<u>Topic: Assessing the efficiency of different Chinese input methods</u>

Research Question: To what extent is the Chinese Pinyin IME different from Chinese Stroke IME in terms of speed and efficiency?

Subject: Computer Science

Word Count: 3408

Table of Contents

1. Introduction

During the computer science HL course, the Human-Computer Interactions (HCI) is taught, and that interaction can be achieved through the utilization of input devices. Among them, keyboard plays a major role inside the HCI, which is particularly important for early PCs without mice. While we usually believe that most buttons on the keyboard represent their corresponding characters that users intend to input, such as the roman characters for English users, the Chinese language requires more steps than simply pressing a button on the keyboard to input a needed character.

Having over thousands of characters, PC users are expected to be able to input enormous numbers of Chinese characters through a fixed size keyboard, and this requires the application of input methods editors (IMEs) to filter out the needed words from a gigantic word bank but with a limited number of keys.

In China, the most popular input method editor belongs to the Pinyin IME, a pronunciation-based input method that inputs Chinese characters through entering series of Pinyin, the official romanization system for Standard Chinese. Or at least, everyone around me uses the Pinyin IME for Chinese inputting. On the other hand, countless of alternate options are available as substitutions, such as stroke-based IME and Zhuyin IME. While Zhuyin IME, also a pronunciation-based IME, is conceptually similar to Pinyin IME, stroke-based IMEs allow users to piece the components of a Chinese character together to recognize and input the wanted character.

As a Chinese Pinyin IME user, I became largely interested in investigating the reason why Pinyin IME became the most popular IME, assuming that Pinyin IME has a greater efficiency than Stroke IME. Therefore, the research question arises: “To what extent is a stroke IME different from Chinese Pinyin IME in terms of speed and efficiency?”

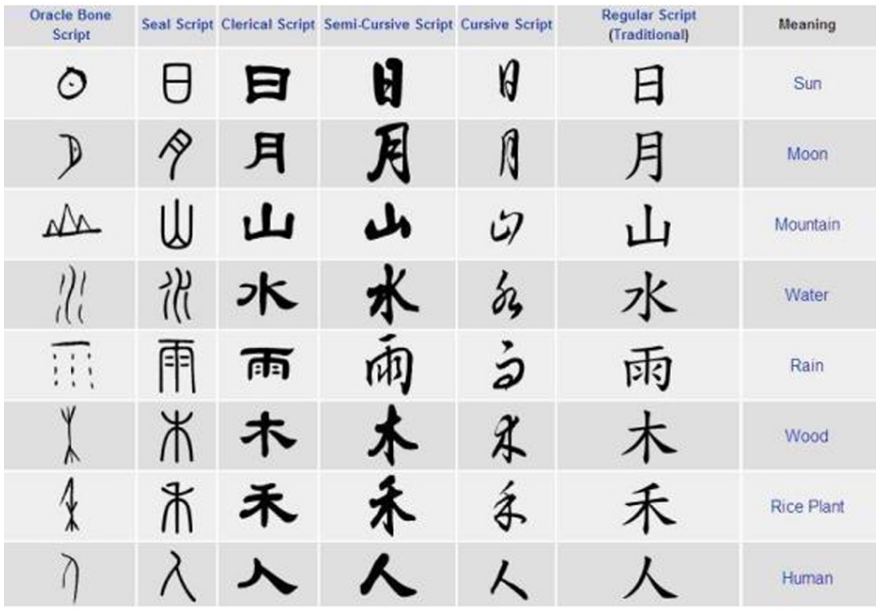
While the paper assesses the efficiency difference of the two IMEs, the main focus is on the investigation of the factor that has contributed to the efficiency of Pinyin IME, if it is more efficient than Stroke IME. The goal is to first find out which IME has the greater efficiency and then to evaluate a possible reason for how the IME is more efficient.

2. Background Information

2.1 Chinese characters

Chinese is the primary language spoken and is used by nearly 1.4 billion citizens of the People's Republic of China for daily communication. Its flexible grammar, perplexing phonemes and particularly, the innumerable characters that exist in the vocabulary of Chinese form its uniqueness.

Chinese characters were initially evolved as hieroglyphs, which are fonts created through the emulation of objects and pictures. The Oracle Bone Script, for instance, was one of the earliest forms of Chinese characters in which were written on oracle bones, which still remains influence on modern Chinese even after millenniums of transformations. [1]



Oracle Bone Script	Seal Script	Clerical Script	Semi-Cursive Script	Cursive Script	Regular Script (Traditional)	Meaning
日	日	日	日	日	日	Sun
月	月	月	月	月	月	Moon
山	山	山	山	山	山	Mountain
水	水	水	水	水	水	Water
雨	雨	雨	雨	雨	雨	Rain
木	木	木	木	木	木	Wood
禾	禾	禾	禾	禾	禾	Rice Plant
人	人	人	人	人	人	Human

Figure 2.1 Examples of Chinese hieroglyphs evolved from Oracle Bone Script [2]

Oracle Bone Scripts established the foundation of basic structure for the Chinese characters, which is one character each having its own unique meaning, and since an individual meaning is expressed through each Chinese character, there is a necessity of having a large number of characters to express a variety of content.

An overwhelming number of characters are presented in the language that surprisingly reaches up to 91,251 Chinese characters. Within the total number of characters, commonly used ones only hold around 7,000 characters, and the frequently used characters occupy only 3,500 characters out of the entire character set. [3]

2.2 Input Method Editor

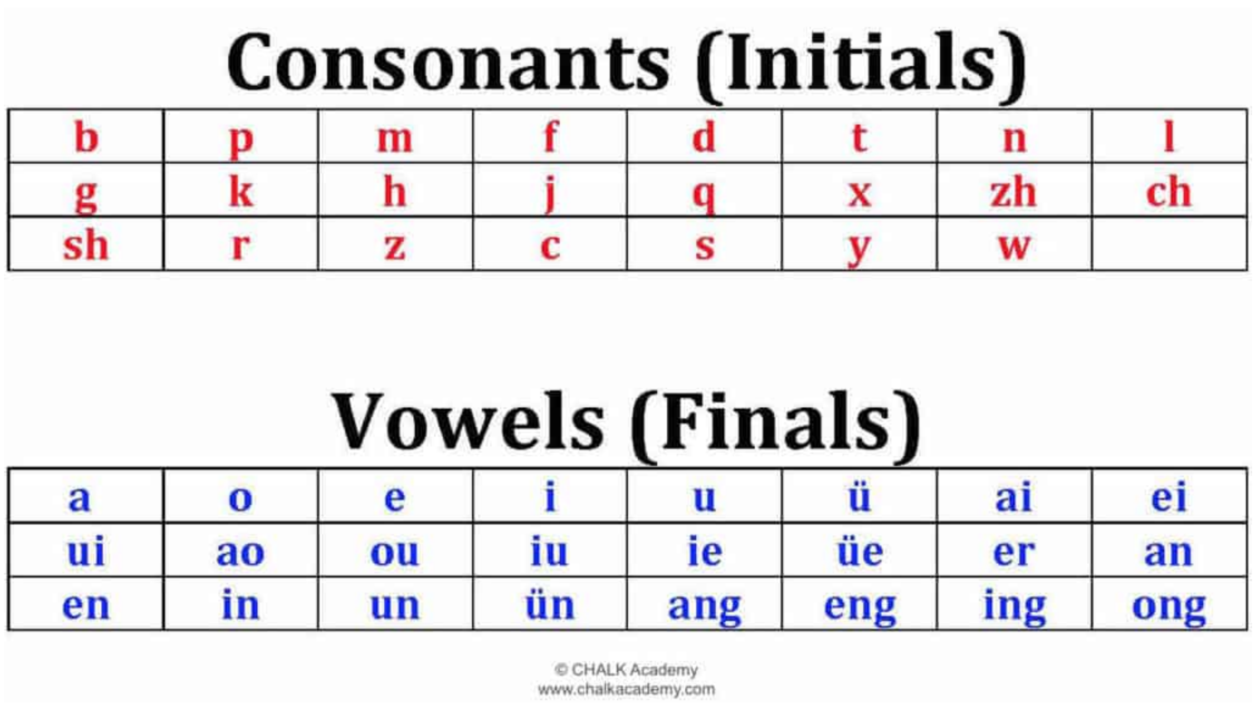
Input Methods are encoding methods that allow the output of multiplex characters or symbols through inputting any data.

The demand of input methods originated from the limitations of keyboard size. Initially designed primarily for English language, keyboard only features a composition of up to around 110 keys, which is limited when inputting logograms like Japanese and Chinese which have over thousands of characters. [4]

Having over thousands of Chinese characters, the Chinese language requires input methods to input them on a device.

2.3 Chinese Pinyin IME

Every character in Chinese has a monosyllabic pronunciation that does not interfere with other nearby characters, while some characters in Western languages may have several different pronunciations when combined as different words. For instance, letter A in English can be pronounced as [ei] or [æ] depending on its circumstance. Instead, the pronunciation of each Chinese character is composed of a consonant and a vowel or diphthong, the combination of two vowels in a single sound, such as zhuang, zhuo and tiao.



Consonants (Initials)

b	p	m	f	d	t	n	l
g	k	h	j	q	x	zh	ch
sh	r	z	c	s	y	w	

Vowels (Finals)

a	o	e	i	u	ü	ai	ei
ui	ao	ou	iu	ie	üe	er	an
en	in	un	ün	ang	eng	ing	ong

© CHALK Academy
www.chalkacademy.com

Figure 2.3.1 Consonants and vowels in Chinese characters presented in Pinyin [5]

Chinese Pinyin, or Hanyu Pinyin are the phonetic notations for Chinese characters where each word in Chinese is able to convert itself into Pinyin based upon its pronunciation. In consequence, Pinyin serves mainly as a tool of assisting the learning of Chinese pronunciation. As Chinese Pinyin mostly borrows the roman characters that are vastly presented on the QWERTY keyboard, the Chinese Pinyin becomes one of the most commonly used Chinese input methods, which is more likely due to the education of Pinyin being already taught for recent generations of people during their elementary school.

The Pinyin IME is an input method that uses Pinyin to select wanted Chinese characters. Using the Pinyin IME, user enters a Chinese character by typing down Pinyin that corresponds to the pronunciation of that character and then selects the exact character through a pop-up bar. The reason why a further selection is required is due to the existence of the dozens of characters which have the same pronunciation but have different meanings, and this becomes the downside of Pinyin IME.

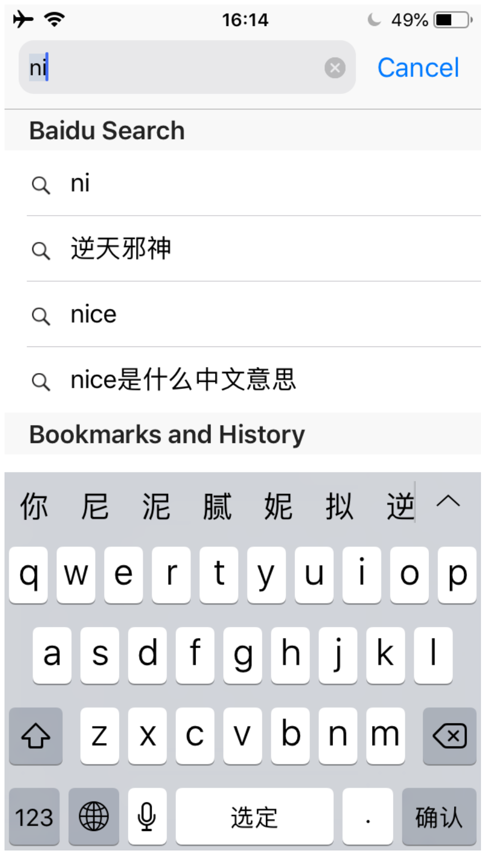


Figure 2.3.2 User inputting a character "你" (you) using Pinyin IME on an iPhone

Usually, a single character of Chinese only requires a maximum of 6 Pinyin characters to input, but it does not output the exact wanted character. The downside that the Pinyin IME owns is its inaccuracy. Typing down the corresponding Pinyin displays dozens of characters that share the same Pinyin for further selection, causing the waste of time to choose the accurate one. In addition, standard Chinese has four main tones that are unable to be labeled through typing. This results in an even higher inaccuracy while using Pinyin IME.

