数理解析研究所講究録
第123巻 1971年 20-31

20

Hash Coding の理論と応用

電総研　古川 康一

§1. はじめに

hash addressing は，表検索に用いられる技術の1つである。今，情報はキーとデータから成っているものとし，その表を計算機の記憶領域に置いて，キーで検索を行うものとする。情報のとり込み方を，キーと番地とその内容の関係から捉えてみると，図1のような3つの方法が考えられる。a)の index table では，キーのビット系列をそのまま番地として用いる。これは、検索は早いが，通常大きな記憶領域を必要とする。b)の sequential table では，キーと番地が直接的関係をもたず，或る順序で表をうめる。このとき，キーはデータと同様に記憶させておく必要がある。この方法はa)とは逆に、記憶領域は少くてすむが，検索が遅くなる。c)の hash table は，a)とb)の中間で，キーの情報の一部分は番地にあり，残りはその内容部にある。この方法は，a)とb)の長所を合わせもっている。

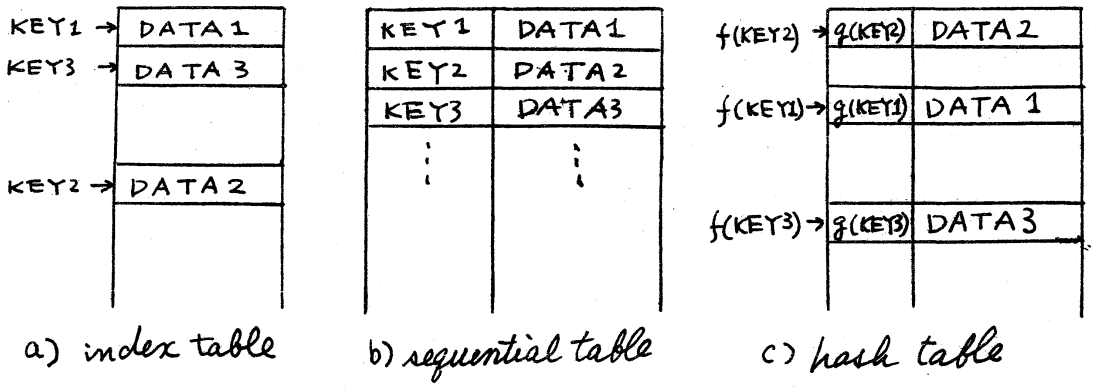


図1. 各テーブルの構成

§2. では, hash addressing を実現する変換 (hash coding) の具体的な方法, 問題点などを概説する。§3. では, hash addressing を集合演算に応用したときの改良されたアルゴリズムを与え, その効率の評価を行う。

§2. hash coding の方法[1]

hash coding は, キーの集合から表のエントリのインデックスへの写像と考えることができる。この写像は1対1ではないので, 異ったキーが同じインデックスへ写像されることが起こる。この現象をコンフリクト (またはコリジョン) という。良い hash coding とは, このコンフリクトが, 統計的に少ない方法である。それは, 各エントリに対する code が等確率に発生される場合である。また, キーの集合のもつ規則性を反映しないような方法が良い。これらの点を考慮した代表的な hash coding の方法を3つ挙げる。

(1) キーのビット系列の一部分に定数またはそれ自身をかけ、その結果のビット系列の部分系列をcodeとする。

(2) キーのビット系列の部分系列を加えたものをcodeとする。

(3) キーを表の大きさで割り、余りをcodeとする。

また、キーの集合が与えられている場合には、その集合に依存した写像を適当に見つけ出すことによって、より効率のよい表を作ることができる。[2)]

次に、コンフリクトを解消する方法について述べる。[1)] これは、サーチ・アルゴリズムと呼ばれている問題で、大きく2つの方法に分けられる。1つは、コンフリクトを起こした項目を同じ表の他の場所におく方法である。他の1つは、コンフリクトを起こした項目を、あらかじめ用意された領域において、同じhash codeをもつ項目をlinked listとして管理する方法である。この方法はdirect chaining methodと呼ばれ、サーチの効率は他の方法より良い。しかしながら、この方法では余分なポインタ領域を必要とし、自由領域の管理も必要となる。以後の議論は、主に前者について行う。

