Code Drafting, Part 4: Adaptive Tables

Preceding articles in this series [1-3] presented various tables for code drafting. Any of these tables can be used for a given string. Granted, if the result is unsatisfactory, another table can be tried, but the tables are fixed.

A different approach is to construct a code table tailored to a given string. Such a code table need only have the characters that occur in the string, and these characters can be distributed in a balanced manner over the shaft pairs based on the number of times they occur in the string, rather than according to frequencies found in large bodies of text.

Here is an example, take from Edgar Allan Poe's poem *Annabel Lee*:

> we loved with a love that was
> more than love

The characters that occur in this string, arranged in order of decreasing occurrence, are:

character	*count*
␣	9 (blanks)
e	5
a	4
o	4
t	4
h	3
l	3
v	3
w	3
d	1
i	1
m	1
n	1
r	1
s	1

Constructing the code table starts by assigning the character with the most occurrences to the first shaft pair, the character with the next most occurrences to the next shaft pair, and so on.

Assigning the first four characters produces

characters	*shaft pairs*	*count*
␣	1,2	9
e	2,3	5
a	3,4	4
o	4,1	4

The process continues with the remaining characters. Since the count for t is 4, it cannot be added to shaft set (1,2), since that would produce too large a count for that shaft set. Instead, t is added to the next available shaft set, and the process continues.

␣	1,2	9
et	2,3	9
ah	3,4	7
ol	4,1	7

The next character is v with a count of 3, so it cannot be added to shaft sets (1,2) or (2,3), but instead is added to shaft set (3,4). Similarly, w is added to shaft set (4,1).

␣	1,2	9
et	2,3	9
ahv	3,4	10
olw	4,1	10

The next character is d with a count of 1, so it can be added to shaft pair (1,2), and so on:

␣d	1,2	10
eti	2,3	10
ahvm	3,4	11
olwn	4,1	11

The last two characters are added to the first two shaft pairs, completing the table:

␣dr	1,2	11
etis	2,3	11
ahvm	3,4	11
olwn	4,1	11

Since the number of characters in the string used here is evenly divisible by 4, the counts come out even and the table is perfectly balanced. A pattern for this string derived from this code table appears at the end of this article.

On the other hand, the technique of producing balanced code tables based on the frequency of occurrence of characters in large bodies of text produces this table, which is not as well balanced:

characters	*shaft pairs*	*frequency*
bd	1,2	0.227
aei	2,3	0.227
hlmnrs	3,4	0.227
otvw	4,1	0.318

If the number of characters in a string is not evenly divisible by 4, the resulting code table cannot be perfectly balanced, but the process described above assures that it will be as nearly balanced as possible.

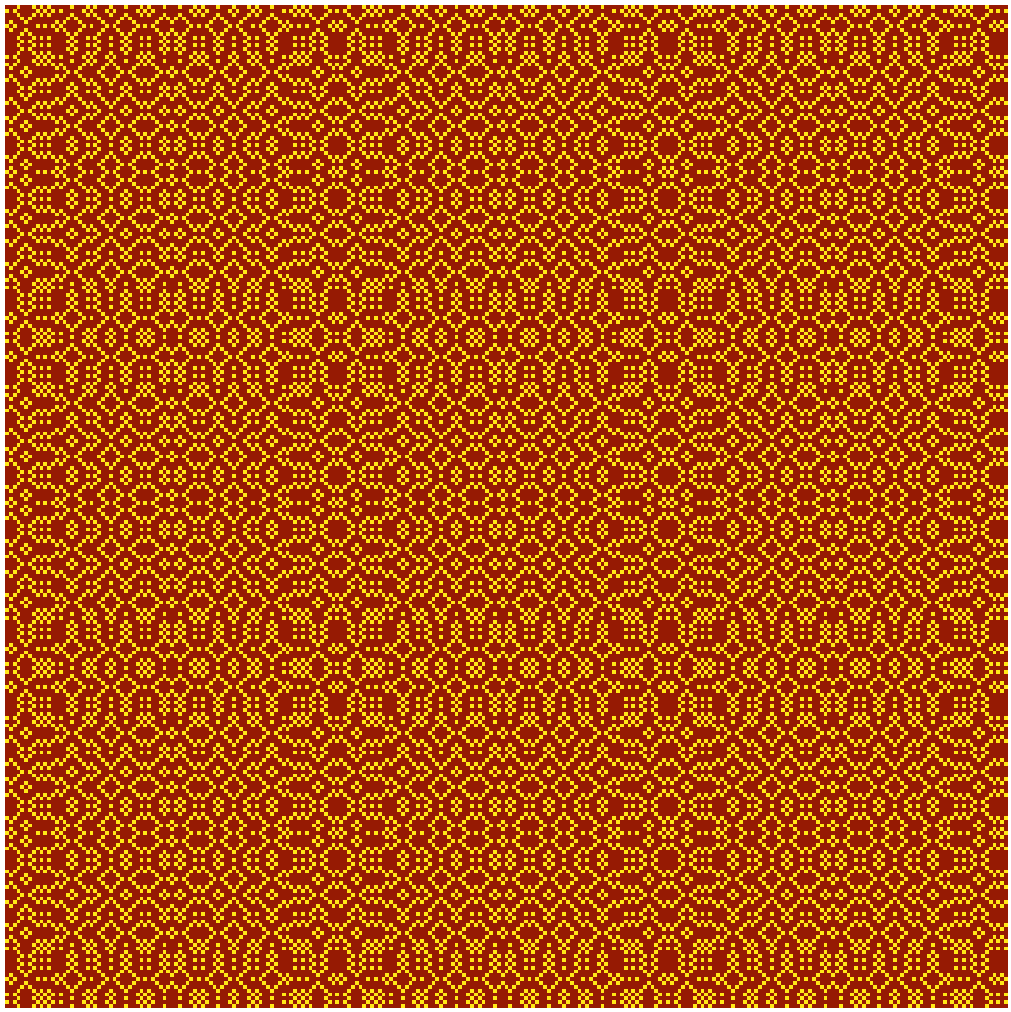
References

1. *Code Drafting, Part 1: Introduction*, Ralph E. Griswold, 2004: http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_cd1.pdf

2. *Code Drafting, Part 2: Balanced Code Tables*, Ralph E. Griswold, 2004: http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_cd2.pdf

3. *Code Drafting, Part 3: A Larger Character Set*, Ralph E. Griswold, 2004: http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_cd3.pdf

Ralph E. Griswold
Department of Computer Science
The University of Arizona
Tucson, Arizona



Annabel Lee