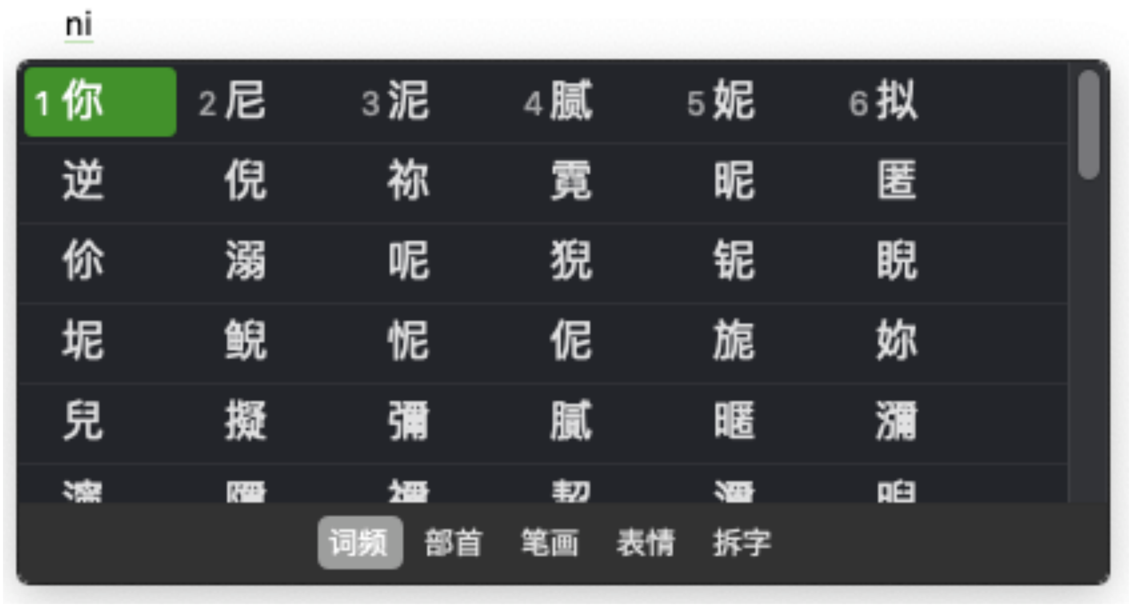


Figure 2.3.3 User selecting a character "你" using Pinyin "ni" out of all possible choices

2.4 Chinese Stroke IME

Stroke input method was initially intended to serve as an easy-to-learn IME for users without knowledge of Chinese Pinyin and want to master a Chinese input method in the least amount of time. The method uses 6 buttons to perform action of inputting Chinese. Each button refers to a specific type of stroke that is vastly presented on Chinese characters and is usually located at a corresponding number key if the computer includes a physical number pad.

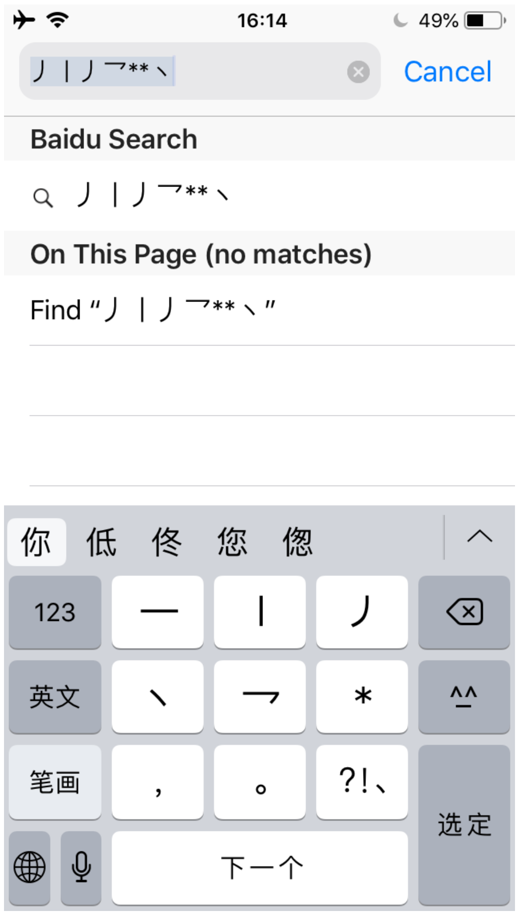


Figure 2.4.1 User inputting a character "你" (you) using Stroke IME on an iPhone

When inputting, users enter the strokes of a Chinese character in its correct stroke order. The keys contain 5 components of Chinese characters:

1. A horizontal stroke (一)
2. A vertical stroke (丨)
3. A long diagonal stroke (丿)
4. A very short dash stroke (丶)
5. A horizontal stroke that has a downwards hook on the right (乛)
6. An asterisk that corresponds to any stroke.

A disadvantage of stroke input method is its low fault tolerance. The strokes inputted must be in the correct stroke order, meaning that a single mistake in stroke order can potentially cause the output of a completely different character. This is especially obvious when each character out of thousands of others has its unique stroke order for memorization, increasing a certain amount of difficulties for users who do not follow along with the correct stroke orders during handwriting.

However, despite entering the character needs a large number of presses, the high accuracy stands out when user masters the IME. Due to the theory that each Chinese character has its own distinguishing strokes and stroke order, inputting correct strokes in the right order returns the exact character that user intends to output, whereas inputting the correct Pinyin while using Pinyin IME may outcome dozens of possible characters.

In addition, the stroke IME does not necessarily require user to enter all the strokes for a character to input itself. The number of strokes entered into the IME can be reduced when the inputted strokes are already enough to filter a majority of characters and leave dozens for selections. Without entering any extra strokes, user can select the character handily through a glance at the pop-up bar. However, this technique rarely used since experienced users are likely to be capable of recognizing all the characters’ strokes and entering them handily, requiring no selection needed at the pop-up bar as it possibly takes even longer time than inputting all the memorized strokes.

2.5 Hidden Markov Model

Typically, Chinese IMEs are heavily dependent on Hidden Markov Model (HMM) to assist the efficiency of selecting Chinese characters. HMM a probability model served as the representation of possibilities for a sequence of data. [6] The model is referred to a type of Markov Model with hidden states that are unable to be directly observed. HMM is mainly adopted in the areas of machine translation and speech recognition.

HMM is consisted of variables sets with two different states, hidden states *Q* and observations *O*, and three probabilities, transition probabilities *A*, emission probability *B* and initial state probability *Π*. The probabilities and variable sets can be put into calculations to solve for one another. [7]

Problems	**Given…**	**Find…**	**Solutions**
Evaluation problem	Observation sequences and an HMM model	The probability of observations	Forward and backward algorithms
Decoding problem	Observation sequences and an HMM model	The most likely state sequence that outcomes the observations	Posterior decoding and Viterbi algorithm
Learning problem	Observation sequences and an HMM model	The model parameters $\{A, B, \pi\}$ to maximize $p\{O\|\lambda\}$	Baum-Welch algorithm

Figure 2.5.1 Three basic problems associated with HMM and their solutions [7]

In terms of Chinese IME, the decoding problem, as an example, can be involved to calculate the most likely chosen character from the Pinyin or strokes inputted. As each set of Pinyin can return dozens of Chinese characters, the character with maximum probability of being selected will be displayed at front while other characters are shown in the sequence of probability descending, allowing less time consumption for users to select characters.

In a case where the user uses Pinyin IME to input a Chinese phrase “今天天气很好”(The weather is good today), the Pinyin inputted and user expected outputs are listed in the table below:

Pinyin Input	**Possible Character Outputs**	**User Intended Output**
jin	金 进 近 仅 **今** 斤 紧…	今
tian	**天** 甜 田 填 添 舔 恬…	天
tian	**天** 甜 田 填 添 舔 恬…	天
qi	起 祺 其 七 起 **气** 期…	气
hen	狠 恨 **很** 痕 拫 噷 佷…	很
hao	郝 号 豪 浩 **好** 耗 昊…	好

Figure 2.5.2 Pinyin Inputted and user expected character output table

The posterior decoding assumes that the inputted Pinyin are observations O and the outputted characters are hidden states Q. To obtain the probability of the model to output a character given the inputted Pinyin, the equation $\lambda_t(i)$ is given by

$$\lambda_t(i) = P(\, i_{th} possible\ character\ match\ for\ t_{th}\ Pinyin\ inputted \mid o)$$

Which can be derived into

$$\lambda_t(i) = P(\, q(t) = q_i \mid o) \qquad [7]$$

Where $P(\, q(t) = q_i)$ is the probability of outputting an $\mathrm{i_{th}}$ character for $\mathrm{t_{th}}$ Pinyin inputted and $i = 1, \ldots, N, t = 1, \ldots, T.$ [7]

Then, finding the character output with the maximum probability of being selected is given by equation $q(t)$:

$$q(t) = \arg\max P\{\lambda_t(i)\} \qquad [7]$$

Furthermore, the computation of probabilities for choosing the characters also relies on cloud data, providing vast number of literary samples to determine which character is more frequently used out of all characters with the same Pinyin. For instance, Sogou Chinese Pinyin IME is one of the major Pinyin IME used in China, occupying 43.2% of Chinese Pinyin IME market share in 2019. [8] The IME initially added the feature of cloud computing input in 2010. When inputting, Sogou IME will automatically obtain cloud data to ensure more accurate input results, which is usually done by calculating the probabilities $q(t)$ of each character through the examination of large amount of cloud samples. [9]

By doing so, the probabilities $q(t)$ of each character being selected are calculated by the model and then are used in arranging the order of characters from the most likely selected ones at the front and the least likely selected ones at the last, effectively reduces the time consumed by user selecting characters.

3. Experiment Methodology

The investigation will be conducted through two components of codes, each for evaluating the efficiency of one type of Chinese IMEs. The assessment of IME efficiency and speed is determined through the number of presses which is required to input a sample text and is further achieved through the comparison between the two Chinese IMEs, Stroke IME and Pinyin IME. In this case, the IME that requires fewer presses to input the sample text demonstrates the higher efficiency in time and speed. The base codes are discovered from online sources and modified to achieve basic functionality, and they are available in Appendix section.

3.1 Independent Variables

The three independent variables used in the investigation are three sample texts for IME efficiency evaluation and are excerpts from translated novels and biography or Chinese works.

Sample	Name	Author
1	*Death's End in Remembrance of Earth's Past trilogy [10]*	Cixin Liu
2	*Steve Jobs [11]*	Walter Isaacson
3	*The Little Prince [12]*	Antoine De Saint-Exupéry

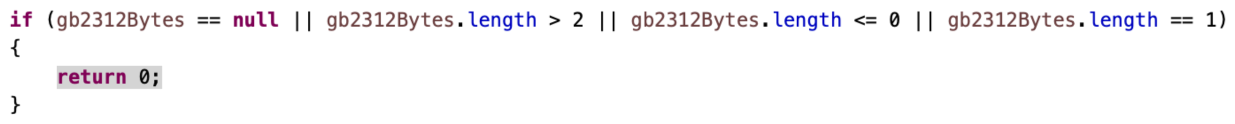
Figure 3.1.1 Book name and author of the three excerpts samples

3.2 Dependent Variables

The dependent variables are the stroke count and Pinyin count of each sample inputted.

3.3 Preprocessing

In order to count the strokes and Pinyin number properly, punctuations within the sample texts are removed since the extract of GB2312, the official Chinese character coding set of the People’s Republic of China, that I use in Chinese characters conversion does not include punctuation. Arabic numerals and Latin characters are also avoided in passage selection due to the similar reason that gb2312 does not consist them, and Chinese stroke IME itself also does not support English typing. In some parts of the sample texts, Arabic numbers are exchanged with Chinese characters.



```
if (gb2312Bytes == null || gb2312Bytes.length > 2 || gb2312Bytes.length <= 0 || gb2312Bytes.length == 1)
{
    return 0;
}
```

Figure 3.3.1 Returning 0 if the character is not included in GB2312, such as punctuations, numerals and Latin characters.

3.4 Hypothesis

Personally, I believe that the Chinese Pinyin IME will be more efficient than Chinese stroke IME. As a Chinese Pinyin IME user, I find the Pinyin IME being very speedy at typing Chinese in practice, which is perhaps due to my years of experiences with the Pinyin IME.

3.5 The Experiments

Both Stroke IME and Pinyin IME use database to store the stroke count and Pinyin for each Chinese character. For instance, the stroke count database, which indicates stroke counts for all Chinese characters, is externally inputted in the form of an array and follows the standard of GB2312, the official Chinese character coding set of the People’s Republic of China. To return the total number of presses required to input the sample text through both IMEs, the process of calculating stroke counts and Pinyin counts is done separately for each individual Chinese character at a time and is then added up after a complete repetition. The flow charts below demonstrate the basic procedures for computing the needed number of presses using Stroke IME and Pinyin IME to input the sample text.

