Hešování s lineárním přidáváním - neúspěšný případ

6. dubna 2001, Martin Kopečný

Vkládání a vyhledávání záznamů

Algoritmus hešování s lineárním přidáváním pracuje s hešovací funkcí h a tabulku s M adresovatelými pozicemi. Při přístupu do tabulky (vkládání, vyhledávání, mazání klíče) nejprve použije hešovací funkci k určení počátečního indexu, na kterém by se daný klíč mohl nacházet a v případě neúspěchu začne tabulkou cyklicky procházet ve snaze nalézt zadaný klíč na ostatních pozicích.

Cílem tohoto referátu je odhadnout průměrný počet testů (přístupů do tabulky) potřebných v neúspěšném případě vyhledávání zadaného klíče K.

Předpoklady

Jednotlivé položky tabulky budeme značit `TABLE[i]`, pro $0 \leq i < M$, a budou dvojího typu — *prázdné* a *obsazené*. Obsazené položky tabulky obsahují klíč `KEY[i]` a případná další data. S tabulkou je svázána hodnota N udávající aktuální počet obsazených pozic tabulky.

V algoritmu je použita hešovací funkce $h(K)$, která k zadanému klíči K udává počáteční index i do tabulky, od kterého začíná sekvenční prohledávání. Předpokládejme, že funkce $h(K)$ rozděluje hodnoty rovnoměrně.

Algoritmus A

A1 `hešování`
Vypočti hodnotu $h(K)$ a ulož ji do i $(0 \leq i < M)$

A2 `porovnání`
Jestliže `KEY[i]`$= K$, algoritmus končí *úspěšně*.
Jestliže je `TABLE[i]` prázdná, jdi na krok A4.

A3 `posun`
Do i přiřadíme hodnotu $i - 1$.
Pokud je nyní $i < 0$, nastav $i = i + M$ a pokračuj krokem A2

A4 `vložení` (*neúspěšné hledání*)
Pokud je $N = M - 1$, potom algoritmus končí *přetečením*.
Jinak nastav $N = N + 1$, označ `TABLE[i]` jako obsazenou a ulož klíč K do `KEY[i]`.

Tabulka s M adresami je schopná pojmout max. $M-1$ klíčů. Tedy v tabulce zbývá i při maximálním naplnění jedna položka volná. Toho je využito pro zjednodušení kroku 2 a 3, kde se nemusí hlídat počet průchodů celou tabulkou a je zaručeno, že se algoritmus A vždy zastaví.

V praxi bylo ověřeno, že algoritmus A pracuje dobře do té doby, než se hodnota N začne blížit celkovému počtu všech adres v tabulce. Nejlepšího výkonu se dosahuje při zaplnění do 75% ($\alpha = \frac{N}{M}$ ... *load factor*).

Důvodem snížení výkonu při vyšším zaplnění je stále častější prohledávání zaplněných úseků. Následující příklad demonstruje tuto vlastnost algoritmu (tabulka má parametry $M = 19$, $N = 9$):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
		X	X		X	X						X	X	X	X		X	

Další klíč K bude vložen na jednu z 10 volných pozic v tabulce. Tyto pozice však nejsou stejně pravděpodobné. Na pozici 8 bude umístěn klíč pouze v případě, že $h(K) = 8$. Avšak např. na pozici 11 bude umístěn klíč v případě, že $11 \leq h(K) \leq 15$. Tedy pozice 11 bude pět krát pravděpodobnější.

Mazání záznamů

Pro úplnost si ukažme ještě algoritmus na vypouštění klíče z tabulky. Jak si jistě každý uvědomí, nelze klíč z tabulky odstranit pouhým změněním typu políčka `TABLE[i]` z obsazeného na prázdné. Pokud by totiž za tímto polem byla uložena data, která měla být zahešována na již obsazené pole ležící před polem obsahující mazaný klíč, byla by tato data — díky způsobu hledání klíčů v algoritmu A — ztracena.

