On Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression on Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression

To cite this article: M A Budiman and D Rachmawati 2017 IOP Conf. Ser.: Mater. Sci. Eng. 180 012062

View the article online for updates and enhancements.

Related content

- Improvement Text Compression Performance Using Combination of Burrows Wheeler Transform, Move to Front, and Huffman Coding Methods
  Mohammada Aprilianto and Maman Abdurohman

- Samawa Language: Part of Speech Tagset and Tagged Corpus for NLP Resources
  Trienani Hariyanti, Saori Aida and Hiroyuki Kameda

- Computational studies of compressed and sheared electrorheological fluid
  A Lukkarinen and K Kaski
On Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression on Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression

M A Budiman* and D Rachmawati

1Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia

*mandrib@usu.ac.id and dian.rachmawati@usu.ac.id

Abstract. This research aims to study the efficiency of two variants of variable-length codes (i.e., Goldbach G0 codes and Even-Rodeh codes) in compressing texts. The parameters being examined are the ratio of compression, the space savings, and the bit rate. As a benchmark, all of the original (uncompressed) texts are assumed to be encoded in American Standard Codes for Information Interchange (ASCII). Several texts, including those derived from some corpora (the Artificial corpus, the Calgary corpus, the Canterbury corpus, the Large corpus, and the Miscellaneous corpus) are tested in the experiment. The overall result shows that the Even-Rodeh codes are consistently more efficient to compress texts than the unoptimized Goldbach G0 codes.

1. Introduction

Data compression is a technique to reduce the size of data in order to store it much more compactly and also to decrease its transfer time. In modern computing systems, characters or symbols are usually encoded in ASCII. Each symbol that appears on a computer screen has a different ASCII code. Since the length of each ASCII code in binary is 8, there are 28 unique symbols in the ASCII table.

Data compression can be divided into two types: lossless and lossy compressions. In lossless compression, the compressed data can always be reconstructed back to the original data. On the other hand, in lossy compression, the compressed data cannot be reverted back to the original data since there are some information losses due to some approximation methods used within the algorithms. Therefore, lossless compression is more appropriate to compress multimedia (such as animations, audio, images, and video) and lossless compression is more suitable to compress data where content changes are strictly not allowed (such as text files) [4].

There are many methods of lossless compression, but overall, they have the same principle that is shrinking the data size by removing redundancies. Fixed Length Code (FLC) is a code which has the same number of bits for each symbol. (A well-known example of FLC is the ASCII code itself: each symbol symbol is represented in a binary number of length 8.)
The opposite of FLC is Variable Length Code (VLC). VLC is a code which uses different number of bits for expressing a symbol. As a result, it is intuitively expected that VLC may have bigger (in other word: better) ratio of compression than FLC. With the aim to decompress unambiguously, VLC must follow the prefix property, i.e. no code is the prefix of other codewords [3]. Two VLC algorithms discussed in this research are the Goldbach G0 codes and the Even-Rodeh codes.

The purpose of this research is to study the efficiency of Goldbach G0 codes and Even-Rodeh codes in compressing texts. The parameters being studied are: (1) the compression ratio which is the ratio of the size of the uncompressed data to the size of the compressed data; (2) the space savings (the percentage of savings); and (3) the bitrate (the average number of bits used for encoding one symbol, which is the size of the compressed bits divided by the number of unique symbols in each text.

2. Method
Goldbach G0 codes was developed by Peter Fenwick [2] based on the Goldbach conjecture. The Goldbach conjecture states that every even integer larger than four can be expressed as the sum of two odd primes [6]. For example, \(8 = 3 + 5\), \(20 = 3 + 17 + 7 + 13\), and \(100 = 3 + 97 = 11 + 89 = 17 + 83 = 29 + 71 = 41 + 59 = 47 + 53\). In 2001, this conjecture was known to be true until \(4.1014\) [5].

To generate Goldbach G0 codes [2], suppose that we have an array of the first seven odd prime numbers \(P = [3, 5, 7, 11, 13, 17, 19]\). It is clear that \(P[0] = 3\), \(P[1] = 5\), \(P[2] = 7\), …, \(P[6] = 19\). Let list \(I = [0, 0, 0, 0, 0, 0, 0]\) which will act as a ‘map’ for the corresponding indices of \(P\).