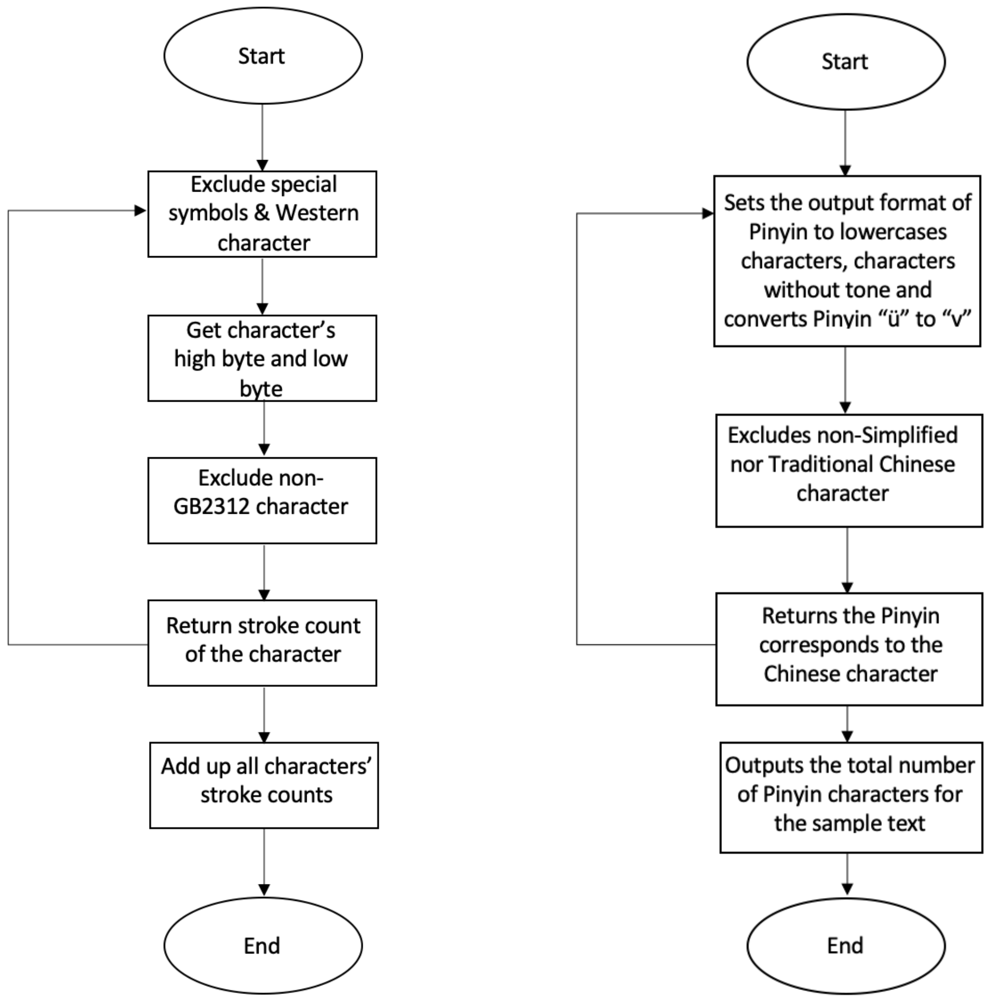
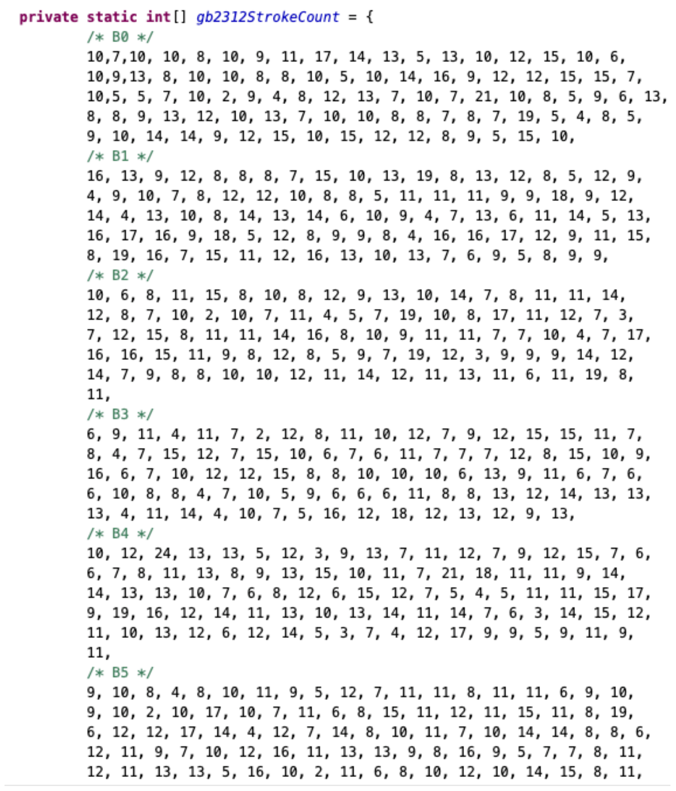

Start
Exclude special symbols & Western character
Get character's high byte and low byte
Exclude non-GB2312 character
Return stroke count of the character
Add up all characters' stroke counts
End
Start
Sets the output format of Pinyin to lowercases characters, characters without tone and converts Pinyin "ü" to "v"
Excludes non-Simplified nor Traditional Chinese character
Returns the Pinyin corresponds to the Chinese character
Outputs the total number of Pinyin characters for the sample text
End

Figure 3.5.1 Flowcharts for the basic procedures for calculating the needed number of presses using Stroke IME (left) and Pinyin IME (right).

3.6 Programs Break Down

3.6.1 The Stroke IME

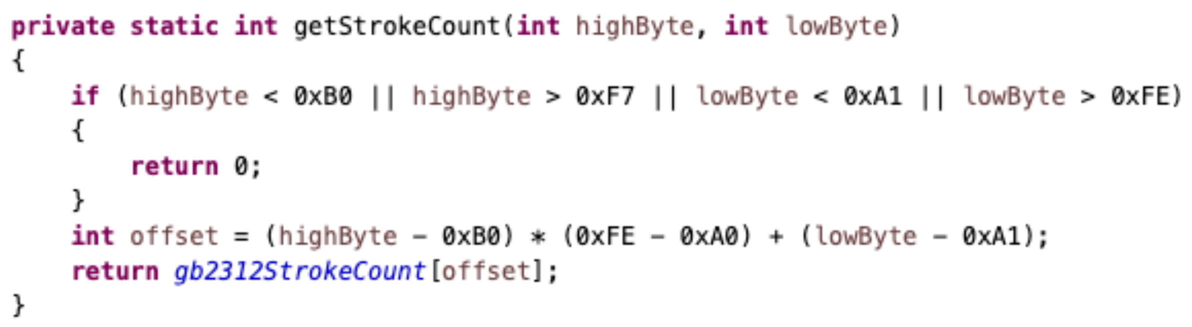
The calculations of strokes for each character are achieved through an array that contains all GB2312 characters' stroke counts, which is arranged in the order of those having smallest to greatest bytes. Since GB2312 requires 2 bytes to represent a Chinese character, or 16 bits, the bytes can be divided into a high byte and a low byte, where high byte holds the most significant part of the integer. [13] For instance, Chinese character "一" (one) has a GB2312 code of "D2BB", so the high byte and low byte would respectively be D2 and BB.



```
private static int[] gb2312StrokeCount = {
        /* B0 */
        10,7,10, 10, 8, 10, 9, 11, 17, 14, 13, 5, 13, 10, 12, 15, 10, 6,
        10,9,13, 8, 10, 10, 8, 8, 10, 5, 10, 14, 16, 9, 12, 12, 15, 15, 7,
        10,5, 5, 7, 10, 2, 9, 4, 8, 12, 13, 7, 10, 7, 21, 10, 8, 5, 9, 6, 13,
        8, 8, 9, 13, 12, 10, 13, 7, 10, 10, 8, 8, 7, 8, 7, 19, 5, 4, 8, 5,
        9, 10, 14, 14, 9, 12, 15, 10, 15, 12, 12, 8, 9, 5, 15, 10,
        /* B1 */
        16, 13, 9, 12, 8, 8, 8, 7, 15, 10, 13, 19, 8, 13, 12, 8, 5, 12, 9,
        4, 9, 10, 7, 8, 12, 12, 10, 8, 8, 5, 11, 11, 11, 9, 9, 18, 9, 12,
        14, 4, 13, 10, 8, 14, 13, 14, 6, 10, 9, 4, 7, 13, 6, 11, 14, 5, 13,
        16, 17, 16, 9, 18, 5, 12, 8, 9, 9, 8, 4, 16, 16, 17, 12, 9, 11, 15,
        8, 19, 16, 7, 15, 11, 12, 16, 13, 10, 13, 7, 6, 9, 5, 8, 9, 9,
        /* B2 */
        10, 6, 8, 11, 15, 8, 10, 8, 12, 9, 13, 10, 14, 7, 8, 11, 11, 14,
        12, 8, 7, 10, 2, 10, 7, 11, 4, 5, 7, 19, 10, 8, 17, 11, 12, 7, 3,
        7, 12, 15, 8, 11, 11, 14, 16, 8, 10, 9, 11, 11, 7, 7, 10, 4, 7, 17,
        16, 16, 15, 11, 9, 8, 12, 8, 5, 9, 7, 19, 12, 3, 9, 9, 9, 14, 12,
        14, 7, 9, 8, 8, 10, 10, 12, 11, 14, 12, 11, 13, 11, 6, 11, 19, 8,
        11,
        /* B3 */
        6, 9, 11, 4, 11, 7, 2, 12, 8, 11, 10, 12, 7, 9, 12, 15, 15, 11, 7,
        8, 4, 7, 15, 12, 7, 15, 10, 6, 7, 6, 11, 7, 7, 7, 12, 8, 15, 10, 9,
        16, 6, 7, 10, 12, 12, 15, 8, 8, 10, 10, 10, 6, 13, 9, 11, 6, 7, 6,
        6, 10, 8, 8, 4, 7, 10, 5, 9, 6, 6, 6, 11, 8, 8, 13, 12, 14, 13, 13,
        13, 4, 11, 14, 4, 10, 7, 5, 16, 12, 18, 12, 13, 12, 9, 13,
        /* B4 */
        10, 12, 24, 13, 13, 5, 12, 3, 9, 13, 7, 11, 12, 7, 9, 12, 15, 7, 6,
        6, 7, 8, 11, 13, 8, 9, 13, 15, 10, 11, 7, 21, 18, 11, 11, 9, 14,
        14, 13, 13, 10, 7, 6, 8, 12, 6, 15, 12, 7, 5, 4, 5, 11, 11, 15, 17,
        9, 19, 16, 12, 14, 11, 13, 10, 13, 14, 11, 14, 7, 6, 3, 14, 15, 12,
        11, 10, 13, 12, 6, 12, 14, 5, 3, 7, 4, 12, 17, 9, 9, 5, 9, 11, 9,
        11,
        /* B5 */
        9, 10, 8, 4, 8, 10, 11, 9, 5, 12, 7, 11, 11, 8, 11, 11, 6, 9, 10,
        9, 10, 2, 10, 17, 10, 7, 11, 6, 8, 15, 11, 12, 11, 15, 11, 8, 19,
        6, 12, 12, 17, 14, 4, 12, 7, 14, 8, 10, 11, 7, 10, 14, 14, 8, 8, 6,
        12, 11, 9, 7, 10, 12, 16, 11, 13, 13, 9, 8, 16, 9, 5, 7, 7, 8, 11,
        12, 11, 13, 13, 5, 16, 10, 2, 11, 6, 8, 10, 12, 10, 14, 15, 8, 11,
```

Figure 3.6.1.1 An array that contains all GB2312 characters' stroke counts with hexadecimal numbers annotated in between the array, indicating the characters with the same high bytes are below the number.

