Připomínka: Složitost KMI/ALM?, KMI/VSL

Θ-notace a $\mathcal{O}$-notace

$$\Theta(g(n)) = \{f(n) \mid \text{existují kladné konstanty } c_1, c_2 \text{ a } n_0, \text{ t.ž.}$$
$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ pro všechna } n \geq n_0\}$$

Místo $f(n) \in \Theta(g(n))$ píšeme $f(n) = \Theta(g(n))$.
Říkáme, že $g(n)$ je asymptoticky těsná hranice (asymptotically tight bound) pro $f(n)$.

$$\mathcal{O}(g(n)) = \{f(n) \mid \text{existují kladné konstanty } c \text{ a } n_0, \text{ t.ž.}$$
$$0 \leq f(n) \leq c g(n) \text{ pro všechna } n \geq n_0\}$$

Místo $f(n) \in \mathcal{O}(g(n))$ píšeme $f(n) = \mathcal{O}(g(n))$.
Říkáme, že $g(n)$ je asymptotická horní hranice (asymptotic upper bound) pro $f(n)$.

Prelude: Harmonická čísla

(budeme je potřebovat dnes i příště)

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k} = \ln n + \mathcal{O}(1) \qquad (*)$$

Jak se (*) dokáže: aproximací integrály
Když se dá sumace vyjádřit jako $\sum_{k=m}^{n} f(k)$, kde $f(k)$ je monotónně klesající funkce, můžeme ji aproximovat:

$$\int_{m}^{n+1} f(x)dx \leq \sum_{k=m}^{n} f(k) \leq \int_{m-1}^{n} f(x)dx$$

Dolní hranice:

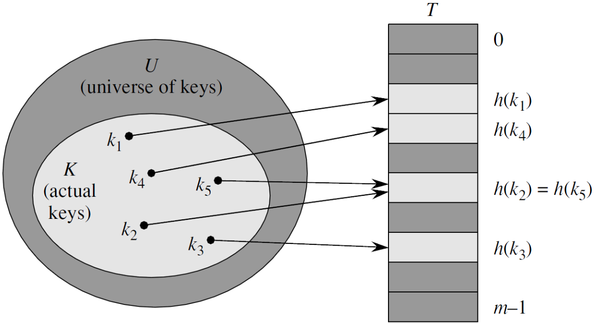
$$\sum_{k=1}^{n} \frac{1}{k} \geq \int_{1}^{n+1} \frac{1}{x} dx = \ln(n+1)$$

Horní hranice:

$$\sum_{k=2}^{n} \frac{1}{k} \leq \int_{1}^{n} \frac{1}{x} dx = \ln n \quad \text{a tedy} \quad \sum_{k=1}^{n} \frac{1}{k} \leq \ln n + 1$$

Připomínka: Hašování KMI/ALM?

Už známe...

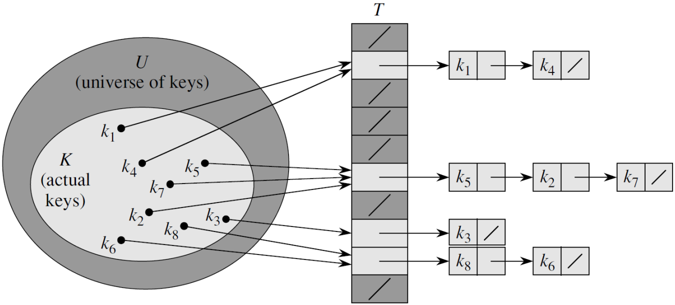


ukradený obrázek z Cormena

Řešení kolizí:

- řetězení,
- otevřené adresování.

Hašování s řetězením



ukradený obrázek z Cormena

Analýza hašování s řetězením

Nejhorší případ: hrůza. Průměrný případ?

Faktor zaplnění α (load factor) je n/m, tedy průměrný počet prvků v řetězu. Průměrný výkon závisí na tom, jak dobře (v průměru) hašovací funkce distribuuje klíče do m slotů. Prozatím budeme předpokládat, že daný prvek je se stejnou pravděpodobností nahašován do libovolného z m slotů, nezávisle na tom, kam byly hašovány ostatní prvky.
Toto nazýváme předpoklad *jednoduchého uniformního hašování* (simple uniform hashing), **SUH**.

