An Efficient Accessing Technique of Chinese Characters Using Boshiamy Chinese Input System

Chin-Chen Chang*, Tung-Shou Chen, and Yi Lin***

*Department of Computer Science and Information Engineering
National Chung Cheng University, Chiayi, Taiwan 62107
E-mail:{ccc,yl87}@cs.ccu.edu.tw

**Department of Information Management
Taichung Institute of Technology, Taichung, Taiwan 404
E-mail: ts.chen@taiwan.com

Abstract

In this paper, a new efficient technique for Chinese character retrieval is proposed. This technique designs a minimal perfect hashing function based on the Chinese remainder theorem for a simply and widely used Chinese input system, called Boshiamy Chinese input system. The hashing function accepts the input key sequence of Boshiamy and calculates the address of the corresponding Chinese character. The Chinese character therefore can be retrieved through the computed address directly. Consequently, this technique is suitable for Boshiamy Chinese input system and could retrieve Chinese characters very efficiently.

Keywords: Chinese input; perfect hashing; Chinese remainder theorem; Boshiamy.

1. Introduction

To make Chinese character input go faster and more smoothly, there are several Chinese input systems, such as Dayi, phonetic transcription, and Cangjia, have been devised. Generally, a Chinese character is coded as a sequence of symbols, called *key sequence*, in Chinese input systems. After the user types a key sequence from the keyboard, the corresponding Chinese input system will process this key sequence and generate the corresponding Chinese character to the user. A brief diagram of Chinese input system is shown in Figure 1.

A Chinese input system always maintains a table to keep the mapping between key sequences and Chinese characters. How to access this table efficiently is an important issue for Chinese input systems. Therefore, there are several literatures [1-5] devoted to this problem. Hashing is a widely used skill among these techniques.

In this paper, we intend to design an effective and efficient accessing technique for Boshiamy Chinese input system [7]. Boshiamy is a famous Chinese input system. Boshiamy codes a Chinese character as a key sequence like the other Chinese input systems, and it retrieves a Chinese character by the mapping between key sequences and Chinese characters. In this new accessing technique, the skill of hashing is also employed to maintain the mapping since it is always supposed to be an efficient accessing method of data from key sequences to Chinese characters.

In 1984, a letter-oriented minimal perfect hashing function is proposed by Chang and Lee [6]. This hashing function is a one-to-one mapping between keywords and their address space. It is an effective and efficient method. For this reason, we employ this hashing function in our new accessing technique. It helps us to retrieve Chinese characters from the mapping table directly.

In Section 2, we shall have a brief review of the minimal perfect hashing function method [6]. After that, our accessing technique for Boshiamy Chinese input system will be described in Section 3. In Section 4, the analyses of our technique are discussed. Finally, the conclusions will be made in Section 5.

2. A Letter-Oriented Minimal Perfect Hashing Function

Chang and Lee proposed a letter-oriented minimal perfect hashing function in [6]. Given a set of keywords $\{W_1, W_2, W_3, \ldots, W_n\}$, Chang and Lee extracted distinct character pairs from this word set. The distinct character pair of W_i is

Proceedings of the 5th International Workshop Information Retrieval with Asian Languages

Copyright ACM 1-58113-300-6/00/009 ... $5.00

defined to be $dp(W_i) = (a_i, b_i)$, where both a_i and b_i belong to $\{A, B, C, \ldots, Z\}$, and $(a_i, b_i) \neq (a_j, b_j)$ if $i \neq j$. Next, sort the distinct character pairs in their lexical order, and separate the distinct character pairs into groups $\{G_1, G_2, G_3, \ldots, G_r\}$ according to the same first element of the pairs. Let G_i be the collection of the distinct character pairs with the same first element a_i, and the number of distinct character pairs in group G_i is denoted by S_i.