キーKが与えられたとき、それに相当する表の番地を得る場合に、もしKがすでに表中にあれば、その番地が答となる。もしなければ、表中の未だ使われていない場所を割り当てねばならない。それには、空のエントリが見つかるまで、次々

と、codeを発生し続けなければならない。今、i番目のcodeを$h_i(K)$とすると、

$$h_i(K) = h_0(K) + d(i) \qquad i \geq 1 \qquad (1)$$

と表わせる。ここで$d(i)$がどのような性質をもてばよいかは、次にあげる3つの例によって明らかにされる。

(1) Linear Search　$d(i) = i$または$d(i) = ix$とする方法で、図3のようにコンフリクトは必ずクラスタ(cluster)を生じ、そのクラスタ自身は、他のコンフリクトの処理を遅くさせる。

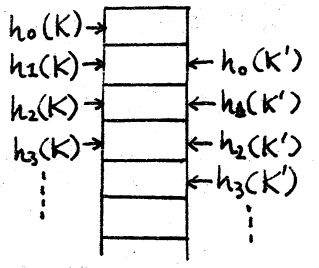


図3. Linear Searchでの1次クラスタ

(2) Random Search　表の大きさ以下の周期をもつ擬似乱数系列をr_i, $i=1,2\cdots$とするとき、$d(i) = r_i$にとる方法で、この方法は、(1)のようなクラスタは生じない。そのために空のセルを見つけるまでの回数(サーチ回数)は、(1)よりもずっと少なくなる。ただし、一回当りのサーチの手間は、(1)よりもかかる。またこの場合でも、図4のような2次クラスタを生んでいる。

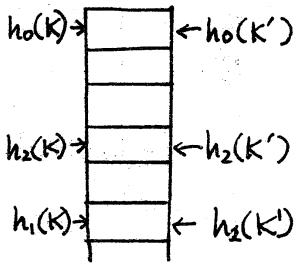


図4. Random Searchでの2次クラスタ

(3) Quadratic Search[3]　$d(i) = ax_i + bx_i^2$とする方法で、(2)に比べて$d(i)$が簡単に求まる。また、表の大きさが素数ならば、$d(i)$は表の半分のエントリに対するcodeを

発生する。この方法を改良したもので、すべてのエントリに対するcodeを発生するアルゴリズム[4),5)]、および2次クラスタを生じないアルゴリズム[6)]が発表されている。

以上の考察から、結論として、良いサーチは、クラスタを生じないこと、一回のサーチの手間がかからないこと、記憶領域が充分にとれればDirect chaining methodが最も良いことなどが云える。これらの各方法の平均サーチ回数の、表の占有率(load factor)に対するグラフを図5に示す。