Pro $j = 0, 1, \ldots, m-1$, označme délku seznamu $T[j]$ jako n_j, takže

$$n = n_0 + n_1 + \cdots + n_{m-1},$$

a očekávaná hodnota n_j je $E[n_j] = \alpha = n/m$.

Předpokládáme, že $h(k)$ je počítáno v čase $\mathcal{O}(1)$, takže čas hledání prvku s klíčem k je lineárně závislý na délce $n_{h(k)}$ seznamu $T[h(k)]$.
Podíváme se na to, kolik prvků musíme prozkoumat při úspěšném a neúspěšném hledání.

Věta

V hašovací tabulce, ve které jsou kolize řešeny řetězením, neúspěšné hledání zabere očekávány čas $\Theta(1 + \alpha)$ za předpokladu **SUH**.

Důkaz.

Za předpokladu **SUH** je libovolný klíč k, který doposud není uložený v tabulce se hašuje se stejnou pravděpodobností do kteréhokoli z m slotů.
Očekávaný čas neúspěšného hledání je klíče k je očekávaný čas hledání do konce seznamu $T[h(k)]$, který má očekávanou délku $E[n_{h(k)}] = \alpha$.
Tedy očekávaný počet prvků zkoumaných při neúspěšném hledání je α, a celkový čas (včetně času pro výpočet $h(k)$ je $\Theta(1 + \alpha)$). $\square$

Věta

V hašovací tabulce, ve které jsou kolize řešeny řetězením, úspěšné hledání zabere očekávány čas $\Theta(1+\alpha)$ *za předpokladu* **SUH**.

Důkaz

Předpokládáme, že prvek který hledáme, je se stejnou pravděpodobností kterýkoli z n prvků uložených v tabulce.
Počet zkoumaných prvků během úspěšného hledání prvku x je $1+$ počet prvků, které jsou v seznamu (ve kterém je x) před x.
To je tedy počet prvků, které byly vloženy po x (vkládáme na začátek seznamu).
Nechť x_i označuje $i$tý prvek vložený do tabulky pro $i = 1, 2, \ldots, n$ a nechť $k_i = \mathrm{key}[x_i]$.
Pro klíče k_i a k_j definujeme náhodnou proměnnou $X_{ij} = I\{h(k_i) = h(k_j)\}$.
Z předpokladu **SUH** máme $Pr\{h(k_i) = h(k_j)\} = 1/m$, a tedy $E[X_{ij}] = 1/m$.

Takže očekávaný počet zkoumaných prvků při úspěšném hledání je:

$$E\left[\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}X_{ij}\right)\right]=\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}E\left[X_{ij}\right]\right)=$$

$$=\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}\frac{1}{m}\right)=1+\frac{1}{nm}\sum_{i=1}^{n}(n-i)=$$

$$=1+\frac{1}{nm}\left(\sum_{i=1}^{n}n-\sum_{i=1}^{n}i\right)=1+\frac{1}{nm}\left(n^2-\frac{n(n+1)}{2}\right)=$$

$$=1+\frac{n-1}{2m}=1+\frac{\alpha}{2}-\frac{\alpha}{2n}$$

Takže celkově je to $\Theta(2+\alpha/2-\alpha/2n)=\Theta(1+\alpha)$.

Univerzální hašování (universal hashing)

Pro každou pevně danou hašovací funkci můžeme vybrat n klíčů tak, aby se přiřadily stejnému slotu, a tím dostaneme průměrný čas hledání $\Theta(n)$.
Jediný efektivní způsob, jak to vylepšit je vybírat hašovací funkci náhodně, způsobem, který je nezávislý na klíčích, které mají být uloženy — univerzální hašování.
Hlavní idea: náhodně vybrat z dobře navržené třídy hašovacích funkcí na začátku spuštění.
Nechť $\mathcal{H}$ je konečná kolekce hašovacích funkcí, která zobrazuje dané universum U klíčů do množiny $\{0, 1, \ldots, m-1\}$. Taková kolekce se nazývá *univerzální*, pokud pro každý pár různých klíčů $k, l \in U$, počet hašovacích funkcí h, pro které je

$$h(k) = h(l)$$

je nejvýše $|\mathcal{H}/m|$.