Then Chang and Lee defined the function $D(a_i)$ to be $\sum_{j=1}^{i-1} S_j$ that represents the sum of the numbers of character pairs in $G_1, G_2, G_3, \ldots, G_{i-1}$. Besides, based on the Chinese Remainder Theorem, Chang and Lee also defined a function $P(x)=y$, where $x \in \{A, B, C, \ldots, Z\}$ and y is a prime number, and if $x \neq x'$, then $P(x) \neq P(x')$. Here the Chinese Remainder Theorem is defined as follows: Given $2n$ integers $r_1, r_2, r_3, \ldots, r_n$ and $m_1, m_2, m_3, \ldots, m_n$. If $m_1, m_2, m_3, \ldots, m_n$ are relatively prime to each other, there exists a constant C for $C \equiv r_1 \pmod{m_1}$, $C \equiv r_2 \pmod{m_2}$, $C \equiv r_3 \pmod{m_3}$, ..., and $C \equiv r_n \pmod{m_n}$. Thus, for each a_i, Chang and Lee calculated a constant for each a_i. For each a_i, they defined this constant as $C(a_i)$. $C(a_i)$ must satisfy the requirement of Chinese Remainder Theorem such that $C(a_i) \equiv 1 \pmod{P(b_1)}$, $C(a_i) \equiv 2 \pmod{P(b_2)}$, $C(a_i) \equiv 3 \pmod{P(b_3)}$, ..., and $C(a_i) \equiv n' \pmod{P(b_n)}$. According to the definitions of D, P, and C, Chang and Lee proposed a letter-oriented minimal perfect hashing function $H(a, b) = D(a) + (C(a) \bmod P(b))$, where (a, b) is the given character pair.

An example in [4] is employed here to illustrate this skill as follows: Consider the instruction set of MAC-1 {LOAD, STOD, ADDD, SUBD, JPOS, JZER, JUMP, LOCO, LODL, STOL, ADDL, SUBL, JNEG, JNZE, CALL, PSHI, POPI, PUSH, $POP, RETN, SWAP, INSP, DESP}. Their distinct character pairs can be obtained by extracting the second and the last letters from the instruction set. The 23 distinct character pairs are (O, D), (T, D), (D, D), (U, D), (P, S), (Z, R), (U, P), (O, O), (O, L), (T, L), (D, L), (U, L), (N, G), (N, E), (A, L), (S, I), (O, I), (U, H), (P, P), (E, N), (W, P), (N, P), and (E, P). The 23 distinct character pairs are divided into 11 groups, based on their first elements as shown in Table 1. Then, $D(x)$, $P(x)$, and $C(x)$ is calculated like the previous description. The table containing $D(x)$, $P(x)$, and $C(x)$ is shown in Table 2. If 'ADDL' needs to be searched, then the extracted distinct character pair is (D, L). From Table 2, we have D(D)=1, C(D)=113, and P(L)=37. H(D, L) = D(D) + (C(D) mod P(L)) = 1 + (113 mod 37) = 3. So, 'ADDL' can be retrieved from the third entry of Table 1.

3. Our Scheme for Boshiamy Chinese Input Method

Boshiamy [7] codes a Chinese character as a key sequence which contains one to four keys. Each key is an English letter. For example, the characters '高', '中', '台', and '誠' are coded as '*Q*', '*C Γ*', '*U O O*', and '*I A Q N*'. The user types the key sequences from the keyboard, and Boshiamy maps the input key sequences to the corresponding Chinese characters. To maintain the mapping between key sequences and Chinese characters, we have to reserve 26^4 memory addresses intuitively since there are 26 letters in the English alphabet and we have to save all the possible combinations. This is a waste of memory space. Obviously, there are some addresses which do not contain Chinese characters in fact. To remedy the above problem, we can construct a character table as shown in Table 3. The character table contains two columns, the key sequences of Boshiamy and Chinese characters. Each character is arranged in each row compactly, and the character table is sorted in lexical order of the key sequences. The character table is employed to maintain the mapping between key sequences and Chinese characters. However, if the sequential and binary searches are the only choices on the table, the Chinese character accessing of Boshiamy shall be inefficient. Therefore, in this paper, we design a minimal perfect hashing function based on Chang and Lee's method to speed up the accessing of Chinese characters using Boshiamy Chinese input system. The new technique is now described below.

At first, an extra symbol β is employed in the proposed technique. β means a blank. Our technique uses 27 symbols that are 26 letters in the English alphabet and a β. If the input key sequence of Boshiamy does not contain four symbols, some β's are inserted into the tail of the key sequence. Therefore, each input key sequence of Boshiamy is automatically converted into the key sequence with exact four symbols after the above process. For example, if the input key sequence is '*Q*', we treat this sequence as '*Q ββ β*'; if the input key sequence is '*C Γ*', we treat it as '*C I β β*', and so on.

