Design and Implementation of an I/O System for Chinese Word Processing

Xin Cao

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

March 1987

ISBN 0-315-35562-X

ABSTRACT

Design and Implementation of an I/O System for Chinese Word Processing

Xin Cao

A study of Chinese word processing system with a capacity of handling 4864 Chinese characters is conducted. An implementation of the system is carried out on both CYBER170/835 and VAX11/780 computer systems. In the processing system, five encoding methods, based on a combination of three basic schemes of phonics, radical and shape, are designed for individual preferences. A conventional English keyboard is adopted, and some modifications have been made in order to facilitate the input of a mixed text of English and coded Chinese. The best performance among these encoding methods is expected to require an average of 3.54 key-punches/character with 1.4% conflict code rate. A division hash function has been applied in order to achieve a high rate in retrieving the character from the pattern data base. The problems of hash collision and encoding collision are resolved by using a two-dimensional chaining technique. An efficient hash table size of 4337 has been obtained together with a space utilization and a time complexity of 1.55 and 1.56, respectively.

All output data can be edited in desired formats and sizes, and can be monitored over a CRT display. A file editing system is also available to experienced users. Hard copies of the input text can be reproduced in a high resolution of 50*50 character matrices.

內 容 提 要

本文介紹了一個具有處理4864個字的中文輸入輸出系統的設計.這個中文處理系統已經在計算機系統CYBER170/835,VAX/780和IBM/AT上得到實現.在這個處理系統中,有五種輸入方法可供選擇.它們是建立在拼音碼,字根碼和字形碼的基礎上.中文輸入鍵盤是利用現有的英文鍵盤經過合理調整后實現的.在五種輸入方法中,最有效的輸入方法平均每輸入一個字的按鍵次數是3.54,最少的異字同碼碰撞率是1.4%.該系統采用了混列取模法,以提高從字庫中檢索中文字的效率,而由混列取模法和輸入編碼造成的碰撞問題,則用二向串接技术加以適當的解決.經過實驗,找出了經濟有效混列表的大小值應為4337,並且其時間和空間的利用率分別是1.55和1.56.

中文輸出的編輯可以在銀光屏上進行,熟練操作員還可以直接利用文件編譯系統.每個中文字的輸出由50*50点陣形成.

ACKNOWLEDGEMENTS

I wish to express my gratitude and deep appreciation to Dr. C. Y. Suen, my thesis supervisor, for initiating this project and providing continued guidance throughout the investigation.

I wish to thank the Natural Sciences and Engineering Research Council of Canada, and the Department of Education of Quebec for their financial support for this project.

I also wish to express my great appreciation to Dr. S. Lin, and Mrs. Y. Y. Zhou, my aunt, for sponsoring me to study in Canada and providing substantial financial support since I left China.

I wish to thank all my colleaques and friends for their kindness and help offered during all the time I stay in Montreal.

Finally, I wish to express my deep appreciation to my family for their understanding and consistent supports both financially and spiritually.

TABLE OF CONTENTS

CONTENTS PAGE

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Chinese is a language with a long history in that it can be traced back five thousand years, and its main linguistic feature, ideographs different in essence from the alphabetic writing, has not changed. Up to this very day, Chinese remains as one of the few languages which does not benefit fully from the ever-advancing technology of electronic data-processing and tele-communications. The reason, perhaps, is due to the fact that Chinese is written in the form of a set of so-called ideographic characters, which amount almost to a hundred thousand, and perplexes most people, even those who are experts in data-processing of English which has only 26 letters in its repertoire.

Exploration for Chinese word processing (CWP) system has been tackled by computer specialists and computational linguists since the 1960's. Great advances have been brought about through their efforts. Generating, displaying, storing and printing Chinese character are no longer the crucial problems. It is time now to develop data-processing systems for Chinese. This thesis proposes to design just such a system.

Generally speaking, a CWP system consists of several functions, such as input, central processing, and output. Each one will need some special treatments in design. In input, the input encoding that maps a two dimensional ideographic Chinese character to a one dimensional symbolic coded string, is the central problem to be settled. And a special attention has to be paid to the arrangment of keyboard. A rapid retrieval of a coded character string to locate its physical location in character data base, is the focus in the central processing. And the high speed displaying, printing and formatting constitute the main considerations in output. It is these objectives that this thesis attempts to explore.

Unlike English and other western languages, which are purely phonetic languages based on symbols, Chinese is comprised of characters in that each character forms one ideogram and has one phonetic syllable. According to the characteristic of Chinese language, a lot of researchers have put great efforts to translate the ideographic characters into symbolic coded strings. These efforts have led to hundreds of proposals of Chinese input encoding methods. Nowadays, more than 30 of them have been put into commercial uses [1]. And these input encoding methods could be classified into three major classes, that is, the radical method, the phonetic method, and the mixed method of phonics and radical.

The radical encoding method is usually based on the information of a character pattern. A Chinese character is basically composed of strokes. A radical can have one or several strokes, and it is the fundamental component in the Chinese characters. In Xinhua dictionary [2], more than three hundreds of them have been used to index the Chinese characters. In radical encoding method, a character is decomposed into radicals according to certain sequence, and the selected radicals or their corresponding codes are then entered into the processing system in order to retrieve the correct character. Work pertaining to the radical encoding method can be found in Huang et al.[3], Kiang et al.[4], and so on.

In 1982, Huang et al.[3] introduced a three corner encoding method. According to his analysis, a Chinese character can be decomposed into a sequence of the pattern corners. The corner sequence goes from the upper left corner to the lower right one in a "Z" route. According to the shapes in the three major corners, every character is coded by digit codes from 1 to 99 instead of the basic symbols (radicals) of Chinese characters. A character can be input by using three pairs of codes (six digits). The low conflict code rate of 3%, the ratio of the sum of characters with the same codes but different character patterns and the total number of characters in the data base, shows that this method is very valuable. However, the memorization of 99 symbol codes and the familiarization with the sophisticated pattern decomposition rules constitute the main disadvantages of this method.

1) Data input is efficient. The average times of keyboard punch for inputting a character is generally less than in other input schemes.

2) There will be no serious problem for people with different dialects to use a Chinese character radical input system, and there is a possibility of inputting unknown characters by people who do not even know Chinese.

3) There is a comparative uniqueness of the characters, once the rules or codes are thoroughly applied.

However, from our points of view, the Chinese character radical method is somewhat difficult to understand and master. People need special training to master the encoding system. The reason to say so is obvious:

1) There exist diversing ways of decomposing each single character into its "basic" patterns or radicals (components).

2) It is difficult for an ordinary user to decide instantly the decomposition, without mention the difficulty of memorizing the code for each pattern or the patterns of each code.

These disadvantages are the main obstacle to popularize the use of the radical input method.

Compared with the radical method, phonetic input encoding scheme is more natural and straight forward.

People who are familiar with certain phonetic spelling system will have little difficulty in mastering the use of the corresponding phonetic input encoding method.

However there are still several severe problems which block the way for development of the phonetic input system. They can be summarized as follows:

1) Problem of homonyms. There exist a large number of homonyms in Chinese phonetic spelling systems. The number of homonyms for a specific pronunciation could be more than 100 [2]. For example, there are 81 different Chinese characters having the same pronunciation "ji" in the 4864 samples of Chinese characters in Suen's phonetic spelling system [6].
2) Problems of different Chinese phonetic spelling systems. There are several different phonetic spelling systems in use nowadays, such as Suen, Wade, Wade-Giles, Gwoyeu Romatzyh, Yale, Liu and Pinyin system. People who are familiar with one phonetic spelling system may feel quite uncomfortable to use another one. Therefore, a phonetic input method would be limited to those who know that specific phonetic spelling system.
3) Problem of dialects. In China, people in different regions speak different dialects and they may not be able to pronounce a character correctly all the time. Thus, they will come across the difficulty of coding a character according to a certain phonetic spelling

latter.

In recent years, studies have been made by Ooka et al.[9], Zhang [10] and so on, to combine the two schemes mentioned above in order to minimize the undesirable ambiguity.

In Ooka's method, the input code of a character consists of two parts: One is the basic pronunciation of the character in Pinyin system, another part is a supplemental basic pronunciation of one of the radicals constituting that specific character. There are several rules regarding to the selection of a proper radical in order to reduce the number of homonyms.

In Zhang's scheme, a character is decomposed first into a set of smaller elements which are still independent, pronunciable characters, or a set of radicals with assigned pronunciations according to the Pinyin system. Then these derived smaller elements are used as input codes.

The conflicting rate of the code in such input schemes is relatively low. Nevertheless, because of difficulties in learning and memorizing both the phonetic spelling system and the complex radical decomposition rules, this scheme, somewhat, seems difficult to master.

In this thesis, specially in chapter three, our new input encoding method in CWP system is discussed in detail.

Rather than a single input encoding method, five encoding methods based on the combinations of three basic schemes (radical, phonetic spelling and shape) have been designed. With this system, the users can operate in any of the five schemes according to their preference.

The other aspects, such as keyboard design, central processing unit, and the output are briefly discussed in chapter two. The results of performance of our CWP system, regarding to input, central processing and output, are presented and discussed quantitatively in chapter four. And, finally, in chapter five, a conclusion and some suggestions for extension of the work done in this thesis are presented.

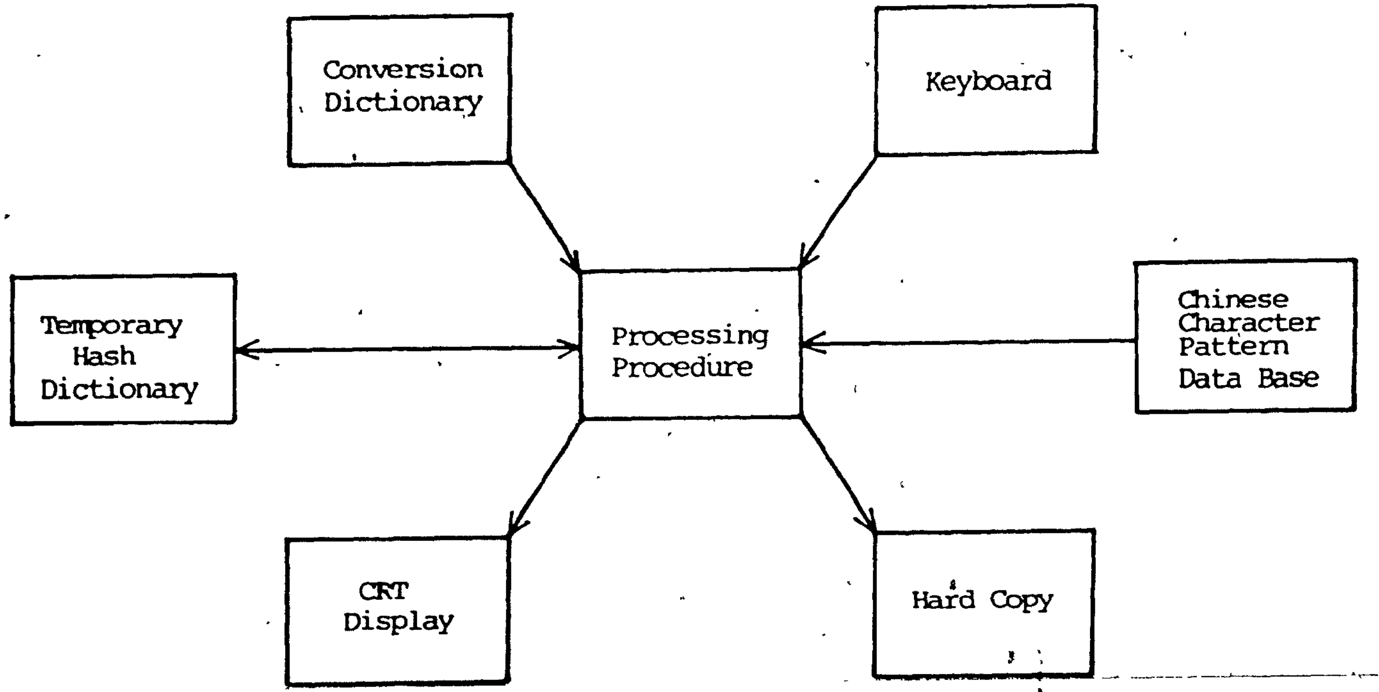


Figure 1. Block Diagram of the Chinese Word Processing System

homonyms, it is extremely difficult to find a simple and easy memorizable coding technique. A compromise between easy coding and less collision has to be made prior to the design of such a system. In the next section, an appropriate input coding technique is discussed together with the corresponding collision handling method.

2.2. INPUT CODING AND COLLISION HANDLING

2.2.1. Collision Handling

As mentioned before, an efficient Chinese character input coding method does not only require easy learning, but also less chances of collision. There exist a number of methods to handle collision in Chinese character input coding. For example, if only the phonetic spelling of Chinese character is used as an input code, a large number of collisions will occur. To avoid or reduce the number of collisions, there are several choices. One is to supply some additional information, such as the use of an additional radical code, supplemental pronunciation, and so on. In such case, the use of additional information will increase the complexity of coding. Another way to handle the collision is the use of display method. However the efficiency in input is reduced because of the additional time needed to display the collision characters. In this thesis, the aforementioned collision handling methods are combined and used in order to achieve an optimal result of less collision and ease in coding.

One of the input coding strategies used in this thesis, is based on Chinese character phonetic spelling. As is well known, there are lots of Chinese characters which have the same pronunciation but different structures and meanings. Let us consider the case in which some additional characteristics of Chinese character are added to their phonetic spelling to distinguish them. In this situation, the confusion caused by homonyms can be reduced drastically. One of the valuable characteristics is the tone. There are five tones in the pronunciation of a Chinese character, which are categorized as high tone, rising tone, low tone, falling tone and light tone. For example, there are 81 Chinese characters having the basic phonetic spelling "ji" out of the 4864 samples of Chinese characters in Suen's phonetic system. After a high tone is added, the number of homonyms left is only 25, which is one third of the original amount. Another characteristic which can be used is based on the structure of Chinese characters. In this way, one can take out some parts of the character, assign certain codes to these parts, and use the codes of these parts as additional information to distinquish the homonyms. For example, in the above mentioned 81 homonyms of "ji", if the code of "言" is added to the basic phonetic spelling, there will be only 4 homonyms left, which are "計", "讥", "記" and "詰", and they account for only one twentieth of the original ones. If the code corresponding to "月" is added to the phonetic spelling "ji", a single character "肌" will be produced. That which part should be draw off and how to code them will be discussed in greater detail later.

Another coding strategy used in this thesis is one based on the structural information of Chinese characters. In this case, the structural characteristics of Chinese characters are classified and coded as the fundamental input information, and the additional phonetic spelling is used to avoid or reduce the number of collisions. Since all Chinese characters can be decomposed into a set of character radicals (more precise definition will be given in chapter 3), it is realizable to use radicals to identify different Chinese characters. The problem is how to classify radicals in such a way that a Chinese character can be identified efficiently, because the defined radicals must not only be easy to learn and memorize, but also they produce the least number of radical codes and with lowest collision in identifying Chinese characters. In this thesis, 202 radicals are drawn from 4864 Chinese characters after careful analyses and statistical calculations, 34 codes are used to index them. More detailed rules about the selection of these radicals from a given Chinese character are discussed in chapter 3.

It could be seen in chapter 4 that, collision rate will be 12.3% if only radical codes are used to identify Chinese characters. A lower rate of collision can be expected if some additional pieces of information are added to. For example, if phonetic spelling of characters is added to their radical codes, a 2.4% collision rate is obtained, which is only one sixth of that when only radical codes are applied to identify Chinese characters.

2.2.2. Radical Input Coding Strategy

In this section, the radical coding strategy will be emphasized. The coding methods, such as phonetic code and shape code will be discussed in chapter 3.

Because of the characteristics of Chinese character, the radical is always regarded as a major and important piece of information in the input code design. As a result, the radical code becomes a major factor in the attempt to gain high input speed, less character collision and so on.