Věta

Uvažujme, že hašovací funkce h je vybrána z univerzální třídy hašovacích funkcí a je použita k hašování n klíčů do tabulky T velikosti m s řetězením. Pokud klíč k není v T, pak očekávaná délka $E[n_{h(k)}]$ seznamu, do kterého se nahašuje k je nejvýše α. Pokud klíč k je v T, pak očekávaná délka $E[n_{h(k)}]$ seznamu, který obsahuje k je nejvýše $1 + \alpha$.

Důkaz

Pro každou dvojici k, l různých klíčů definujeme náhodnou proměnnou

$$X_{kl} = I\{h(k) = h(l)\}$$

Dle definice, dvojice klíčů koliduje s pravděpodobností $1/m$, takže

$$Pr\{h(k) = h(l)\} \leq 1/m,$$

a tedy $E[X_{kl}] \leq 1/m$.

Definujeme pro každý klíč k náhodnou proměnnou Y_k, která se rovná počtu klíčů jiných než k, které se hašují do stejného slotu, tedy

$$Y_k = \sum_{l \in T, l \neq k} X_{kl}.$$

Takže máme

$$E[Y_k] = E\left[\sum_{l \in T, l \neq k} X_{kl}\right] = \sum_{l \in T, l \neq k} E\left[X_{kl}\right] \leq \sum_{l \in T, l \neq k} \frac{1}{m}.$$

Zbytek záleží na tom, jestli je klíč k v T nebo ne.

- ne Pokud $k \notin T$, pak $n_{h(k)} = Y_k$ a

$$|\{l \mid l \in T \text{ a } l \neq k\}| = n.$$

  Tedy $E[n_{h(k)}] = E[Y_k] \leq n/m = \alpha$.

- ano Pokud $k \in T$, pak protože klíč k se objevuje v seznamu $T[h(k)]$ a počet Y_k nezahrnuje klíč k, máme $n_{h(k)} = Y_k + 1$ a

$$|\{l \mid l \in T \text{ a } l \neq k\}| = n - 1.$$

  Takže $E[n_{h(k)}] = E[Y_k] + 1 \leq (n-1)/m + 1 = 1 + \alpha - 1/m < 1 + \alpha$.

Z toho máme důsledek, že jakákoli sekvence operací (při univerzálním hašování) může být provedena v dobrém očekávaném čase.

Důsledek

Použitím univerzálního hašování a řešení kolizí řetězením v tabulce s m sloty, zabere očekávaný čas $\Theta(n)$ jakákoli sekvence n operací INSERT, SEARCH, *a* DELETE, *která obsahuje $\mathcal{O}(m)$ operací* INSERT.

Důkaz.

Protože počet vložení je $\mathcal{O}(m)$, máme $n = \mathcal{O}(m)$ a tedy $\alpha = \mathcal{O}(1)$. INSERT a DELETE zaberou konstantní čas a dle předchozí věty očekávaný čas pro SEARCH je $\mathcal{O}(1)$. Dle linearity očekávání je očekávaný čas pro celou sekvenci operací $\mathcal{O}(n)$. $\square$

Návrh univerzální třídy hašovacích funkcí.

- Vybereme prvočíslo p tak, aby všechny klíče byly v rozmezí 0 až $p-1$ (včetně). Označme

$$\mathbf{Z}_p^* = \{1, 2, \ldots, p-1\} \quad \text{a} \quad \mathbf{Z}_p = \{0, 1, 2, \ldots, p-1\}$$

- Pro $a \in \mathbf{Z}_p^*, b \in \mathbf{Z}_p$ definujeme hašovací fukci $h_{a,b}$ jako

$$h_{a,b}(k) = ((a \cdot k + b) \mod p) \mod m$$

  Např: pro $p = 17$, $m = 6$, máme $h_{3,4}(8) = 5$.