The high byte and low byte are then used to locate the character's stroke count in the array. Revealed in figure 3.6.1.2, as the array only contains stroke counts for characters with high bytes that are within the range from B0 to F7, characters that are outside of the range are excluded and thus return 0 for stroke counts. This also applies to characters with low bytes that are not within the range from A1 to FE.



```java
private static int getStrokeCount(int highByte, int lowByte)
{
    if (highByte < 0xB0 || highByte > 0xF7 || lowByte < 0xA1 || lowByte > 0xFE)
    {
        return 0;
    }
    int offset = (highByte - 0xB0) * (0xFE - 0xA0) + (lowByte - 0xA1);
    return gb2312StrokeCount[offset];
}
```

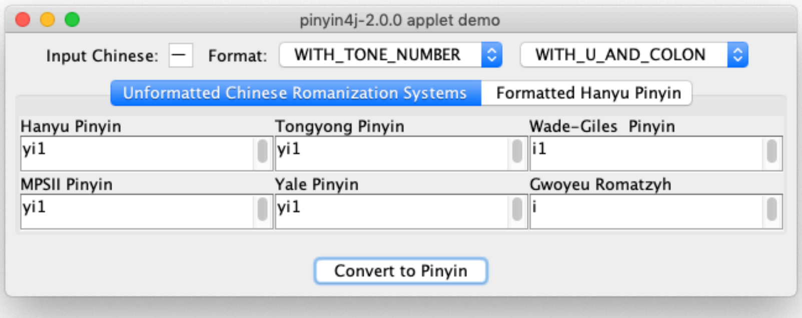
Figure 3.6.1.2 Method of getStrokeCount(int highByte, int lowByte)

The integer, offset, is the location of inputted character's stroke count that is stored in the array gb2312StrokeCount[]. Since the gb2312StrokeCount[] array begins from characters with high bytes of B0, and in each high byte, there are FE – A0 numbers of characters, so offset can locate its final result to those with the same high bytes using `(highByte – 0xB0) * (0xFE – 0xA0)`. Then, the exact location within the category of high byte is calculated by `(lowByte – 0xA1)`, which returns the stroke count of the character using "`return gb2312StrokeCount[offset];`"

As a result, by repeating the above process for each character in the entire inputted passage using a for loop, the program is able to return the total number of presses required for Stroke IME to input the passage.

3.6.2 The Pinyin IME

Calculating the Pinyin count of each character inside a passage primarily relies upon a finished JAR file named “pinyin4j-2.0.0”. [14] The application outputs six types of Pinyin of the user’s inputted Chinese character.


pinyin4j-2.0.0 applet demo

Input Chinese: 一 Format: WITH_TONE_NUMBER WITH_U_AND_COLON

Unformatted Chinese Romanization Systems | Formatted Hanyu Pinyin

Hanyu Pinyin: yi1
Tongyong Pinyin: yi1
Wade-Giles Pinyin: i1
MPSII Pinyin: yi1
Yale Pinyin: yi1
Gwoyeu Romatzyh: i

Figure 3.6.2.1 Finding the Pinyin of character “一”(one) using the JAR file

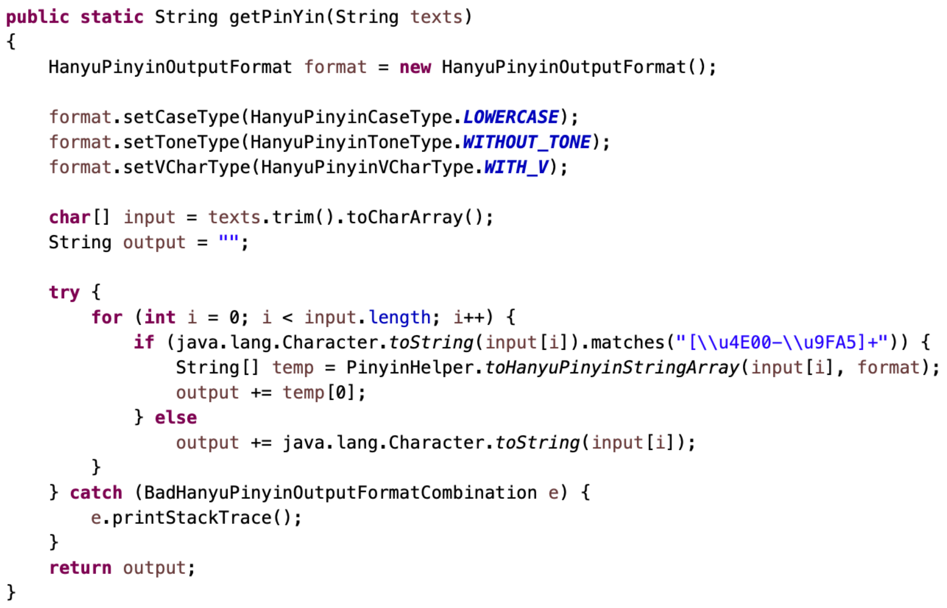
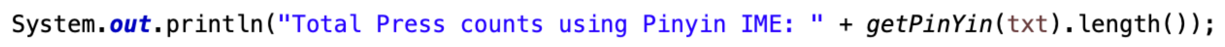

```java
public static String getPinYin(String texts)
{
    HanyuPinyinOutputFormat format = new HanyuPinyinOutputFormat();

    format.setCaseType(HanyuPinyinCaseType.LOWERCASE);
    format.setToneType(HanyuPinyinToneType.WITHOUT_TONE);
    format.setVCharType(HanyuPinyinVCharType.WITH_V);

    char[] input = texts.trim().toCharArray();
    String output = "";

    try {
        for (int i = 0; i < input.length; i++) {
            if (java.lang.Character.toString(input[i]).matches("[\\u4E00-\\u9FA5]+")) {
                String[] temp = PinyinHelper.toHanyuPinyinStringArray(input[i], format);
                output += temp[0];
            } else
                output += java.lang.Character.toString(input[i]);
        }
    } catch (BadHanyuPinyinOutputFormatCombination e) {
        e.printStackTrace();
    }
    return output;
}
```

Figure 3.6.2.2 Method of getPinYin(String texts)

Demonstrated in figure 3.6.2.2, the program first initializes its Pinyin format to all lowercase, without tone and with v (Pinyin has a special character “ü” that can also be exchanged with character “v”). While String texts is the excerpts passage, it is broken down into a character array. In the for loop, the program detects if the character matches one of the Chinese characters, then it outputs the Pinyin of the character, and otherwise, it would return the character itself.



```
System.out.println("Total Press counts using Pinyin IME: " + getPinYin(txt).length());
```

Figure 3.6.2.3 Outputs the total number of presses using Pinyin IME in the main method

After receiving all the Pinyin of the passage, the total number of Pinyin characters can be obtained using “.length()", displayed in figure 3.6.2.3.

4. Experiment Results

4.1 Results Table

		Presses required for (IME)	
Sample Text	Word Count	Pinyin IME	Stroke IME
#1 Three Body	408	1258	2850
#2 Steve Jobs	380	1134	2746
#3 Little Prince	374	1074	2685
Average	387.33	1155.33	2760.33

Figure 4.1.1 The word counts and presses required for sample texts using the two IMEs

4.2 Results Graph

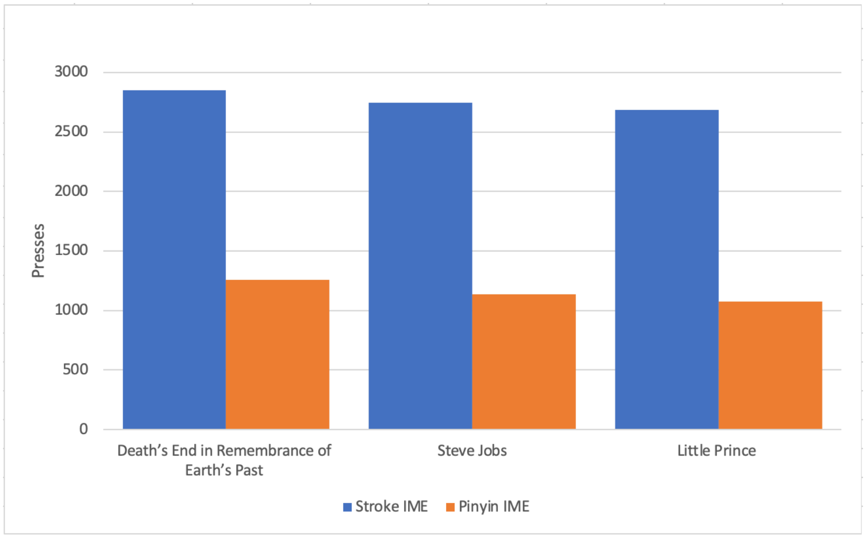


Figure 4.2.1 The presses needed for entering sample texts using the two IMEs in bar graph

5. Evaluation

Although the results data demonstrate vast differences of efficiency in terms of the number of presses, the computation of presses for the two Chinese IMEs in inputting a sample text does not guarantee an accurate concept of efficiency comparison. There is a flaw that causes the huge uncertainty to the results, where entering Pinyin into the IME does not occupy the entire time for inputting. As explained earlier, user must select the exact character to input from a group of characters after inputting Pinyin, since there are often dozens or hundreds of Chinese characters with the same pronunciations displayed for selection, whereas entering the full strokes of a Chinese character in stroke IME bypasses the further selection but at a cost of spending much more time.

The results compare the time necessary to input raw Pinyin and strokes to recognize Chinese characters but neglect the time taken for the selection of characters. The selection increases the time required for inputting with Pinyin IME, which is more than the time returned through the experiment. This is a huge uncertainty that impacts the factuality of obtained results, since the time consumed when selecting characters is unable to be calculated due to a variety of reasons as there is no defined unit to measure the time it takes, and it is rather down to user's ability to react quickly when he/she finds the needed character. But after all, as previously mentioned, it is largely minimized by the Hidden Markov Model by arranging the mostly likely selected characters at the front and the least ones at the back when user is choosing a character from an inputted Pinyin.

In my personal experience with Pinyin IME, I rarely encounter situations where I am in need of browsing through pages and pages of possible characters to find the right one besides when inputting sophisticated characters from literatures. In most cases, only a glance is needed to input a character. In addition, despite that there are around 100,000 characters in Chinese, statistics have shown that 1,000 common characters can cover around 92% of written data, while 3,000 characters can reach up to 99% of them. [15] This supports the reason why the further selection of characters does not occupy too much time and can be perhaps neglected to a certain extent.

6. Conclusion

The aim of the essay is to assess the efficiency of Chinese Pinyin IME and Chinese Stroke IME. Based on the results, we can approximately ensure that Pinyin IME requires less presses to input an excerpt than Stroke IME needed, which means that Pinyin IME has a greater efficiency than Stroke IME, proving the reason why it is the most popular IME in the Chinese market.

Throughout the investigation, we found out a major difference between the two IMEs is that Pinyin IME has a selection process after inputting Pinyin, which requires the utilization of HMM to minimize the time consumed by the process. This is what contributes to the great efficiency of Pinyin IME, or else user may be always struggling to find a character from hundreds of characters. As algorithms are being improved, newly developed algorithms such as the Recurrent Neutral Network (RNN) are gradually attempted by many Pinyin IMEs to provide greater accuracy and efficiency in the past few years, but the Hidden Markov Model is still the most mature algorithm in the field of Pinyin IME, emphasizing the importance of algorithm that impacts the efficiency of Pinyin IME. [16]

On the other hand, despite the low efficiency of Stroke IME, it still serves its role of being an easy-to-learn IME for people who do not know Pinyin and are willing to be proficient in an IME using the least amount of time, but at the cost of spending more time to input characters than Pinyin IME.

After all, I hope this paper can present some useful information about Chinese IMEs in general and project the effect of the statistical model on the Pinyin IME by comparing the most popular used Pinyin IME with the less commonly used Stroke IME.

7. References

[1] "The Origin of Chinese Characters." *Chinasage*, www.chinasage.info/chinese-characters.htm. Accessed 14 Dec. 2020.

[2] Chang, Hao-Ting. “Evolution of Chinese Pictogram Characters.” *Rain*, 2011, ciid.dk/education/portfolio/idp11/courses/generative-design/final-projects/rain/.

[3] *List of Commonly Used Characters in Modern Chinese*. Ministry of Education of the People's Republic of China, 1988.

[4] “How Many Keys Are on a Computer Keyboard?” *Computer Hope*, 7 Oct. 2019, www.computerhope.com/issues/ch001598.htm.

[5] “Pinyin Consonants and Vowels.” *When Should My Child Learn Hanyu Pinyin?*, Betty, 15 July 2018, chalkacademy.com/when-should-child-learn-pinyin/.

[6] Jia, Jianfeng, et al. “Research on Chinese Whole Sentence Pinyin IME Based on HMM.” *Research and Development*, 2008.

[7] Shokhirev, Nikolai. “Hidden Markov Models.” *Hidden Markov Models - HMM*, www.numericalexpert.com/tutorials/hmm/hmm.php.

[8] Shan, Renyu. "Input Method Industry Data Analysis: Sogou Input Method Accounted for 43.2% of the Market Share of Users in 2019." *IiMedia*, 20 May 2020, www.iimedia.cn/c1061/71552.html.