Next, we construct two kinds of tables beforehand. They are an indirect hashing table and 26^2 direct tables. These tables will be employed to help the accessing of the character table. Let the input key sequence of Boshiamy be '*a b c d*'. Here *a*, *b*, *c*, and *d* denote four symbols, and they must belong to $\{\beta, A, B, C, \ldots, Z\}$. The key sequence is divided into two parts '*a b*' and '*c d*' first. For the first part, we design an indirect hashing function to calculate a value *m* according to '*a*' and '*b*', and the content of the indirect hashing table. For the second part, a direct hashing table is selected from 26^2 direct hashing tables $T_{AA}, T_{AB}, T_{AC}, \ldots, T_{ZY}, T_{ZZ}$. After the determination of a direct hashing table T_{ab}, a direct hashing function is implemented to retrieve the corresponding Chinese characters according to *m*, '*c*', and '*d*'. In the following subsections, we shall describe the contents of indirect and direct hashing tables and the implements of indirect and direct hashing functions in detail. The overview of our scheme is shown in Figure 2.

3.1 Part One of Proposed Technique

First, we should construct the indirect hashing table. Let the key sequence of Boshiamy be '*a b c d*'. Key sequences can be divided into 26×27 groups by '*a b*'. Thy are $\overbrace{A\beta, AA, AB, \cdots, ZY, ZZ}^{26\times27}$ groups. Let the groups be G_1, G_2, G_3, ..., and $G_{26\times27}$, and the number of key sequences in the groups are S_1, S_2, S_3, ..., and $S_{26\times27}$, respectively. Then, the base address m_i of each groups G_i is calculated as $\sum_{j=1}^{i-1} S_j$, like the function D of Chang and Lee's method. We record the base address of each group in the indirect hashing table as shown in Table 4.

Next, an indirect hashing function is designed to access the indirect hashing table. The indirect hashing function is defined as $HI(a, b) = 27 \times (R(a)-2)+R(b)$, where $R(x)$ is the lexical order of x. Here, we define $R(\beta)$ to be 1, $R(A)$ to be 2, $R(B)$ to be 3, ..., $R(Z)$ to be 27. By searching Table 4 with the function HI, the base address m of the corresponding group could be retrieved. We could retrieve directly the Chinese character from the content of the character table pointed by the base address m if '*b*' is 'β'. Otherwise, we shall pass the base address m into the next part.

3.2 Part Two of Proposed Technique

In the second part, we have to construct 26^2 direct hashing tables T_{AA}, T_{AB}, T_{AC}, ..., T_{ZY}, T_{ZZ} first. These direct hashing tables are constructed by Chang and Lee's method [6]. In the following, we brief how to construct these direct hashing table T_{AA} in detail. If an input key sequence of Boshiamy is '*A A c d*', (*c d*) will be one of the elements in {(*A*, β), (*D*, *B*), (*D*, *K*), (*D*, *O*), (*J*, *A*), (*J*, *Q*), (*J*, *W*), (*K*, β), (*M*, β), (*O*, β), (*O*, *B*), (*O*, *M*), (*P*, β), (*Q*, *A*), (*Q*, *B*), (*Q*, *E*), (*Q*, *I*), (*Q*, *N*), (*Q*, *O*), (*Q*, *Q*), (*Q*, *S*), (*Q*, *V*), (*T*, β), (*X*, *H*), (*X*, *I*), (*X*, *V*), (*X*, *X*), (*X*, β), (*A*, β), (*Y*, *O*), (*Y*, β), (β,β)} according to the definition of the key sequences of Boshiamy. Next, $P(x)$, $D(x)$, $C(x)$ of T_{AA} are calculated according to the letter-oriented minimal perfect hashing scheme, which has been explained in Section 2. T_{AA} is then constructed and shown in Table 5. Similarly, the tables T_{AB}, T_{AC}, T_{AD}, ..., T_{ZY}, T_{ZZ} can be constructed similar to T_{AA}.