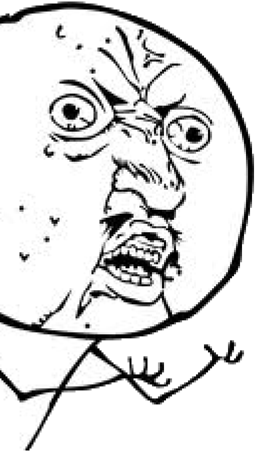
- Třída všech takových hašovacích funkcí (pro dané p a m) je

$$\mathcal{H}_{p,m} = \{h_{a,b} \mid a \in \mathbf{Z}_p^*, b \in \mathbf{Z}_p\}.$$

Každá $h_{a,b}$ zobrazuje $\mathbf{Z}_p$ na $\mathbf{Z}_m$.
Velikost m je libovolná (bude se hodit později).
V $\mathcal{H}_{p,m}$ je $p(p-1)$ hašovacích funkcí.

Věta

Třída $\mathcal{H}_{p,m}$ hašovacích funkcí je univerzální.



Y R U UNIVERSAL???

Navrhnout to bylo docela jednoduché, ale budeme potřebovat pár výsledků z teorie čísel, abychom dokázali, že to tak skutečně je.

Věta

Pokud a, b jsou celá čísla a alespoň jedno je nenulové, pak $\gcd(a, b)$ je nejmenší kladný prvek množiny $\{ax + by \mid x, y \in \mathbf{Z}\}$ lineárních kombinací a, b.

Důkaz.

Nechť s je nejmenší kladná lineární kombinace a, b a nechť $s = ax + by$ pro nějaké $x, y \in \mathbf{Z}$
Pak máme:

$$\begin{aligned} a \mod s &= a - qs \\ &= a - q(ax + by) \\ &= a(1 - qx) + b(-qy), \end{aligned}$$

takže $a \mod s$ je také lineární kombinace a, b.
Ale protože $a \mod s < s$, dostáváme, že $a \mod s = 0$,
protože s je nejmenší kladná lineární kombinace. Takže $s \mid a$, a analogicky $s \mid b$ a tedy $\gcd(a, b) \geq s$.
Protože $\gcd(a, b)$ dělí a, b a s je lineární kombinace a, b, dostáváme, že $\gcd(a, b)$ dělí s.
Z $\gcd(a, b) \mid s$ a $s > 0$ dostáváme $\gcd(a, b) \leq s$. □

Věta

Pro jakákoli celá čísla $a, b, p \in \mathbf{Z}$ *platí:*

$$\textit{pokud } \gcd(a,p) = 1 \textit{ a } \gcd(b,p) = 1, \textit{ pak } \gcd(ab,p) = 1.$$

Důkaz.

Z předchozí věty vyplývá, že existují čísla x, y, x', y', t.ž.

$$ax + py = 1 \quad bx' + py' = 1$$

Vynásobením těch dvou rovnic dostáváme:

$$ab(xx') + p(ybx' + y'ax + pyy') = 1.$$

Takže 1 je kladná lineární kombinace ab a p.
Použijeme předchozí větu. $\square$

Důkaz univerzálnosti třídy $\mathcal{H}_{p,m}$

Uvažujme dva různé klíče $k, l \in \mathbf{Z}_p$.
Nechť pro danou hašovací funkci $h_{a,b}$ platí

$$r = (ak + b) \mod p$$
$$s = (al + b) \mod p$$

Nejdříve si ukážeme, že $r \neq s$: Všimněme si, že

$$r - s \equiv a(k - l) \pmod p.$$

Z toho vyplývá, že $r \neq s$ protože p je prvočíslo a a i $(k - l)$ jsou nenulové modulo p, takže jejich součin musí být také nenulový modulo p dle předchozí věty.
Takže během výpočtu kterékoli $h_{a,b} \in \mathcal{H}_{p,m}$ jsou k a l zobrazeny na různé hodnoty r a s modulo p – tady ještě nejsou žádné kolize.

