数理解析研究所講究録
461巻 1982年 113-126

連想値を用いる完全ハッシング関数の存在条件について

玉越靖司　　有川節夫

(九州大学総合理工学研究科)

1. はじめに

ハッシュ法は，キー比較コストが定数に近い探索法である。 特に，計算機システムのコマンドのような特定のキーの集合に対して，その集合の1つの要素について探索する場合，非常に効果的である。 このような静的なキーの集合に対しては，ハッシング関数を工夫することによって，キー比較コストを減少させることが可能である。 そこで，望ましいハッシング関数として，キー比較回数が1つのキーに対してただ1回だけである完全ハッシング関数（perfect hashing function）[1]が考えられる。

完全ハッシング関数の1つの方法として，連想値（associated value）を用いる方法[2]がある。 この方法は，関数値の計算に割り算を含まずに整数の足し算だけでキーのハッシュ値を計算するので，計算機での実行に際して演算時間が短くなるという利点を持つ。 しかしながら，この方法ではJaeschkeら[3]の指摘を待つまでもなく，完全ハッシング関数を持たないようなキーの集合が存在する。 そこで本稿では，まず，与えられたキーの集合に対して，連想値を用いる完全ハッシング関数が存在するための必要十分条件について考察した。

完全ハッシング関数が存在すれば，ハッシュテーブルの大きさMはキーの集合の要素の数n以上であるが，このとき，ハッシュテーブルのうちでM－n個のバケットは使われていない。 格納率（loading factor；n／M）が1のとき，その関数は最小完全ハッシング関数（minimal perfect hashing function）[1]と呼ばれ，計算機の記憶域コストの面からは理想的である。 こうした連想値を用いる最小完全ハッシング関数について，存在性の問題が可解であることを示した。

§2では，我々が定義した型[4]の連想値を用いるハッシング関数について，その完全な関数の存在条件，格納率に関する問題，そして最小完全な関数の存在性に関する決定問題について述べる。 §3では，Cichelli[2]が定義した型の関数について，§2と同様の議論が可能であることを述べ，完全な関数の存在条件を示す。 さらに§4では，関数のコストを計算機による関数値計算の演算時間の面から，連想値を

用いないハッシング関数と比較して検討する。

2. 連想値を用いる完全ハッシング関数

2. 1. 諸定義

以下の議論で必要な文字の有限集合を，$\Sigma = \{\gamma_1, \gamma_2, \cdots, \gamma_m\}$ とし，$I = \{w_1, w_2, \cdots, w_n\} \subset \Sigma^{\ell}$ をキーの集合と呼ぶ。 Iの要素をキーと呼ぶ。

注意1. Iに属するすべてのキーの長さが等しいと仮定しても一般性を失わない。 これは，長さの異なるキーの集合に対しても，その最長のキーの長さℓより短いキーには＿（空白）という文字を埋めて，長さをℓにすることができるからである□

定義1. 文字x，$y \in \Sigma$に対して

$$d(x, y) = \begin{cases} 0 & (x = y) \\ 1 & (x \neq y) \end{cases} \square$$

定義2. キーu，$v \in I$に対して

$$u = v \iff \sum_{k=1}^{\ell} d(u^{(k)}, v^{(k)}) = 0$$

ここに，$w^{(i)}$は，キーwのi番目の文字とする□

定義3. キー$w \in I$のうち，p個の文字；$w^{(i_1)}, w^{(i_2)}, \cdots, w^{(i_p)}$ $(1 \leq p \leq \ell,\ s < t \Rightarrow 1 \leq i_s < i_t \leq \ell)$ の中で，$\gamma \in \Sigma$と等しい文字の個数を次のように書く。

$$|w|_{\gamma}^{(i_1, \cdots, i_p)} = p - \sum_{k=1}^{p} d(w^{(i_k)}, \gamma) \square$$

例1. $w = ABCBC$ とする（$\ell = 5$）。

$$|w|_A^{(1, 3, 5)} = 1,\ |w|_B^{(1, 3, 5)} = 0,\ |w|_C^{(1, 3, 5)} = 2 \square$$