After the construction of these 26^2 direct hashing tables, one of the 26^2 direct hashing tables should be selected by the first two symbols of the input key sequence (i.e. '*a b*'). If '*a b*' is '*A A*', then T_{AA} will be used in the second part; if '*a b*' is '*AB*', then T_{AB} will be used; and so on. After the determination of the hashing table T_{ab}, a direct hashing function HD is defined as $HD(m, c, d) = m + D(c) + (C(c) \bmod P(d))$, where the function P, D, and C are defined on the selected table T_{ab}, and m is the base address retrieved from the first part. By using this HD, the corresponding Chinese character located on the $HD(m, c, d)$-th entry of the character table can then be retrieved. Note that the key sequence of the character table is useless when we access the corresponding Chinese character. Therefore, it can be discarded in the character table.

3.3 Algorithms of Proposed Technique

Algorithm 1: [construction of character table, indirect hashing table, and direct hashing tables]

Step 1: Record the key sequences of Boshiamy and the Chinese characters in the character table.

Step 2: Sort each row of the character table in lexical order of the key sequences.

Step 3: Group each row in the character table into 26×27 groups according to the first two symbols of the key sequence. The 26×27 groups are $A\beta$, AA, AB, $\cdots$, ZY, ZZ, as well as the numbers from 1 to 26×27.

Step 4: Calculate the base address m_i for each group G_i, and record the base address of each group in the indirect hashing table.

Step 5: For each group, construct a corresponding direct hashing table as follows.

Step 6: Extract the last two symbols of each key sequence in the group. According to the extracted pairs of symbols, the direct hashing table can be constructed by using the letter-oriented minimal perfect hashing scheme.

Step 7: Discard the columns of the key sequence in the character table.

Algorithm 2: [retrieval of Chinese character]

Input: a key sequence of Boshiamy

Output: a corresponding Chinese character

Step 1: Accept the input key sequence of Boshiamy.

Step 2: If the key sequence does not contain four symbols, then β's are inserted into the tail of the key sequence to make the length of the key sequence to be four. After that, the key sequence contains four symbols '*a b c d*'.

Step 3: Partition '*a b c d*' into two pair (*a*, *b*) and (*c*, *d*).

Step 4: According the indirect table, as shown in Table 4, a base address m is retrieved by the indirect hashing function HI. $HI(a, b) = 27 \times (R(a)-2)+R(b)$.

Step 5: If '*b*' is 'β', then the corresponding Chinese character can be retrieved through address m from the character table; otherwise, continue to perform the following steps.

Step 6: Select a direct hashing table T_{ab} according to 'a' and 'b'. Then, the functions D, C, and P can be determined by the selected table T_{ab}.

Step 7: The direct hashing function is defined as HD(m, c, d) = m + D(c) + (C(c) mod P(d)). HD computes an address, and the corresponding Chinese character can be retrieved through this address from the character table.

3.4 Examples of Proposed Technique

Example 3.1

Consider the Chinese character '台'. The input key sequence of Boshiamy of '台' is '*U O O*'. The key sequence '*U O O*' contains only three symbols, so we treat the key sequence as '*U O O* β'. The sequence '*U O O* β' is partitioned into two pairs (*U*, *O*) and (*O*, β). In the first part, HI is employed to retrieve the based address m from the indirect hashing table shown in Table 4. HI(*U*, *O*) = 27 × (R(*U*)-2)+R(*O*) = 27 × 20 + 16 = 556. Hence m is 10445. In the second part, because the first two symbols of the key sequence are '*U*' and '*O*', the direct hashing table T_{UO} will be used. According to T_{UO} shown in Table 6, HD(m, *O*, β) = m + D(*O*) + (C(*O*) mod P(β)) = 10445 + 9 + (127097 mod 2) = 10455. Therefore, the Chinese character '台' can be retrieved from the 10455th entry of the character table.