Suppose we would like to encode number 3, so we set \(N = 3\). Then, we compute \(M = 2(N + 3) = 12\). Looking back at \(P\), there are two distinct odd primes that can be added together to get the value of 12, which are 5 and 7. In \(P\), the corresponding indices of 5 and 7 are 1 and 2. Thus, we set the indices 1 and 2 of \(I\) as 1, so now \(I = [0, 1, 1, 0, 0, 0, 0]\). We remove the tailing zeros, so the list \(I = [0, 1, 1]\). Hence, the codeword of \(N = 3\) is ‘011’.

The Even-Rodeh codes are explained as follows [1]. If \(N < 4\), then let \(c\) is the binary representation of \(N\) and \(lc\) is the length of \(c\); the codeword of \(N\) is \((3 – lc)\) times ‘0’ prepended to \(c\). Thus, if \(N = 2\), then \(c = ‘10’\), \(lc = 2\), so the codeword of \(N = 2\) is \((3 – 2)\) times ‘0’ prepended to ‘10’, which is ‘010’. If \(N >= 4\) and \(N < 8\), then the codeword is simply the binary representation of \(N\) prepended to ‘0’. Therefore, if \(N = 5\), then the codeword is ‘101’ prepended to ‘0’, which is ‘1010’. If \(N >= 8\), then let \(c\) is the binary representation of \(N\), \(lc\) is the length of \(c\), and \(bc\) is the representation of \(lc\) in binary; the codeword of \(N\) is \(bc\) prepended to \(c\) and prepended again to ‘0’. Hence, if \(N = 9\), then \(c = ‘1001’\), \(lc = 4\), \(bc = ‘100’\), so the codeword is ‘100’ prepended to ‘1001’ and prepended again to ‘0’, which is ‘10010010’.

In this research, we conduct an experiment on using the Goldbach G0 codes and the Even-Rodeh codes for compressing texts. The texts are derived from the files which are included in five corpora: the Artificial corpus, the Calgary corpus, the Canterbury corpus, the Large corpus, and the Miscellaneous corpus (http://www.corpus.canterbury.ac.nz.descriptions/).

3. Results and Discussion
The results of the experiment are tabulated in Table 1 and Table 2 as follows.

| Files   | Compressed (bits) | Uncompressed (bits) | Compression Ratio | Space Savings (%) | Bitrate (bits/symbol) |
|---------|-------------------|---------------------|-------------------|-------------------|-----------------------|
| aaa.txt | 200008            | 800000              | 3.999             | 74.999            | 200008                |
| alphabet.txt | 703840        | 800000              | 1.137             | 12.02             | 27070.769             |
| random.txt | 1199808        | 800000              | 0.667             | -                 | 49.976                |
| bib    | 778576           | 890088              | 1.1432            | 12.528            | 9612.049              |
| book1  | 939432           | 1391128             | 1.481             | 32.469            | 9612.049              |
| book2  | 3614872          | 4886848             | 1.352             | 26.028            | 37654.916             |
| geo    | 20040            | 15880               | 0.792             | -                 | 18747                 |
| news   | 2629352          | 3016872             | 1.147             | 12.845            | 26830.122             |
Table 1. Cont.