[9] . Li Yunfeng. "In the Cloud Era, Input Methods Should Also be 'In The Cloud’." *Computer enthusiasts* 000.008(2010):42-42.

[10] Liu, Cixin. *Death's End*. Head of Zeus, 2016.

[11] Isaacson, Walter. *Steve Jobs*. Translated by Yanqi Guan, Zhongxin, 2011.

[12] Saint-Exupéry Antoine de. *The Little Prince*. Translated by Zhenni Lin and Zhencheng Ma, YiLin, 2010.

[13] Kjell, Bradley. "High Bytes and Low Bytes." *Introduction to Computer Science Using Java*, Central Connecticut State University, chortle.ccsu.edu/java5/Notes/chap85/ch85_12.html.

[14] (n.d.). Retrieved from http://pinyin4j.sourceforge.net

[15] Zhiping Zhu, "Chinese second language teaching of optimization and classification of Chinese characters in overseas elementary and middle schools." *The second International Symposium on Literacy Education,* 2006.

[16] Yi, Shu. "Recurrent Neural Network Components -- RNN and RNN Layer (Detailed Analysis of RNN Principle and Implementation)." *CSDN*, 20 Sept. 2020, blog.csdn.net/sinat_35907936/article/details/108276948.

8. Appendix

8.1 **Codes for Pinyin IME** (Java, retrieved from https://blog.csdn.net/qq_40083897/article/details/85779162)

```
import net.sourceforge.pinyin4j.PinyinHelper;

public static void main(String[] args) {

        String txt = “”;
        System.out.println("Total Press counts using Pinyin IME: " + getPinYin(txt).length());

}

public static String getPinYin(String texts) {
        HanyuPinyinOutputFormat format = new HanyuPinyinOutputFormat();

        format.setCaseType(HanyuPinyinCaseType.LOWERCASE);
        format.setToneType(HanyuPinyinToneType.WITHOUT_TONE);
        format.setVCharType(HanyuPinyinVCharType.WITH_V);

        char[] input = texts.trim().toCharArray();
        String output = "";

        try {
            for (int i = 0; i < input.length; i++) {
                if (java.lang.Character.toString(input[i]).matches("[\\u4E00-\\u9FA5]+")) {
                    String[] temp = PinyinHelper.toHanyuPinyinStringArray(input[i], format);
                    output += temp[0];
                } else
                    output += java.lang.Character.toString(input[i]);
            }
        } catch (BadHanyuPinyinOutputFormatCombination e) {
            e.printStackTrace();
        }
        return output;
    }
```