Example 3.2

Consider the Chinese character '誠'. The input key sequence of Boshiamy of '誠' is '*I A Q N*'. The key sequence '*I A Q N*' contains four symbols, so we do not insert any 'β' into the key sequence. The sequence '*I A Q N*' is partitioned into two pairs (*I*, *A*) and (*Q*, *N*). In the first part, HI is employed to retrieve the base address m from the indirect hashing table shown in Table 4. HI(*I*, *A*) = 27 × (R(*I*)-2)+R(*A*) = 27 × 8 + 2 = 218. Hence m is 3615. In the second part, because the first two symbols of the key sequence are '*I*' and '*A*', the direct hashing table T_{IA} will be used. According to T_{IA} shown in Table 7, HD(m, *Q*, *N*) = m + D(*Q*) + (C(*Q*) mod P(*N*)) = 3615 + 9 + (3008706 mod 47) = 3625. Therefore, the Chinese character '誠' can be retrieved from the 3625th entry of the character table.

4. Analysis and Discussions

In this section, we analyze and discuss the performance of our technique. Based on Boshiamy, each Chinese character is coded as one, two, three, or four symbols. Each symbol is an English letter. Since there are 26 letters in the English alphabet, if we save all the possible combinations, there are 26^4 memory addresses. But Boshiamy only contains 12762 key sequences. Of course, we do not intend to save these 12762 key sequences in 26^4 memory addresses. In our technique, we only have to reserve a character table, an indirection hashing table and 26^2 direct hashing tables. The character table contains one column and 12762 rows, and each row contains a Chinese character only. Hence there are 12762 memory space units occupied. The indirect hashing table contains 26 × 27 base addresses. There are 26 × 27 memory space units occupied. As for the direct hashing table, it contains 27 columns for β, *A*, *B*, ..., and *Z*, as well as 3 rows for functions D, P, and C. The function P is identical for each direct hashing table in fact. We do not have to store it repeatedly. Hence there are 27 × 2 memory space units for each direct hashing table and 27 memory space units for the function P. Therefore, the total reserved memory space units add up to 12762 + 26 × 27 + 26^2 × 27 × 2 + 27 = 49995. Since 49995 is extremely smaller than 26^4, our technique is effective in the use of memory space.

On the other hand, the loading factor is defined as the ratio between the number of key sequences and the size of address space. The number of key sequences in Boshiamy is 12762. In our technique, the last address of the Chinese character is located on '䶮', whose Boshiamy code is '*Z Z R B*'. According to our technique, the address of '䶮' in the character table can be calculated as follows. HI(*Z*, *Z*) = 27 × (R(*Z*)-2)+R(*Z*) = 27 × 25 + 27 = 702. The corresponding base address m is 12753. HD(m, *R*, *B*) = m + D(*R*) + (C(*R*) mod P(*B*)) = 12753 + 7 + (17 mod 5) = 12762. Hence, the size of the address space of our technique is 12762, and the loading factor is 12762/12762 = 1. That means the addressing of the memory space in our technique is very efficient. Thus we conclude our technique is suitable for Boshiamy Chinese input system.

We should emphasize here that, in our technique, the character table, indirect hashing table, and direct hashing tables need to be pre-processed. When a user inputs a key sequence, our method uses an indirect hashing function and a direct hashing function to retrieve the corresponding Chinese character. The indirect hashing function only contains one multiplication, one subtraction, and one addition. The direct hashing function only contains two additions and one modulation. These are all simple calculations. Therefore, the retrieving of Chinese characters using our method is very fast.

5. Conclusions

An efficient technique for Chinese characters retrieval using Boshiamy Chinese input system is proposed in this paper. Our technique is inspired by Chang and Lee's letter-oriented minimal perfect hashing scheme. Moreover, this scheme has been modified to suit the Boshiamy Chinese input system. Overall, our new technique is efficient for the following four reasons: First, the hashing function has no collision occurred in our technique; it is also a minimal perfect hashing function. Second, the requirement of memory space is economic in our technique. Third, since the loading factor is 1, the addressing of memory space is efficient. Finally, the operations of our

hashing functions are simple. The corresponding Chinese character can be retrieved efficiently. Therefore, we can say that this technique is efficient and effective for retrieving Chinese characters using Boshiamy Chinese input system.

References

1. Chen, J. N. and Chang, C. C., "A Chinese Character Retrieval Scheme Using Shuang Pinyin," Journal of Information Science and Engineering, Vol. 8, pp. 487-507, 1992.