Navíc, každá z $p(p-1)$ dvojic (a, b) dá jako výsledek jinou dvojici (r, s), protože můžeme řešit:

$$a = ((r-s)((k-l)^{-1} \mod p) \mod p,$$
$$b = (r - ak) \mod p,$$

$((k-l)^{-1} \mod p)$ označuje unikátní inverzi modulo p čísla $k-l$.
Protože je pouze $p(p-1)$ dvojic (r, s), t.ž. $r \neq s$, existuje bijekce mezi dvojicemi (a, b) s $a \neq 0$ a dvojicemi (r, s) s $r \neq s$.
Pro danou hodnotu r, ze zbývajících $p-1$ možných hodnot s je počet hodnot s, které splňují

$$s \neq r \quad \text{and} \quad s \equiv r \pmod m$$

nejvýše

$$\lceil p/m \rceil - 1 \leq ((p+m-1)/m) - 1 = (p-1)/m$$

Pravděpodobnost, že s koliduje s r, když jsou redukovány modulo m, je nejvýše

$$((p-1)/m)/(p-1) = 1/m$$

Takže pro jakoukoli dvojici různých $k, l \in \mathbf{Z}_p$ platí

$$Pr\{h_{a,b}(k) = h_{a,b}(l)\} \leq 1/m, \text{ a tedy } \mathcal{H}_{p,m} \text{ je univerzální.}$$

Otevřené adresování (open addressing)

- všechny prvky jsou uloženy v samotné hašovací tabulce; hašovací tabulka může být naplněna, tj. α nemůže překročit hodnotu 1.
- při vyhledávání se systematicky zkoumají sloty tabulky, dokud není nalezen hledaný element nebo není jasné, že v tabulce není.
- nepotřebujeme seznamy a ukazatele, místo nich se počítá sekvence slotů, které mají být prozkoumány = sondování (probing)

To, které sloty budou zkoumány, je závislé na klíči. Hašovací funkce je tedy

$$h : U \times \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}$$

Pro každý klíč k potřebujeme posloupnost sond (probe sequence)

$$\langle h(k, 0), h(k, 1), \ldots, h(k, m-1) \rangle,$$

která je permutací $\langle 0, 1, \ldots, m-1 \rangle$.

V následující analýze budeme používat předpoklad *uniformního hašování*, (uniform hashing), **UH**.
To jest, že každý klíč bude jako posloupnost sond mít se stejnou pravděpodobností libovolnou z $m!$ permutací $\langle 0, 1, \ldots, m-1 \rangle$.

Tři techniky jsou obvykle používány k výpočtu sekvence sond pro otevřené adresování:

- *lineární sondování* (linear probing)
- *kvadratické sondování* (quadratic probing)
- *dvojité hašování* (double hashing)

Lineární sondování

Máme *pomocnou hašovací funkci* $h' : U \to \{0, 1, \ldots, m-1\}$. Používáme hašovací funkci

$$h(k, i) = (h(k') + i) \mod m$$

pro $i = 0, 1, \ldots, m-1$.
Pro daný klíč k, je nejprve prozkoumán $T[h'(k)]$, pak slot $T[h'(k)+1], \ldots, T[m-1]$, pak zase od $T[0]$, až dojdeme k $T[h'(k)-1]$.

Problém: *primární shlukování* (primary clustering): shluky vznikají, protože prázdný slot, kterému předchází i plných slotů bude zaplněn jako další s pravděpodobností $(i+1)/m$.

Kvadratické sondování

Máme *pomocnou hašovací funkci* $h' : U \to \{0, 1, \ldots, m-1\}$ a pomocné konstanty $c_1, c_2 \neq 0$. Používáme hašovací funkci

$$h(k,i) = (h(k') + c_1 i + c_2 i^2) \mod m$$

pro $i = 0, 1, \ldots, m-1$.
Pracuje to lépe než lineární sondování, ale aby byla hašovací tabulka plně využita, musí být omezeny hodnoty c_1, c_2, m.

Problém: pokud mají dva klíče stejnou iniciální pozici $h(k_1, 0) = h(k_2, 0)$, pak mají stejnou celou sondovací sekvenci, tedy $h(k_1, i) = h(k_2, i)$ pro $i = 0, \ldots, m-1$. To vede k slabší podobě shlukování – tzv. *sekundární shlukování* (secondary clustering).