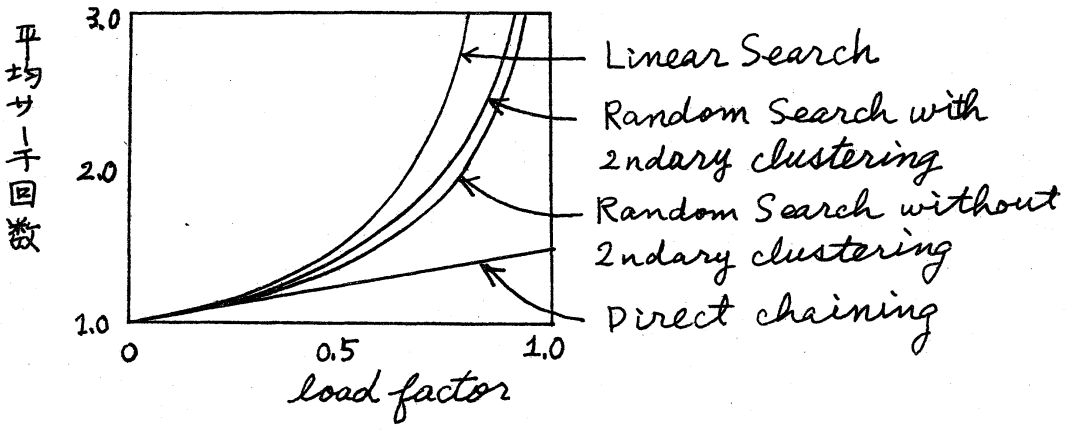


図5. 各サーチの平均サーチ回数

§3. hash addressingの集合演算への応用[7)]

キーの集合A, Bを与えて、その合併集合や共通集合を求める等の集合演算にhash addressingを使うことができる。そのとき、基本演算は一方の集合の各要素が他方の集合に含まれているかどうかを知ることである。これは、一方の集合の

すべての要素をhash tableに登録して，他方の集合の各要素で検索すればよい。この検索のときに，その表に含まれていない要素に対して，出来るだけ早く含まれていないということがわかった方がよい。この時間を棄却時間(reject time)と呼ぶことにする。ここでは，わずか1ビットを用いて棄却時間を短くするアルゴリズムを与える。それは，表を作るphaseと検索するphaseに分け，作るphaseでコンフリクトを起こしたエントリにマークをつけておき（1ビット），検索するとき，サーチはマークがあるかぎり続けるという方法である。普通，その表に含まれていない要素に対するサーチは空のセルが見つかるまで行われるが，この方法では，マークのない空でないセルにアクセスしてもサーチは終了する。したがって，その分だけ棄却時間が短縮される。

次に，このアルゴリズムによる棄却時間を求める。計算の仮定としては，クラスタは全く生じないものとする。今，hash tableの大きさをNとする。kコの要素を登録したときに，jコのエントリにマークがつく確率を$P_k(j)$とする。$P_k(j)$に対して，次の漸化式が成り立つ。

$$P_{k+1}(j) = P_k(j)\left\{\frac{N-k}{N} + \frac{j}{N}\cdot\frac{N-k}{N-1} + \frac{j(j-1)}{N(N-1)}\cdot\frac{N-k}{N-2} + \cdots \right.$$
$$\left. \cdots + \frac{j(j-1)\cdots\cdots 1}{N(N-1)\cdots(N-j+1)}\cdot\frac{N-k}{N-j}\right\}$$

$$+P_k(j-1)\left\{\frac{k-j+1}{N}\cdot\frac{N-k}{N-1}+\binom{2}{1}\frac{(j-1)(k-j+1)}{N(N-1)}\cdot\frac{N-k}{N-2}\right.$$
$$+\binom{3}{2}\frac{(j-1)(j-2)(k-j+1)}{N(N-1)(N-2)}\cdot\frac{N-k}{N-3}+\cdots\cdot$$
$$\left.\cdots+\binom{j}{j-1}\frac{(j-1)(j-2)\cdots(k-j+1)}{N(N-1)\cdots(N-j+2)}\cdot\frac{N-k}{N-j+1}\right\}$$

$$+P_k(j-2)\left\{\frac{(k-j+2)(k-j+1)}{N(N-1)}\cdot\frac{N-k}{N-2}+\binom{3}{1}\frac{(j-2)(k-j+2)(k-j+1)}{N(N-1)(N-2)}\cdot\frac{N-k}{N-3}\right.$$
$$\left.+\cdots\cdots+\binom{j}{j-2}\frac{(j-2)(j-3)\cdots(k-j+1)}{N(N-1)\cdots(N-j+3)}\cdot\frac{N-k}{N-j+2}\right\}$$

$$+\cdots\cdots\cdot$$

$$+P_k(0)\cdot\frac{k(k-1)\cdots(k-j+1)}{N(N-1)\cdots(N-j+1)}\cdot\frac{N-k}{N-j}\qquad(2)$$