定義4. $h : I \to N$ の1対1写像：Iに対する完全ハッシング関数と呼ぶ。 ここに，Nは0以上の整数の全体である□

定義5. Iに対する完全ハッシング関数hが

$$\max h(I) - \min h(I) = n - 1$$

をみたすとき，hをIに対する最小完全ハッシング関数と呼ぶ□

定義6. $a : \Sigma \rightarrow Z$: aを連想といい，$a(\tau)$ を文字τの連想値と呼ぶ。 ここに，Zは整数の全体である□

定義7. aを連想とするとき，

$$h(w) = a(w^{(i_1)}) + a(w^{(i_2)}) + \cdots + a(w^{(i_p)}) \qquad (w \in I)$$

で定義される関数 $h : I \rightarrow N$ を，連想値を用いるハッシング関数といい，今後，簡単にA－ハッシング関数と呼ぶことにする□

例2. キーの集合として，英語の月の名前を考える。

Function : $h(w) = a(w^{(2)}) + a(w^{(3)})$

Associated Values :

a(A)= 4	a(B)= 5	a(C)= 2	a(D)= 0	a(E)= 0	a(F)= 0
a(G)= 3	a(H)= 0	a(I)= 0	a(J)= 0	a(K)= 0	a(L)= 6
a(M)= 0	a(N)= 0	a(O)= 5	a(P)= 1	a(Q)= 0	a(R)= 6
a(S)= 0	a(T)= 6	a(U)= 0	a(V)= 6	a(W)= 0	a(X)= 0
a(Y)= 5	a(Z)= 0	a(_)= 0			

I (Set of keys)	Hash Values
w_1 = JANUARY__	$h(w_1)$ = 4
w_2 = FEBRUARY_	$h(w_2)$ = 5
w_3 = MARCH____	$h(w_3)$ = 10
w_4 = APRIL____	$h(w_4)$ = 7
w_5 = MAY_____	$h(w_5)$ = 9
w_6 = JUNE_____	$h(w_6)$ = 0
w_7 = JULY_____	$h(w_7)$ = 6
w_8 = AUGUST___	$h(w_8)$ = 3
w_9 = SEPTEMBER	$h(w_9)$ = 1
w_{10}= OCTOBER__	$h(w_{10})$= 8
w_{11}= NOVEMBER_	$h(w_{11})$= 11
w_{12}= DECEMBER_	$h(w_{12})$= 2 □

2．2．完全な関数の存在条件

A－ハッシング関数では，完全ハッシング関数を存在させ得ないようなキーの集合Ｉがある（例3）。

例3．　キーの集合として，$I=\{AA, AB, BA\}$　（$\ell=2$，$w_1=AA$，$w_2=AB$，$w_3=BA$）を考える□

（証明）　i）　$p=1$，$i_1=1$　のとき，

$h(w_1)=h(w_2)=a(A)$　となる。

ii）　$p=1$，$i_1=2$　のとき，

$h(w_1)=h(w_3)=a(A)$　となる。

iii）　$p=2$，$i_1=1$，$i_2=2$　のとき，

$h(w_2)=h(w_3)=a(A)+a(B)$　となる。

従って，完全なA－ハッシング関数は存在しない□

定理1．　キーの集合Ｉに対して，完全なA－ハッシング関数が存在する。

$\Longleftrightarrow \exists\ p, i_1, \cdots, i_p$ ：　i）　$1\leqq p\leqq \ell$

ii）　$s<t \Longrightarrow 1\leqq i_s<i_t\leqq \ell$

iii）　任意のu，$v\in I$，$u\neq v$　に対して

$$\sum_{k=1}^{m}\left|\,|u|_{\tau_k}^{(i_1,\cdots,i_p)}-|v|_{\tau_k}^{(i_1,\cdots,i_p)}\right|\neq 0$$ □

（証明）

$\Longleftarrow$）　$a(\tau_j)=(p+1)^{j-1}$　$(1\leqq j\leqq m)$　とすると，

$$h(w)=|w|_{\tau_1}^{(i_1,\cdots,i_p)}\times 1+|w|_{\tau_2}^{(i_1,\cdots,i_p)}\times(p+1)+\cdots+|w|_{\tau_m}^{(i_1,\cdots,i_p)}\times(p+1)^{m-1}.$$

となり，$0\leqq|w|_{\tau_j}^{(i_1,\cdots,i_p)}\leqq p$　より，$h(w)$ は，m桁の $(p+1)$ 進数として表わされる。　しかも，

$$\sum_{k=1}^{m}\left|\,|u|_{\tau_k}^{(i_1,\cdots,i_p)}-|v|_{\tau_k}^{(i_1,\cdots,i_p)}\right|\neq 0$$