Odstranění tohoto problému lze provést dvěma způsoby. Pokud je operace mazání poměrně řídká, stačí zavést nový typ políčka v tabulce — *smazáno* — a označit takto pole s odstraňovaným klíčem.

Zároveň je však třeba poopravit i algoritmus A. Pokud je vkládán (vyhledáván) klíč K, je nutné považovat smazaná pole za pole obsazená a v případě, že je hledání neúspěšné, stačí vložit klíč na první nalezené pole typu smazáno.

Toto řešení má však jisté nevýhody. Tabulka se každým vložením stále více zaplňuje, k uvolňování však v podstatě nikdy nedochází. Jednou obsazené pole se už navždy bude při vyhledávání chovat jako pole obsazené.

Druhým řešením je provádět současně s vymazáním klíče i *reorganizaci* tabulky. Ta trvá sice déle než jednoduché označení pole za smazané, ale může výrazně zkrátit hledání v tabulce.

Algoritmus D

D1 `uvolnění pole`
Označ `TABLE[i]` jako prázdné a nastav $j = i$.

D2 `dekrementace` i
Nastav $i = i - 1$ a pokud je i záporné nastav $i = i + M$.

D3 `oprava tabulky`
Pokud je `TABLE[i]` prázdné, algoritmus končí.
Jinak nastav $r = h($`KEY[i]`$)$ (původní heš. hodnota je nyní uložena v r).
Pokud je $i \leq r < j$ nebo $r < j < i$ nebo $j < i \leq r$ jdi na krok D2.

D4 `přesun záznamu`
Nastav `TABLE[j]` = `TABLE[i]` a pokračuj krokem D1.

Algoritmus D předpokládá na vstupu tabulku zkonstruovanou algoritmem A a odstraňuje z ní klíč z pozice i. Lze dokázat, že tato dodatečná úprava nezhorší chování algoritmu.

Analýza složitosti neúspěšného případu vyhledání

Ke studiu algoritmu hešování s lineárním přidáváním je třeba zvolit správný pravděpodobnostní model. Jak už bylo ukázáno na předchozím příkladu, nelze předpokládat, že jsou všechna pole tabulky při vyhledávání (vkládání) stejně pravděpodobná.

Zvolíme tedy pravděpodobnostní model takový, ve kterém budeme předpokládat, že každá z M^N možných *hešovacích posloupností*

$$a_1\ a_2\ \ldots\ a_N, \qquad 0 \leq a_j < M \tag{1}$$

je stejně pravděpodobná. Prvek a_j označuje počáteční adresu do tabulky, která byla vypočítána funkcí $h(K_j)$ při vkládání $j$tého klíče.

Průměrný počet testů potřebných při vkládání $(N + 1)$ního klíče do tabulky (za předpokladu, že jsou všechny hešovací posloupnosti (1) stejně pravděpodobné), budeme označovat C'_N. Je to průměrný počet testů potřebný při neúspěšném vyhledání, na jehož začátku bylo v tabulce N záznamů.

Označme $f(M, N)$ počet hešovacích posloupností (1), které ponechávají v tabulce velikosti M po vložení N záznamů pozici 0 prázdnou. Díky cyklické symetrii lineárního testování v algoritmu A je pole 0 stejně často prázdné jako jakákoliv jiná pozice.

$$f(M, N) = \left(1 - \frac{N}{M}\right) M^N \tag{2}$$

Označme $g(M,N,k)$ počet hešovacích posloupností (1), pro které algoritmus A ponechává v tabulce velikosti M po vložení N záznamů pozici 0 prázdnou, pozici 1 až k obsazenou a pozici $k+1$ prázdnou (zbylých $N-k$ záznamů je libovolně uloženo na pozicích $(k+2)$ – $(M-1)$). Tedy

$$g(M,N,k) = \binom{N}{M} f(k+1,k)\ f(M-k-1,N-k) \tag{3}$$

Nakonec si ještě označme P_k pravděpodobnost, že při vkládání $(N+1)$ního klíče do tabulky velikosti M bude potřeba přesně $k+1$ testů. Tedy