Dvojité hašování

používá hašovací funkci

$$h(k, i) = (h_1(k) + ih_2(k)) \mod m,$$

kde h_1, h_2 jsou pomocné hašovací funkce. První sonda jde na pozici $T[h_1(k)]$, následující sonda je odsazena o hodnotu $h_2(k)$, modulo m.
Hodnota $h_2(k)$ musí být nesoudělná s velikostí hašovací tabulky m, aby byla prohledána celá tabulka.
Jednoduchý způsob, jak to udělat:

- vzít m jako mocninu 2 a navrhnout h_2, t.ž. výsledkem bude vždy liché číslo;
- (nebo) m zvolit jako prvočíslo a navrhnout h_2, t.ž. výsledkem bude kladné číslo $< m$.

Dvojité hašování je lepší než lineární sondování či kvadratické sondování, protože generuje $\Theta(m^2)$ posloupností sond místo $\Theta(m)$.

Analýza otevřeného adresování

Věta

Pro hašovací tabulku s otevřeným adresováním s faktorem zaplnění $\alpha = n/m < 1$ je očekávaný počet sond při neúspěšném hledání nejvýše $1/(1-\alpha)$ za předpokladu **UH**.

Důkaz

Při neúspěšném hledání, každá sonda – až na poslední – zkoumá obsazený slot, který neobsahuje hledaný klíč; a poslední je prázdný.
Definujme náhodnou proměnnou X jako počet sond potřebných při neúspěšném hledání.
Dále definujeme jev A_i pro $i = 1, \ldots,$ je jev: „existuje i-tá sonda a zkoumá obsazený slot".
Pak jev $\{X \geq i\}$ je průnik jevů $A_1 \cap A_2 \cap \cdots \cap A_{i-1}$.

$$Pr\{A_1 \cap A_2 \cap \cdots \cap A_{i-1}\} = Pr\{A_1\} \cdot Pr\{A_2 \,|\, A_1\} \cdot Pr\{A_3 \,|\, A_1 \cap A_2\} \cdots Pr\{A_{i-1} \,|\, A_1 \cap A_2 \cap \cdots \cap A_{i-2}\}$$

(viz KMI/PRAS)

Protože máme n prvků a m slotu, je $Pr\{A_1\} = n/m$.
Pro $j > 1$: $Pr\{A_j \mid A_1 \cap \cdots \cap A_{j-1}\} = (n - j + 1)/(m - j + 1)$,
protože hledáme jeden ze zbývajících $(n - j + 1)$ prvků v jednom z $(m - (j + 1))$ neprozkoumaných slotů (s předpokladem **UH**).

Dále, protože $n < m$ implikuje $(n - j)/(m - j) \leq n/m$ pro všechna $0 \leq j < m$, platí všechna $0 \leq i \leq m$:

$$Pr\{X \geq i\} = \frac{n}{m} \cdot \frac{n-1}{m-1} \cdot \frac{n-2}{m-2} \cdots \frac{n-i+2}{m-i+2} \leq \left(\frac{n}{m}\right)^{i-1} = \alpha^{i-1}.$$

No a teď můžeme ohraničit očekávané množství sond:

$$E[X] = \sum_{i=1}^{\infty} Pr\{X \geq i\} \leq \sum_{i=1}^{\infty} \alpha^{i-1} = \sum_{i=0}^{\infty} \alpha^{i} = \frac{1}{1-\alpha}.$$

Důsledek

Vkládání prvku do hašovací tabulky s otevřeným adresováním s faktorem zaplnění α vyžaduje v průměru nejvýše $1/(1-\alpha)$ sond, s předpokladem **UH**.

Důkaz.

Prvek je vložen do tabulky jenom, pokud je tam místo, tedy $\alpha < 1$.
Vložení klíče je vlastně neúspěšné hledání následované umístěním klíče do prvního prázdného slotu, který je nalezen.
Tedy očekávaný počet sond je nejvýše $\frac{1}{1-\alpha}$. $\square$

Věta

Mějme tabulku s otevřeným adresováním a s faktorem zaplnění $\alpha < 1$, očekávané množství sond při úspěšném hledání je nejvýše.

$$\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$$

za předpokladu **UH** *a předpokladu, že každý klíč v tabulce bude hledán se stejnou pravděpodobností.*

Důkaz

Hledání klíče k bude následovat stejnou posloupnost sond, jako když byl klíč k vkládán. Dle předchozího důsledku, pokud k bylo $(i+1)$-ní klíč vložený do tabulky, očekávané množství sond při hledání k je nejvýše $(1/(1-i/m)) = m/(m-i)$.