2. Liu, C. C., Chang, C. C., and Buehrer J., "Encoding and Accessing Chinese Words Using Mandarin Phonetic Spellings," Computer Processing of Chinese and Oriental Languages, Vol. 6, No. 2, pp. 195-204, Dec. 1992.

3. Sung, S. Y., "Chinese Words Accessing Based on Phonetic Input," Proceedings of International Conference on Chinese and Oriented-Language Computing, pp.338-392, Aug. 1989.

4. Chang C. C. and Wu H. C., "A Fast Chinese Characters Accessing Technique Using Mandarin Phonetic Transcriptions," Proceedings of 3rd International Conference in Chinese Information Processing, Beijing, China, pp. 13-19, Oct. 1992.

5. Chang, C. C. and Lee, C. F., "Retrieving Chinese Characters Using DAYI Chinese Input Method," Proceedings of 17th International Conference on Computer Processing of Oriental Languages, Hong Kong, pp. 387-391, Apr. 1997.

6. Chang, C. C. and Lee, R. C. T., "A Letter-oriented Minimal Perfect Hashing Scheme," The Computer Journal, Vol. 29, No. 3, pp. 277-281, 1986.

7. 嘸蝦米輸入法, 劉重次編著, 二十一版, 1999, 行易有限公司.

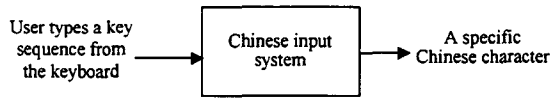

Figure 1. A brief diagram of a Chinese input system

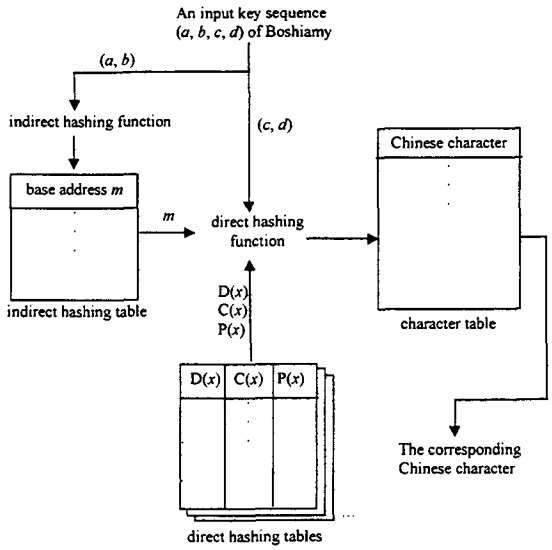

Figure 2. The overview of our scheme

Table 1: The instructions of MAC-1 are divided into 11 groups

Address	Character pairs	Instruction word	Group
1	(A, L)	CALL	G_1
2	(D, D)	ADDD	G_2
3	(D, L)	ADDL	
4	(E, N)	RETN	G_3
5	(E, P)	DESP	
6	(N, E)	JNZE	
7	(N, G)	JNEG	G_4
8	(N, P)	INSP	
9	(O, D)	LOAD	
10	(O, I)	POPI	G_5
11	(O, L)	LODL	
12	(O, O)	LOCO	
13	(P, P)	$POP	G_6
14	(P, S)	JPOS	
15	(S, I)	PSHI	G_7
16	(T, D)	STOD	G_8
17	(T, L)	STOL	
18	(U, D)	SUBD	
19	(U, H)	PUSH	G_9
20	(U, L)	SUBL	
21	(U, P)	JUMP	
22	(W, P)	SWAP	G_{10}
23	(Z, R)	JZER	G_{11}

Table 2: The D(x), C(x), and P(x) for retrieved MAC-1 instructions

x	D(x)	C(x)	P(x)
A	0	1	2
B			3
C			5
D	1	113	7
E	3	1592	11
F			13
G			17
H			19
I			23
J			29
K			31
L			37
M			41
N	5	7635	43
O	8	7477	47
P	12	2280	53
Q			59
R			61
S	14	1	67
T	15	113	71
U	17	227109	73
V			79
W		1	83
X			89
Y			97
Z	22	1	101