より，任意の，Ｉの異なる2要素u，vに対して，

$h(u)\neq h(v)$

となる。

$\Longrightarrow$)　完全なA－ハッシング関数hが存在する。

$\Longleftrightarrow$ $\exists a(r_1), \cdots, a(r_m) : \forall u, v \in I, u \neq v$ に対して,

i)　$h(w) = a(w^{(i_1)}) + a(w^{(i_2)}) + \cdots + a(w^{(i_p)})$　　$(w \in I)$

ii)　$h(u) \neq h(v)$

より明らか□

注意2.　ある　p, $i_1, \cdots, i_p$ に対して, 完全なA－ハッシング関数の存在を決定するには, O（n log n）　の時間がかかる。　これは, 文字の取り出し（n）, ソート（n log n）, そして比較（n）に要する時間である□

2．3．格納率について

定理1の証明で示したような連想値の割り当て方では, 完全ハッシング関数の格納率がかなり低下すると予想される。　この方法と, Cichelli [2] が示したようなバックトラッキングによる連想値の割り当ての方法で, 格納率を比較してみる。

いま, m個の異なる文字から重複を許してp個を取り出すすべての組み合わせを表わす文字列の集合Iを考え（例4）, これを　m－p飽和集合と呼ぶ。

例4.　　3－3飽和集合Iは, Σ＝｛A, B, C｝　とするとき,

I＝｛AAA, AAB, ABB, BBB, AAC, ABC, BBC, ACC, BCC, CCC｝である□

注意3.　m－p飽和集合Iの要素の数nは,

$n = {}_mH_p = (m+p-1)!/\{p!\,(m-1)!\}$ である□

m－p飽和集合Iに対して, バックトラッキングによって決定される連想値の割り当ては,

$$a(r_j) = (p^{j-1}-1)/(p-1) + C$$

$(1 < p,\ 1 \leq j \leq m,\ C$は任意の整定数$)$

である。　このことは, 次のようにして説明できる。　すなわち,

$$a(r_1) < a(r_2) < \cdots < a(r_m)$$

と仮定できる。　実際, Iの, 文字に対する対称性から, $i \neq j$　ならば　$a(r_i) \neq a(r_j)$　でなければならない。

いま，a（γ_1）＝C　とする。　次に，γ_1 と γ_2 だけからなるキーに対して，完全性を保ち，かつハッシュテーブルを可能な限り小さくするために　a（γ_2）＝C＋1　を割り当てる。　ここで，γ_1 と γ_2 だけからなるキーの中で，ハッシュ値が最も大きいものは（C＋1）×p　となる。　さらに，γ_1，γ_2，γ_3 だけからなるキーに対して，完全性を保ち，かつハッシュテーブルを可能な限り小さくするために　a（γ_3）＝C＋1＋p　を割り当てる。　こうして　a（γ_i）＝C＋x　が割り当てられれば　a（γ_{i+1}）＝C＋x＋p^{i-1}が割り当てられる（例5）。

例5.　　例4に対するバックトラッキングによる連想値の割り当ては，次のようになる。

Function : $h(w) = a(w^{(1)}) + a(w^{(2)}) + a(w^{(3)})$

Associated Values : a(A)= 0 , a(B)= 1 , a(C)= 4

I (Set of keys)	Hash Values
w_1 = AAA	$h(w_1)$ = 0
w_2 = AAB	$h(w_2)$ = 1
w_3 = ABB	$h(w_3)$ = 2
w_4 = BBB	$h(w_4)$ = 3
w_5 = AAC	$h(w_5)$ = 4
w_6 = ABC	$h(w_6)$ = 5
w_7 = BBC	$h(w_7)$ = 6
w_8 = ACC	$h(w_8)$ = 8
w_9 = BCC	$h(w_9)$ = 9
w_{10}= CCC	$h(w_{10})$ = 12 □