Zprůměrováním přes všech n klíčů v tabulce dostáváme průměrný počet sond při úspěšném hledání:

$$\frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i} = \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} = \frac{1}{\alpha}(H_m - H_{m-n}),$$

kde $H_i = \sum_{j=1}^{i}\frac{1}{j}$ je i-té harmonické číslo.
Použijeme aproximaci integrálem:

$$\begin{aligned}
\frac{1}{\alpha}(H_m - H_{m-n}) &= \frac{1}{\alpha}\sum_{k=m-n+1}^{m}\frac{1}{k} < \\
&< \frac{1}{\alpha}\int_{m-n}^{m}\frac{1}{x}dx = \\
&= \frac{1}{\alpha}\ln\frac{m}{m-n} = \\
&= \frac{1}{\alpha}\ln\frac{1}{1-\alpha}.
\end{aligned}$$

Dokonalé hašování

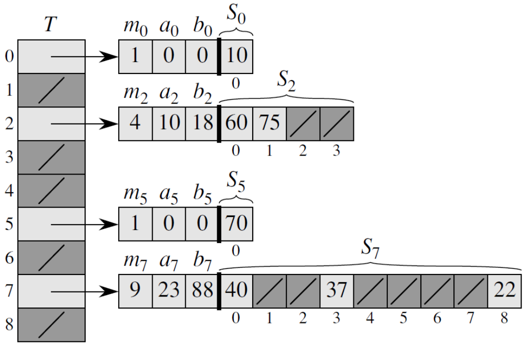
Hašování může mít skvělý výkon *v nejhorším případě* v případě, že je množina klíčů *statická*. Např.

- množina rezervovaných slov v programovacím jazyce,
- množina jmen souborů na CD-ROM.

Základní myšlenka: Použít dvojúrovňové schéma s univerzálním hašováním na obou úrovních.

- první úroveň – v podstatě stejné jako hašování s řetězením.
- druhá úroveň – místo seznamů použijeme *sekundární hašovací tabulky* S_j s asociovanou hašovací funkcí h_j. Vhodným výběrem můžeme zajistit, aby na sekundární úrovni nebyly žádné kolize.

Ukradený obrázek



Abychom zajistili, že na druhé úrovni nebudou žádné kolize, potřebujeme $m_j = n_j^2$, kde m_j je velikost sekundární tabulky ve slotu j, n_j je počet klíčů, které se tam nahašují.

To se může zdát hodně, ale uvidíme, že při vhodné volbě hašovací funkce na první úrovni bude očekávané množství použité paměti $\mathcal{O}(n)$.

Tu funkci vezmeme z $\mathcal{H}_{p,m}$. Klíče, které se hašují do slotu j jsou přehašovány do sekundární tabulky S_j velikosti m_j použitím hašovací funkce z $\mathcal{H}_{p,m_j}$

V následujím pujde o dvě věci:

- jak zajistit, že na druhé úrovni nebudou kolize.
- dokázat, že předp. množství použité paměti je $\mathcal{O}(n)$.

Věta

Pokud uložíme n klíčů do hašovací tabulky velikosti $m = n^2$ s použitím hašovací funkce náhodně vybrané z univerzální třídy hašovacích funkcí, pak pravděpodobnost, že nastane kolize je menší než $1/2$.

Důkaz.