Table 3: The character table

Address	Key sequence	Chinese character	Group
0	(*A*, *β*, *β*, *β*)	對	G_1
1	(*A*, *A*, *β*, *β*)	寸	
2	(*A*, *A*, *A*, *β*)	鑫	
.	.	.	G_2
.	.	.	
.	.	.	
31	(*A*, *A*, *Y*, *β*)	鈐	
32	(*A*, *A*, *Y*, *O*)	鋡	
33	(*A*, *B*, *A*, *β*)	佘	
34	(*A*, *B*, *B*, *E*)	鐵	
35	(*A*, *B*, *B*, *O*)	鋡	
.	.	.	G_3
.	.	.	
.	.	.	
57	(*A*, *B*, *T*, *O*)	錒	
58	(*A*, *B*, *U*, *β*)	鈆	
.	.	.	.
.	.	.	.
.	.	.	.
12754	(*Z*, *Z*, *E*, *W*)	篷	
12755	(*Z*, *Z*, *F*, *β*)	艋	
12756	(*Z*, *Z*, *G*, *β*)	筮	
.	.	.	$G_{26\times 27}$
.	.	.	
.	.	.	
12761	(*Z*, *Z*, *R*, *β*)	篾	
12762	(*Z*, *Z*, *R*, *B*)	簏	

Table 4: The indirect hashing table

Address	Key sequence (a, b)	Base address m	Group
1	(*A*, *β*)	0	G_1
2	(*A*, *A*)	1	G_2
3	(*A*, *B*)	31	G_3
.	.	.	.
.	.	.	.
.	.	.	.
26	(*A*, *Y*)	543	G_{26}
27	(*A*, *Z*)	570	G_{27}
28	(*B*, *β*)	588	G_{28}
29	(*B*, *A*)	589	G_{29}
30	(*B*, *B*)	602	G_{30}
.	.	.	.
.	.	.	.
.	.	.	.
53	(*B*, *Y*)	924	G_{53}
54	(*B*, *Z*)	934	G_{54}
.	.	.	.
.	.	.	.
.	.	.	.
676	(*Z*, *β*)	12319	G_{675}
677	(*Z*, *A*)	12320	G_{676}
678	(*Z*, *B*)	12340	G_{677}
.	.	.	.
.	.	.	.
.	.	.	.
701	(*Z*, *Y*)	12744	G_{701}
702	(*Z*, *Z*)	12753	G_{702}

Table 5: The direct hashing table of T_{AA}

x	D(x)	C(x)	P(x)
β	0	1	2
A	1	1	3
B			5
C			7
D	2	7476	11
E			13
F			17
G			19
H			23
I			29
J	5	8725	31
K	8	1	37
L			41
M	9	1	43
N			47
O	10	347	53
P	13	1	59
Q	14	4966972282327	61
R			67
S			71
T	23	1	73
U			79
V			83
W			89
X	24	10204987	97
Y	29	161	101
Z			103

Table 6: The direct hashing table of T_{UQ}

x	D(x)	C(x)	P(x)
β			2
A	0	1	3
B	1	53	5
C			7
D	3	1	11
E			13
F	4	239	17
G			19
H	6	1	23
I			29
J			31
K	7	1	37
L			41
M	8	1	43
N			47
O	9	127097	53
P			59
Q			61
R	14	1	67
S			71
T	15	1	73
U			79
V			83
W	16	1	89
X			97
Y			101
Z	17	1	103

Table 7: The direct hashing table of T_{IA}

x	D(x)	C(x)	P(x)
β	0	1	2
A			3
B			5
C			7
D	1	3553	11
E			13
F			17
G			19
H			23
I			29
J	3	1	31
K	4	1	37
L	5	1	41
M	6	1	43
N			47
O	7	1	53
P	8	1	59
Q	9	3008706	61
R			67
S			71
T	13	1	73
U	14	1	79
V			83
W			89
X	15	5337	97
Y	17	1	101
Z			103

Table 8: The direct hashing table of T_{ZZ}

x	D(x)	C(x)	P(x)
β			2
A			3
B			5
C			7
D			11
E	0	1	13
F	1	1	17
G	2	1	19
H			23
I			29
J	3	1	31
K			37
L			41
M			43
N			47
O	4	1	53
P			59
Q	5	342	61
R	7	17	67
S			71
T			73
U			79
V			83
W			89
X			97
Y			101
Z			103