| Files   | Compressed (bits) | Uncompressed (bits) | Compression Ratio | Space Savings (%) | Bitrate (bits/symbol) |
|---------|-------------------|---------------------|-------------------|-------------------|-----------------------|
| obj1    | 2576              | 8784                | 3.409             | 70.674            | 59.906                |
| obj2    | 15776             | 16664               | 1.056             | 5.328             | 143.418               |
| paper1  | 338424            | 425288              | 1.257             | 20.425            | 3562.357              |
| paper2  | 454640            | 657592              | 1.446             | 30.863            | 4996.044              |
| paper3  | 262928            | 372208              | 1.416             | 29.359            | 130.095               |
| paper4  | 75904             | 106288              | 1.4               | 28.586            | 948.8                 |
| paper5  | 75000             | 95632               | 1.275             | 21.574            | 824.176               |
| paper6  | 246040            | 304840              | 1.239             | 19.289            | 2645.591              |
| pic     | 1369024           | 4105728             | 2.999             | 66.656            | 8610.214              |
| progc   | 281080            | 316888              | 1.127             | 11.299            | 3055.217              |
| progl   | 426632            | 573168              | 1.343             | 5.566             | 4903.816              |
| progp   | 314048            | 395032              | 1.258             | 20.501            | 130.095               |
| trans   | 726192            | 733536              | 1.01              | 1.001             | 7335.273              |
| alice29.txt | 796768           | 1187840            | 1.491             | 32.923            | 11066.222             |
| asyoulik.txt | 748536           | 1001432            | 1.338             | 25.253            | 1007.882              |
| cp.html | 174696            | 196824              | 1.127             | 11.299            | 3055.217              |
| fields.c | 73528            | 89200               | 1.213             | 17.569            | 816.978               |
| grammar.lsp | 21136            | 29768               | 1.408             | 28.998            | 278.105               |
| kennedy.xls | 1096             | 1984                | 1.81              | 44.758            | 29.622                |
| lcet10.txt | 2326104          | 3353880             | 1.442             | 30.644            | 28025.349             |
| plraban12.txt | 2501528          | 3769272             | 1.507             | 33.634            | 31664.911             |
| ptt5    | 1369024           | 1405728             | 1.258             | 20.501            | 3528.629              |
| sum     | 726192            | 733536              | 1.01              | 1.001             | 7335.273              |
| xargs.1 | 26128            | 33816               | 1.294             | 22.735            | 353.081               |
| bible.txt | 20349624         | 32379136            | 1.591             | 37.152            | 323009.905            |
| E.coli  | 13877680          | 37109520            | 2.674             | 62.603            | 469420                |
| world192.txt | 15421152         | 19266248            | 1.249             | 19.958            | 165818.839            |
| pi.txt  | 4296792           | 8000000             | 1.862             | 46.29             | 429679.2              |

Table 2. The experimental results of the Even-Rodeh codes.
In Table 1 and Table 2, it can be noted that in most of the cases, the Even-Rodeh codes have bigger compression ratio and space savings than the Goldbach G0 codes. The overall bitrates of the Even-Rodeh codes are also lower than those of the Goldbach G0 codes. Thus, it is reasonable to conclude that in general, the Even-Rodeh codes are more efficient than the Goldbach G0 codes in compressing texts.

Negative values of space savings and compression ratio below 1 in both Table 1 and Table 2 suggest that the size of the compressed text are larger than the original (uncompressed) text. This may happen when the number of unique symbols in a text is too many.

4. Conclusion
The conclusion of this research is that in the vast majority of cases of text compression, the Even-Rodeh codes clearly outperform the Goldbach codes in terms of compression ratio, the space savings, and bitrate.

5. Acknowledgments
We gratefully acknowledge that this research is funded by the Ministry of Research and Technology and Higher Education Republic of Indonesia. The support is under the research grant Talenta USU of Year 2016 Contract Number 90/UN5.2.3.1/PPM/SP/2016.

References
[1] Even S and Rodeh M 1978 Economical encoding of commas between strings Communications of the ACM Commun. ACM 21 315–7.
[2] Fenwick P 2002 Variable-length integer codes based on the Goldbach conjecture, and other additive codes IEEE Transactions on Information Theory 48 2412–7.
[3] Joseph N R, Planichamy J and Sandanam D 2016 Variable length integer codes based on radix number system conversion Electronics Letters 52 1385–7.
[4] Kaufman Y and Klein S Semi-lossless text compression Data Compression Conference, 2004. Proceedings. DCC 2004
[5] Richstein J 2000 Verifying the Goldbach conjecture up to $4 \times 10^4$ Mathematics of Computation 70 1745–50.

[6] Saouter Y 1998 Checking the odd Goldbach conjecture up to $10^{20}$ Mathematics of Computation 67 863–7.