$$P_k = \frac{g(M,N,k) + g(M,N,k+1) + \cdots + g(M,N,N)}{M^N} \tag{4}$$

Průměrný počet testů potřebných při vkládání $(N+1)$ního klíče do tabulky potom bude vyjádřen vztahem

$$C'_N = \sum_{k=0}^{N} (k+1) P_k \tag{5}$$

Dosazením výrazu (4) do rovnice (5) a vynásobením obou stran rovnosti výrazem M^N dostáváme

$$\begin{aligned} M^N C'_N &= \sum_{k=0}^{N} (k+1)\big(g(M,N,k) + g(M,N,k+1) + \cdots + g(M,N,N)\big) \\ &= \sum_{k=0}^{N} \binom{k+2}{2} g(M,N,k) \\ &= \frac{1}{2}\left(\sum_{k=0}^{N} (k+1) g(M,N,k) + \sum_{k=0}^{N} (k+1)^2 g(M,N,k)\right) \end{aligned} \tag{6}$$

Poznámka: obdobná úprava, která byla použita při přechodu mezi 1. a 2. řádkem rovnosti (6), je podrobněji naznačena ve výrazu (7).

Všimněme si nejprve první sumy ve výrazu (6)

$$\begin{aligned} \sum_{k=0}^{N} (k+1) g(M,N,k) &= [\text{ součin v sumě a sumu rozepišme do sloupců }] \\ &= g(M,N,0) + g(M,N,1) + \cdots + g(M,N,N) \\ &\qquad + g(M,N,1) + \cdots + g(M,N,N) \\ &\qquad\qquad \ddots \\ &\qquad\qquad + g(M,N,N) \\ &= \sum_{k=0}^{N} g(M,N,k) + g(M,N,k+1) + \cdots + g(M,N,N) \end{aligned}$$

$$
\begin{aligned}
&= M^N \sum_{k=0}^{N} \frac{g(M,N,k) + \cdots + g(M,N,N)}{M^N} \\
&= M^N \sum_{k=0}^{N} P_k \\
&= M^N
\end{aligned} \tag{7}
$$

Ještě než se zaměříme na druhou sumu ve výrazu (6), připravme si několik užitečných vztahů a tvrzení, které se nám budou později hodit.

Nejprve si dosazením vztahu (2) upravme výraz (3):

$$
\begin{aligned}
g(M,N,k) &= \binom{N}{M} \left(1 - \frac{k}{k+1}\right) (k+1)^k \left(1 - \frac{N-k}{M-k-1}\right) (M-k-1)^{N-k} \\
&= \binom{N}{M} (k+1)^{k-1} (M-k-1)^{N-k-1} (M-N-1)
\end{aligned} \tag{8}
$$

Pro další výpočet budeme též potřebovat následující tvrzení, které si zde ovšem nebudeme dokazovat. [Důkaz lze nalézt např. v knize J. Riordan: *Combinatorial Identities* (New York: Wiley, 1968), str. 18–23.]

Tvrzení 1 *Nechť je funkce $s(n,x,y)$ tvaru*

$$
s(n,x,y) = \sum_{k \geq 0} \binom{n}{k} (x+k)^{k+1} (y-k)^{n-k-1} (y-n)
$$

Potom lze funkci vyjádřit rekurentním vztahem ve tvaru

$$
s(n,x,y) = x(x+y)^n + n s(n-1, x+1, y-1) \tag{9}
$$

□

Jako poslední pomocný vztah si označme

$$
\begin{aligned}
Q_r(M,N) &= \binom{r}{0} + \binom{r+1}{1} \frac{N}{M} + \binom{r+2}{2} \frac{N(N-1)}{M^2} + \cdots \\
&= \sum_{k \geq 0} \binom{r+k}{k} \frac{N}{M} \frac{N-1}{M} \cdots \frac{N-k+1}{M}
\end{aligned} \tag{10}
$$