After a careful analysis we know that the most basic strokes of Chinese characters are ㇔, ㇐, ㇑, ㇒, ㇏, ㇀, ㇖, ㇚, ㇂, ㇕, ㇗ [5], which can constitute up to thousands of Chinese characters. Let us call these basic strokes. In view of the structure of Chinese characters, the numbers of strokes constituting different characters are not the same; and can vary from one to more than 30. It is not economical and efficient to input those strokes as codes to identify characters though less collision would be expected by doing so. For that reason the selection of radicals and their corresponding codes, determination of the number of radicals to be used, and effect of coding sequence should be discussed in detail for when considering of efficiency and ease of use. It must bear in mind that we are looking for such a coding strategy which should give considerations to both efficiency and ease at the same time.

kind of collision, "𠬝" is not selected as a radical. For the same reason, radicals "衤", "扌" and "言" have the same first phoneme of "y". When the radical code of "曷" is combined together with "y", the collision characters "褐", "揭" and "謁" will be produced. Therefore, "扌" and "言" will not be defined as radicals. The selection of radicals in these cases is rather arbitrary and heavily depends on experiences.

In some situations, it is impossible to simply eliminate the radicals from the dictionary and to use the first phoneme of pronunciation as the only means to avoid collision characters. In such cases, the ideograph code of radicals should be adopted too. For example, the radicals, "月" and "衤" have the same "y" as the first phoneme. When radical codes for "甫" are input together with "y", the collision characters of "脯" and "補" will be resulted. Hence, in order to distinguish these characters, the ideograph code "a" of the radical "月" is adopted while the first phoneme of "衤" is used as its radical code.

Some more examples are given as follows.

(1) English dot	Radical	Pronunciation	Code
left-side dot	ノ		
right-side dot	丶	dian	d
tilting-up dot	㇀		
extended-tilting-up dot	✓		

(2) English horizontal	Radical	Pronunciation	Code

plain horizontal	一		
slant horizontal	㇀	hen	h
slant rising	㇀		
plain rising	一		
(3) English vertical	Radical	Ideograph	Code
upright vertical	丨		
slant vertical	\ /	I	i
vertical-rising	㇙		
vertical-hook	亅		
(4) English left curved	Radical	Pronunciation	Code
slant leftfalling	㇒		
vertical leftfalling	㇓	pie	p
tilting-down dot	'		
plain leftfalling	㇒		
(5) English right curved	Radical	Pronunciation	Code
slant rightfalling	㇏	na	n
plain rightfalling	㇏		
(6) English turning	Radical	Pronunciation	Code
horizontal-leftfalling	㇇		
horizontal-turning-hook	㇆	rje	rj
horizontal-turning	㇕		
(7) English concave	Radical	Pronunciation	Code
vertical-horizontal-vertical	凵		
vertical-roundbending-hook	㇟	au	au
slant hook	㇂		
plain bending-hook	㇄		

(8) English turning	Radical	Pronunciation	Code
horizontal-turning-turning-turning-hook	㇡	lao	1
horizontal-turning-turning-leftfalling	㇋		

(9) English	Radical	Ideograph	Code
sun	日	Q	q

(10) English	Radical	Ideograph	Code
moon	月	A	a

(11) English	Radical	Pronunciation	Code
ear	耳	er	er

It should be noticed that since several radicals may have the same code, the number of radical codes is undoubtedly less than the number of radicals. A complete list of radicals employed in our system and their corresponding codes is given in Table 1.

2. Number of Radicals

The number of radicals used to identify each Chinese character is another important factor. Let n_c denote the total number of radical codes and n_k denote the number of keys used to identify each character. If each Chinese character is identified by one key, there will be n_c choices for a total of n_c radical codes. If n_k keys are used to identify each character, we have at most,

$$\sum_{i} n_c! / (n_c-i)! \qquad (i=1,n_k)$$

Table 1. List of Radicals and Their Corresponding Codes

Radical Codes	Radical Pronunciation or Ideograph	Radicals
a	A	月 , 冃 , 目 , 月
au	au	冂 , 凵 , 乚 , ㇃
b	ba	丷 , 灬 , 刂 , 八 ハ) , 儿
	bao	宀 , 冖
c	C	工 , コ , 匚 , ㄊ , ユ
ch	chi	七 , 匕 , 乂
		其
d	dao	刀 , 刃 , ⺈ , 刂
	die	丶 (㇀ , ㇀ , ㇀) , 亠
ds	dsi	止 , ⺊
e	E	E , ヨ , ⺕ , 王 , 彐
er	er	儿 , ル
		阝 , 卩 , 耳
		二 , 冫
f	F	F , ㄱ , ュ
	fen	丰 , 牛 扌 , 丰 , 才 , 午 , 扌
g	ge	戈 , 戈
	gen	艮 , 艮
	gong	工 , 弓 , 弓 , 勹
	guang	广
h	hen	一 (一 , 一)
	hu	虍
	huo	火

(Table 1 continue)

Radical Codes	Radical Pronunciation or Ideograph	Radicals
rj	rje	㇇, フ, ㇆, 勹, ヲ
	rjou	舟, 舟
	rju	⺮
	rjua	⺤, ⺥
s	shen	身
	shi	尸, 戶
		十, ナ (ナ), 卝
		豕, 豖
	si	厶, 山, 罒, 罒, 皿
	shui	水, 氺, 氺
t	T	丅
	tu	土 (扌), 士
ts	tsao	艹, 廾, 㐅, 廿, 龷
	tsun	寸
u	U	毌, 毋, 罒
v	V	く, 𠃋, 𠃊, ㇀, 𠂊
		凸
w	wang	王, 王
	wen	文, 攵, 夂, 夊
x	X	乂, 㐅
	xi	襾, 西
		夕, 夕
	xiao	小, 小, 少, ⺍
	xin	心, 忄

choices.

For example, for a total number of radical codes n_C=26, if two keys are used to identify Chinese characters, there will be 676 choices; if four keys are used, there will be 375,076 choices; and if six keys are used, there will be 174,034,276 choices; and so on.

From the above discussion, it can be concluded that on the one hand, in order to reduce collision characters in a large set of Chinese characters, the keys used to identify each character has to be large enough for a certain number of radical codes. On the other hand, to get a high input speed, we have to limit the number of keys used to input each character. Once again, a compromise should be made.

On analysis, we can see that most Chinese characters are differently shaped in terms of structure. The difference may show in one of the structural parts on the left or right, top or bottom, inside or outside, etc.
For example,

"槽" and "嘈" are different on the left side;
"际" and "陪" are different on the right side;
"吉" and "召" are different on the top part;
"琴" and "瑟" are different on the bottom part;
"闾" and "闯" are different on the inside;
"问" and "回" are different on the outside.

This suggests that it is unnecessary to use all the radicals of a character to identify it. Instead, few radicals selected from the whole radical sets of a character are enough to tell itself apart from the other characters. A simulation has been performed on the character set of size 4864 by varying the length of codes and evaluating the total number of collision characters for different approaches in encoding. From this experiment, we have obtained the knowledge that, a maximum of 2 radicals in phonetic spelling based method and a maximum of 4 radicals in character structure based method will best suit the compromise purpose. More details on the rules used to select such radicals will be discussed in chapter 3.

3. Radical Extraction and Radical Sequence

In the previous section, we have already discussed the construction of radicals from given characters, and Table 1 shows the complete list of radicals designed in this way. In this section, a counterpart problem will be reviewed shortly. It is "how to extract radicals listed in Table 1 from a given character". Unfortunately, as shown below, there are many ways to extract radicals from a given character.

For example, the radicals of the character "仁" can be extracted as (丿 , 丨, 一, 一), or (丿 , 丨, 二), or (亻, 二) and so on. However, the various choices imply that certain reduced.

In chapter 3, more detailed discussions and examples on this subject will be given in order to enable readers to understand better this important process.

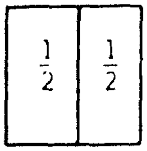
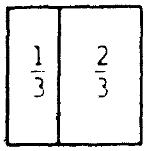
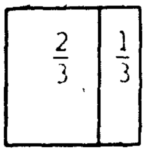
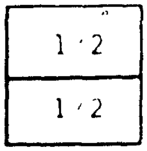
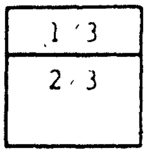
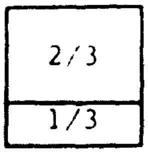
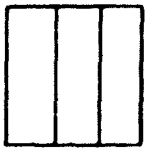
As far as the radical sequence of a character is concerned in this chapter, we only mention that it mainly depends on the Chinese writing stroke-order. More specific description can be found in chapter 3 as well.

In our input system, as known from the last section, five input methods are related to three fundamental schemes, that is, radcial code, phonetic code and shape code. As defined in Table 1, 34 radical codes are formed from the letters of the English alphabet. In Suen's Phonetic System [6], most phonetic symbols are also English letters except letter "u:". And as given in Table 2, the shape codes are ASCII codes as well. These facts make it possible for us to use an available English keyboard directly. However, due to the very high average number of keystrokes in such a design, the input efficiency is relatively low.

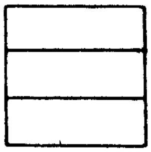
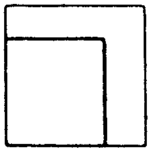
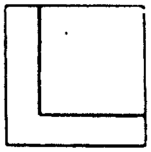
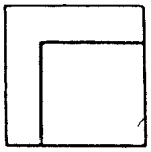
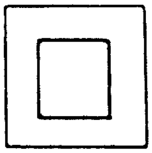
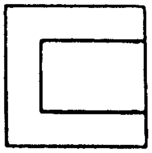
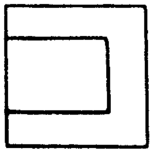
Considering the characteristic of radical codes, phonetic symbols and shape codes, it is more efficient to design a keyboard according to the consonants, vowels and diphthongs in Suen's Phonetic System and to use some additional radical codes which are not phonetic symbols. In this way, 39 phonetic symbols and four radical codes c, q, v, and z, should be arranged on a keyboard. Therefore, a total of 43 keys need to be considered in addition to other alphanumerical numbers and special symbols which already exist on the ordinary English keyboard.

Fig.3 shows a proposed keyboard design based on the above consideration. As shown in Fig.3, 26 capital English letters are kept as it is on the lower shift of the keyboard. The remaining phonetic symbols and radical codes are rearranged on the upper shift of the keyboard.

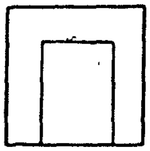
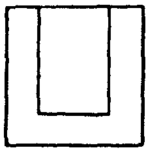
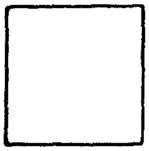
Table 2. Shape Classifications of Chinese Characters and Their Corresponding Shape Codes.

Class	Character Shape	Shape Code	Character Samples
1		"	林 , 兆
2		`	陈
3		`	剖
4		:	昌
5		;	系 , 示
6		!	照 , 黄
7		'	微 , 小 , 候 , 亂

(Table 2 continue)

Class	Character Shape	Shape Code	Character Samples
8		%	憲，亮，蔡 公
9		?	或，可，氣
10		＼	道，起
11		/	夜 在，瓜
12		@	國，母，
13		[	匡
14		]	句，馬

(Table 2 continue)

Class	Character Shape	Shape Code	Character Samples
15		⌢	局，兩，商，向，風
16		—	凶，心、，一
17		+	坐，巫，包，少
18	Crossed Radicals	#	五，我，里，民，西
19	Non-crossed Radicals	*	看，上，用，成

2.4. METHOD OF ACCESSING THE DATA BASE

After we have discussed the encoding technique in our CWP system, the next step is to retrieve the Chinese character from a prepared Chinese character data base.

Naturally speaking, the Chinese character data base is very large. A commercial Chinese character data base may contain 5000 - 15000 Chinese characters. In our experimental one, the 5000 most frequently used characters are used. Even so, an efficient access algorithm is crucial to achieve a high throughput in CWP system. Besides, an index is absolutely required to access such a large data base.

There are a lot of algorithms dealing with techniques to search an index. Some of them are based on comparison, such as, Binary Searching, Fibonacci Searching, Sequential, etc. The best performance we may expect from these algorithms, is about 1 for space utilization (defined as [space used] / [space required]), Log(n) for time complexity (n is the volume of the index, and the time complexity is the average time for searching an item in the index). For number of characters n=1000, the time complexity is Log(n)=10.0. While for n=5000, the average time for searching an item in the index is equal to Log(n)=12.3. It is obvious that, these algorithms are inadequate for our consideration.

2.5. CHARACTER DATA BASE AND DICTIONARY

2.5.1. Character Data Base

The Chinese character fonts are implemented on two different computer systems, CYBER170 CRM and VAX11 RMS. After pre-scan, a Chinese character is digitized into a 50*50 binary matrix. The character matrices are then stored in the computer system as a character pattern data base.

On CYBER170 CRM, 4864 Chinese character matrices are stored in a random-access file, using GRANPAK software which is a random-access I/O. The file accessed by GRANPAK is composed of a fixed CDC logical record length whose size is pre-specified by users. Fig.4 shows the logical structure of the random-access file. The file is made up of the so-called pages. And each page consists of a "Page No." and a "Buffer". While the "Page No." is the page number of the random-access file used to identify each character, the "Buffer" is a data structure whose contents are 50*50 binary values used to store characters. The "Page No." is also equivalent to the character identification number (ID) which will be discussed in a later section.

On VAX11 RMS , relative file organization and direct record access method are applied for character data base design. The data contents for each character in the file are also represented by 50 by 50 binary values.

Page No.	Buffer
0	50 x 50 Binary
1	
⋮	
4863	

Figure 4. Structure of Chinese Character Pattern Data Base

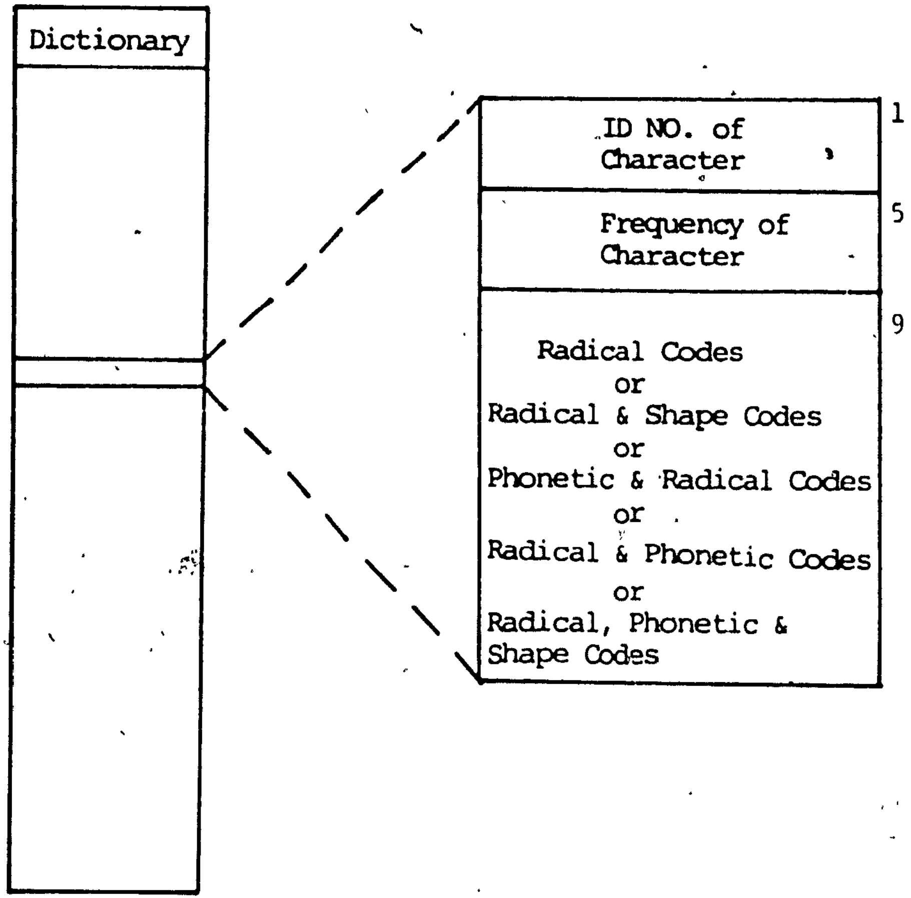

Dictionary
ID NO. of Character
1
Frequency of Character
5
Radical Codes or Radical & Shape Codes or Phonetic & Radical Codes or Radical & Phonetic Codes or Radical, Phonetic & Shape Codes
9

Figure 5. Structure of Conversion Dictionary

with the hash method. The temporary dictionary is also called a hash table. It keeps the same information as the conversion dictionary does besides a multiplicity count, but its data structure is completely different as shown in Fig.6.

