značkujeme, pak n je součet počtu prvků a označkovaných přihrádek

Očekávaný počet porovnání při operaci Insert je

- $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ pro úplně náhodný systém (Knuth, 1963)
- konstantní pro $\log(n)$-nezávislý systém (Schmidt, Siegel, 1990)
- $\mathcal{O}\left(\frac{1}{\epsilon^{\frac{13}{6}}}\right)$ pro 5-nezávislý systém (Pagh, Pagh, Ruzic, 2007)
- $\mathcal{O}(\log n)$ pro 4-nezávislý systém (Pǎtraşcu, Thorup, 2010) 1
- $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ pro tabulkové hešování (Pǎtraşcu, Thorup, 2012)
- $\mathcal{O}(\log n)$ pro multiply-shift

1. Existuje 4-nezávislý hešovací systém a posloupnost operací Insert nezávislá na vybrané hešovací funkci taková, že očekávaná složitost je $\Omega(\log n)$.

Lineární přidávání: Analýza

Počet prvků od dané přihrádky do nejbližší volné přihrádky

Jestliže $n/m = \alpha < 1$ a systém hešovacích funkcí je úplně náhodný, pak očekávaný počet porovnání klíčů je $\mathcal{O}(1)$. ①

Důkaz

1. Nechť $1 < c < \frac{1}{\alpha}$ a $q = \left(\frac{e^{c-1}}{c^c}\right)^{\alpha}$
   - Platí $0 < q < 1$ ②
2. Nechť $p_t = P[|\{x \in S;\ h(x) \in T\}| = t]$ je pravděpodobnost, že do dané množiny přihrádek T velikosti t je zahešováno t prvků. Pak $p_t < q^t$. ③
   - Nechť X_i je náhodná proměnná indikující, zda prvek i je zahěšován do T
   - Nechť $X = \sum_{i \in S} X_i$ a $\mu = E[X] = t\alpha$
   - Platí $c\mu = c\alpha t < t$
   - Chernoff: $p_t = P[X = t] \leq P[X > c\mu] < \left(\frac{e^{c-1}}{c^c}\right)^{\mu} = q^{\frac{\mu}{\alpha}} = q^t$
3. Nechť b je nějaká přihrádka. Nechť p'_k je pravděpodobnost, že přihrádky b až $b + k - 1$ jsou obsazeny a $b + k$ je první volná přihrádka. Pak $p'_k < \frac{q^k}{1-q}$. ④
   - $p'_k < \sum_{s=0}^{\infty} p_{s+k} < q^k \sum_{s=0}^{\infty} q^s = \frac{q^k}{1-q}$
4. Očekávaný počet porovnání klíčů je $\sum_{k=0}^{m} kp'_k < \frac{1}{1-q} \sum_{k=0}^{\infty} kq^k = \frac{q}{(1-q)^3} = \mathcal{O}(1)$ ⑤

1. Neúspěšná operace Find musí dojít až k volné přihrádce a též operace Insert, pokud cestou není přihrádka označená operací Delete. Úspěšná operace Find může porovnat méně prvků. Složitost operace Delete se v různých verzí liší, ale v rozumných implementacích dojde nejhůře k nejbližší volné přihrádce. Knuth spočítal očekávanou složitost přesně, ale výpočet je náročný.
2. Zjevně $q > 0$. Chceme dokázat, že $\frac{e^{c-1}}{c^c} = e^{c-1-c\log c} < 1$. Musíme tedy dokázat, že $c - 1 - c\log c < 0$ pro $c > 1$. Pro $c = 1$ máme $c - 1 - c\log c = 0$, abychom dokázali ostrou nerovnost pro $c > 1$, ukážeme, že funkce $c - 1 - c\log c$ je pro $c > 1$ klesající. Derivace $1 - \log c - 1$ je záporná pro $c > 1$.
3. Zde uvažujeme prvky, které hešovací funkce zobrazí do daných přihrádek, a nikoliv prvky, které se do daných přihrádek dostanou vlivem lineárního přidávání.
4. Tedy přihrádky $b - s$ až $b + k - 1$ jsou obsazeny pro nějaké s. Indexy přihrádek počítáme modulo m.
5. Gabriel's staircase:
`https://en.wikipedia.org/wiki/Arithmetico-geometric_sequence`

Multiply-shift

Multiply-shift

- Předpokládáme, že $|U| = 2^w$ a $m = 2^l$
- $h_a(x) = (ax \mod 2^w) >> (w - l)$
- $\mathcal{H} = \{h_a;\ a \text{ je liché } w\text{-bitové číslo } \}$