この式の $P_k(i)$ の係数は，k コの要素を登録したときに i コ（$0\le i\le j$）のエントリにマークがあったとき，次の1コの要素を追加してマークが j コになる確率である。その $l+1$ 項は，　　そのときに l コのすでにマークのあるエントリを途中でアクセスした場合の確率である。

(2)式は，計算によって次式のようになる。(Appendix参照)

$$P_{k+1}(j)=\frac{N-k}{(N)_{j+1}}\cdot\left\{P_k(0)\cdot k(k-1)\cdots\cdots\cdots(k-j+1)\right.$$
$$+P_k(1)\cdot N(k-1)\cdots\cdots\cdots(k-j+1)$$
$$+P_k(2)\cdot N(N-1)(k-2)\cdots\cdots(k-j+1)$$
$$+\cdots\cdots\cdots$$
$$+P_k(j-1)\cdot N(N-1)\cdots(N-j+2)(k-j+1)$$
$$\left.+P_k(j)\cdot N(N-1)\cdots\cdots\cdots(N-j+1)\right\}\qquad(3)$$

$j=1,2,\cdots,k$　　$(N)_{j+1}=N(N-1)\cdots\cdots(N-j)$

一方 $j=0$ に対しては

$$P_{k+1}(0)=\frac{N-k}{N}P_k(0) \tag{4}$$

(3)式より

$$(N-i+1)P_{k+1}(i+1)-(k-i)P_{k+1}(i)=(N-k)P_k(i+1) \tag{5}$$

(5)式を $i=0,1,\cdots,j$ について加え，(4)式を代入すると，

$$P_{k+1}(j)=\frac{N-k}{N-j}\left(\sum_{i=0}^{j}P_k(i)-\sum_{i=0}^{j-1}P_{k+1}(i)\right) \tag{6}$$

ところで，Nコのエントリのうち jコがマークされているときの棄却時間 $t(j)$ は，

$$t(j)=\frac{N-1}{N-j+1} \tag{7}$$

で与えられる。[1] これより，kコの要素が登録されているときの平均棄却時間 $\bar{t}_k$ は，

$$\bar{t}_k=E_k\left(\frac{N-1}{N-j+1}\right)=\sum_{j=0}^{k-1}P_k(j)\cdot\frac{N-1}{N-j+1} \tag{8}$$

で与えられる。$P_k(j)$ を，(6)式を用いて数値計算し，$\bar{t}_k$ を求めた。図6は，コンフリクト・マークをつけないときの棄却時間との比を示している。ここで、コンフリクト・マークのないときの棄却時間は，7式の $t(k)$ で与えられる。

次に，この方法を *direct chaining method* と比較してみる。今，kコの要素を登録したとき、iコの要素のチェインのできているエントリの数を n_i とすると，棄却時間 t_d は，

$$t_d=\frac{n_0}{N}+\sum_{i=1}^{k}\frac{n_i}{N}\cdot i=\frac{n_0}{N}+\frac{k}{N} \tag{9}$$

となる。n_0 は空のセルの数である。Nと kが充分に大きいと

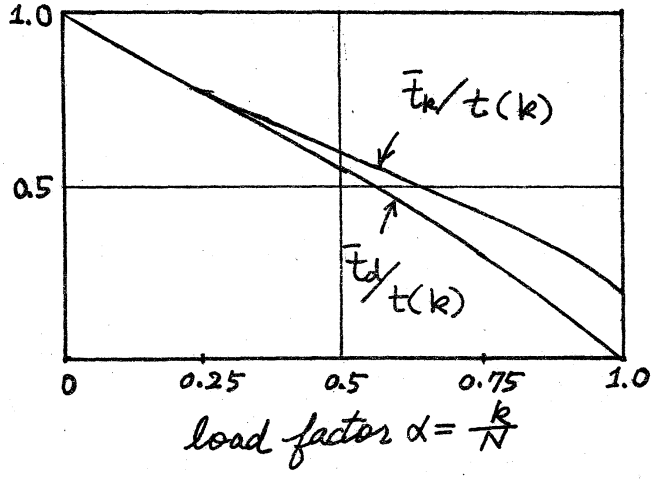
ま、n_0は $\lambda = Ne^{-\frac{k}{N}}$ のポアソン分布