To set up a hash table, character codes in the conversion dictionary are read first. By using the proper hash function (division function), a character code is mapped into a bucket in the hash table. In certain cases, it happens that more than one code with different code numbers or with the same code number but different serial numbers are mapped into one bucket. This kind of situation is called collision.. And the former is caused by hash function while the later by encoding. A two-dimensional chaining technique is applied to handle such complex problem as shown in Fig.6. The chaining sequence in the first case depends on the reading sequence from the conversion dictionary, while the latter depends on the character frequency.

The way to retrieve a character input from the keyboard is very similar to that in set-up of the hash table. The input code is mapped into a corresponding bucket by hash function. A comparison of character codes will be carried out along the main chain. As long as the desired character code is found in the main chain, the comparison will stop. Otherwise, the comparison process will continue or go to the

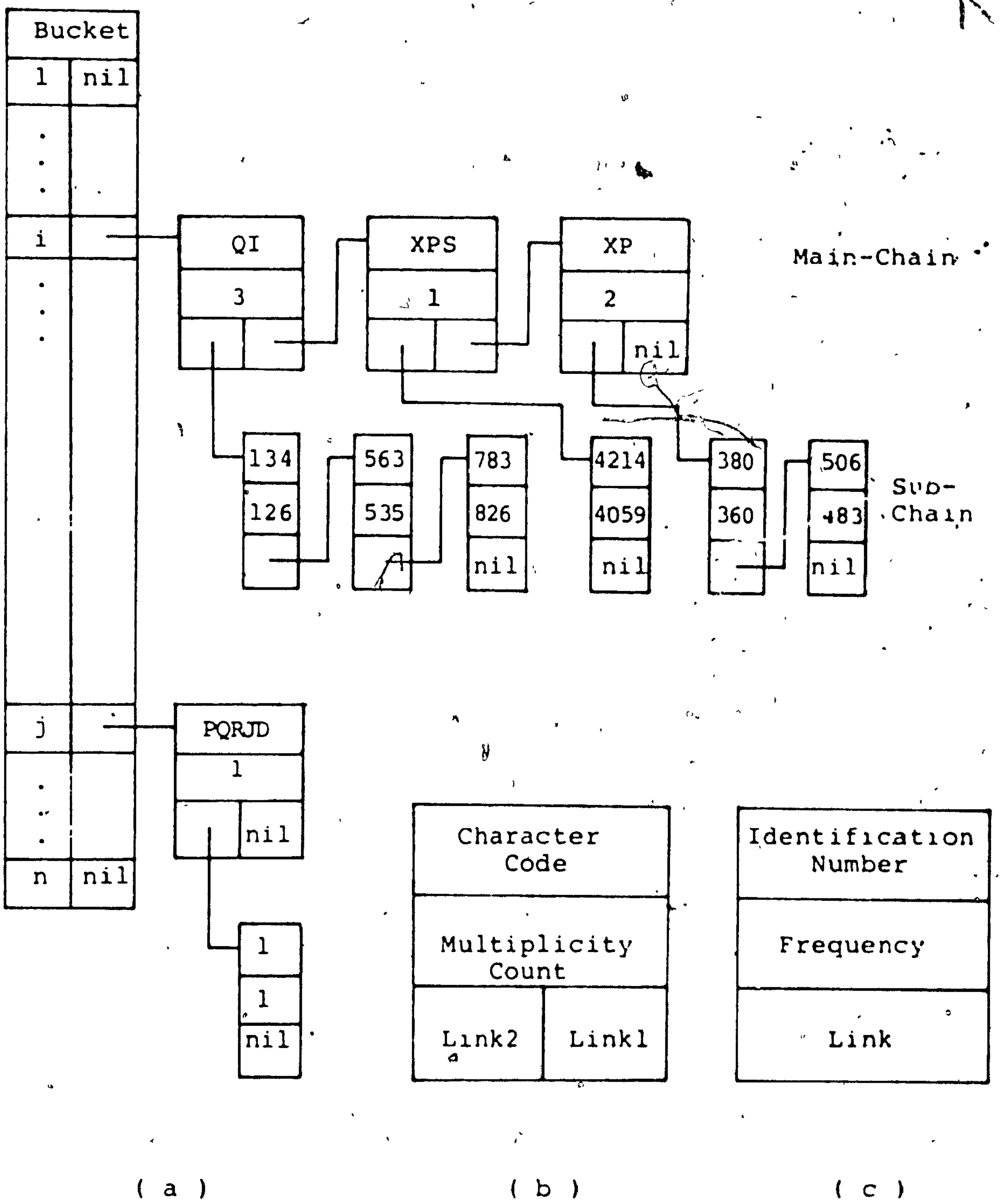


Figure 6 a. Structure of Tempqrary Hash Dictionary.
b. Main-Chain Node Structure.
c. Sub-Chain Node Structure.

end of chain. When the end of the chain is reached and the desired code is not found, information will be displayed saying that this is an invalid code. In case the desired character code has been found in the main chain, the existence of character collision need to be determined. If the value of character multiplicity is greater than one, or if there is a character encoding collision, a display technique is used to overcome this difficulty. A list of collision characters will be displayed on the screen, one can then choose the required character by specifying the order of the corresponding character, or by inputting a specified code to indicate that none of them is the proper one.

2.6. EDITING SYSTEM

2.6.1. Edit

In our CWP system, two types of editing system have been designed. One is the screen edit and the other file edit. In the screen editing system, the user communicates with the computer directly. The system is working fully interactively. The screen is divided into three parts: working space, monitor area, and collision display area. Fig.7 shows the arrangement of the screen display.. The top part (working space) is limited to 15 columns by 8 rows of Chinese characters. Under the user's control, the required Chinese characters will be printed out in this part of the screen. The bottom part (monitor area) is the code input area. The information for code matched, rejected or collision will be shown in this area. The middle part is the place for display of collision characters. Whenever a character collision is found during the code input, all collision characters will be displayed. After the order of collision character is specified by the user, collision characters will be erased immediately.

For the file editing system, the users communicate with the computer only during code input or the occurrence of character collisions. There are two separated routines in this system, which are code check and character display. The process is not the same as in the screen editing system where only one routine is used for both checking code and displaying character. When the code checking routine is running, two files, input code and code address, are produced. The display routine will retrieve the Chinese characters from the character data base according to the code address file, and will display characters on the screen (maximum size is 20 by 20 characters).. If any error occurs in the output, the user can modify the code address file directly or indirectly through the modification of the code file.

Both editing systems have their own advantages and disadvantages. The screen editing system is more convenient, but it has a relatively low input speed because of its fully interactive characteristic; the file editing system is faster in editing but it is more difficult to modify the code if an error occurs in the input file.

Both editing systems have been implemented on CYBER170/835 and VAX11/780. The display terminal used is a Tektronix 4027 and a hard copy can be obtained from Tektronix 4631 hard copy unit for CYBER170/835 and from Versatec for VAX11/780.

2.6.2. Character Generator and Editing Format

Our CWP system is capable of processing 4864 different Chinese characters and 128 ASCII codes. Each of them consists of a 50*50 binary matrix and is stored in the data base. Any one of them can be reproduced by different resolutions and different geometric sizes. Some examples for different resolutions are given in Fig.8. The required resolutions or sizes can be specified prior to the code input.

Following is a list of special formats in editing.

1. Specifying character size: 1 by 1 scale is used to incorporate 50 lines vertically on screen and 50 dots horizontally for each character. The actual size of character will depend on the resolution of the display equipment used. The scales can be specified before any code inputs. Rectangular character pattern can be produced by specifying different scales in the horizontal and vertical directions. Scales could be less than 1, so that the character will be shrunk, or greater than 1, so that the character will be enlarged.
2. To display Chinese characters: enter the input code directly according to the chosen input encoding method.
3. To display English letters: type sign "+" for displaying upper-case English letters and sign "-" for displaying

中文信息處理

中文信息處理

中文信息處理

中文信息處理

中文信息處理

(b)

lower-case letter.

For example:

to input upper-case letter "A": type "+a";

to input lower-case letter "z": type "-z".

4. To display special symbols: just type the symbol and followed with a blank.
5. Edit format for specifying new page, paragraph, new line, a blank and erasing: type sign "$" first and followed with "n", "p", "l", "b", and "e", respectively.
6. Exit: type 0.

3.1. DEFINITIONS

Radical

A Chinese character is usually composed of one or more strokes. A radical consists of a group of strokes, which may or may not be pronunciable, and may or may not be isolated from other parts of a character. The word "isolated" means that there is no stroke connecting one group of strokes and the other parts of a character. A radical is usually defined as a common set of strokes in characters. Sometimes, a character itself may be a radical.

An isolated radical is the radical which can be isolated and extracted from other parts of a character. A non-isolated radical can not be isolated and taken out from the other parts of a character.

Largest Radical

The largest radical is the one which has the largest number of strokes in a specific character. Though being a radical itself, the largest radical can be composed from smaller radicals.

Smallest Element

The smallest element is a group of strokes which is pronunciable and can be isolated from other parts of a character. A character cannot be seen as a smallest element. A smallest element itself may be composed from one or more radicals.

Divisible Character

A divisible character is a character consisting of several smallest elements and (or) isolated radicals.

Indivisible Character

A character consists of one radical or several nonisolated radicals. There is no smallest element in it.

Main Component (MC)

In divisible characters, the main component is the smallest element or radical(s), whose position, either the leftmost or topmost part of a character, depends on the structure of the character. In indivisible characters, the character itself is the main component.

Subordinate Component (SC)

Once the main component of a character has been taken out, all remaining parts form the subordinate component of a character.

Divisible Component

A divisible component consists of one or more isolated radicals.

Indivisible Component

An indivisible component consists of at least two nonisolated radicals.

Radical Code (RC)

A radical code is the representation of a radical. It is usually composed of the first phoneme of the pronunciation of a radical or the ideograph code of a radical.

Shape Code (ShC)

A shape code is a symbol which represents the composition of a Chinese character.

Phonetic Code (PC)

The first phoneme of pronunciation of a Chinese character is defined as its phonetic code.

Fig.9 shows the relationships among these definitions.

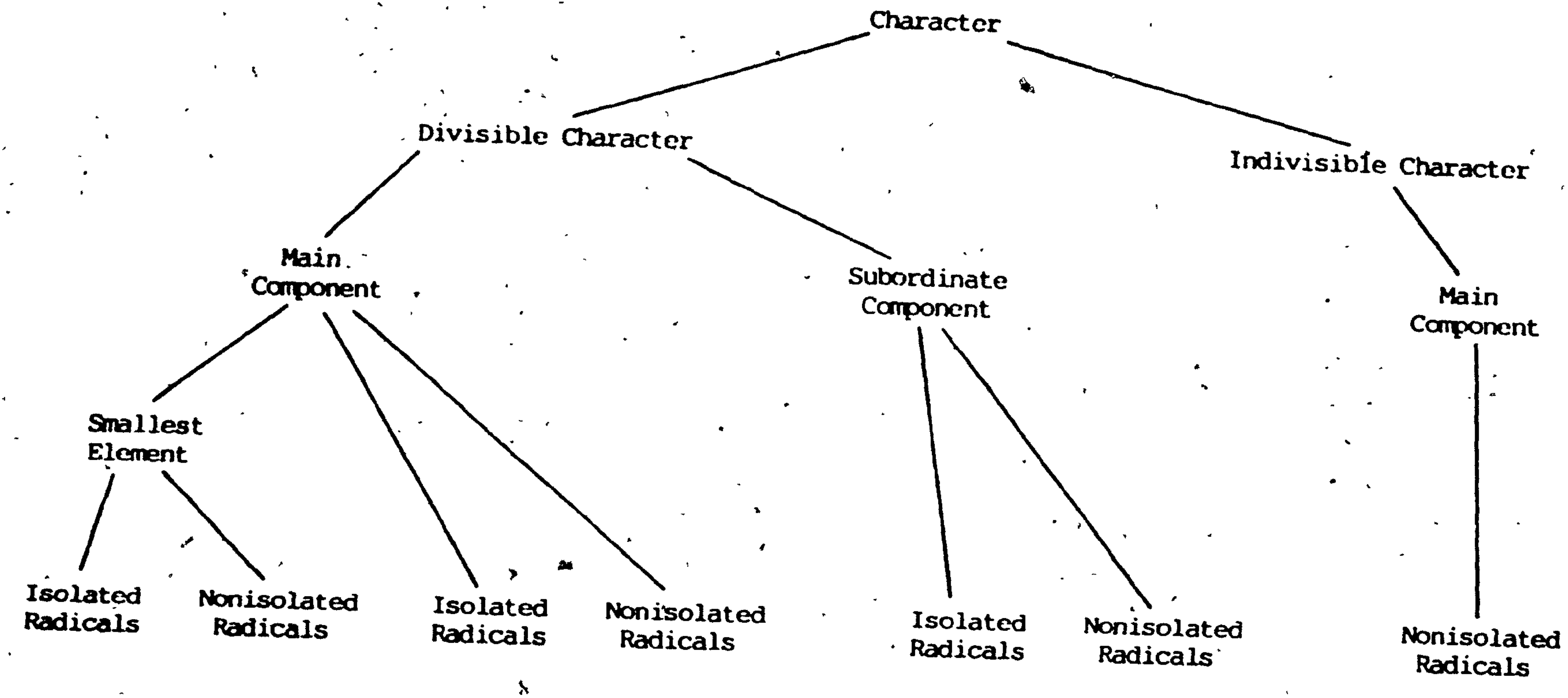


Figure 9. The Relationships Among the Different Components of a Character

3.2. INPUT CODE DESIGN

It has been recognized for a long time that, the criteria for a Chinese input encoding system are: a) how easy an input encoding is, and b) how small the chance of collision is. And there exists a conflict between these two criteria. Since the easier an encoding is, the more collisions will occur. Incorporated with this characteristic, five input encoding methods are discussed in this section. Among them, the Radical method in section 3.2.1 is the most fundamental one in our design. The others are the proper combinations of the Radical method with shape code, and/or with phonetic code. As the complexity in encoding increases, the collision in input decreases. A further discussion on the performance of these encoding methods will be given in the next chapter.

3.2.1. Radical Code Method

In the Radical Method, input keys are radical codes (RC). There are 202 radicals in this method and they have been extracted from 4864 most common Chinese charaters. 34 keys are used as the radical code to index the 202 radicals (refer to Table 1). A Chinese character can then be identified by

a. Different radicals.

b. The number of radicals.

c. The order of radical sequence.

The maximum number of radical codes in this method of inputting each individual Chinese character is 4.

In the rest of this section, the rules for character and component decomposition, and for input code construction are discussed in sequence. A Chinese character is encoded as follows. First, it is divided into two parts called the main component and the subordinate component. The components are then decomposed further into a sequence of radicals. Then, some of the radicals are properly chosen and converted into its corresponding codes as input information.