8.2 **Codes for Stroke IME** (Java, retrieved from https://blog.csdn.net/tsyj810883979/article/details/6500709)

```
import java.io.FileNotFoundException;

public static void main(String[] args) throws FileNotFoundException {

        String txt = “”;

        int totalStrokeCount = 0;
        for (int i = 0; i < txt.length(); i++)
                {
                totalStrokeCount = totalStrokeCount +
getStrokeCount(String.valueOf(txt.charAt(i)));
                }
                System.out.println("Total stroke count: " + totalStrokeCount);

            }

public static int getStrokeCount(String character)
        {
          try {
             byte[] gb2312Bytes = character.getBytes("gb2312");
                 if (gb2312Bytes == null || gb2312Bytes.length > 2 || gb2312Bytes.length <= 0 ||
gb2312Bytes.length == 1)
                    {
                        return 0;
                    }
                    if (gb2312Bytes.length == 2)
                    {
                        int highByte = 256 + gb2312Bytes[0];
                        int lowByte = 256 + gb2312Bytes[1];
                        return getStrokeCount(highByte, lowByte);
                    }

                } catch (UnsupportedEncodingException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                return 0;
            }

            private static int getStrokeCount(int highByte, int lowByte)
            {
                if (highByte < 0xB0 || highByte > 0xF7 || lowByte < 0xA1 || lowByte > 0xFE)
                {
                    return 0;
                }
                int offset = (highByte - 0xB0) * (0xFE - 0xA0) + (lowByte - 0xA1);
                return gb2312StrokeCount[offset];
            }

private static int[] gb2312StrokeCount = {
                    /* B0 */
                    10,7,10, 10, 8, 10, 9, 11, 17, 14, 13, 5, 13, 10, 12, 15, 10, 6,
                    10,9,13, 8, 10, 10, 8, 8, 10, 5, 10, 14, 16, 9, 12, 12, 15, 15, 7,
                    10,5, 5, 7, 10, 2, 9, 4, 8, 12, 13, 7, 10, 7, 21, 10, 8, 5, 9, 6, 13,
                    8, 8, 9, 13, 12, 10, 13, 7, 10, 10, 8, 8, 7, 8, 7, 19, 5, 4, 8, 5,
                    9, 10, 14, 14, 9, 12, 15, 10, 15, 12, 12, 8, 9, 5, 15, 10,
                    /* B1 */
                    16, 13, 9, 12, 8, 8, 8, 7, 15, 10, 13, 19, 8, 13, 12, 8, 5, 12, 9,
                    4, 9, 10, 7, 8, 12, 12, 10, 8, 8, 5, 11, 11, 11, 9, 9, 18, 9, 12,
                    14, 4, 13, 10, 8, 14, 13, 14, 6, 10, 9, 4, 7, 13, 6, 11, 14, 5, 13,
                    16, 17, 16, 9, 18, 5, 12, 8, 9, 9, 8, 4, 16, 16, 17, 12, 9, 11, 15,
                    8, 19, 16, 7, 15, 11, 12, 16, 13, 10, 13, 7, 6, 9, 5, 8, 9, 9,
                    /* B2 */
                    10, 6, 8, 11, 15, 8, 10, 8, 12, 9, 13, 10, 14, 7, 8, 11, 11, 14,
                    12, 8, 7, 10, 2, 10, 7, 11, 4, 5, 7, 19, 10, 8, 17, 11, 12, 7, 3,
                    7, 12, 15, 8, 11, 11, 14, 16, 8, 10, 9, 11, 11, 7, 7, 10, 4, 7, 17,
                    16, 16, 15, 11, 9, 8, 12, 8, 5, 9, 7, 19, 12, 3, 9, 9, 9, 14, 12,
                    14, 7, 9, 8, 8, 10, 10, 12, 11, 14, 12, 11, 13, 11, 6, 11, 19, 8,
                    11,
                    /* B3 */
                    6, 9, 11, 4, 11, 7, 2, 12, 8, 11, 10, 12, 7, 9, 12, 15, 15, 11, 7,
                    8, 4, 7, 15, 12, 7, 15, 10, 6, 7, 6, 11, 7, 7, 7, 12, 8, 15, 10, 9,
                    16, 6, 7, 10, 12, 12, 15, 8, 8, 10, 10, 10, 6, 13, 9, 11, 6, 7, 6,
                    6, 10, 8, 8, 4, 7, 10, 5, 9, 6, 6, 6, 11, 8, 8, 13, 12, 14, 13, 13,
                    13, 4, 11, 14, 4, 10, 7, 5, 16, 12, 18, 12, 13, 12, 9, 13,
                    /* B4 */
                    10, 12, 24, 13, 13, 5, 12, 3, 9, 13, 7, 11, 12, 7, 9, 12, 15, 7, 6,
                    6, 7, 8, 11, 13, 8, 9, 13, 15, 10, 11, 7, 21, 18, 11, 11, 9, 14,
                    14, 13, 13, 10, 7, 6, 8, 12, 6, 15, 12, 7, 5, 4, 5, 11, 11, 15, 17,
                    9, 19, 16, 12, 14, 11, 13, 10, 13, 14, 11, 14, 7, 6, 3, 14, 15, 12,
                    11, 10, 13, 12, 6, 12, 14, 5, 3, 7, 4, 12, 17, 9, 9, 5, 9, 11, 9,
                    11,
                    /* B5 */
                    9, 10, 8, 4, 8, 10, 11, 9, 5, 12, 7, 11, 11, 8, 11, 11, 6, 9, 10,
                    9, 10, 2, 10, 17, 10, 7, 11, 6, 8, 15, 11, 12, 11, 15, 11, 8, 19,
                    6, 12, 12, 17, 14, 4, 12, 7, 14, 8, 10, 11, 7, 10, 14, 14, 8, 8, 6,
                    12, 11, 9, 7, 10, 12, 16, 11, 13, 13, 9, 8, 16, 9, 5, 7, 7, 8, 11,
                    12, 11, 13, 13, 5, 16, 10, 2, 11, 6, 8, 10, 12, 10, 14, 15, 8, 11,
                    13,
                    /* B6 */
                    2, 7, 5, 7, 8, 12, 13, 8, 4, 6, 5, 5, 12, 15, 6, 9, 8, 9, 7, 9, 11,
                    7, 4, 9, 7, 10, 12, 10, 13, 9, 12, 9, 10, 11, 13, 12, 7, 14, 7, 9,
                    12, 7, 14, 12, 14, 9, 11, 12, 11, 7, 4, 5, 15, 7, 19, 12, 10, 7, 9,
                    9, 12, 11, 9, 6, 6, 9, 13, 6, 13, 11, 8, 12, 11, 13, 10, 12, 9, 15,
                    6, 10, 10, 4, 7, 12, 11, 10, 10, 6, 2, 6, 5, 9, 9, 2,
                    /* B7 */
                    9, 5, 9, 12, 6, 4, 9, 8, 9, 18, 6, 12, 18, 15, 8, 8, 17, 3, 10, 4,
                    7, 8, 8, 5, 7, 7, 7, 7, 4, 8, 8, 6, 7, 6, 6, 7, 8, 11, 8, 11, 3, 8,
                    10, 10, 7, 8, 8, 8, 9, 7, 11, 7, 8, 4, 7, 7, 12, 7, 10, 8, 6, 8,
                    12, 12, 4, 9, 8, 13, 10, 12, 4, 9, 11, 10, 5, 13, 6, 8, 4, 7, 7, 4,
                    15, 8, 14, 7, 8, 13, 12, 9, 11, 6, 9, 8,
                    /* B8 */
                    10, 11, 13, 11, 5, 7, 7, 11, 10, 10, 8, 11, 12, 8, 14, 9, 11, 18,
                    12, 9, 12, 5, 8, 4, 13, 6, 12, 4, 7, 6, 13, 8, 15, 14, 8, 7, 13, 9,
                    11, 12, 3, 5, 7, 9, 9, 7, 10, 13, 8, 11, 21, 4, 6, 9, 9, 7, 7, 7,
                    12, 7, 16, 10, 10, 14, 10, 16, 13, 15, 15, 7, 10, 14, 12, 4, 11,
                    10, 8, 12, 9, 12, 10, 12, 9, 12, 11, 3, 6, 9, 10, 13, 10, 7, 8, 19,
                    /* B9 */
                    10, 10, 11, 3, 7, 5, 10, 11, 8, 10, 4, 9, 3, 6, 7, 9, 7, 6, 9, 4,
                    7, 8, 8, 9, 8, 8, 11, 12, 11, 8, 14, 7, 8, 8, 8, 13, 5, 11, 9, 7,
                    8, 9, 10, 8, 12, 8, 5, 9, 14, 9, 13, 8, 8, 8, 12, 6, 8, 9, 6, 14,
                    11, 23, 12, 20, 8, 6, 3, 10, 13, 8, 6, 11, 5, 7, 9, 6, 9, 8, 9, 10,
                    8, 13, 9, 8, 12, 13, 12, 12, 10, 8, 8, 14, 6, 9, 15, 9, 10, 10, 6,
                    10, 9, 12, 14, 7, 12, 7, 11, 12, 8, 12, 7, 16, 16, 10, 7, 16, 10,
                    11, 6, 5, 5, 8, 10, 17, 17, 14, 11, 9, 6, 10, 5, 10, 8, 12, 10, 11,
                    10, 5, 8, 7, 6, 11, 13, 9, 8, 11, 14, 14, 15, 9, 15, 12, 11, 9, 9,
                    9, 10, 7, 15, 16, 9, 8, 9, 10, 9, 11, 9, 7, 5, 6, 12, 9, 12, 7, 9,
                    10, 6, 8, 5, 8, 13, 10, 12, 9, 15, 8, 15, 12,
                    /* BB */
                    8, 8, 11, 7, 4, 7, 4, 7, 9, 6, 12, 12, 8, 6, 4, 8, 13, 9, 7, 11, 7,
                    6, 8, 10, 7, 12, 10, 11, 10, 12, 13, 11, 10, 9, 4, 9, 12, 11, 16,
                    15, 17, 9, 11, 12, 13, 10, 13, 9, 11, 6, 9, 12, 17, 9, 12, 6, 13,
                    10, 15, 5, 12, 11, 10, 11, 6, 10, 5, 6, 9, 9, 9, 8, 11, 13, 9, 11,
                    17, 9, 6, 4, 10, 8, 12, 16, 8, 11, 5, 6, 11, 6, 13, 15, 10, 14,
                    /* BC */
                    6, 5, 9, 16, 4, 7, 10, 11, 12, 6, 7, 12, 13, 20, 12, 3, 9, 10, 6,
                    7, 13, 6, 9, 2, 10, 3, 13, 7, 16, 8, 6, 11, 8, 11, 9, 11, 11, 4, 5,
                    9, 7, 7, 7, 10, 6, 14, 9, 6, 8, 10, 5, 9, 12, 10, 5, 10, 11, 15, 6,
                    9, 8, 13, 7, 10, 7, 6, 11, 7, 13, 10, 8, 8, 6, 12, 9, 11, 9, 14,
                    12, 8, 10, 13, 9, 11, 11, 9, 14, 13, 12, 9, 4, 13, 15, 6,
                    /* BD */
                    10, 10, 9, 8, 11, 12, 10, 8, 15, 9, 9, 10, 6, 19, 12, 10, 9, 6, 6,
                    13, 8, 15, 12, 17, 12, 10, 6, 8, 9, 9, 9, 20, 12, 11, 11, 8, 11, 9,
                    7, 9, 16, 9, 13, 11, 14, 10, 10, 5, 12, 12, 11, 9, 11, 12, 6, 14,
                    7, 5, 10, 8, 11, 13, 14, 9, 9, 13, 8, 7, 17, 7, 9, 10, 4, 9, 9, 8,
                    3, 12, 4, 8, 4, 9, 18, 10, 13, 4, 13, 7, 13, 10, 13, 7, 10, 10,
                    /* BE */
                    6, 7, 9, 14, 8, 13, 12, 16, 8, 11, 14, 13, 8, 4, 19, 12, 11, 14,
                    14, 12, 16, 8, 10, 13, 11, 10, 8, 9, 12, 12, 7, 5, 7, 9, 3, 7, 2,
                    10, 11, 11, 5, 6, 13, 8, 12, 8, 17, 8, 8, 10, 8, 8, 11, 7, 8, 9, 9,
                    8, 14, 7, 11, 4, 8, 11, 15, 13, 10, 5, 11, 8, 10, 10, 12, 10, 10,
                    11, 8, 10, 15, 23, 7, 11, 10, 17, 9, 6, 6, 9, 7, 11, 9, 6, 7, 10,
                    /* BF */
                    9, 12, 10, 9, 10, 12, 8, 5, 9, 4, 12, 13, 8, 12, 5, 12, 11, 7, 9,
                    9, 11, 14, 17, 6, 7, 4, 8, 6, 9, 10, 15, 8, 8, 9, 12, 15, 14, 9, 7,
                    9, 5, 12, 7, 8, 9, 10, 8, 11, 9, 10, 7, 7, 8, 10, 4, 11, 7, 3, 6,
                    11, 9, 10, 13, 8, 14, 7, 12, 6, 9, 9, 13, 10, 7, 13, 8, 7, 10, 12,
                    6, 12, 7, 10, 8, 11, 7, 7, 3, 11, 8, 13, 12, 9, 13, 11,
                    /* C0 */
                    12, 12, 12, 8, 8, 10, 7, 9, 6, 13, 12, 8, 8, 12, 14, 12, 14, 11,
                    10, 7, 13, 13, 11, 9, 8, 16, 12, 5, 15, 14, 12, 9, 16, 12, 9, 13,
                    11, 12, 10, 11, 8, 10, 10, 10, 7, 7, 6, 8, 9, 13, 10, 10, 11, 5,
                    13, 18, 16, 15, 11, 17, 9, 16, 6, 9, 8, 12, 13, 7, 9, 11, 11, 15,
                    16, 10, 10, 13, 11, 7, 7, 15, 5, 10, 9, 6, 10, 7, 5, 7, 10, 4, 7,
                    12, 8, 9,
                    /* C1 */
                    12, 5, 11, 7, 8, 2, 14, 10, 9, 12, 10, 7, 18, 13, 8, 10, 8, 11, 11,
                    12, 10, 9, 8, 13, 10, 11, 13, 7, 7, 11, 12, 12, 9, 10, 15, 11, 14,
                    7, 16, 14, 5, 15, 2, 14, 17, 14, 10, 6, 12, 10, 6, 11, 12, 8, 17,
                    16, 9, 7, 20, 11, 15, 10, 7, 8, 9, 11, 13, 13, 10, 7, 11, 10, 7,
                    10, 8, 11, 5, 5, 13, 11, 14, 12, 13, 10, 6, 15, 10, 9, 4, 5, 11, 8,
                    11, 16,
                    /* C2 */
                    11, 8, 8, 7, 13, 9, 12, 15, 14, 8, 7, 5, 11, 7, 8, 11, 7, 8, 12,
                    19, 13, 21, 13, 10, 11, 16, 12, 8, 7, 15, 7, 6, 11, 8, 10, 15, 12,
                    12, 10, 12, 9, 11, 13, 11, 9, 10, 9, 13, 7, 7, 11, 11, 7, 8, 6, 4,
                    7, 7, 6, 11, 17, 8, 11, 13, 14, 14, 13, 12, 9, 9, 9, 6, 11, 7, 8,
                    9, 3, 9, 14, 6, 10, 6, 7, 8, 6, 9, 15, 14, 12, 13, 14, 11, 14, 14,
                    /* C3 */
                    13, 6, 9, 8, 8, 6, 10, 11, 8, 13, 4, 5, 10, 5, 8, 9, 12, 14, 9, 3,
                    8, 8, 11, 14, 15, 13, 7, 9, 12, 14, 7, 9, 9, 12, 8, 12, 3, 7, 5,
                    11, 13, 17, 13, 13, 11, 11, 8, 11, 15, 19, 17, 9, 11, 8, 6, 10, 8,
                    8, 14, 11, 12, 12, 10, 11, 11, 7, 9, 10, 12, 9, 8, 11, 13, 17, 9,
                    12, 8, 7, 14, 5, 5, 8, 5, 11, 10, 9, 8, 16, 8, 11, 6, 8, 13, 13,
                    /* C4 */
                    14, 19, 14, 14, 16, 15, 20, 8, 5, 10, 15, 16, 8, 13, 13, 8, 11, 6,
                    9, 8, 7, 7, 8, 5, 13, 14, 13, 12, 14, 4, 5, 13, 8, 16, 10, 9, 7, 9,
                    6, 9, 7, 6, 2, 5, 9, 8, 9, 7, 10, 22, 9, 10, 9, 8, 11, 8, 10, 4,
                    14, 10, 8, 16, 10, 8, 5, 7, 7, 10, 13, 9, 13, 14, 8, 6, 15, 15, 11,
                    8, 10, 14, 5, 7, 10, 10, 19, 11, 15, 15, 10, 11, 9, 8, 16, 5,
                    /* C5 */
                    8, 8, 4, 7, 9, 7, 10, 9, 6, 7, 5, 7, 9, 3, 13, 9, 8, 9, 17, 20, 10,
                    10, 8, 9, 8, 18, 7, 11, 7, 11, 9, 8, 8, 8, 12, 8, 11, 12, 11, 12,
                    9, 19, 15, 11, 15, 9, 10, 7, 9, 6, 8, 10, 16, 9, 7, 8, 7, 9, 10,
                    12, 8, 8, 9, 11, 14, 12, 10, 10, 8, 7, 12, 9, 10, 8, 11, 15, 12,
                    13, 12, 13, 16, 16, 8, 13, 11, 13, 8, 9, 21, 7, 8, 15, 12, 9,
                    /* C6 */
                    11, 12, 10, 5, 4, 12, 15, 7, 20, 15, 11, 4, 12, 15, 14, 16, 11, 14,
                    16, 9, 13, 8, 9, 13, 6, 8, 8, 11, 5, 8, 10, 7, 9, 8, 8, 11, 11, 10,
                    14, 8, 11, 10, 5, 12, 4, 10, 12, 11, 13, 10, 6, 10, 12, 10, 14, 19,
                    18, 12, 12, 10, 11, 8, 2, 10, 14, 9, 7, 8, 12, 8, 8, 11, 11, 10, 6,
                    14, 8, 6, 11, 10, 6, 3, 6, 7, 9, 9, 16, 4, 6, 7, 7, 8, 5, 11,
                    /* C7 */
                    9, 9, 9, 6, 8, 10, 3, 6, 13, 5, 12, 11, 16, 10, 10, 9, 15, 13, 8,
                    15, 11, 12, 4, 14, 8, 7, 12, 7, 14, 14, 12, 7, 16, 14, 14, 10, 10,
                    17, 6, 8, 5, 16, 15, 12, 10, 9, 10, 4, 8, 5, 8, 9, 9, 9, 9, 10, 12,
                    13, 7, 15, 12, 13, 7, 8, 9, 9, 10, 10, 11, 16, 12, 12, 11, 8, 10,
                    6, 12, 7, 9, 5, 7, 11, 7, 5, 9, 8, 12, 4, 11, 6, 11, 8, 7, 11,
                    /* C8 */
                    8, 11, 17, 15, 5, 11, 23, 6, 16, 10, 6, 11, 10, 4, 8, 4, 10, 8, 16,
                    7, 13, 14, 12, 11, 12, 13, 12, 16, 5, 9, 22, 20, 20, 20, 5, 9, 7,
                    9, 12, 10, 4, 4, 2, 7, 7, 6, 4, 3, 7, 6, 5, 4, 4, 6, 9, 13, 9, 16,
                    14, 13, 10, 9, 4, 12, 9, 6, 9, 20, 16, 17, 6, 10, 8, 6, 2, 15, 8,
                    6, 15, 13, 12, 7, 10, 8, 10, 15, 9, 11, 13, 17, 13, 14, 3, 8,
                    /* C9 */
                    6, 12, 10, 13, 8, 12, 12, 6, 12, 13, 6, 10, 12, 14, 10, 9, 6, 8, 7,
                    7, 13, 11, 13, 12, 10, 9, 8, 7, 3, 7, 14, 8, 5, 8, 16, 17, 16, 12,
                    6, 10, 15, 14, 6, 11, 12, 10, 3, 8, 14, 11, 10, 12, 10, 6, 3, 14,
                    4, 10, 7, 8, 11, 11, 11, 6, 8, 11, 13, 10, 13, 10, 7, 6, 10, 5, 8,
                    7, 7, 11, 10, 8, 9, 7, 8, 11, 9, 8, 13, 11, 7, 5, 12, 9, 4, 11,
                    /* CA */
                    9, 11, 12, 9, 5, 6, 5, 9, 9, 12, 8, 3, 8, 2, 5, 9, 7, 4, 9, 9, 8,
                    7, 5, 5, 8, 9, 8, 8, 6, 5, 3, 5, 9, 8, 9, 14, 10, 8, 9, 13, 16, 9,
                    5, 8, 12, 8, 4, 5, 9, 9, 8, 8, 6, 4, 9, 6, 7, 11, 11, 8, 14, 11,
                    15, 8, 11, 10, 7, 13, 8, 12, 11, 12, 4, 12, 11, 15, 16, 12, 17, 13,
                    13, 12, 13, 12, 5, 8, 9, 7, 6, 9, 14, 11, 13, 14,
                    /* CB */
                    10, 8, 9, 14, 10, 5, 5, 10, 9, 17, 4, 11, 10, 4, 13, 12, 7, 17, 9,
                    12, 9, 11, 10, 9, 12, 15, 15, 9, 7, 5, 5, 6, 13, 6, 13, 5, 7, 6, 8,
                    3, 8, 10, 8, 10, 9, 7, 6, 9, 12, 15, 16, 14, 7, 12, 9, 10, 10, 12,
                    14, 13, 13, 11, 7, 8, 14, 13, 14, 9, 11, 11, 10, 21, 13, 6, 17, 12,
                    14, 10, 6, 10, 10, 13, 11, 10, 14, 11, 10, 12, 8, 13, 5, 5, 6, 12,
                    /* CC */
                    16, 9, 17, 15, 9, 8, 8, 5, 10, 11, 4, 8, 7, 7, 13, 8, 15, 13, 7,
                    17, 13, 15, 14, 10, 8, 12, 10, 14, 11, 5, 9, 6, 13, 13, 11, 12, 15,
                    10, 16, 10, 15, 11, 15, 10, 11, 10, 13, 10, 11, 10, 9, 11, 10, 5,
                    10, 10, 18, 13, 10, 13, 11, 10, 15, 12, 12, 15, 16, 12, 7, 12, 17,
                    11, 10, 9, 8, 4, 11, 13, 5, 11, 9, 14, 12, 9, 7, 8, 11, 13, 9, 10,
                    8, 4, 7, 9,
                    /* CD */
                    5, 6, 11, 9, 9, 9, 12, 10, 10, 13, 17, 6, 11, 7, 12, 11, 10, 12, 9,
                    12, 11, 7, 5, 10, 5, 7, 9, 8, 10, 10, 10, 11, 3, 6, 8, 12, 6, 11,
                    13, 13, 13, 14, 9, 7, 4, 17, 8, 6, 11, 10, 7, 6, 8, 12, 7, 8, 12,
                    9, 9, 12, 9, 9, 4, 10, 9, 5, 15, 9, 12, 8, 10, 3, 11, 7, 13, 10,
                    11, 12, 11, 8, 11, 3, 12, 7, 4, 3, 8, 6, 8, 8, 11, 7, 6, 9,
                    /* CE */