さて，定理1の証明で示した連想値の割り当て　a（γ_j）＝$(p+1)^{j-1}$　を用いたm－p飽和集合Iに対するA－完全ハッシング関数の格納率　lf　は，

$$lf = {}_m H_p / (p+1)^m$$

となる。　一方，バックトラッキングによる割り当て　a（γ_j）＝$(p^{j-1}-1)/(p-1)$　を用いた場合の格納率は，

$$lf = {}_m H_p / \{(p^m - 1)/(p-1)\}$$

となり，飽和集合のような性質の悪い集合に対するA－完全ハッシング関数の場合では，連想値の割り当て方による格納率の差には，オーダーから見ても大差がない。

2．4．最小完全な関数の決定可能性

完全なA－ハッシング関数は存在するが，最小完全なA－ハッシング関数は存在し得ないようなキーの集合Iがある（例6）。

例6．　キーの集合として，$I = \{w_1, \cdots, w_n\}$ を考える。ここに，$2 \leq n$，$w_k = \tau_k \tau_{k+1}$ $(1 \leq k \leq n-1)$，$w_n = \tau_1 \tau_n$ とする。

Function : $h(w) = a(w^{(1)}) + a(w^{(2)})$

Ex.1 (n=2)

w_1 = AB

w_2 = BA

Ex.2 (n=3)

w_1 = AB　$h(w_1) = 1$　$a(A) = 0$, $a(B) = 1$, $a(C) = 2$

w_2 = BC　$h(w_2) = 3$

w_3 = AC　$h(w_3) = 2$

Ex.3 (n=4)

w_1 = AB　$h(w_1) = 0$　$a(A) = 0$, $a(B) = 0$, $a(C) = 1$,

w_2 = BC　$h(w_2) = 1$　$a(D) = 2$

w_3 = CD　$h(w_3) = 3$

w_4 = AD　$h(w_4) = 2$

Ex.4 (n=5)

w_1 = AB　$h(w_1) = 0$　$a(A) = 0$, $a(B) = 0$, $a(C) = 1$,

w_2 = BC　$h(w_2) = 1$　$a(D) = 2$, $a(E) = 2$

w_3 = CD　$h(w_3) = 3$

w_4 = DE　$h(w_4) = 4$

w_5 = AE　$h(w_5) = 2$

Ex.5 (n=6)

w_1 = AB

w_2 = BC

w_3 = CD

w_4 = DE

w_5 = EF

w_6 = AF

Ex.6 (n=7)

w_1 = AB　$h(w_1) = 1$　$a(A) = 0$, $a(B) = 1$, $a(C) = 1$,

w_2 = BC　$h(w_2) = 2$　$a(D) = 2$, $a(E) = 3$, $a(F) = 3$,

w_3 = CD　$h(w_3) = 3$　$a(G) = 4$

w_4 = DE　$h(w_4) = 5$

w_5 = EF　$h(w_5) = 6$

w_6 = FG　$h(w_6) = 7$

w_7 = AG　$h(w_7) = 4$ □

注意4. 例6のようなキーの集合Iで，$n=4n'+2$（n'は0以上の整数）のとき，最小完全なA－ハッシング関数は存在しない□

（証明） $n=4n'+2$のとき，最小完全なA－ハッシング関数が存在するなら，

$$h(w_1)+h(w_2)+\cdots+h(w_{4n'+2})=b+(b+1)+\cdots+(b+4n'+1)$$

$$=2(4n'^2+3n')+1 \quad (\text{奇数})$$

となる。 一方，完全なA－ハッシング関数は，$p=2$，$i_1=1$，$i_2=2$ の場合だけにしか存在しないことは明らかなので，

$$h(w_1)+h(w_2)+\cdots+h(w_{4n'+2})=2\{a(A)+a(B)+\cdots+a(r_n)\} \quad (\text{偶数})$$

となり，矛盾が生じる。

従って，背理法により，$n=4n'+2$のときには最小完全なA－ハッシング関数が存在しないことが証明された□

次に，最小完全なA－ハッシング関数の存在性に関する決定問題について考えよう。 いま，

$$\mu_{kh}=|w_k|_{r_h}^{(i_1,\cdots,i_p)}$$

とすると，ある p，i_1，$\cdots$，i_p で，最小完全なA－ハッシング関数の存在は，

(1) $A\times x=b$

なるxの存在と同値である。 ここに，

$$A=\begin{pmatrix}\mu_{11}&\mu_{12}&\cdots&\mu_{1m}\\ \mu_{21}&\mu_{22}&\cdots&\mu_{2m}\\ \vdots&&&\vdots\\ \mu_{n1}&\mu_{n2}&\cdots&\mu_{nm}\end{pmatrix} \quad x=\begin{pmatrix}x_1\\ x_2\\ \vdots\\ x_m\end{pmatrix} \quad b=\begin{pmatrix}b\\ b+1\\ \vdots\\ b+n-1\end{pmatrix}$$