RULES FOR INPUT CONSTRUCTION

1. Character Decomposition

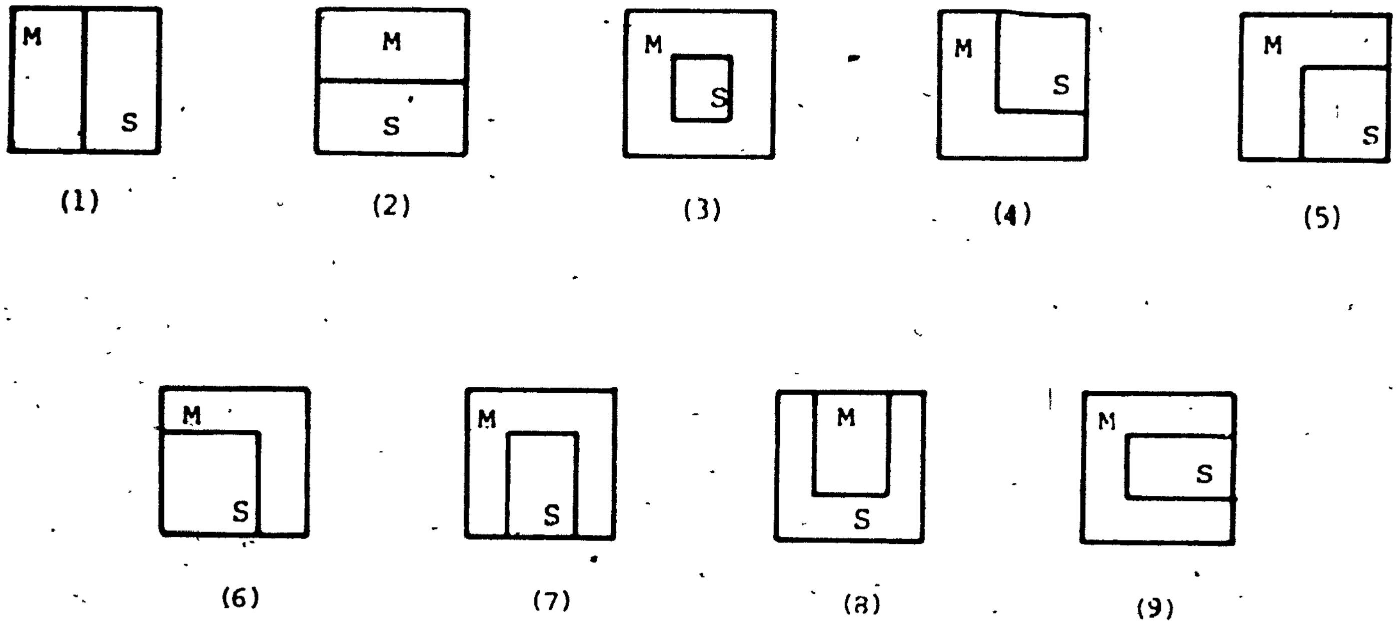


Figure 10. Structure for Divisible Chinese Characters

According to above sequence, in determining the main component of a character, one needs first to determine whether there exists a smallest element in the leftmost part or topmost. If yes, this smallest element is the main component. Otherwise, find whether there exists a largest radical in the leftmost part or topmost. If yes, the main component is this largest radical. If both cases (1) and (2) do not apply, then one has to use two or more isolated radicals as the main component. Once the main component of a character has been determined, the remaining part or strokes form the subordinate component of the character.

For example,

"班": MC is 王, SC is [illegible], case (1)
"贏": MC is 亡, SC is [illegible]; case (1)
"冥": MC is 冖, SC is 具, case (2)
"澈": MC is 氵, SC is [illegible], case (2)
"微": MC is 彳, SC is [illegible], case (3)
"[illegible]": MC is [illegible], SC is 欠, case (3)

B. Indivisible Characters

An indivisible character is itself a main component.

For examples,

"不": MC is 不.
"水": MC is 水.
"秀": MC is 秀.
"看": MC is 看.

" 變 ": MC is " 䜌", radicals are " 幺 ", " 言 " and " 幺 ";

SC is " 欠 ", radical is " 欠 ".

" 想 ": MC is " 相 ", radicals are " 木 " and " 目 ";

SC is " 心", radical is " 心".

Note that above examples apply to the divisible component only.

For an indivisible component, the decomposition rule still applies. But, special care should be taken when there exists a common stroke such that with this stroke, the previously found radical is the largest one; and with this stroke, the next largest possible radical also can be concluded from the remaining part of the component. In this situation, we let this common stroke belong to the next radical in order to avoid collision.

For example :

" 甥 ", its MC is " 生 ", radicals are " 𠂉 " and " 土 ";

its SC is " 男 ", radicals are " 田 " and " 力 ".

" 青 ", its MC is " 龶 ", radicals are " 一 " and " 土 ";

its SC is " 月 ", radical is " 月 ".

" 棘 ", its MC is " 朿 ", radicals are " 木 " and " 冂 ";

its SC is " 朿 ", radicals are " 木 " and " 冂 ".

" 拜 ", its MC is " 手 ", radicals are " 丿 " and " 扌 ";

its SC is " 手 ", radicals are " 二 " and " 丰 ".

" 未 ", its MC is " 未 ", radicals are " 一 " and " 木 ".

This is an example for an indivisible character.

If a character itself is a radical according to Table 1, then it should be decomposed further into radicals except itself if such a radical exists.

For example:

" 水 ", its MC is " 水", radicals are " 亅 ", " ㇇ " and " 乀 ".

" 力 ", its MC is " 力 ", radicals are " ㇆ " and " 丿 ".

" 王 ", its MC is " 王 ", radicals are " 一 " and " 土 ".

3. Constructing Radical Sequence

Having had in mind the above decomposition rules for character and component, we are now ready to construct a sequence of radicals decomposed from a given character.

(a). The radical decomposing sequence is conducted from left to right:

For example:

"斑" -- (王, 文, 王)

"纵" -- (糸, 人, 人)

(b). The sequence is conducted from top to bottom:

For example:

"喜" -- (士, 口, 䒑, 口)

"氣" -- (𠂉, 一, 乙, 夕, 一, 水)

"得" -- (丿, 亻, 日, 一, 寸)

"彦" -- (丶立, 厂, 丿, 丿, 丿)

(c). The sequence is conducted from outside to inside:

"由" -- (曰 , |)

(f). Interruption of stroke-order within a radical:

For example:

"東" -- (木 , 口)
"柬" -- (木 , 口 , 丷)
"夾" -- (一 , 人 , 人 , 人)
"秉" -- (丿 , 木 , ヨ)
"曹" -- (廿 , 曰 , 曰)
"喬" -- (土 , 人 , 人 , 口 , 口)

(g). The first and common stroke of two radicals will belong to the former radical:

For example:

"我" -- (丿 , 扌 , 乀 , 丶)

4. Input Code Construction

As stated in the beginning of this section, at most 4 radicals will be used in input coding after the decomposition and the construction sequence. The remaining part of this section will show how to extract the input radicals from the radicals sequence constructed as given before.

Case 1: The main component has only one radical, then the input radicals are selected as follows:

Input Information = FRMC (+ FRSC (+ SRSC (+ LRSC)))

Where FRMC: First Radical of Main Component;

FRSC: First Radical of Subordinate Component;

SRSC: Second Radical of Subordinate Component;

LRSC: Last Radical of Subordinate Component.

And the brackets used are for the special cases in which a character may consist of only the main component, or both main and subordinate components, while the subordinate component consists of one, or two, and or more radicals. For example:

"叭", its MC is " 口 ", radical is " 口 ";
its SC is " 八", radical is " 八",
input information = (口 , 八)

" 脂 ", its MC is " 月 ", radical is " 月 ";
its SC is " 旨 ", radicals are " 匕 " and " 日 ";
input information = (月 , 匕 , 日)

" 咬 ", its MC is " 口 ", radical is " 口 ";
its SC is " 交 ", radicals are " 亠 ", " 八 " and " 乂 ";
input information = (口 , 亠 , 八 , 乂)

" 腦 ", its MC is " 月 ", radical is " 月 ";
its SC is " 𡿺 ", radicals are " く ", " く ", " く ", " 丿 ", " 口 " and " 乂 ";
input information = (月 , く , く , 乂)

Case 2: The main component consists of two or more radicals and the character itself is the main component. This is the case for indivisible characters.

Input Information = FRMC + SRMC (+ TRMC (+ LRMC))

Where FRMC: First Radical of Main Component;

"朩";

its SC is "夅", radicals are "又" and "丰";

input information = (ノ , 朩 , 又 , 丰)

"彩": its MC is "采", radicals are "丿", "丷" and "朩";

its SC is "彡", radicals are "ノ", "ノ" and "ノ";

input information = (ノ , 朩 , ノ , ノ)

"露": its MC is "雨", radicals are "一", "冂" and "⺢";

its SC is "路", radicals are "口", "止", "夂" and "口";

input information = (一 , ⺢ , 口 ; 口)

Note: To avoid heavy collision in the input code the "丶" is considered only in the constructing sequence of following characters; 求, 甫, 术, 瓦; 犬, 氏, 兔, 鬼, 我, 弋, but neglected otherwise.

For example:

"球", its radicals are (王 , 一 , 氺)

"埔", its radicals are (土 , 一 , 冂 , 丰)

"求", its radicals are (一 , 氺 , 丶)

"甫", its radicals are (一 , 冂 , 丰 , 丶)

character. Classes (7) and (8) mean that a character can be divided into three or more isolated parts from left to right (or from top to bottom).

(f) Class (17) is applied to such divisible characters which are not included in classes (1) to (16).

(g) Both classes (18) and (19) apply to indivisible characters. However, class (18) is specially devoted to such indivisible characters where some of its radicals cross each other. And class (19) is applied to those indivisible characters where the radicals do not cross each other.

For example in class (18), the radicals of "万" are "丆" and "工", where "丆" crosses the other radical "工". For the character "生", radical "𠂉" crosses "工". In class (19), for example, the radicals of "看" are "丿", "龵" and "目". They do not cross each other. For the character "土", radical "一" does not cross with radical "工".

3.2.3. Phonetic Spelling and Radical Code Method

In this method, the information for inputting a Chinese character consists of phonetic spelling and radical code. The input codes used in this method have the following format,

Input Information = PS + RC

Where

PS : Suen's phonetic spelling symbols. The maximum number of letters of PS is 4.

RC : Radical codes extracted from character. The maximum number of letters of RC is 2.

PS is the phonetic spelling of a Chinese character. Depending on the phonetic system used, a Chinese character may have different phonetic spellings but the same pronunciation. For example, take character "親", its phonetic spelling in Suen's system is "chin", while in Pinyin system is "qin". In our input encoding system, Suen's phonetic system is adopted. In this phonetic system, the basic phonemes are represented by 23 English letters (i.e. the alphabet with the exception of "q", "v" and "z"), and a German letter um-laut "u:". There are 22 consonants or double consonants, 2 semi-vowels, 8 vowels or vowel types and 5 diphthongs (listed in Table 3). For the vocabulary used, a total of 402 different basic pronunciations (refer to Table 4) appear in this system.

Table 4. List of Phonemic Strings in Suen's Phonetic System

1 b a
2 b o
3 b ai
4 b ei
5 b au
6 b a n
7 b u n
8 b a ng
9 b u ng
10 b i
11 b i eh
12 b i au
13 b i e n
14 b i n
15 b i ng
16 b u
17 p a
18 p o
19 p ai
20 p ei
21 p au
22 p ou
23 p a n
24 p u n
25 p a ng
26 p u ng
27 p i
28 p i eh
29 p i au
30 p i e n
31 p i n
32 p i ng
33 p u
34 m a
35 m o
36 m uh
37 m ai
38 m ei
39 m au
40 m ou
41 m a n
42 m u n
43 m a ng
44 m u ng
45 m i
46 m i eh
47 m i au
48 m i e n
49 m i n
50 m i ng
51 m u
52 f a
53 f o
54 f ei
55 f ou
56 f a n
57 f u n
58 f a ng
59 f u ng
60 f u
61 f a
62 d uh
63 d ai
64 d ei
65 d au
66 d ou
67 d a n
68 d a ng
69 d u ng
70 d i
71 d i eh
72 d i au
73 d iu
74 d i e n
75 d i ng
76 d u
77 d w o
78 d w ei
79 d w a n
80 d oon
81 d oong
82 t a
83 t uh
84 t ai
85 t au
86 t ou
87 t a n
88 t a ng
89 t u ng
90 t i
91 t i eh
92 t i au
93 t i e n
94 t i ng
95 t u
96 t w o
97 t w ei
98 t w a n
99 t oon
100 t oong
101 n a
102 n uh
103 n ai
104 n ei
105 n au
106 n a n
107 n u n
108 n a ng
109 n u ng
110 n i
111 n i eh
112 n i au
113 n iu
114 n i e n
115 n i n
116 n i a ng
117 n i ng
118 n u
119 n w o
120 n w a n
121 n oong
122 n u:
123 n u:eh
124 l a
125 l o
126 l uh
127 l ai
128 l ei
129 l au
130 l ou
131 l a n
132 l a ng
133 l u ng
134 l i
135 l i a
136 l i eh
137 l i au
138 l iu
139 l i e n
140 l i n
141 l i a ng
142 l i ng
143 l u
144 l w o
145 l w a n
146 l oon
147 l oong
148 l u:
149 l u:eh
150 l u:e n
151 l u:n
152 g a
153 g uh
154 g ai
155 g ei
156 g au
157 g ou
158 g a n
159 g u n
160 g a ng
161 g u ng
162 g u
163 g w a
164 g w o
165 g w ai
166 g w ei
167 g w a n
168 g oon
169 g w a ng
170 g oong
171 k a
172 k uh
173 k ai
174 k au
175 k ou
176 k a n
177 k u n
178 k a ng
179 k u ng
180 k u
181 k w a
182 k w o
183 k w ai
184 k w ei
185 k w a n
186 k oon
187 k w a ng
188 k oong

(Table 4 continue)

189	h a
190	h uh
191	h ai
192	h ei
193	h au
194	h ou
195	h a n
196	h u n
197	h a ng
198	h u ng
199	h u
200	h w a
201	h w o
202	h w ai
203	h w ei
204	h w a n
205	H oon
206	h w a ng
207	h oong
208	j i
209	j i a
210	j i eh
211	j i, au
212	j iu
213	j i e n
214	j i n
215	j i a ng
216	j i ng
217	j u:
218	j u:eh
219	j u:e n
220	j u:n
221	j u:oong
222	chi
223	chi a
224	chi eh
225	chi au
226	chi u
227	chi e n
228	chi n
229	chi a ng
230	chi ng
231	chu:
232	chu: eh
233	chu: e n
234	chu: n
235	chu: oong
236	x i
237	x i a
238	x i eh
239	x i au
240	x iu
241	x i e n
242	x i n
243	x i a ng
244	x i ng
245	x u:
246	x u: eh
247	x u: e n
248	x u: n
249	x u: oong
250	rji
251	rja
252	rjuh
253	rjai
254	rjau
255	rjou
256	rja n
257	rju n
258	rja ng
259	rju ng
260	rju
261	rjw a
262	rjw o
263	rjw ai
264	rjw ei
265	rjw an
266	rjoon
267	rjw a ng
268	rjoong
269	roi
270	roa
271	rouh
272	roai
273	roau
274	roou
275	roa n
276	rou n
277	roa ng
278	rou ng
279	rou
280	row o
281	row ei
282	row a n
283	rooon
284	row a ng
285	rooong
286	shi
287	sha
288	shuh
289	shai
290	shei
291	shau
292	shou
293	sha n
294	shu n
295	sha ng
296	shu ng
297	shu
298	shw a
299	shw o
299	shw o
300	shw ai
301	shw ei
302	shw a n
303	shoon
304	shw a ng
305	r i
306	r uh
307	r au
308	r ou
309	r a n
310	r u n
311	r a ng
312	r u ng
313	r u
314	r w o
315	r w ei
316	r w a n
317	r oon
318	r oong
319	dsi
320	dsa
321	dsuh
322	dsai
323	dsei
324	dsau
325	dsou
326	dsa n
327	dsu n
328	dsa ng
329	dsu ng
330	dsu
331	dsw o
332	dsw ei
333	dsw a n
334	dsoon
335	dsoong
336	tsi
337	tsa
338	tsuh
339	tsai
340	tsau
341	tsou
342	tsa n
343	tsu n
344	tsa ng
345	tsu ng
346	tsu
347	tsw o
348	tsw ei
349	tsw a n
350	tsoon
351	tsoong
352	s i
353	s a
354	s uh
354	s uh
355	s ai
356	s au
357	s ou
358	s a n
359	s u n
360	s a ng
361	s u ng
362	s u
363	s w o
364	s w ei
365	s w a n
366	s oon
367	s oong
368	a
369	o
370	uh
371	ai
372	au
373	ou
374	a n
375	u n
376	a ng
377	er
378	y i
379	y a
380	y o
381	y eh
382	y ai
383	y au
384	y ou
385	y e n
386	y i n
387	y a ng
388	y i ng
389	w u
390	w a
391	w o
392	w ai
393	w ei
394	w a n
395	w u n
396	w a ng
397	w u ng
398	y u:
399	y u: eh
400	y u: e n
401	y u: n
402	y u: oong

		(shei)
Consonant+Vowel+Diphthong	謝	x+u+eh (xueh)
Consonant+Vowel+Vowel	胛	j+i+a (jia)
Consonant+Semi-vowel+Vowel	托	t+w+o (two)
Consonant+Semi-vowel+Vowel+ Nasal-consonant	況	k+w+a+ng (kwang)
Consonant+Semi-vowel+Diphthong	鬼	g+w+ei (gwei)

b). Starting with Semi-vowel:

Syllabic Structure	Smaple	Phonetic
Semi-vowel + Vowel	一	y+i (yi)
Semi-vowel + Diphthong	傜	y+au (yau)
Semi-vowel+Vowel+Nasal-consonant	偃	y+e+n (yen)
Semi-vowel+Vowel+Vowel Nasal-consonant	鉞	y+u+eh (yueh)

c). Starting with vowel or diphthong:

Syllabic Structure	Smaple	Phonetic
Vowel	嗄	a (a)
Vowel+Nasal-consonant	庵	a+n (an)
Diphthong	捱	ai (ai)

2. RADICAL CODE CONSTRUCTION

a). A maximum number of two radicals are extracted from the sequence of radicals of a character (see section 3.2.1).