$$p(n_0;\lambda) = e^{-\lambda}\frac{\lambda^{n_0}}{n_0!}$$

に従うことが知られている。[8] n_0の期待値は，λで与えられるので，$k/N = \alpha$ とすると，(9)の期待値は，

$$\bar{t}_d = e^{-\alpha} + \alpha \qquad (10)$$

となる。図6に，$\bar{t}_d$ と $t(k)$ の比も合わせてのせてある。



$\bar{t}_k$：コンフリクト・マークをつけたときの棄却時間

$t(k)$：コンフリクト・マークのないときの棄却時間

$\bar{t}_d$：direct chaining method での棄却時間

($N = 1024$)

図6．棄却時間の比の load factor に対するグラフ

図6．より、load factor が 60～70% になると，コンフリクト・マークをつけたときの棄却時間は，マークのないときの約半分になることがわかる。しかしこの場合にも，direct chaining method がほぼ効率の上限を与えていることがいえる。コンフリクト・マークのビット数を増やすと，この上限に近づくことが予想されるが，どのように近づくかは，明らかではない。

参考文献

1. Morris, R. Scatter storage techniques. CACM 11, 1 (Jan. 1968), PP. 38~44
2. 古川, 山本 Hash codeによる命令表の作製法 昭和44年電気四学会連合大会予稿集
3. Maurer, W.D. An improved hash code for scatter storage. CACM 11, 1 (Jan. 1968), PP. 35~38
4. Radke, C.E. The use of quadratic residue research. CACM 13, 2 (Feb. 1970), PP. 103~105
5. Day, A.C. Full table quadratic searching for scatter storage. CACM 13, 8 (Aug. 1970), PP. 481~482
6. Bell, J.R. The quadratic quotient method: A hash code eliminating secondary clustering. CACM 13, 2 (Feb. 1970), PP. 107~109
7. 古川, 山崎 Hash addressingによる集合演算. 昭和46年電子通信学会全国大会発表予定
8. Feller, W An introduction to probability theory and its applications. Vol I. John Wiley, 1957, pp. 91~95

Appendix. (2)式から(3)式への誘導

$$f(a,b,c) = 1 + \frac{b}{a-1}\binom{c+1}{1} + \frac{b(b-1)}{(a-1)(a-2)}\binom{c+2}{2} + \cdots\cdots + \frac{b(b-1)\cdots\cdots 1}{(a-1)(a-2)\cdots\cdots(a-b)}\binom{c+b}{b} \quad (1a)$$

とおいて、このfを用いて(2)式を書きなおすと、

$$P_{k+1}(j) = P_k(j)\cdot f(N,j,0)\cdot\frac{N-k}{N} + P_k(j-1)\cdot f(N-1,j-1,1)\cdot\frac{N-k}{N}\cdot\frac{k-j+1}{N-1} + \cdots\cdots + P_k(0)\cdot f(N-j,0,j)\cdot\frac{N-k}{N}\cdot\frac{k(k-1)\cdots(k-j+1)}{(N-1)\cdots\cdots(N-j)} \quad (2a)$$

となる。関数fの一般項$f(N-j+i, i, j-i)$を求めると

$$f(N-j+i,i,j-i) = \frac{N(N-1)\cdots\cdots(N-i+1)}{(N-j+i-1)(N-j+i-2)\cdots\cdots(N-j)}, \quad (i\geq 1) \quad (3a)$$