とし，x_1，$\cdots$，x_m は整数，bは0以上の整数で，**b**はその要素の置換を許すものとする。 また，Aに関して，

$$\mu_{1\,1}+\mu_{1\,2}+\cdots+\mu_{1\,m}=\mu_{2\,1}+\mu_{2\,2}+\cdots+\mu_{2\,m}$$

$$=\cdots$$

$$=\mu_{n\,1}+\mu_{n\,2}+\cdots+\mu_{n\,m}=p$$

なる関係がある。 そこで，もしこの方程式の解x_1，$\cdots$，x_m，およびbが1組見つかれば，$x_1'=x_1+1$，

$x_2' = x_2 + 1$, …, $x'_m = x_m + 1$, $b' = b + p$ として，やはりこの方程式の別の解x_1'，…，x'_m，およびb'の組を作ることが可能で，従って無限個の解の組が存在することになる。 このことから，もし，この方程式の解が存在するならば，$0 \leq b < p$ となる解が必ず存在する。 つまり，bは有限個について調べればよい。

さて，（1）が解（有理数解）を持つための必要十分条件は，

（2） $\mathrm{rank}A = \mathrm{rank}(Ab)$

である。 従って，この条件（2）は（1）が整数解を持つための必要条件である。 この必要条件（2）を満たす

$$\mathrm{rank}A = \mathrm{rank}(Ab) = r$$

について，

i） $r \leq m$ のとき，解（有理数解）は一意に決まり，それが整数解であるかどうかは決定可能である。

ii） $r > m$ のとき，（1）は連立一次不定方程式となる。 1つの一次不定方程式 $a_1 x_1 + a_2 x_2 + \cdots + a_m x_m = b$ が整数解を持つための必要十分条件は，

（3） $(a_1, a_2, \cdots, a_m) \mid b$

である。 ここに，$(a_1, \cdots, a_m)$は，$a_1, \cdots, a_m$の最小公倍数である。 （i）は，n個の一次不定方程式より成っていて，これを次のように順次解いていく。

まず，第1の一次不定方程式が条件（3）を満たしていれば，任意の整数のパラメータを含む一般解を得る。この一般解を第2の一次不定方程式に代入して，同様に一般解を求め，これを次々に第nの一次不定方程式にまで繰り返せば，（1）の，整数の一般解が得られる［5］。 もし，この繰り返しの途中で，条件（3）を満たさない一次不定方程式があれば（1）に整数解は存在しない。

以上のことから，次の結果が得られる。

定理2． キーの集合Iに対して，最小完全なA－ハッシング関数が存在するか否かを問う問題は，決定可能である□

（証明） 長さℓのキーから何番目の文字を取り出すかという組み合わせは，$2^{\ell} - 1$通りで，有限である。その1つ1つの場合に対して，bにおけるbはp通り，また置換はn！通りで，いずれも有限である。

こうして各要素が具体的に決まった方程式（1）については，整数解の存在性が可解であったので，上記の

すべての場合をつくせば，キーの集合Ｉに対して最小完全なＡ－ハッシング関数が存在するか否かという問題は決定可能である□

3．Ｃｉｃｈｅｌｌｉの完全ハッシング関数

本節では，Ｃｉｃｈｅｌｌｉが定義したハッシング関数について，§２と同様の議論が可能であることを述べ，与えられたキーの集合Ｉに対してＣｉｃｈｅｌｌｉの完全ハッシング関数が存在するための必要十分条件を示す。

3．1．定義

本節では，キーの集合として，$I = \{w_1, w_2, \cdots, w_n\} \subset \Sigma^+$ を考える。 すなわち，同じキーの集合の中で，必ずしもキーの長さは等しくないものとする。