The corresponding radical codes and the corresponding phonetic spelling are used to input a Chinese character.

b). The two radicals are so chosen such that the first radical at leftmost (or topmost) and the last radical at rightmost (or bottommost) of a character. The principle in the determination of radicals, is the same as given in section 3.2.1.

For example:

"得": the first radical is "丿" and last radical is "寸";

the corresponding codes are "p" and "ts".

"想": the first radical is "木" and last radical is "心";

the corresponding codes are "m" and "x".

"问": the first radical is "门" and last radical is "口";

the corresponding codes are "k" and "o".

"匕": the first radical is "丿" and last radical is "凵";

the corresponding codes are "p" and "au".

In using Suen's phonetic system, 31 different phonetic codes have been extracted from the first phoneme of 402 different basic pronunciations. These phonetic codes can be classified into four groups, i.e. those starting with (a) consonant, (b) semi-vowel, (c) vowel, and (d) diphthong. For example:

a). Starting with consonant:

"乞", its radicals are "ノ" and "乚",

radical codes are "p" and "au";

its phonetic spelling is "chi", phonetic code is "ch";

so, input keys = pau + ch.

b). Starting with semi-vowel:

"一", its radicals is "一", radical codes is "h";

its phonetic spelling is "yi", pnonetic code is "y";

so, input keys = h + y.

c). Starting with vowel:

"安", its radicals are "宀", and "女",

its radical codes are "b", and "n";

its phonetic spelling is "an", phonetic code is "a";

so, input keys = bn + a.

d). Starting with diphthong:

"捱", its radicals are "扌", "厂", "土" and "土",

radical oodes are "f", "rc", "t" and "t";

its phonetic spelling is "ai", pnonetic code is "ai";

so, input keys = frctt + ai.

code is "x";

its shape code is ":";

so, input keys = mqx + x + :.

3.3. SUMMARY OF THE INPUT ENCODING METHODS

A brief summary of the input encoding methods designed is given as follows.

1. Radical Code Method
   - a. Maxmum input keys are 4.
   - b. 34 keys are used to index the 202 radicals.
   - c. The users need to be familiar with the radicals and the decomposition rules.

2. Radical and Shape Code Method
   - a. Maximum input keys are 5.
   - b. 34 keys are used to index 202 radicals and additional 19 shape codes are used.
   - c. The users need to be familiar with the radicals and the decomposition rules; need to consider the structure of characters.

3. Phonetic Spelling and Radical Code Method
   - a. Maximum input keys are 5.
   - b. 34 keys are used to index 202 radicals.
   - c. The users need to be familiar with Suen's phonetic system; need to be familiar with the radicals.

4. Radical and Phonetic Code Method
   - a. Maximum input keys are 5.
   - b. 34 keys are used to index 202 radicals and additional

phonetic codes are used.

c. The users need to be familiar with the radicals and the decomposition rules; need to be familiar with Suen's phonetic system.

5. Radical, Phonetic and Shape Code Method

a. Maximum input keys are 6.

b. 34 keys are used to index 202 radicals and additional shape and phonetic code are used.

c. The users need to be familiar with the radicals and the decomposition rules; need to be familiar with Suen's phonetic system; and need to consider the structure of Chinese characters.

CHAPTER FOUR

PERFORMANCE EVALUATION AND SIMULATION

In this chapter, the performance of the Chinese word processing system is evaluated quantitatively from several aspects, such as, the input encoding methods (section 4.1), the keyboard function (section 4.2), and the access of data base (section 4.3). A systematic simulation is performed in section 4.4 as well in order to demonstrate the performance of our CWP system.

4.1. EVALUATION OF INPUT ENCODING METHODS

To any Chinese word processing system, the Chinese input method is an indispensable part. In other words, all systems must possess their own input encoding methods to process Chinese information.

Five encoding methods have been implemented in our system. They are Radical Code method, Phonetic Spelling and Radical Code method, Radical and Phonetic Code method, Radical and Shape Code method, and Radical, Phonetic and Shape Code method. They can be chosen by users at their own convenience. However, different methods will certainly affect both the design of the Chinese terminals and the arrangement of the internal code, and result in different input speeds and efficiencies.

The main advantages and disadvantages of these five encoding methods have already been discussed in the previous chapters. In the following, a quantitative comparison will be made for the aspects of

1). Number of symbols in input methods,

2). Number of codes used to key in a character,

3). And conflict code rate.

1. Number of symbols in input methods

It has been discussed that the five input methods are, to some extent, combinations of three basic encoding schemes: radical, shape and phonetic spelling. Among these three schemes, except radical scheme, there is a one-to-one correspondence between the codes and shapes and between the codes and pronunciations. From the users' point of view, however, they still need to memorize correct codes. It could be seen that, for radical scheme, each code generally does not just represent a single radical. For instance, radicals "月", "爿", "目", and "自" have the same code "a". And the radical code "ch" represents several radicals, such as, "乜", "匕", "乂", and "其".

In practice, the user not only has to learn every code, but also to remember the many symbols on each code. The larger is the number of codes, the greater is the effort required to memorize them. And as the number of radicals increases on each code, it will give the user additional burdens. We would like to estimate these efforts and burdens quantitatively.

Suppose we have two codes A and B, and we need to represent a symbol (radical, shape, pronunciation) 1. There are then two ways to represent 1, either on code A or on code B. If we assign two symbols 1 and 2 to the codes A and B, then there are 2^2 ways of doing it. In general, let n_c denote the number of codes and n_s the number of symbols, if we have n_c codes to represent n_s symbols, the total variation is n_c^{ns}. From such large variation, there is only one correct way, and the effort to memorize the correct one can be measured by

$$E = \log_2 n_c^{n_s} = n_s \log_2 n_c \qquad (4.1)$$

because if there are two (2^1), four (2^2), or eight (2^3) possibilities, one needs to choose once, twice, or three times respectively.

For a particular encoding method the total efforts made could be estimated as the summation of efforts of the individual corresponding schemes, i.e.,

$$E_t = \sum_i E_i \qquad (4.2)$$

For example, in Radical, Phonetic and Shape Code method, the total effort is equal to the summation of efforts of radical scheme, phonetic scheme, and shape scheme. Table 5 lists the results for those five methods. From Table 5, we note that the effort made to memorize the correct code in Radical scheme is so large compared with that of Phonetic and Shape schemes, that the total efforts for each kind of the encoding methods are all in the same magnitude. Following gives the list of the five encoding methods in the increasing order of their total efforts.

Radical Code method,

Radical and Shape Code method,

Radical and Phonetic Code method,

Phonetic and Radical Code method,

Radical, Phonetic and Shape Code method.

Table 5 Comparison of the Difficulties in Using the Various Input Encoding Methods.

Input Method	Number of Symbols (n_s)	Number of Codes (n_c)	Total Effort (E_t)
RC	202	34	1.03×10^3
RSC	202 +19	34 + 19	1.11×10^3
PRC	39 + 202	39 + 34	1.24×10^3
RPC	202 + 31	34 + 31	1.18×10^3
RPSC	202 + 31 + 19	34 + 31 + 19	1.26×10^3

This sequence does not take into the consideration of initial familiarity of codes. For example, phonetic spelling might be familiar by almost all users. Therefore, the above order might be different if this factor is included in calculating the efforts involved.

Ordinarily the code designer would prefer to represent symbols of similar kinds by the same code to facilitate memory. Our analysis does not take this into account. Hence our value E_t probably overestimates the actual difficulties in learning each of the methods.

2. Number of codes to key in a character

It is obvious that the number of codes used to key in a character is another important aspect affecting the input speed. The fewer the number of codes is used, the faster the input is. There are two aspects related to this subject, the maximum code length and the average code length. They are defined as follows:

Maximum code length -- the maximum number of codes per character.

average code length -- total number of codes for all characters in data base / total number of characters in data base.

Table 6 gives a summary of those five encoding methods based on the above two criteria. It could be seen that the maximum code length for these five encoding methods varies from 4 to 6, and the average code length for them is within the range of 3.54 to 5.54. The Radical Code method has the shortest maximum and average code lengths which are 4 and 3.54 respectively, while Radical, Phonetic and Shape Code method has the longest maximum and average code lengths, which are 6 and 5.54 respectively.

3. Conflict code rate

Before we proceed to this subject, some terminologies need to be defined first.

(1) Collision frequency -- for each particular character, the number of characters which have the same code but different shapes.

(2) Collision number -- the total number of characters which have the same collision frequency.

(3) Conflict code rate -- sum of characters having value of collision frequency greater than one / total number of characters in the data base.

To explain these definitions, let us look at the following examples.

Assume a character set,

(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)

Table 6. Encoding Performance Evaluation in Terms of Code Length.

No. of keys / Code length / Method	1	2	3	4	5	6	Max. Code Length	Ave. Code Length
RC	1	438	1340	3085	-	-	4	3.54
RSC	-	1	438	1340	3085	-	5	4.54
PRC	-	-	74	2266	2013	511	6	4.61
RPC	-	1	438	1340	3085	-	5	4.54
RPSC	-	-	1	438	1340	3085	6	5.54

has the corresponding codes

$(c_0, c_1, c_1, c_2, c_3, c_1, c_2, c_4, c_3, c_5)$

i.e. a_1, a_2, and a_5 have the same code, a_3 and a_6 have the same code, and so on. Then by the definitions we may say that, characters a_1, a_2 and a_5 have the collision frequency of 3, characters a_3 and a_6, and a_4 and a_8 both have the collision frequency of 2, and characters a_0, a_7 and a_{10} have the collision frequency of 1. The collision number is 3 for characters having a collision frequency of 3, the collision number is 4 for characters having a collision frequency of 2, and the collision number is 3 for characters having a collision frequency of 1. Finally, the conflict code rate is equal to (3+4)/10 = 0.7.

It is obvious that the higher the value in collision number is, the more chances in the occurrence of collision there are. And as stated in chapter 2, whenever a collision occurs in input, a display of collision characters is necessary in order to allow the operator to pick up the right one from them. This implies that a smaller value of collision frequency will give rise a shorter time in displaying those collision characters. When the value of collision frequency equals to 1, there will be no collision. Hence, it is not necessary to display the characters at all.

Table 7 shows the statistical results of the above criteria for the different encoding methods. It can be seen that the Radical, Phonetic and Shape Code method has very

Table 7. Statistics of Encoding Collision in Input Methods.

Collision Number / Collision Frequency Method	1	2	3	4	5	Conflict Code Rate
RC	4265	514	57	28	-	12.3 %
RSC	4573	268	15	8	-	6.0 %
PRC	4404	418	42	-	-	9.5 %
RPC	4749	112	3	-	-	2.4 %
RPSC	4798	66	-	-	-	1.4 %

few collisions and its values of collision frequency are less than 2. Hence, it has a low conflict code rate of only 1.4%. The Radical Code method and Phonetic Spelling and Radical Code method have more severe collision problems in encoding, and their maximum values of collision frequency are 4 and 3, respectively. Therefore, higher conflict code rates are expected from them, 12.3% and 9.5%, respectively. It can also be seen from the Table 7 that, though the Radical and Shape Code method has a larger value of collision frequency (=4) than that of Phonetic and Radical Code method (=3), its conflict code rate is almost half of the latter. And Table 7 also shows that the Radical and Phonetic Code method has the second least value of conflict code rate (2.4%), and its maximum value of collision frequency is 3.

By contrasting Table 6 with Table 7, it is interesting to note that a method having lower conflict code rate will have a higher value in maximum and average codes. This could be understood as a fact that price must be paid if we want to reduce the conflict code rate.

Overall speaking, we can conclude that the Radical and Phonetic Code method and Radical and Shape Code method are the best among these five encoding methods. They have the lower values of both collision frequency and collision number, therefore a lower value of conflict code rate, excluding Radical, Phonetic and Shape Code method. Their maximum and average code lengths are within an acceptable range. At the same time, efforts made to memorize their coding methods are less than the others, excluding Radical Code method.

This conclusion can be best explained in Table 8 in which the normalized scores from 0 to 1 (the best to the worst), are assigned to evaluate performances of the five encoding methods according to the criteria listed in the Tables 5 to 7. The normalized scores are defined as follows:

$$(P_{normal})_{ij} = P_{ij}/(P_i)_{max}$$

where P_{ij} is the value of the i^{th} performance index (one of the performance criteria) for j^{th} encoding method (one of the encoding methods) and $(P_i)_{max}$ is the maximum value for i^{th} criteria. The total score is just the summation of all the normalized scores for one encoding method, i.e.,

$$(P_j)_{total} = \sum_i (P_{normal})_{ij}$$

A method having lower total scores can be considered as the one having better performances. It can be seen that RPC method and RSC method have better performance in terms of their lower total scores of 2.784 and 3.021 respectively.

Table 8 Summary of the Evaluation Made on the Input Design.

Score / Normalized Criteria / Method	Total Efforts E_t	Max. Length	Ave. Length	Conflict Code Rate	Total Score
RC	0.817	0.667	0.639	1	3.123
RSC	0.881	0.833	0.819	0.488	3.021
PRC	0.984	1	0.832	0.772	3.588
RPC	0.937	0.833	0.819	0.195	2.784
RPSC	1	1	1	0.114	3.114

4.2. KEYBOARD FUNCTION EVALUATION

As mentioned in Chapter two, a standard English keyboard is adopted in our keyboard design. To incorporate the characteristics of our encoding methods, some modifications have been made on the standard English keyboard as shown in Fig.3. In this section, based on a simple mathematic model developed by Lo S.Y.[16], a quantitative measurement has been made on the so designed keyboard for the different encoding methods. The analysis here is concentrated on the following objectives:

1). Learning stage.

2). Typing stage.

3). Typing speed.