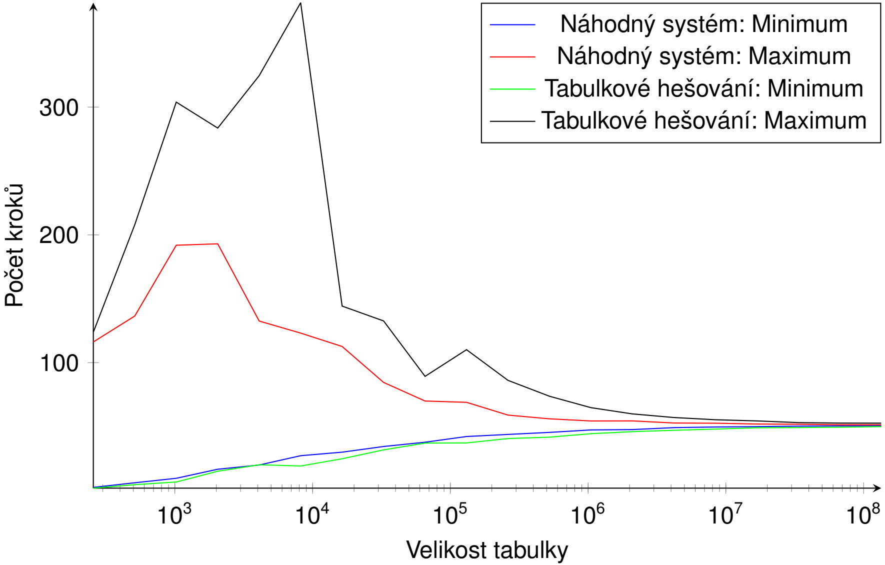
Implementace v C

```
uint64_t hash(uint64_t x, uint64_t l, uint64_t a)
{ return (a*x) >> (64-l); }
```

Vlastnosti systému multiply-shift

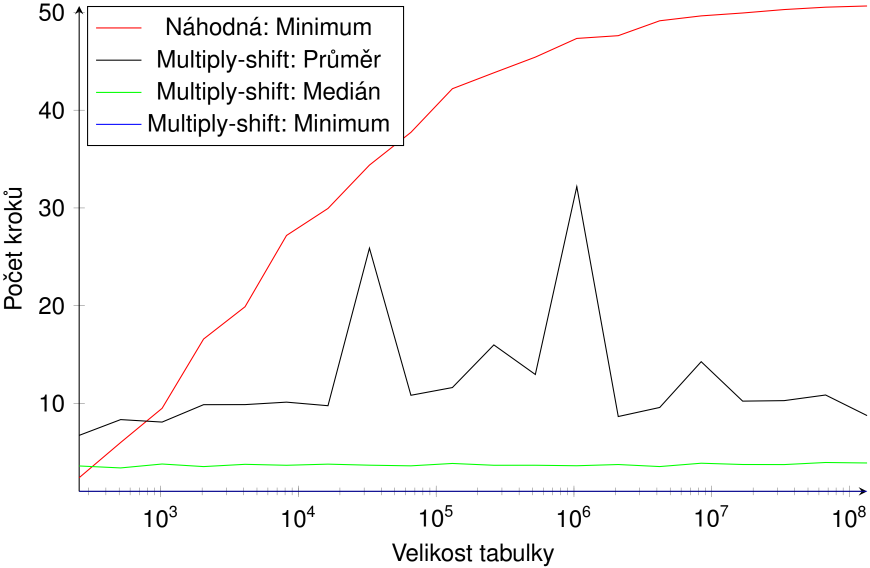
- 2-universální
- Velmi rychlý na reálných počítačích
- V praxi často používaný
- Celý výpočet musí být proveden v neznaménkových celočíselných typech, protože ze součinu *ax* potřebujeme získat posledních *w* bitů