Existuje $\binom{n}{2}$ párů klíčů, které mohou kolidovat; každý pár koliduje s pravděpodobností $1/m$, pokud je h vybraná z univerzální třídy $\mathcal{H}$ hašovacích funkcí. Nechť X je náhodná proměnná, která představuje počet kolízi. Pokud platí $m = n^2$, pak očekávané množství kolízi je

$$E[X] = \binom{n}{2} \cdot \frac{1}{n^2} = \frac{n^2 - n}{2} \cdot \frac{1}{n^2} < 1/2.$$

Použijeme Markovovu nerovnost $Pr\{X \geq t\} \leq E[X]/t$ pro $t = 1$ a je vymalováno. □

Věta

Pokud uložíme n *klíčů v hašovací tabulce velikosti* $m = n$ *použitím hašovací funkce* h *náhodně vybrané z univerzální třídy hašovacích funkcí, pak*

$$E\left[\sum_{j=0}^{m-1} n_j^2\right] < 2n,$$

kde n_j *je počet klíčů hašovaných do slotu* j.

Důkaz

Začneme následující rovností, která platí pro libovolné nezáporné celé číslo a:

$$a^2 = a + 2\binom{a}{2}.$$

Platí, že

$$
\begin{aligned}
E\left[\sum_{j=0}^{m-1} n_j^2\right] &= E\left[\sum_{j=0}^{m-1}\left(n_j + 2\binom{n_j}{2}\right)\right] \\
&= E\left[\sum_{j=0}^{m-1} n_j\right] + 2E\left[\sum_{j=0}^{m-1}\binom{n_j}{2}\right] \\
&= E\left[n\right] + 2E\left[\sum_{j=0}^{m-1}\binom{n_j}{2}\right] \\
&= n + 2E\left[\sum_{j=0}^{m-1}\binom{n_j}{2}\right]
\end{aligned}
$$

Suma $\sum_{j=0}^{m-1} \binom{n_j}{2}$ je vlastně celkový počet kolizí.
Podle vlastností univerzálního hašování, očekávaná hodnota této sumy je nejvýše

$$\binom{n_j}{2}\frac{1}{m} = \frac{n(n-1)}{2m} = \frac{n-1}{2},$$

protože $m = n$.
Takže

$$E\left[\sum_{j=0}^{m-1} n_j^2\right] \leq n + 2\frac{n-1}{2} = 2n - 1 < 2n.$$

Důsledek

Pokud uložíme n klíčů v hašovací tabulce velikosti $n = m$ použitím hašovací funkce h náhodně vybrané z univerzální třídy hašovacích funkcí a nastavíme velikost každé sekundární tabulky na $m_j = n_j^2$ pro $j = 0, 1, \ldots, m-1$, pak očekávané množství paměti potřebné k uložení všech sekundárních hašovacích tabulek v perfektním hašování je méně než $2n$.

Důkaz.

Protože $m_j = n_j^2$ pro $j = 0, 1, \ldots, m-1$, předchozí věta dává

$$E\left[\sum_{j=0}^{m-1} m_j\right] = E\left[\sum_{j=0}^{m-1} n_j^2\right] < 2n.$$

□

Důsledek

Pokud vybereme n klíčů v hašovací tabulce velikosti $m = n$ použitím hašovací funkce h náhodně vybrané z univerzální třídy hašovacích funkcí, a nastavíme velikost každé sekundární tabulky na $m_j = n_j^2$ pro $j = 0, 1, \ldots, m-1$, pak pravděpodobnost že celková paměť použitá pro sekundární hašovací tabulku překročí $4n$ je méně než 0.5.

Důkaz.

Použijeme Markovovu nerovnost, $Pr\{X \geq t\} \leq E[X]/t$, na nerovnost z předchozího důkazu s $X = \sum_0^{m-1} m_j$ a $t = 4n$:

$$Pr\left\{\sum_{j=0}^{m-1} m_j \geq 4n\right\} \leq \frac{E\left[\sum_{j=0}^{m-1} m_j\right]}{4n} < \frac{2n}{4n} = \frac{1}{2}$$

□

Takže po otestování několika náhodně vybraných hašovacích funkcí najdeme takovou, která využívá rozumné množství paměti.

ZÁPOČTOVÝ ÚKOL

Cactus Kev's Poker Hand Evaluator
`http://www.suffecool.net/poker/evaluator.html`

LATE BREAKING NEWS!!! Paul Senzee of Florida decided that he could speed up my evaluator by using a pre-computed perfect hash function instead of a binary search for those final 4888 hand values. He says he obtained a speedup factor of 2.7x!

Úkol: udělat to, co udělal Paul Senzee.