1. Learning stage.

Two aspects which are concerned in Lo's model about learning stage. One is the difficulty faced by the beginner; the other is the difficulty faced by the experienced typist. Let L_1 and L_2 represent the difficulties faced by the beginner and experienced typist respectively, n_k the number of keys used on the keyboard, then the difficulties faced by the beginner can be evaluated as follows,

$$L_1 = \log_2 n_k \quad (4.3)$$

and difficulties faced by the experienced typist is

$$L_2 = \log_2(n_k!) \qquad (4.4)$$

These two models could be interpreted as follows.

For a beginner, to learn typing means to learn to decide which key to type every time. If one has to choose a correct key from two keys, he needs to make only one choice. If one chooses the correct key from four keys, it is necessary to make 2 choices. From eight keys to get the correct one, one has to make 3 choices. In general from n_k, one needs to make

$$L_1 = \log_2 n_k$$

choices. The keyboard based on radical and phonetic code encoding method, contains 43 keys, with $L_1 = 5.43$. For the standard English keyboard, which has 26 keys for the Latin alphabet, the difficulty to learn is $L_1 = 4.7$. All our other input methods can be calculated in such a fashion. Their values are listed in the Tables 9 and 10.

For someone who wants to become an experienced typist he must learn all key positions by heart. To assign one symbol on one key, there is only one way. To assign two symbols using two keys, there are 2! ways. To assign three symbols using three keys or in general n_k symbols using n_k keys there are 3! and $n_k!$ ways respectively. To obtain all n_k symbols correctly from n_k keys by a random procedure, the chance of getting it right is only $1/n_k!$. Hence, the difficulty of learning to master the whole keyboard can be

Table 9. List of Parameter Values for Various Input Methods.

Input Method	No. of Keys n_k	No. of Symbols n_s	No. of Codes per character k_s
RC	34	202	3.54
RSC	53	221	4.54
PRC	43	241	4.61
RPC	37	205	4.54
RPSC	56	224	5.54
English	26	52	5.00

Table 10 Comparison of the Difficulties Occurred in Various Input Methods in Keyboard Design.

Input Method	Keyboard		No. of Steps in Analysis in Typing		Typing Speed $1/(k_s n_k)$
	Beginner $L_1 = \log_2 n_k$	Expert $L_2 = \log_2 n_k!$	P	$r_p = \frac{P}{P(\text{English})}$	
RC	5.09	127.8	47.28	2.0	8.31×10^{-3}
RSC	5.73	231.3	71.93	3.05	4.16×10^{-3}
PRC	5.43	175.3	85.45	3.62	5.04×10^{-3}
RPC	5.21	143.3	78.17	3.31	5.95×10^{-3}
RPSC	5.81	248.7	102.82	4.36	3.22×10^{-3}
English	4.70	88.4	23.6	1.00	7.69×10^{-3}

measured by

$$L_2 = \log_2(n_k!)$$

We have listed the values of L_2 for the different encoding methods in Table 10. For a 43 key keyboard based on radical and phonetic code, L_2 = 175.34 while for a 26 key English keyboard, L_2 = 89.9. Hence, the mental effort required to memorize the large keyboard is needed twice more than that for an English keyboard.

2. The typing stage

During our encoding, the brain must perform a detailed analysis on the Chinese character after the image of the printed word is received in order to match the subcomponent or phoneme with the symbols on the keys. Differently, for English encoding, the printed alphabet matches exactly with symbols on the keys.

The stages of the different encoding schemes from receiving a Chinese character to type in its code into computer can be clearly illustrated as follows:

Phonetic scheme:

Chinese character -- sound -- phonetic code -- typing

Radical scheme:

Chinese character -- radical -- sound -- radical code -- typing.

Shape scheme:

Chinese character -- shape -- shape code -- typing

which can be compared with the typing stages for English,

English letter -- typing

Every stage requires some brain's dynamic activities. For example, for the phonetic scheme, to recognize the characters one has

$$\log_2 n_w \tag{4.5}$$

choices where n_w is the total number of Chinese characters. For sounds we have

$$\log_2 n_s \tag{4.6}$$

where n_s is the number of different sounds in Chinese. And for each sound there are k_w ambiguities. To analyze these homonyms we have

$$\log_2 k_w \tag{4.7}$$

choices. Finally for the typing we have

$$l_p \log_2 n_{pk}$$

where n_{pk} and l_p are the number of keys and the average length of phonetic codes respectively in each phonetic coded Chinese character.

Therefore typing in Chinese characters in the phonetic scheme requires the total effort of P_p, which is the sum of all these calculations:

$$P_p = \log_2 n_w + \log_2 n_s + \log_2 k_w + l_p \log_2 n_{pk} \qquad (4.8)$$

Along the same reasoning, the total effort P_r to type in Chinese characters in the radical scheme is

$$P_r = \log_2 n_w + \log_2 n_r + \log_2 n_s + \log_2 n_{rc} + l_r \log_2 n_{rk} \qquad (4.9)$$

where n_w, n_r, n_s, n_{rc}, l_r and n_{rk} are the number of Chinese characters, the number of radicals, the number of sounds, the number of radical codes, the average length of radical codes per radical coded Chinese character, and the number of keys used in Radical method, respectively.

Similarly the total effort P_s to type in Chinese characters in the shape scheme is,

$$P_s = \log_2 n_w + \log_2 n_s + \log_2 n_{sc} + l_s \log_2 n_{sk} \qquad (4.10)$$

where n_w, n_s, n_{sc}, l_s and n_{sk} are the number of Chinese characters, the number of shapes, the number of shape codes, the average length of shape codes per shape coded Chinese character, and the number of keys used in Shape method, respectively.

While typing texts in English which has only 26 letters in the alphabet, searching through the file and making a correct match need at most 4.8 times ($2^{4.8}>26$) of comparison. For an English word which has an average length of approximately five letters, then the total effort can be evaluated as

$$P_E = 5*\log_2 26 = 24 \tag{4.11}$$

The numerical indications obtained by applying the above equations for each of the encoding methods are summarized in Table 10.

While there might be some other rules and inherent regulations in the scheme, which are not included in the above estimate. We believe the estimate has nevertheless included the most important features of these schemes. The additional rules and regulations would only make the P value greater.

3. Speed of typing

Normally one expects that the typing speed V is inversely proportional to the number of codes per character k_s:

$$V = 1/k_s \tag{4.12}$$

There are other factors, such as, the size of the keyboard, the typing speed for ordinary people, trained people and professional, the labor intensity, and so on, which also affect the typing speed.

If only the number of codes per character k_s and the number of keys n_k are considered in our measurement, then V is a function of two variables, i.e.,

$$V = V(k_s, n_k) \tag{4.13}$$

As we know that, the typing speed will decrease as the number of keys increases. We can conclude that

$$V = 1/(k_s n_k) \qquad (4.14)$$

Table 10 summarizes the values of eq.(4.14) for the keyboards of the different input encoding methods. We have also compared the values with those of English. In decreasing order of difficulties they are:

Radical, Phonetic and Shape Code method,

Radical and Shape Code method,

Phonetic Spelling and Radical Code method,

Radical and Phonetic Code method,

Radical Code method.

4.3. PERFORMANCE OF THE ACCESS METHOD

As mentioned in Chapter two, a hash table has been built up in order to shorten the data access time, where a division hash function was used and a chaining scheme was applied to overcome difficulties arisen in hashing collision. The hash indexing method used may be briefly restated as follows:

Let x_i be one kind of codes (radical, or phonetic, or shape), then, for a particular encoding method, a character code may have the following form

$$y_c = x_1 x_2 x_3 \ldots x_n$$

where n is the number of codes per character, for different characters the value of n may be different, and for the same character but different methods the value of n may be varied also. x_i could have one of the forms from "A" to "Z" in phonetic and radical schemes or one of the forms from the ASCII special characters as in shape scheme.

When such alphabet based code is received by the computer, it is transformed into a numerical based code as follows,

$$y_n = O(x_1) \oplus O(x_2) \oplus \cdots \oplus O(x_n)$$

Where $O(x_i)$ is the numerical value of the order of character x_i, most computer languages have this function defined. $\oplus$ is an operation defined as $(a \oplus b)=ab$. For example, $(13 \oplus 51)=1351$.

Thus the obtained numerical based code can then be applied to the division hash function, such that

$$y'_n = H_d(y_n)$$

where H_d is the division hash function defined as

$$H_d(x) = x \text{ MOD } m$$

Here MOD means modulus operation, i.e. the right hand side of the equation gives the remainder from integer division of x by m. And the value of $H_d(x)$ is in the range

$$0 \le H_d(x) \le m-1$$

The hash table is composed of m buckets which correspond to the maximum value of $H_d(x)$. Each bucket consists of several records, and one of them is the absolute address of Chinese characters in the character pattern data base.

For a given Chinese character, when an encoding method is chosen, its $H_d(x)$ value could be determined as above. The absolute address of its character pattern could be found in the bucket H_d. In most cases, there are several different characters with the same H_d value. This is called hash collision. A chaining scheme is applied to solve such a problem. It just simply chains them in the order they appear. When a character is hashed to such a bucket, the code has to be compared one by one till the right character has been found or the chain has been exhausted, that is, the input code could not be found in the chain, and hence it will be rejected, subsequently.

There is another kind of collision called encoding collision, which has been discussed in Chapter 2. We will not discuss it in this section because it does not affect the performance of the hash function.

Let us go back to the hash collision. It is not difficult to understand that the more collisions are on one bucket, the longer the chaining is, and a longer time will be used in the comparison process. As is well known from any standard data structure text, the number of hash collisions in a division function could be reduced or even eliminated by a proper selection of the value of the prime number m.

It is easily understood that if m is equal to the maximum value of y_n, there will be no hash collision at all. However, the space occupied by a hash table will have an unreasonable size with most of them unused. The objective in our design is to find the best value m so that it will confine both time complexity and space utilization close to 1. The time complexity and space utilization are defined as,

Time complexity -- average time used to match a character code in terms of the number of comparisons.

Space utilization -- space used /space occupied or number of buckets having codes / total number of buckets in the table.

Since the data base is static rather than dynamic, i.e., there are no changes in data during the execution of our CWP system, an optimal value m could be obtained through an experimental search. Figures 11a to 11e give the results for such a searching effort for time complexity and space utilization versus the sizes of the hash table for Radical Code method, Radical and Shape Code method, Phonetic and Radical Code method, Radical, Phonetic and Shape Code method, respectively.

As shown in Fig.11, when the prime number is less than 5000, which is close to the value of the total number of characters, the value of time complexity is reduced drastically with the increase of the value of prime number (bucket size), and is kept minimal decline for the values of prime number greater than 5000. The value of space utilization is almost linearly proportional to the value of the prime number. The value of the prime number at the intersection point of time complexity and space utilization is very close to the total number of characters. Comparing the results shown in Fig.11a to 11e, it can be seen that the tendency of time complexity and space utilization versus prime number and the value of prime number at the intersection point differ slightly from one encoding method to another. Table 11 lists the values of the prime number at the intersection point and the value of the corresponding time complexity and space utilization, and the maximum chain