                    20, 13, 6, 4, 7, 10, 7, 11, 11, 4, 14, 11, 7, 11, 8, 6, 6, 7, 7, 5,
                    14, 8, 9, 9, 12, 17, 7, 12, 11, 11, 15, 3, 14, 12, 10, 4, 9, 7, 7,
                    14, 10, 6, 13, 10, 8, 9, 13, 10, 12, 7, 14, 8, 12, 7, 7, 7, 9, 4,
                    6, 9, 9, 4, 7, 11, 7, 7, 4, 8, 4, 10, 4, 14, 6, 9, 7, 5, 13, 11, 8,
                    4, 5, 10, 9, 8, 14, 8, 6, 11, 8, 12, 15, 6, 13, 10,
                    /* CF */
                    12, 10, 7, 11, 15, 3, 11, 14, 11, 13, 6, 12, 17, 11, 10, 3, 13, 12,
                    11, 9, 7, 12, 6, 8, 15, 9, 7, 17, 14, 13, 9, 8, 9, 3, 12, 10, 6,
                    11, 13, 6, 5, 14, 6, 9, 8, 11, 11, 7, 9, 8, 13, 9, 9, 8, 13, 7, 13,
                    11, 12, 9, 10, 8, 8, 9, 11, 22, 9, 15, 17, 12, 3, 12, 10, 8, 13, 9,
                    8, 9, 9, 15, 13, 6, 11, 11, 12, 15, 9, 10, 18, 12, 10, 10, 11, 10,
                    /* D0 */
                    3, 7, 10, 7, 11, 10, 10, 13, 8, 13, 15, 15, 6, 9, 13, 6, 11, 8, 11,
                    5, 11, 9, 19, 16, 8, 8, 12, 10, 16, 7, 12, 8, 7, 13, 7, 4, 9, 11,
                    9, 13, 12, 12, 6, 6, 9, 7, 6, 6, 16, 8, 7, 8, 8, 5, 4, 10, 6, 7,
                    12, 14, 6, 9, 10, 6, 13, 12, 7, 10, 10, 14, 6, 14, 11, 14, 9, 10,
                    6, 13, 11, 9, 6, 7, 10, 9, 12, 12, 11, 11, 7, 12, 9, 11, 11, 5,
                    /* D1 */
                    9, 19, 10, 9, 13, 16, 8, 5, 11, 6, 9, 14, 12, 6, 8, 6, 6, 6, 10, 6,
                    5, 5, 9, 6, 6, 8, 9, 10, 7, 3, 7, 4, 10, 11, 13, 11, 12, 9, 6, 6,
                    11, 9, 11, 10, 11, 10, 7, 9, 12, 8, 7, 7, 15, 11, 8, 8, 8, 11, 11,
                    9, 14, 10, 12, 16, 6, 9, 12, 10, 9, 12, 10, 11, 10, 9, 5, 10, 10,
                    7, 6, 8, 8, 6, 9, 6, 10, 6, 11, 9, 10, 14, 16, 13, 7, 14,
                    /* D2 */
                    13, 6, 13, 11, 12, 9, 9, 10, 9, 9, 20, 12, 15, 8, 6, 11, 7, 3, 6,
                    11, 5, 5, 6, 12, 8, 11, 1, 12, 7, 12, 11, 8, 6, 6, 13, 6, 12, 11,
                    5, 10, 14, 7, 8, 9, 18, 12, 9, 10, 3, 1, 7, 4, 4, 7, 8, 7, 6, 3, 7,
                    17, 11, 13, 9, 6, 13, 13, 15, 4, 3, 10, 13, 8, 5, 10, 7, 6, 17, 11,
                    8, 9, 9, 6, 10, 9, 6, 8, 7, 11, 11, 11, 7, 4, 4, 11,
                    /* D3 */
                    5, 8, 15, 11, 18, 7, 14, 10, 11, 11, 9, 14, 7, 17, 9, 15, 13, 12,
                    9, 9, 8, 7, 17, 10, 11, 13, 14, 13, 8, 8, 10, 5, 11, 9, 5, 9, 6,
                    11, 7, 4, 5, 7, 10, 7, 8, 12, 7, 6, 4, 5, 7, 12, 9, 2, 5, 6, 11, 3,
                    8, 13, 13, 13, 14, 7, 9, 12, 8, 12, 12, 11, 11, 4, 10, 8, 3, 6, 9,
                    6, 9, 6, 5, 11, 6, 8, 6, 12, 12, 10, 12, 13, 11, 9, 8, 13,
                    /* D4 */
                    10, 12, 12, 10, 15, 5, 10, 11, 10, 4, 9, 10, 10, 12, 14, 7, 7, 10,
                    13, 13, 12, 7, 8, 14, 9, 9, 4, 6, 12, 11, 9, 8, 12, 4, 10, 10, 10,
                    4, 9, 4, 9, 4, 7, 15, 11, 10, 13, 5, 5, 10, 6, 10, 9, 7, 10, 10, 6,
                    6, 9, 19, 12, 16, 10, 10, 12, 14, 17, 12, 19, 8, 6, 16, 9, 20, 16,
                    10, 7, 7, 17, 8, 8, 6, 8, 10, 9, 15, 15, 12, 16, 4, 12, 12, 5, 5,
                    /* D5 */
                    11, 8, 9, 9, 14, 8, 5, 9, 7, 14, 10, 6, 10, 10, 14, 18, 9, 13, 11,
                    8, 10, 8, 14, 11, 10, 22, 9, 5, 9, 10, 12, 11, 15, 11, 14, 14, 7,
                    12, 10, 7, 3, 7, 8, 5, 8, 16, 13, 8, 9, 7, 8, 9, 13, 13, 6, 14, 5,
                    14, 7, 10, 12, 16, 8, 13, 14, 7, 10, 9, 13, 10, 13, 10, 16, 6, 7,
                    8, 8, 10, 7, 15, 10, 15, 6, 13, 9, 11, 8, 9, 6, 8, 16, 9, 5, 9,
                    /* D6 */
                    9, 10, 8, 7, 6, 8, 4, 7, 14, 8, 8, 10, 5, 3, 8, 11, 8, 12, 12, 6,
                    10, 8, 7, 9, 4, 11, 5, 6, 7, 7, 10, 11, 6, 10, 13, 8, 9, 8, 12, 10,
                    13, 8, 8, 11, 12, 8, 11, 4, 9, 8, 9, 10, 8, 9, 8, 9, 6, 6, 6, 8, 6,
                    9, 7, 12, 9, 7, 8, 8, 10, 8, 9, 17, 10, 10, 12, 6, 11, 10, 8, 10,
                    6, 10, 12, 8, 17, 15, 5, 11, 9, 7, 11, 8, 12, 12,
                    /* D7 */
                    7, 8, 9, 8, 7, 4, 9, 4, 9, 8, 15, 14, 15, 10, 6, 12, 6, 15, 6, 7,
                    12, 13, 9, 14, 7, 11, 10, 10, 10, 8, 8, 10, 12, 8, 10, 11, 11, 7,
                    9, 9, 9, 10, 9, 12, 11, 7, 12, 5, 9, 13, 3, 6, 11, 6, 18, 12, 15,
                    8, 11, 9, 7, 7, 7, 9, 12, 10, 7, 8, 11, 9, 7, 7, 8, 10, 20, 16, 15,
                    12, 13, 12, 15, 9, 5, 7, 9, 11, 7, 7, 10, 0, 0, 0, 0, 0,
                    /* D8 */
                    3, 3, 3, 4, 4, 4, 5, 6, 6, 10, 10, 16, 1, 8, 1, 2, 3, 4, 4, 5, 5,
                    6, 9, 11, 14, 14, 19, 1, 8, 14, 2, 6, 4, 7, 7, 11, 14, 4, 6, 10,
                    11, 12, 14, 15, 16, 2, 5, 8, 11, 11, 15, 8, 7, 2, 4, 6, 7, 8, 8, 8,
                    9, 10, 10, 10, 13, 13, 14, 14, 15, 16, 2, 8, 2, 4, 4, 4, 5, 5, 5,
                    5, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7,
                    /* D9 */
                    7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9,
                    9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11,
                    11, 11, 11, 11, 11, 11, 12, 12, 12, 13, 14, 14, 14, 14, 14, 14, 15,
                    15, 5, 6, 7, 7, 9, 17, 6, 8, 4, 12, 16, 17, 18, 21, 2, 9, 9, 11, 6,
                    6, 7, 2, 8, 10, 10, 11, 12, 12, 12, 13, 16, 19, 19, 2, 6, 8, 8,
                    /* DA */
                    10, 2, 10, 10, 2, 5, 5, 5, 6, 6, 6, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8,
                    8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 11,
                    11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 13, 13,
                    14, 14, 14, 15, 15, 19, 2, 8, 2, 5, 5, 6, 6, 7, 7, 7, 7, 8, 9, 9,
                    10, 10, 10, 11, 11, 11, 16, 5, 5, 5, 5, 6, 6, 7, 7, 7, 7,
                    /* DB */
                    7, 7, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 10, 10, 11, 11, 13, 13,
                    13, 14, 14, 16, 19, 17, 5, 7, 5, 7, 7, 8, 10, 10, 11, 15, 9, 17,
                    20, 2, 2, 6, 10, 2, 5, 10, 12, 7, 9, 9, 14, 16, 16, 17, 6, 6, 6, 6,
                    6, 6, 6, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9,
                    9, 9, 9, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11,
                    /* DC */
                    11, 11, 11, 11, 11, 12, 12, 12, 12, 13, 13, 14, 14, 14, 15, 20, 21,
                    22, 3, 5, 5, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
                    7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
                    8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
                    9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
                    /* DD */
                    9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                    10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11,
                    11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
                    11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
                    12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
                    13, 14, 14, 14, 14, 14, 14, 14,
                    /* DE */
                    14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16,
                    16, 16, 16, 16, 16, 17, 17, 17, 17, 17, 18, 19, 19, 19, 20, 20, 22,
                    3, 9, 6, 7, 9, 9, 10, 10, 11, 3, 5, 5, 12, 3, 6, 7, 8, 8, 8, 8, 9,
                    9, 9, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
                    12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 13, 13,
                    13, 14, 14, 14, 14,
                    /* DF */
```

```
14, 15, 15, 15, 15, 16, 16, 16, 17, 17, 19, 23, 25, 3, 7, 8, 12, 5,
5, 5, 5, 5, 5, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
/* E0 */
11, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14,
14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 16,
16, 16, 16, 16, 17, 17, 19, 25, 3, 6, 6, 7, 7, 8, 9, 10, 11, 11,
16, 7, 8, 8, 8, 10, 11, 11,
/* E1 */
11, 12, 14, 14, 15, 15, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 10, 10, 11, 11, 11, 11, 11, 11,
11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 14, 15, 15,
17, 17, 19, 3, 7, 8, 9, 9, 9, 10, 11, 11, 12, 13, 15, 16, 24, 3, 3,
5, 6, 6, 6, 7, 7, 8, 8, 8, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10,
/* E2 */
10, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 14, 14, 15,
15, 16, 17, 20, 6, 14, 12, 14, 3, 3, 6, 7, 7, 7, 7, 7, 8, 9, 10,
10, 11, 12, 12, 13, 13, 14, 15, 15, 25, 5, 7, 7, 8, 9, 9, 11, 11,
11, 11, 12, 13, 14, 15, 16, 16, 17, 3, 5, 6, 6, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9,
/* E3 */
9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11,
11, 12, 12, 12, 12, 12, 12, 12, 13, 13, 14, 15, 15, 15, 16, 16, 18,
8, 17, 4, 6, 7, 7, 7, 7, 9, 9, 10, 10, 10, 11, 11, 11, 11, 11, 11,
12, 12, 13, 13, 13, 14, 3, 4, 8, 3, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
/* E4 */
9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11,
11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15,
15, 15, 15, 15, 16,
/* E5 */
16, 16, 16, 16, 16, 17, 17, 17, 17, 17, 19, 19, 19, 20, 20, 21, 24,
3, 5, 8, 8, 9, 10, 12, 13, 14, 14, 15, 16, 16, 17, 17, 3, 7, 7, 8,
8, 8, 8, 8, 8, 8, 9, 9, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 12,
12, 12, 12, 13, 13, 13, 13, 15, 15, 16, 16, 17, 17, 18, 3, 11, 9,
12, 5, 9, 10, 10, 12, 14, 15, 21, 8, 8, 9, 11, 12, 22, 3, 6, 6, 7,
7, 7, 7,
/* E6 */
7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10,
10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 13, 13,
13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 15, 16, 16, 17, 17, 20,
5, 9, 7, 8, 12, 3, 3, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 10, 11, 11,
11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 19, 20, 3, 6, 6, 6, 6, 6,
/* E7 */
7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 10, 10, 10, 11, 11, 11, 11,
11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 15, 15, 15, 16,
16, 16, 16, 19, 3, 15, 3, 8, 10, 6, 6, 8, 8, 8, 9, 9, 9, 9, 9, 9,
9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 12, 12, 12, 12, 12,
12, 12, 12,
/* E8 */
12, 12, 13, 13, 13, 13, 13, 14, 14, 15, 15, 15, 15, 15, 15, 15, 16,
17, 17, 17, 18, 20, 20, 13, 13, 14, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12,
12,
/* E9 */
12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16,
16, 16, 16, 16, 16, 16, 16, 17, 17, 17, 17, 18, 13, 14, 8, 9, 9, 9,
11, 11, 11, 12, 12, 14, 16, 7, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 10,
10, 10, 10, 11, 12, 12,
/* EA */
12, 12, 13, 15, 16, 10, 5, 8, 11, 12, 12, 13, 13, 13, 14, 14, 8, 9,
12, 16, 16, 17, 4, 6, 6, 7, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9,
10, 10, 10, 10, 10, 10, 11, 11, 12, 13, 13, 14, 14, 16, 18, 18, 20,
21, 9, 9, 9, 9, 10, 10, 10, 10, 11, 11, 11, 12, 12, 14, 9, 10, 11,
12, 13, 14, 15, 15, 9, 13, 6, 8, 9, 11, 11, 12, 12, 12, 13, 14, 10,
11, 12,
/* EB */
14, 17, 10, 10, 12, 12, 12, 13, 15, 16, 16, 22, 5, 6, 7, 7, 9, 10,
10, 11, 13, 4, 11, 13, 12, 13, 15, 9, 15, 6, 7, 7, 7, 8, 8, 8, 8,
8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10,
10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12,
12, 13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 15, 15, 16, 17, 17,
17, 17,
/* EC */
17, 16, 7, 11, 12, 13, 13, 16, 9, 9, 12, 13, 16, 16, 4, 13, 13, 17,
12, 15, 16, 8, 10, 10, 10, 11, 11, 13, 14, 7, 8, 8, 8, 9, 9, 9, 9,
9, 10, 10, 11, 11, 11, 12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 14,
15, 15, 15, 15, 16, 16, 16, 18, 21, 30, 4, 11, 13, 16, 8, 8, 9, 11,
12, 4, 7, 8, 8, 9, 9, 9, 9, 9, 9, 9, 10, 10, 12, 12, 13, 14, 16,
21, 7, 7,
/* ED */
9, 10, 10, 10, 10, 10, 10, 11, 13, 13, 14, 16, 16, 17, 17, 24, 4,
6, 8, 9, 12, 7, 8, 8, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 12, 13, 13, 13, 13,
13, 14, 14, 14, 14, 14, 15, 15, 15, 16, 16, 17, 17, 18, 19, 18, 21,
11, 12, 17, 19, 8, 9, 9, 9, 9, 9, 10, 10, 10, 11, 11, 11, 11, 12,
12, 12, 12, 13, 13,
/* EE */
13, 13, 14, 14, 14, 14, 15, 15, 16, 16, 16, 17, 18, 7, 8, 9, 9, 9,
10, 12, 13, 17, 9, 10, 10, 12, 13, 14, 14, 16, 17, 17, 10, 16, 23,
5, 6, 6, 7, 7, 7, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
11, 11, 11,
/* EF */
11, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15,
15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 17, 17, 17,
17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 19, 20, 14, 9, 12, 13, 9,
9, 10, 10, 11, 12, 12, 12, 13, 13,
/* F0 */
15, 15, 16, 17, 18, 22, 9, 11, 12, 13, 17, 10, 11, 7, 7, 8, 9, 9,
10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12,
13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 15, 15, 16, 16, 16, 17, 17,
17, 17, 18, 18, 22, 5, 7, 7, 8, 8, 9, 9, 10, 10, 10, 10, 10, 10,
10, 10, 11, 11, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 13, 13,
14, 14, 14, 14, 14, 14, 14,
/* F1 */
15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16, 17, 18, 18, 18, 18,
21, 23, 11, 12, 8, 8, 9, 9, 10, 11, 13, 13, 14, 14, 14, 15, 5, 8,
9, 9, 9, 9, 10, 11, 11, 11, 11, 12, 12, 12, 12, 13, 13, 13, 13, 13,
13, 14, 14, 14, 14, 14, 15, 15, 16, 17, 19, 24, 5, 9, 11, 12, 9, 6,
9, 10, 12, 12, 13, 14, 15, 15, 16, 16, 22, 12, 8, 11, 11, 11, 12,
15, 16, 12, 9, 10, 10,
/* F2 */
12, 12, 12, 12, 13, 15, 15, 16, 16, 16, 18, 20, 21, 6, 10, 7, 8, 9,
9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11,
11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 16, 16, 16, 16,
/* F3 */
16, 16, 16, 16, 16, 16, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17,
18, 18, 18, 18, 19, 19, 19, 19, 20, 21, 24, 26, 6, 14, 17, 17, 10,
8, 9, 9, 9, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11,
        11, 11, 11, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 13, 14, 14,
        14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16,
        16, 16, 17, 17, 17, 17, 17, 17,
        /* F4 */
        18, 18, 18, 19, 19, 19, 8, 9, 11, 12, 10, 10, 9, 9, 9, 10, 10, 10,
        10, 11, 11, 11, 11, 12, 13, 13, 14, 15, 17, 18, 19, 10, 10, 11, 13,
        13, 19, 11, 11, 13, 15, 15, 16, 9, 10, 10, 11, 11, 12, 12, 13, 14,
        14, 14, 15, 15, 15, 15, 15, 16, 18, 6, 15, 9, 11, 12, 14, 14, 15,
        15, 16, 17, 6, 12, 14, 14, 17, 25, 11, 19, 9, 12, 13, 13, 23, 11,
        15, 10, 11, 9, 10, 10, 10, 12,
        /* F5 */
        12, 12, 13, 13, 13, 14, 14, 14, 14, 14, 15, 15, 16, 16, 16, 17, 17,
        18, 19, 19, 19, 20, 20, 21, 7, 16, 10, 13, 14, 18, 18, 10, 10, 11,
        11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 13, 13,
        14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16,
        16, 17, 17, 17, 19, 19, 19, 19, 19, 20, 21, 22, 22, 23, 24, 7, 12,
        13, 13, 17, 17, 11, 11, 12, 12, 13,
        /* F6 */
        13, 14, 15, 13, 18, 12, 11, 12, 12, 14, 14, 16, 16, 16, 19, 19, 20,
        22, 10, 13, 13, 13, 14, 14, 15, 15, 17, 8, 12, 20, 8, 10, 10, 13,
        14, 18, 18, 14, 14, 15, 16, 17, 18, 18, 21, 24, 12, 12, 13, 13, 13,
        13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15,
        15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16,
        16, 17, 17, 17, 17, 17, 17, 17, 17,
        /* F7 */
        18, 18, 18, 18, 18, 19, 19, 19, 19, 19, 19, 20, 20, 20, 21, 14, 14,
        15, 15, 16, 18, 18, 18, 19, 19, 13, 13, 14, 14, 14, 15, 15, 17, 17,
        18, 18, 19, 19, 22, 14, 14, 15, 16, 16, 17, 19, 12, 15, 18, 22, 22,
        10, 13, 14, 15, 15, 16, 16, 16, 18, 19, 20, 23, 25, 14, 15, 17, 13,
        16, 16, 17, 19, 19, 21, 23, 17, 17, 17, 18, 18, 19, 20, 20, 20, 20,
        21, 17, 18, 20, 23, 23, 16, 17, 23
        /* F8 */
};
```