Lineární přidávání: Náhodná hešovací funkce a tabulkové hešování ①

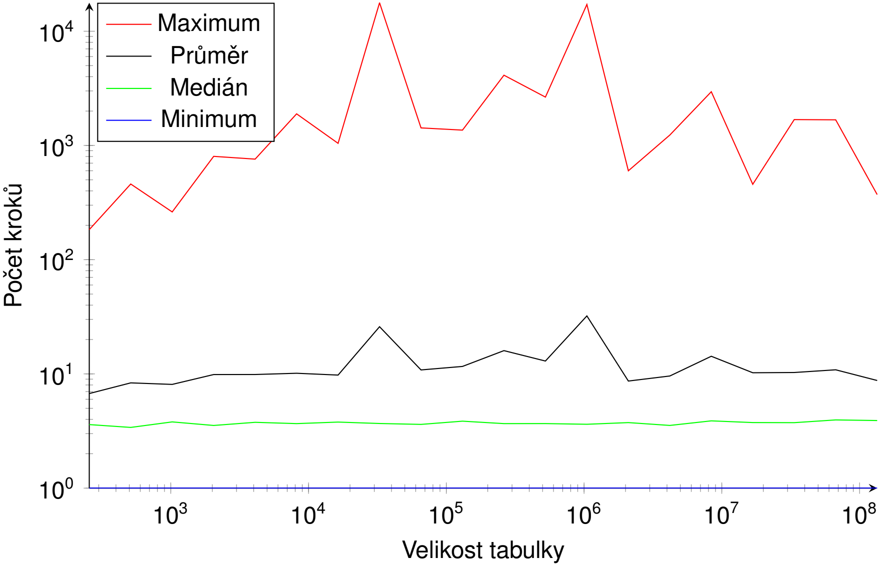


1. Počet kroků při vkládání do tabulky lineárního přidávání při 90% zaplnění. Nejprve vložíme prvky $1, \ldots, \lfloor 0.89m \rfloor$ a poté počítáme průměrný počet kroků při vkládání prvků $\lfloor 0.89m + 1 \rfloor, \ldots, \lfloor 0.91m \rfloor$. Z jednoho experimentu dostaneme jeden průměrný počet kroků a experiment opakujeme 1000-krát pro různé hešovací funkce. Grafy ukazují statistické údaje těchto experimentů.

Lineární přidávání: Náhodná hešovací funkce a Multiply-shift



Lineární přidávání: Multiply-shift



Hešování: Další možnosti

Kvadratické prohledávání

Vložit prvek x do prázdné přihrádky $h(x) + ai + bi^2 \mod m$ s nejmenším možným $i \geq 0$, kde a, b jsou pevné konstanty.

Dvojité hešování

Vložit prvek x do prázdné přihrádky $h_1(x) + ih_2(x) \mod m$ s nejmenším možným $i \geq 0$, kde h_1, h_2 jsou dvě hešovací funkce.

Brentova varianta operace Insert

Jestliže přihrádka

- $b = h_1(x) + ih_2(x) \mod m$ je obsazená prvkem y
- $b + h_2(x) \mod m$ je taky obsazená
- $c = b + h_2(y) \mod m$ je prázdná,

pak přesuneme prvek y to přihrádky c a prvek x vložíme do b. Tímto se zkrátí očekávaná doba hledání.

Hešování posloupností pevné délky

Scalar-mod-prime

- Chceme hešovat d-tici $x_1, \dots, x_d \in \mathbb{Z}_p$, kde p je prvočíslo
- $\left\{x_1, \dots, x_d \to \sum_{i=1}^d a_i x_i \mod p;\ a \in \mathbb{Z}_p^d\right\}$ je 1-universální
- $\left\{x_1, \dots, x_d \to b + \sum_{i=1}^d a_i x_i \mod p;\ a \in \mathbb{Z}_p^d, b \in \mathbb{Z}_p\right\}$ je (2,1)-nezávislý
- $\left\{x_1, \dots, x_d \to \left(b + \sum_{i=1}^d a_i x_i \mod p\right) \mod m;\ a \in \mathbb{Z}_p^d, b \in \mathbb{Z}_p\right\}$ je (2,4)-nezávislý

Důkaz 1-universálnosti ①

- Mějme různé $x, y \in \mathbb{Z}_p^d$ a BÚNO předpokládejme, že $x_1 \neq y_1$
- $P[a \cdot x \equiv_p a \cdot y] = P[a \cdot (x - y) \equiv_p 0] = P\left[a_1 \equiv_p \frac{\sum_{i=2}^d a_i (y_i - x_i)}{x_1 - y_1}\right] = 1/p$ ②

1. Pro 2-nezávislost stačí podobně nahlednout, že a_1, b jsou jednoznačně určené.
2. Náhodná proměnná a_1 musí nabývat jednu konkrétní hodnotu, což nastane s pravděpodobností $1/p$.

Poly-mod-prime pro různě dlouhé řetězce I

- Chceme hešovat řetězec $x_1, \dots, x_d \in \mathbb{Z}_p$, kde p je prvočíslo
- $\left\{ x_1, \dots, x_d \to \sum_{i=0}^{d-1} x_{i+1} a^i \bmod p;\ a \in [p] \right\}$ je d-universální
- Dva různé polynomy stupně nejvýše $d-1$ mají nejvýše d společných bodů, takže existuje nejvýše d kolidujících hodnot a.

Poly-mod-prime pro různě dlouhé řetězce II

- Chceme hešovat řetězec $x_1, \dots, x_d \in U$ do M, kde $p \geq m$ je prvočíslo
- $h_{a,b,c}(x_1, \dots, x_d) = \left(b + c \sum_{i=0}^{d-1} x_{i+1} a^i \bmod p \right) \bmod m$
- $\mathcal{H} = \{ h_{a,b,c};\ a, b, c \in [p] \}$
- $P[h_{a,b,c}(x_1, \dots, x_d) = h_{a,b,c}(x'_1, \dots, x'_{d'})] \leq \frac{2}{m}$ pro různé řetězce délek $d, d' \leq \frac{p}{m}$.