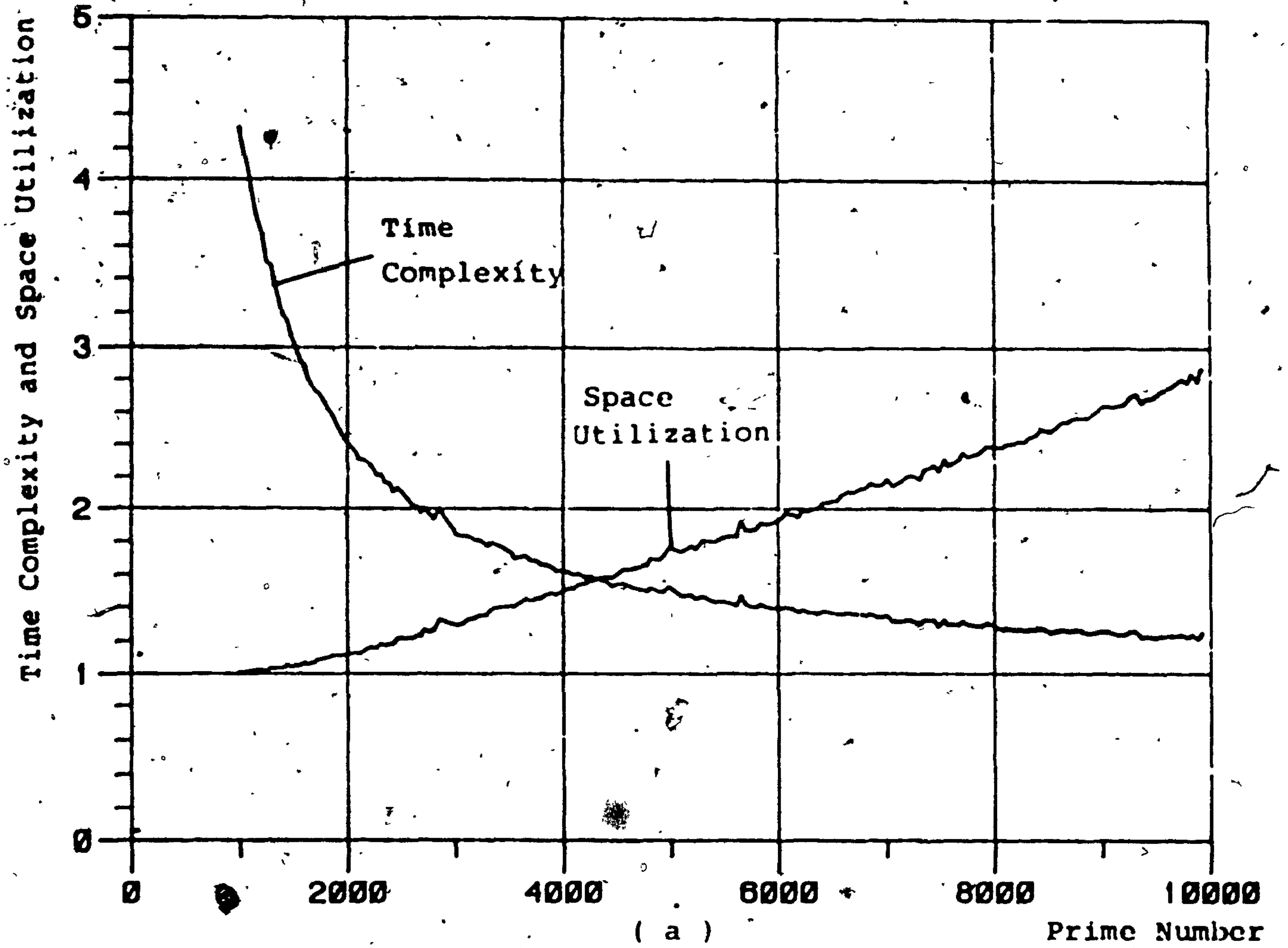
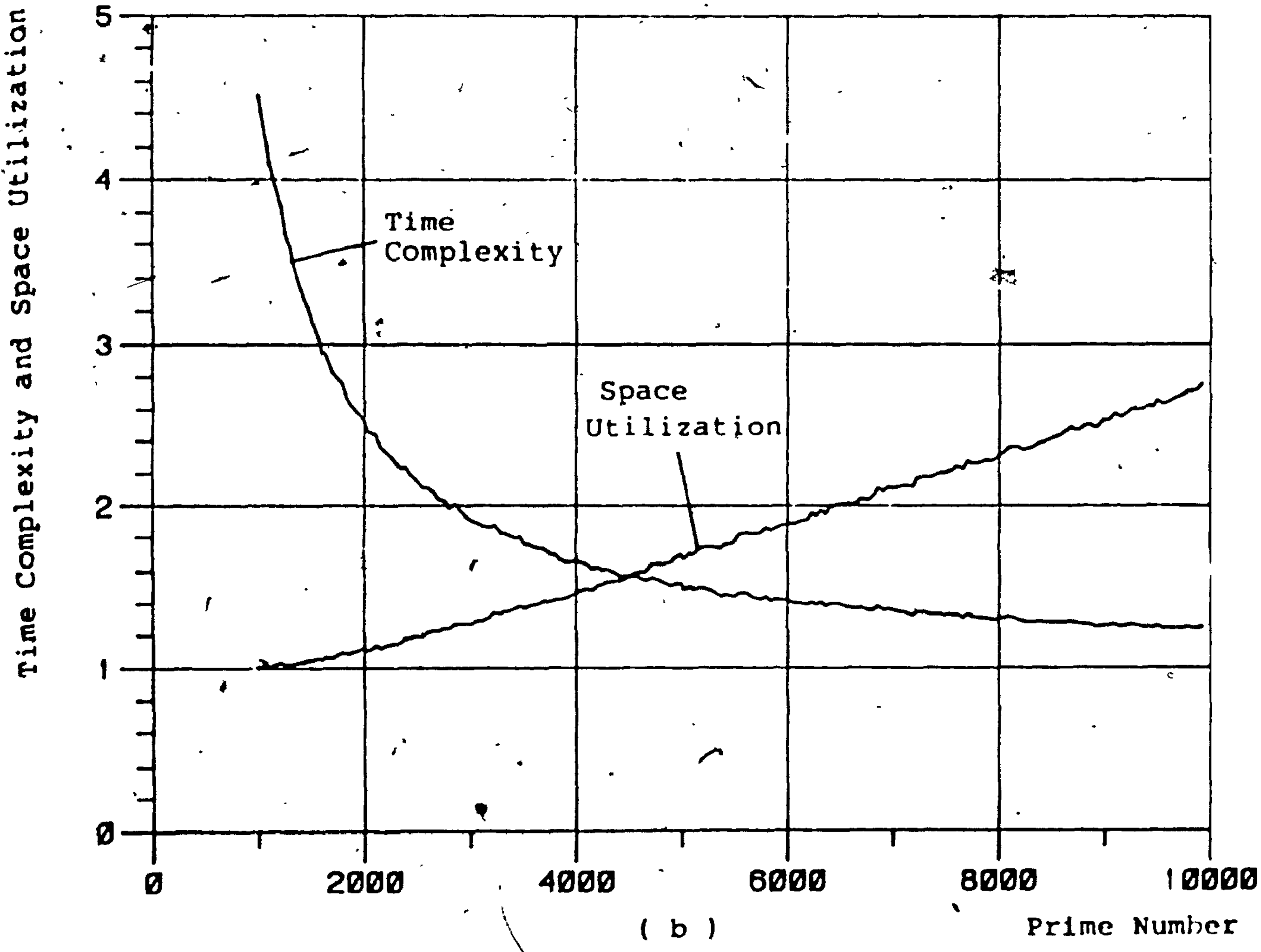
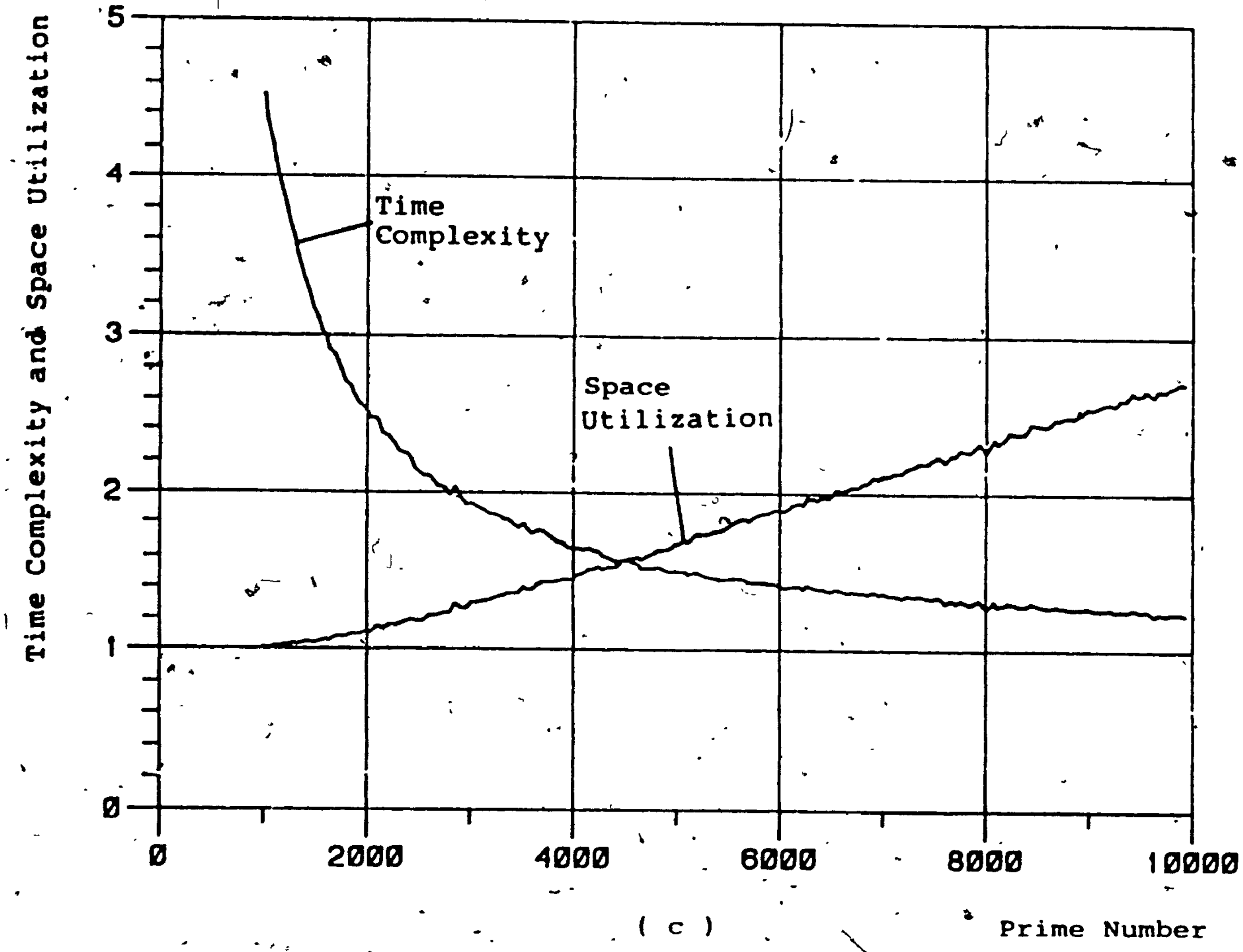


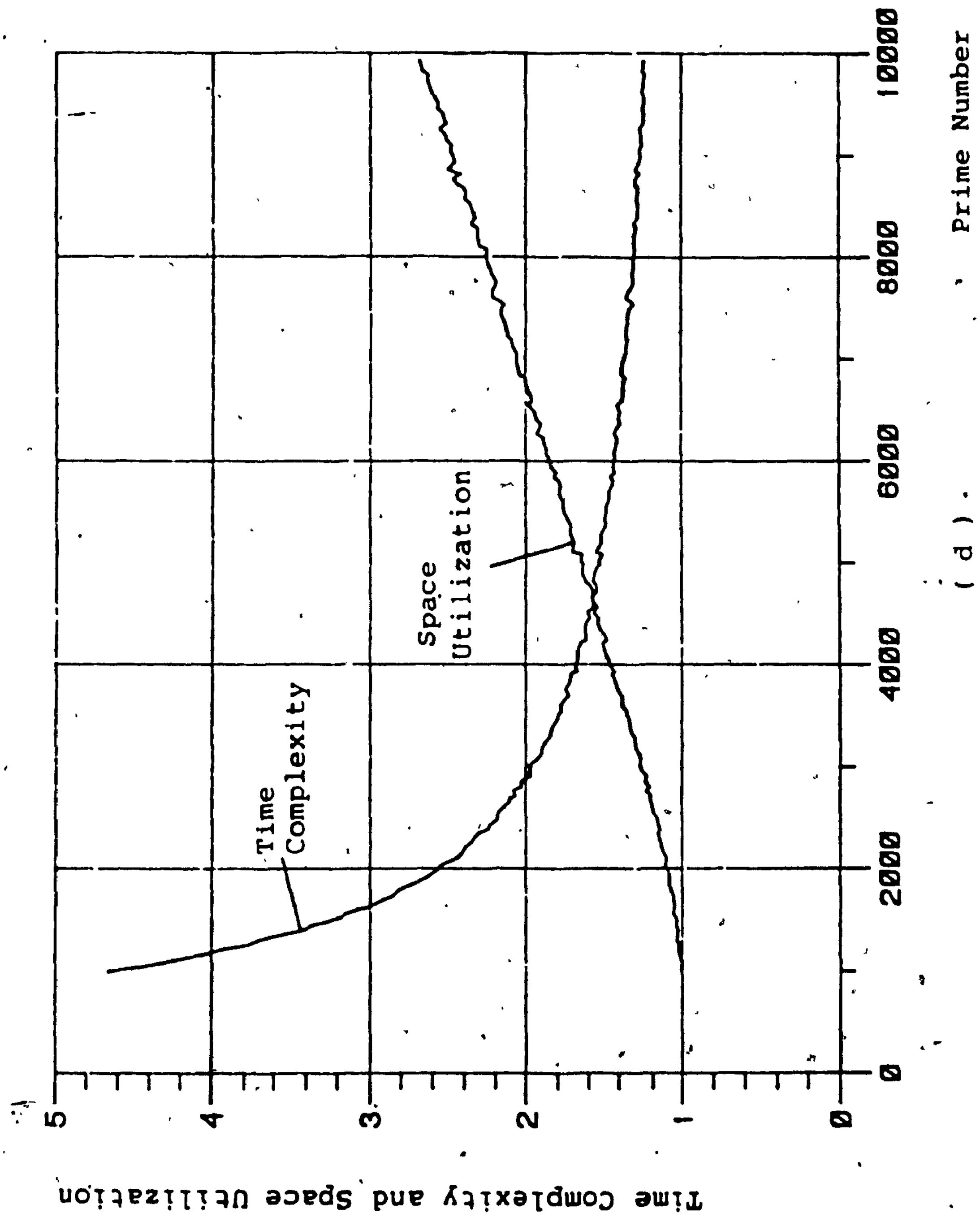
Figure 11. Time Complexity and Space Utilization of Hash Function for (a) Radical method (b) Radical and Shape method (c) Radical and Phonetic method (d) Phonetic and Radical method (e) Radical, Phonetic and Shape method.


Time Complexity and Space Utilization
5
4
3
2
1
Ø
Ø
2ØØØ
4ØØØ
6ØØØ
8ØØØ
1ØØØØ
Prime Number
Time
Complexity
Space
Utilization

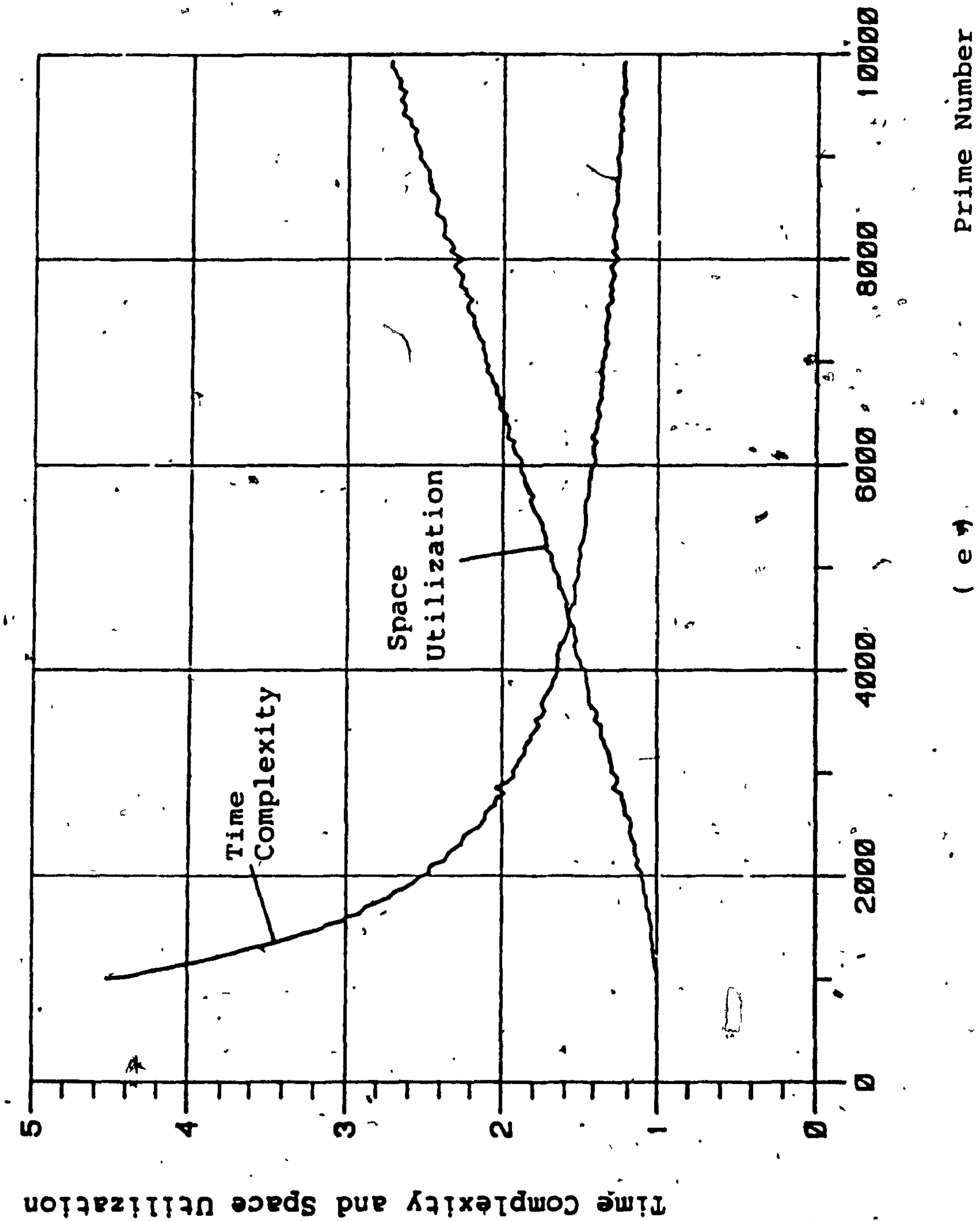
(b)


Time Complexity and Space Utilization
5
4
3
2
1
Ø
Ø
2ØØØ
4ØØØ
6ØØØ
8ØØØ
1ØØØØ
Prime Number
Time
Complexity
Space
Utilization

(c)


Time Complexity and Space Utilization
5
4
3
2
1
0
0
2000
4000
6000
8000
10000
Prime Number
Time Complexity
Space Utilization

(d)


Time Complexity and Space Utilization
0
1
2
3
4
5
0
2000
4000
6000
8000
10000
Prime Number
Time Complexity
Space Utilization

(e)

Table 11. Optimal Bucket Size and Its Corresponding Time Complexity & Space Utilization

Method	Prime Number (Bucket Size)	Max. Chain Length	Time Complexity	Space Utilization
RC	4337	6	1.55	1.56
RSC	4447	6	1.57	1.55
PRC	4663	5	1.54	1.59
RPC	4723	7	1.55	1.57
RPSC	4751	6	1.55	1.58

length at that particular value of prime number is also given.

4.4. Simulation of the Chinese Character I/O Processing System

In this section, the simulation of the Chinese character I/O processing system is discussed. Fig.12 gives a detailed flowchart for the Chinese word processing system. There are two phases in this processing routine. One is the preprocessing phase, the other is the processing phase.

As shown in Fig.12, when the system starts an execution, what should be done first is to determine an encoding method which is going to be used by entering the proper code into the system. A following procedure is then taken up to calculate the corresponding optimal size of the hash dictionary. Having determined the size of the hash dictionary, the processing routine proceeds further to establish the temporary hash dictionary using the conversion dictionary. Up to now the preprocessing phase has been completed.