Všimněme si, že platí

$$
N^k - \binom{k}{2} N^{k-1} \leq N(N-1)\cdots(N-k+1) \leq N^k \tag{11}
$$

a odhadněme (10) pro $r = 1$ za pomoci nerovností (11) a *load factor*u $\alpha = N/M$

$$\sum_{k\geq 0}\binom{k+1}{k}\left(N^k - \binom{k}{2}N^{k-1}\right) \leq \;\; Q_1(M,N) \;\; \leq \sum_{k\geq 0}\binom{k+1}{k}\frac{N^k}{M^k}$$

$$\sum_{k\geq 0}\binom{k+1}{k}\alpha^k - \frac{\alpha}{M}\sum_{k\geq 0}\binom{k+1}{k}\binom{k}{2}\alpha^{k-2} \leq Q_1(M,\alpha M) \leq \sum_{k\geq 0}\binom{k+1}{k}\alpha^k$$

$$\frac{1}{(1-\alpha)^2} - \frac{1}{M}\binom{3}{2}\frac{\alpha}{(1-\alpha)^4} \leq \;\; Q_1(M,N) \;\; \leq \frac{1}{(1-\alpha)^2} \qquad (12)$$

Nyní již máme připravena všechna pomocná tvrzení a značení a můžeme se konečně zaměřit na dopočítání vztahu pro C'_N. Dosadíme (8) do 2. sumy ve vztahu (6).

$$\begin{aligned}
&\sum_{k=0}^{N}(k\,+\,1)^2\; g(M,N,k)\\
&\qquad = \sum_{k=0}^{N}(k+1)^2\binom{N}{M}(k+1)^{k-1}(M-k-1)^{N-k-1}(M-N-1)\\
&\qquad = \sum_{k=0}^{N}\binom{N}{M}(k+1)^{k+1}(M-k-1)^{N-k-1}(M-N-1) \qquad (13)
\end{aligned}$$

Všimněme si ale, že výraz (13) je vyjádřením funce $s(n,x,y)$ pro parametry $N, 1, M-1$. Toho využijeme a na (13) aplikujeme výsledek (9) z tvrzení 1:

$$\begin{aligned}
\sum_{k=0}^{N}(k+1)^2 g(M,N,k) \;&=\; s(N,1,M-1)\\
&=\; M^N + N\big(s(N-1,2,M-2)\big)\\
&=\; M^N + N\big(2M^{N-1} + (N-1)(3M^{N-2}+\ldots)\big)\\
&=\; M^N + 2NM^{N-1} + 3N(N-1)M^{N-2} + \ldots\\
&=\; M^N\left(1 + 2\frac{N}{M} + 3\frac{N(N-1)}{M^2} + \ldots\right)\\
&=\; M^N Q_1(M,N) \qquad (14)
\end{aligned}$$

Tedy pokud dosadíme do výrazu (6) výsledky (7) a (14) dostáváme základní vztah pro C'_N

$$\begin{aligned}
M^N C'_N \;&=\; \frac{1}{2}\left(\sum_{k=0}^{N}(k+1)g(M,N,k) + \sum_{k=0}^{N}(k+1)^2 g(M,N,k)\right)\\
&=\; \frac{1}{2}\big(M^N + M^N Q_1(M,N)\big)
\end{aligned}$$

$$= M^N \frac{1}{2}\big(1 + Q_1(M,N)\big)$$

$$C'_N = \frac{1}{2}\big(1 + Q_1(M,N)\big) \tag{15}$$

Konečně, pokud ve vztahu (15) použijeme odhadu (12) založeném na faktoru naplnění $\alpha = N/M$, dostáváme vztah pro očekávaný počet testů potřebných v neúspěšném případě vyhledávání v tabulce velikosti M při N uložených klíčích:

$$C'_N \approx \frac{1}{2}\left(1 + \left(\frac{1}{1-\alpha}\right)^2\right)$$