となることを、iについての帰納法を用いて証明することができる。(ただしi=0のときはfは1となる)。先ずfを書きなおすと、

$$f(N-j+i,i,j-i) = 1 + \frac{j-i+1}{N-j+i-1}\binom{i}{1} + \frac{(j-i+1)(j-i+2)}{(N-j+i-1)(N-j+i-2)}\binom{i}{2} + \cdots\cdots + \frac{(j-i+1)(j-i+2)\cdots\cdots j}{(N-j+i-1)(N-j+i-2)\cdots\cdots(N-j)}\binom{i}{i}$$

となる。i=1のとき

$$f(N-j+1,1,j-1) = 1 + \frac{j}{N-j}\binom{1}{1} = \frac{N}{N-j}$$

故に(3a)は成り立つ。

今 $i=h$ のときに (3a) が成り立ったとする。$i=h+1$ に対して

$$f(N-j+h+1,\ h+1,\ j-(h+1))$$
$$= 1+\frac{j-h}{N-j+h}\binom{h+1}{1}+\frac{(j-h)(j-h+1)}{(N-j+h)(N-j+h-1)}\binom{h+1}{2}+\cdots\cdots$$
$$+\frac{(j-h)(j-h+1)\cdots\cdots j}{(N-j+h)(N-j+h-1)\cdots(N-j)}\binom{h+1}{h+1}$$

ここで $\binom{h+1}{l}=\binom{h}{l-1}+\binom{h}{l}$ を代入すると.

$$\text{右辺}=1+\frac{j-h}{N-j+h}\left\{\binom{h}{0}+\binom{h}{1}\right\}+\frac{(j-h)(j-h+1)}{(N-j+h)(N-j+h-1)}\left\{\binom{h}{1}+\binom{h}{2}\right\}$$
$$+\cdots\cdots+\frac{(j-h)(j-h+1)\cdots\cdots j}{(N-j+h)(N-j+h-1)\cdots(N-j)}\left\{\binom{h}{h}+0\right\}$$
$$=\frac{j-h}{N-j+h}\left\{1+\frac{j-h+1}{N-j+h-1}\binom{h}{1}+\cdots\cdots\cdots\right\}$$
$$+1+\frac{j-h}{N-j+h}\binom{h}{1}+\frac{(j-h)(j-h+1)}{(N-j+h)(N-j+h-1)}\binom{h}{2}+\cdots\cdots$$
$$=\frac{j-h}{N-j+h}\cdot f(N-j+h,\ h,\ j-h)+f(N-(j-1)+h,\ h,\ (j-1)-h)$$
$$=\frac{j-h}{N-j+h}\cdot\frac{N(N-1)\cdots(N-h+1)}{(N-j+h-1)\cdots\cdots(N-j)}+\frac{N(N-1)\cdots\cdots(N-h+1)}{(N-j+h)\cdots\cdots\cdots(N-j+1)}$$
$$=\frac{N(N-1)\cdots\cdots(N-h+1)(N-h)}{(N-j+h)(N-j+h-1)\cdots(N-j+1)(N-j)}$$

となり，$i=h+1$ に対しても成り立つことがわかった。

この f を (2a) に代入する。(項の順序を逆にする)

$$P_{k+1}(j)=P_k(0)\cdot\frac{N-k}{N}\cdot\frac{k(k-1)\cdots\cdots(k-j+1)}{(N-1)\cdots\cdots\cdots(N-j)}$$
$$+P_k(1)\cdot\frac{N}{N-j}\cdot\frac{N-k}{N}\cdot\frac{(k-1)\cdots\cdots(k-j+1)}{(N-1)-\cdots\cdots-(N-j+1)}$$
$$+P_k(2)\cdot\frac{N(N-1)}{(N-j)(N-j+1)}\cdot\frac{N-k}{N}\cdot\frac{(k-2)\cdots\cdots(k-j+1)}{(N-1)\cdots\cdots(N-j+2)}$$
$$+\cdots\cdots\cdots$$
$$+P_k(j)\cdot\frac{N(N-1)\cdots\cdots(N-j+1)}{(N-j)(N-j+1)\cdots\cdots(N-1)}\cdot\frac{N-k}{N}$$

この式を変形すれば，(3) 式が得られる。