Subsequently, the second phase - information processing starts.

The second processing routine first reads the character code from the keyboard and displays it on the screen. As stated in the last section, the entered character code is then converted to its numerical equivalent. A hash division function can then be

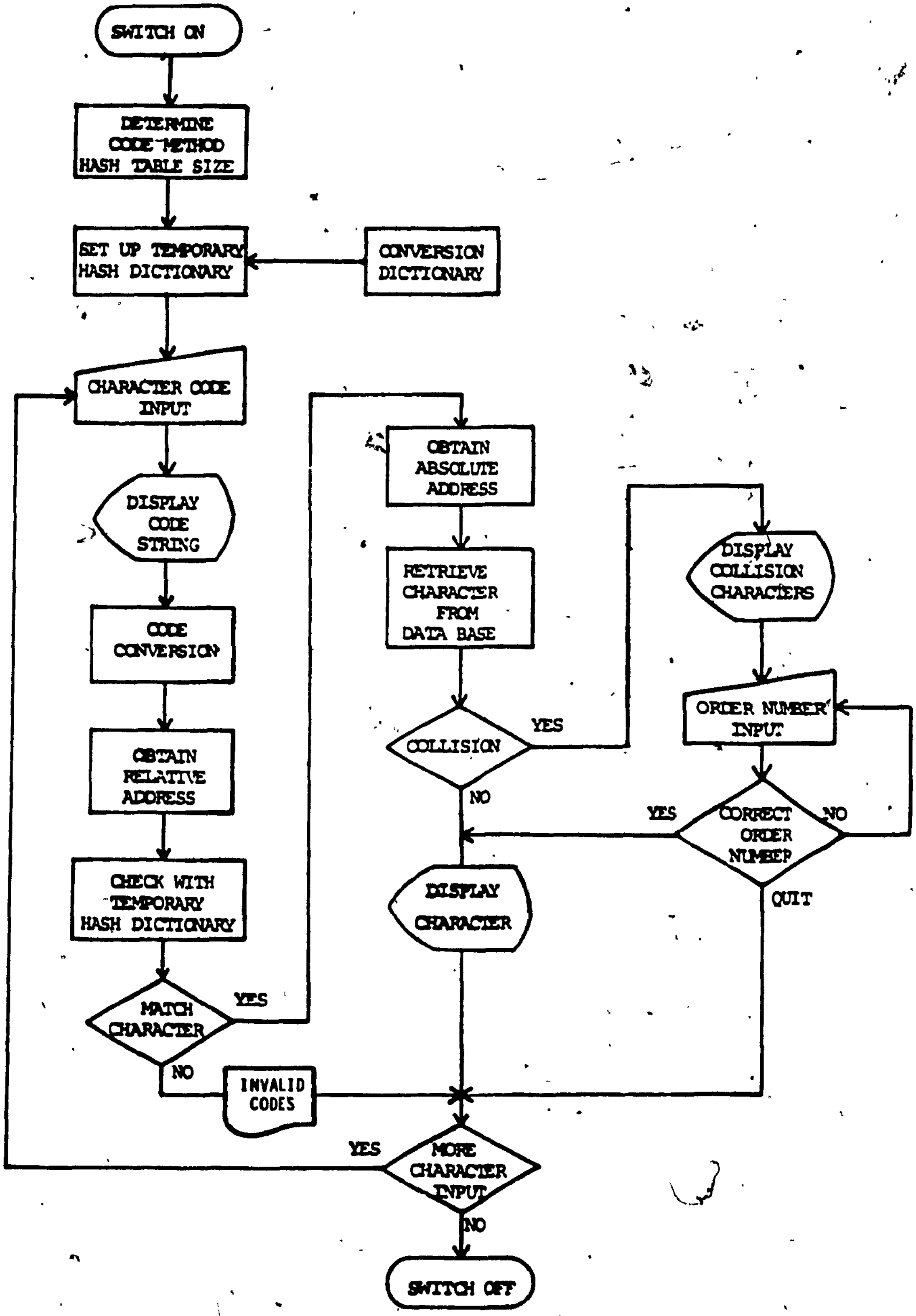


Figure 12. Flow Chart of the Chinese Word Processing Routine.

applied to it to obtain a relative address of the character in the temporary dictionary. A searching in the previous built hash dictionary is performed along its main chain. If there such a code does not exist in the temporary dictionary, a message is sent to inform the user that the encoding fails because the character wanted is not stored in this data base. However, when the code is matched in the main chain, the routine proceeds further to determine if there exists a collision. As long as no collision is found (multiplicity count = 1), the absolute address of the input character in the character pattern data base can be obtained from the sub-chain, and the character pattern is then retrieved from the pattern data base and displayed on the screen (for screen editing system) or the absolute address is written in a particular file which can be used later to display the entire output (for file editing system). Otherwise, when collision happens (multiplicity count > 1), a display of the whole string of collision characters is carried out by retrieving the absolute addresses of these characters from the sub-chain one by one. The desired character can be chosen by specifying its corresponding order in the display list, and it is then displayed in the working area of the screen. In case of the given order is out of range, the user has to re-enter the correct one. If none of the collision characters displayed is the right one, the user can quit from this display process.

In the situations that the wanted character has not been found, the processing routine terminates, or the desired character has been displayed in the workirtg area to ask user for further information. If there no more characters are to be processed, the system will stop running. Otherwise, it repeats again asking the user to input the next character code from the keyboard for processing.

It turns out that, the process starting from the character code input till the desired character pattern being retrieved from the data base, takes approximately 1.0 and 14 milliseconds on CYBER170/835 and VAX11/780, respectively. The display speed on screen is about 200 words per minute and on hard copy unit (Versatec) 500 words per minute. The time required to determine the hash table size and build up the temporary hash dictionary in the preprocessing phase is approximately 6.6 and 14 seconds on CYBER and VAX systems respectively.

As far as the total efficiency is concerned for the processing routine, the file editing system is much faster than the screen editing system.

CHAPTER FIVE

CONCLUSIONS AND SUGGESTIONS

In this thesis, a Chinese word processing system which consists of character encoding methods, keyboard, character pattern data base, conversion and temporary dictionaries, processing procedures, and output, has been designed for and implemented on both CYBER170/835 and VAX11/780 computer systems.

Five different encoding methods based on the combinations of three basic schemes, i.e., phonetic, radical and shape, have been designed for various situations and different preferences. The results show that the Radical and Phonetic Code method is the best among those five encoding methods, judged from the encoding collision rate, and the average and maximum code lengths. It has a low conflict code rate of only 2.4% and its maximum and average code lengths are 5 and 4.5, respectively.

The number of keys varies slightly for the different encoding methods. So long as the character codes are alphabetic, only a small modification has to be made on the English keyboard to accommodate a mixed input of English and Chinese texts, and to reduce the total number of keys on the keyboard.

In the character pattern data base, 4864 Chinese characters are stored in the form of 50 by 50 binary matrices and in the descending order of their frequency values. Random access to such data base is permitted by CRM on CYBER system and by RMS on VAX system. The average time in retrieving a character pattern from the data base is about 1.0 millisecond on CYBER and 14 milliseconds on VAX.

In order to find the absolute address of a character in the character pattern data base, two dictionaries are used. One is the conversion dictionary established prior to the execution of processing procedure; another is the temporary dictionary established during the execution of processing procedure. Division hash function and chaining technique are used in order to facilitate indexing and to solve hashing collision problems. The static characteristic of the data base enables an attempt to find the best size of hash table for each of the five different encoding methods, which will give the best performance in terms of time complexity and space utilization.

The processing procedures in our CWP system is written entirely in Pascal language on both computer systems. The system works interactively and the collision characters resulted from an encoding method are displayed on the screen in order to enable the user to choose the required one.

Two different editing systems, screen editing and file editing, are available on both computer systems. Screen editing is more convenient for the beginner who is not so familiar with the encoding methods, since each inputed character will be displayed on the screen in such editing system. File editing only displays collision characters and it might be suitable for those users who are more familiar with the encoding methods. However, the former editing gives a lower input speed, while the latter a higher input speed.

The results obtained from the simulation of our CWP system with a data base of 4864 Chinese characters, confirms the success of our design.

As suggested below, further improvements on such designed Chinese word processing system are possible.

1). As the Chinese character data base is static, an effort to search for a perfect hash function might be made to enhance the performance. In this way, the values for both space utilization and time complexity can be brought closer to 1.

2). A modification on the present word input format might be carried out in order to allow line input, which will finally allow coded text processing.

3). Knowledge information, such as relations between characters, may be used as a medium to distinguish homonyms. It is really an attractive research area which would involve the knowledge in relational data base.

4). Though an incorrect input character will be rejected by the present system, a new kind of knowledge based data base, might not only lead to the correct code but also reduce the chances of a code rejection. This could be the focus for further research.

REFERENCES

[1] Chen C.K. and Gong R.W., "Evaluation of Chinese Input Methods", Computer Processing of Chinese and Oriental Languages, Vol.1, No.4, pp.236-247, Nov. 1984.

[2] XINHUA Dictionary, SHANG-WU Publication, Beijing, 1980, (in Chinese).

[3] Huang K.T., "Three-Corner Coding Method -- the Digitized Chinese Dictionary", System Publication, Taipei, 1977.

[4] Kiang T.Y. and Cheng T.H., "A New Chinese Indexing System Based on Separation of Character into Main and Subordinate Components", Proc. Int. Computer Symposium, Vol.1, pp.130-141, Dec. 1982.

[5] "China Facts and Figures - Chinese Characters", Foreign Languages Press, 17-E-1187, Beijing, 1984.

[6] Suen C.Y., "Computer Aided Design of Mandarin Phonetic System", Proc. Chinese-American Academic and Professional Assoc. 8th Annual Convention, pp.37-38, Nov. 1983.

[7] Wan S.K., Saitou H. and Mori K.I., "Experiment on PINYIN-HANZI Conversion Chinese Word Processor", Computer Processing of Chinese and Oriental Languages, Vol.1, No.4, pp.213-224, Nov. 1984.

[8] Zhou S.P., "A Proposal on Chinese PINYIN Encoding Scheme", Proc. Of the Second National Conf. On

Chinese Information Processing, April, 1983, (in Chinese).

[9] Ooka T. and Chien M.H., "The Practical Application of the Input Method Converting PINYIN to Characters", Proc. Int. Conf. On Text Processing with a Large Character Set, pp.131-136, Oct. 1983.

[10] Zhang S.Y., "A Proposal on a Chinese Encoding Method Based on the PINYIN of Chinese Word and Its Pattern", Proc. Of the Second National Conf. On Chinese Information Processing, April, 1983, (in Chinese).

[11] Suen C.Y., "Computational Analysis of Mandarin", Birkhauser Boston, Basel, Boston, Stuttgart, 1979.

[12] Suen C.Y., "n-Gram Statistics for Natural Language Understanding and Text Processing", IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol.PAMI-1, No.2, pp.164-172, April 1979.

[13] Lum V.Y., Yuen P.S., and Dodd M., "Key-to-Address Transform Techniques, A Fundamental Performance Study on Large Existing Formatted Files", Commun. Of the ACM, Vol.14, No.4, pp.228, April 1971.

[14] Horowitz E. and Sahni S., "Fundamentals of Data Structures", Computer Science Press Inc., 1982.

[15] Lee W.R., "Representaions of Mandarin Syllables and Computation of Their Frequency Distributions", Master Thesis, Concordia University, April 1984.

[16] Lo S.Y., "A Scientific Model for Comparing Various Methods of Inputting Chinese Characters Into Computer",

Computer Processing of Chinese and Oriental Languages, Vol.1, No.4, pp.36-58, Nov. 1984.

[17] Hwa H.R. and Chung C.Y., "An New Chinese Coding Method for Information Processing", Computer Processing of Chinese and Oriental Languages, Vol.1, No.4, pp.248-267, Nov. 1984.

[18] Yang J, "A Psychological View on the Standardization of the structural Elements of Chinese Characters in Information Encoding", Computer Processing of Chinese and Oriental Languages, Vol.2, No.1, pp.59-70, May 1985.

[19] Lee Y.H. and Kwok C.H., "A Database Implementation of an Interactive Chinese Telephone Directory System", Computer Processing of Chinese and Oriental Languages, Vol.1, No.3, pp.195-204, May 1984.

[20] Chen S.C.J., "Data Compression for Chinese/Hanzi Characters Using a Scaling Algorithm", Computer Processing of Chinese and Oriental Languages, Vol.2, No.2, pp.71-88, Oct. 1985.

[21] Hwang Y.W. and Hwang T.Y., "Microcomputer CAI for Chinese Language with Character Generator", Computer Processing of Chinese and Oriental Languages, Vol.1, No.4, pp.268-275, Nov. 1984.

[22] Heinzl J.L., Ji G.J. and Zhang S.Z., "Decentralized Terminal for Input/Output in HANYU-PINYIN", Computer Processing of Chinese and Oriental Languages, Vol.1, No.4, pp.89-100, Oct. 1985.

[23] Sheng T., "Multi-Input HANZI Encoding System", Proc. Of the Second National Conf. On Chinese Information Processing, April, 1983, (in Chinese).

[24] Suen C.Y. and Huang E.M., "Computational Analysis of the Structural Compositions of Frequently Used Chinese Characters", Computer Processing of Chinese and Oriental Languages, Vol.1, No.3, pp.163-175, May 1984.

[25] Kiang T.Y. and Cheng T.H., "The Analysis of Chinese Characters in Terms of 'Basic Components'", Proc. Int. Computer Conf.: Hong Kong 1980, Vol.1, pp.2.1-2.25, Oct. 1980.

[26] Lee C.C. and Chou C.S., "Structure Analysis and Coding of Chinese Characters", Proc. Int. Computer Conf.: Hong Kong 1980, Vol.1, pp.1.1-1.21, Oct. 1980.

[27] Liu I.M., "Recognition of Fragment-Deleted Characters and Words", Computer Processing of Chinese and Oriental Languages, Vol.2, No.2, pp.89-100, Oct. 1985.

[28] Xiao Q.H., "A Study on the HANZI Structure and Its Encoding", Proc. Chinese Information Processing Theory Conf., May 1982, (in Chinese).

[29] Hu X.H., Zhu Z.L., Fu L.F. and Li H.Q., "A Discussion on the HANZI Radical Encoding Method", Proc. Of the Second National Conf. On Chinese Information Processing, April, 1983, (in Chinese).

[30] Li Z.H., "Fundamental of Chinese Information Processing --- 'LIAN YU XIN'", Proc. Of the Second National Conf. On Chinese Information Processing, April, 1983, (in

Chinese).

[31] Tan L.M. and Yhap E.F., "Decomposition of Chinese Characters for On-line Recognition", the First Int. Conf. On Computers and Applications, pp.42-48, June 1984.

[32] Ye Z.M. and Yuan X.Y., "On Phonetic Spelling Chinese --- A Tone Type Natural Language", Proc. Int. Conf. Chinese Information Processing, Vol.2, pp.238-258, Oct. 1983, (in Chinese).

[33] Tian H.W., "A Phonetic Spelling Coding Based Computer System", Proc. Int. Conf. Chinese Information Processing, Vol.1, pp.15-24, Oct. 1983, (in Chinese).

[34] Ho P., "Ho's Code for Multilingual Character Processor and Communications", Proc. Int. Conf. Chinese Information Processing, Vol.1, pp.69-146, Oct.1983.

[35] Li J.K., "HANZI Information and Dictionary", Proc. Int. Conf. Chinese Information Processing, Vol.1, pp.183-193, Oct. 1983, (in Chinese).

[36] Lo S.Y., "On Chinese Information Processing", Proc. Int. Conf. Chinese Information Processing, Vol.1, pp.1-14, Oct. 1983, (in Chinese).

[37] Ye Z.M., "PINYIN Chinese --- A Natural Language for Computers", Proc. Int. Conf. Chinese Information Processing, Vol.1, pp.48-58, Oct. 1983.