8.3 Excerpts Used for Independent Variables

Death's End in Remembrance of Earth's Past trilogy:

String txt = "两颗类木巨行星已经被二维化天王星的轨道在土星之外但由于前者目前正处于太阳的另一侧首先跌落到二维的是土星二维化后的巨行星应该是圆形只是从冥王星上看视线与二维空间平面有一个角度于是它们在视野中变成了椭圆两颗二维行星呈现出清晰的环层结构二维海王星主要有三个环区最外层是蓝色的环看上去十分艳丽像这只眼睛的睫毛和眼影那是由氢气和氦气构成的大气层中部是白色环这是海王星厚达两万千米的地幔曾被行星天文学家称为水氨大洋中心的深色区是行星核由岩石和冰组成质量相当于一个地球二维土星的结构类似只是外侧没有蓝色环每个大环区中还有无数更细小的环区构成精细的结构细看时这两只巨眼变得像两个年轮刚刚锯断的大树"露出的那种崭新的年轮每颗二维行星的附近都有十几个小圆形那是它们被二维化的卫星土星外侧还有淡淡的一个大圆是二维化的土星环太空中仍能够找到太阳仍然是一个刚能看出形状的小圆盘发出无力的黄光而两颗行星远在太阳的另一侧可见它们二维化后面积的巨大"

Steve Jobs:

String txt = “乔布斯夫妇有一位朋友的丈夫是微软的工程师当时在进行平板电脑的研发这位工程师五十岁生日时举办了一场晚宴邀请乔布斯夫妇和盖茨夫妇出席乔布斯有些不情愿地去了其实史蒂夫那天晚上对我挺友好盖茨回忆说但却对寿星不是特别友好那位工程师不停地透露微软平板电脑的情况这让盖茨很恼火他是我们的员工掌握着我们的知识产权盖茨回忆道乔布斯同样很恼火盖茨担心会引发自己不想看到的后果他的担心成真了乔布斯回忆道这个家伙缠着我说微软这款平板电脑软件将如何彻底改变世界淘汰所有的笔记本电脑苹果应该使用他开发的微软软件但是他设计的这个产品完全错了就是因为配有一支手写笔而只要有手写笔这产品就废了晚宴上他跟我说了近十遍这些玩意儿我都烦死了回到家我就说去他妈的让我们告诉他真正的平板电脑应该是什么样第二天乔布斯一来到公司就召集自己的团队说我要做一款平板电脑不要键盘和手写笔用户能够通过手指触摸屏幕输入”

The Little Prince:

String txt = “从此我孤独地生活着没有一个可以推心置腹的朋友这种状况一直延续至六年前六年前我的飞机出了故障发动机里的某个部件被撞坏了我被迫在撒哈拉沙漠降落身边没有机械师没有一个乘客我只好勉为其难自己动手试着修理部件我带的水仅够喝一个星期能否修好飞机关系到我的生死存亡了第一夜我在远离人烟千里之遥的沙漠上睡觉比起那些乘着木排在茫茫大洋中挣扎漂浮的遇险者我更显得孤独无助朝霞出露的时候一个细细的奇妙的声音把我唤醒你不难想象我当时有多惊讶了这细细的奇妙声音说劳驾请你给我画一只绵羊吧你说什么给我画一只绵羊我像遭了雷击一跃而起我使劲揉我的眼睛仔细地看了看只见一个很奇特的小小的人儿他正在那儿注视我呢下面就是以后我给他画的最为成功的一幅肖像画当然它没有他本人可爱俊美这可不能怪我该怪大人是他们在我六岁那年葬送了我的画家生涯除了画打开肚子和没打开肚子的蟒蛇之外我没有画过一张画”