定義８．　キー$w \in I$の長さに関して，次のように定義する。

$\ell(w)$ ：wの長さ（$1 \leq \ell(w)$）□

例７．　$w = ABCBC$ とする。

$\ell(w) = 5$□

定義９．　aを連想とするとき，

$$h(w) = \ell(w) + a(w^{(1)}) + a(w^{(\ell(w))}) \quad (w \in I)$$

で定義される関数$h : I \to N$ をＣｉｃｈｅｌｌｉのハッシング関数という□

例８．　キーの集合として，出現頻度の高い英単語を考える［２］。

Function : $h(w) = \ell(w) + a(w^{(1)}) + a(w^{(\ell(w))})$

Associated Values :

a(A)= 3	a(B)= 15	a(C)= 0	a(D)= 7	a(E)= 0	a(F)= 15
a(G)= 0	a(H)= 10	a(I)= 0	a(J)= 0	a(K)= 0	a(L)= 0
a(M)= 12	a(N)= 13	a(O)= 7	a(P)= 0	a(Q)= 0	a(R)= 12
a(S)= 6	a(T)= 0	a(U)= 15	a(V)= 0	a(W)= 14	a(X)= 0
a(Y)= 9	a(Z)= 0				

I (Set of keys)	Hash values
w_1 = A	$h(w_1)$ = 7
w_2 = AND	$h(w_2)$ = 13
w_3 = ARE	$h(w_3)$ = 6
w_4 = AS	$h(w_4)$ = 11
w_5 = AT	$h(w_5)$ = 5
W_6 = BE	$h(w_6)$ = 17
w_7 = BUT	$h(w_7)$ = 18
w_8 = BY	$h(w_8)$ = 26
w_9 = FOR	$h(w_9)$ = 30
w_{10}= FROM	$h(w_{10})$ = 31
w_{11}= HAD	$h(w_{11})$ = 20
w_{12}= HAVE	$h(w_{12})$ = 14
w_{13}= HE	$h(w_{13})$ = 12
w_{14}= HER	$h(w_{14})$ = 25
w_{15}= HIS	$h(w_{15})$ = 19
w_{16}= I	$h(w_{16})$ = 1
w_{17}= IN	$h(w_{17})$ = 15
w_{18}= IS	$h(w_{18})$ = 8
w_{19}= IT	$h(w_{19})$ = 2
w_{20}= NOT	$h(w_{20})$ = 16
w_{21}= OF	$h(w_{21})$ = 24
w_{22}= ON	$h(w_{22})$ = 22
w_{23}= OR	$h(w_{23})$ = 21
w_{24}= THAT	$h(w_{24})$ = 4
w_{25}= THE	$h(w_{25})$ = 3
w_{26}= THIS	$h(w_{26})$ = 10
w_{27}= TO	$h(w_{27})$ = 9
w_{28}= WAS	$h(w_{28})$ = 23
w_{29}= WHICH	$h(w_{29})$ = 29
w_{30}= WITH	$h(w_{30})$ = 28
w_{31}= YOU	$h(w_{31})$ = 27 □

3．2．ℓ（w）の扱い方に関する注意

キーの集合Ｉの要素のうち，最長のキーの長さを

$$L = max\ \ell(I)$$

と表わす。 いま，１からＬまでＬ通りのキーの長さに関する情報をＬ個の異なる文字の情報に置き換えて，連想値を $a(\ell(w)) = \ell(w)$ と割り当てていると考えることが可能である。 こうすると§２とほぼ同様の議論が可能となる。

さらに，$a(\ell(w))$ を自由に割り当てることを許せば，§２における $p=3$，$m=m+L$ の場合とまったく同様の議論が可能となるが，Ｃｉｃｈｅｌｌｉの方法ではこれは許されていない。

3．3．完全な関数の存在条件

$\ell(w)$ を上のように扱うと，§2とほぼ同様の議論で完全なCichelliのハッシング関数が存在するための必要十分条件を与えることが可能である。

定理3． キーの集合Iに対して，Cichelliの完全ハッシング関数が存在する。

$\Longleftrightarrow$ 任意のu，$v \in I$，$u \neq v$ に対して，

i) $\ell(u) \neq \ell(v)$ または，

ii) $\{u^{(1)}, u^{(\ell(u))}\} \neq \{v^{(1)}, v^{(\ell(v))}\}$ □

(証明)

$\Longleftarrow$) $a(\tau_j) = (L+2)^j$ $(1 \leq j \leq m)$ とすると，

$h(w) = \ell(w) + |w|_{\tau_1}^{(1, \ell(w))} \times (L+2) + \cdots + |w|_{\tau_m}^{(1, \ell(w))} \times (L+2)^m$

となり，$1 \leq \ell(w) \leq L$，$0 \leq |w|_{\tau_j}^{(1, \ell(w))} \leq 2$ より，$h(w)$ は $(m+1)$ 桁の $(L+2)$ 進数として表わされる。 しかも，

i) $\ell(u) \neq \ell(v)$ または，

ii) $\{u^{(1)}, u^{(\ell(u))}\} \neq \{v^{(1)}, v^{(\ell(v))}\}$

より，任意の，Iの異なる2要素u，vに対して，

$h(u) \neq h(v)$ となる。

$\Longrightarrow$) Cichelliの完全ハッシング関数が存在する。

$\Longleftrightarrow$ $\exists a(\tau_1), \cdots, a(\tau_m) : \forall u, v \in I, u \neq v$ に対して

i) $h(w) = \ell(w) + a(w^{(1)}) + a(w^{(\ell(w))})$

ii) $h(u) \neq h(v)$

より明らか□

注意5． キーの集合Iに対してCichelliの完全ハッシング関数の存在を決定するには，$O(n \log n)$ の時間で十分である□

4．関数値の計算に要するコスト

完全なA－ハッシング関数を用いて，1個のキーを探索するための時間的コストTAは，次のように定義するのが妥当である。

$$TA = p\,(t_1 + ta) + t_2 + tc$$

また，これと比較するため，Jaeschke［6］が定義した最小完全なハッシング関数

$$h(w) = \lfloor C/(Dw+E) \rfloor \bmod n$$

の場合に要する時間的コストTJを，

$$TJ = tm + ta + td + t_3 + t_2 + tc$$

と定義する。 ここに，

p：連想値を取り出す文字の個数

ta：整数の足し算1回に要する時間

tm：整数の掛け算1回に要する時間

td：整数の割り算1回に要する時間

tc：キー比較1回に要する時間

t_1：1つの文字の連想値を知るのに要する時間

t_2：ハッシュテーブルからキーを1つ呼ぶのに要する時間

t_3：整数の剰余を1回求めるのに要する時間

C，D，E：整定数

とする。 九州大学大型計算機センターのFACOM M－200におけるFORTRAN－GEによるプログラム実験では，ta＝1としたとき，tm＝1，td＝20，t_1＝1，t_3＝22であった。これらをTA，TJの式に代入すると，

$p < 22$ のとき，$TA < TJ$

となる。 通常，pは1～5程度であるから，A－完全ハッシング関数の方が十分に速いといえる。

また，同様の比較をDividing法［7］によるハッシング関数に対しても行ったが，$p < 11.5$ のとき，A－完全ハッシング関数の方が速いという結果を得た。

5. おわりに

静的なキーの集合に対するA－完全ハッシング関数の特徴を示し，Ｃｉｃｈｅｌｌｉが定義した完全ハッシング関数が存在するための必要十分条件を与えた。 また，関数値を計算するための時間的コストの点で，他のハッシング関数と比較して，A－完全ハッシング関数が優れていることを示した。

今後は，最小完全な関数の存在するキーの集合に対して，その連想値を割り当てる方法，および，A－完全ハッシング関数が存在するか否かを決定する問題の複雑性（ＮＰ完全であることが予想される）等について検討していきたい。

参考文献

[1] Sprugnoli,R.: Perfect Hashing Functions : A Single Probe Retrieving Method for Static Sets,CACM,20,11,(Nov. 1977).

[2] Cichelli,R.J.: Minimal Perfect Hash Functions Made Simple, CACM,23,1,(Jan. 1980).

[3] Jaeschke,G.,and Osterburg,G.: On Cichelli's Minimal Perfect Hash Functions Method,CACM,23,12,(Dec. 1980).

[4] 玉越，有川：連想値を用いた完全ハッシング関数，昭和５６年度電気四学会九州支部連合大会講演論文集.

[5] 高木貞治：初等整数論講義 第2版，共立出版.

[6] Jaeschke,G.: Reciprocal Hashing : A Method for Generating Minimal Perfect Hashing Functions,CACM,24,12,(Dec. 1981).

[7] Martin,J.: Computer Data-Base Organization,2nd edition, Prentice-Hall,Inc.,1977.