UDC 004.065 **DOI: 10.18413/2518-1092-2021-6-4-0-4**

Kostikov V.A.
Maslova M.A.

COMPRESSION OF ENCRYPTED AND HIDDEN DATA. FIXING THE AVERAGE TIME WHEN PERFORMING THE HUFFMAN CODING ALGORITHM

Sevastopol State University, 33 Universitetskaya St., Sevastopol, 299053, Russia

e-mail: my.virus.kaspersky@gmail.com, mashechka-81@mail.ru

Abstract
This paper presents an algorithm for compressing data that has been encrypted or hidden, as well as capturing performance – the average time it takes to encode and decode (compress / decompress) a lossless file up to 30%, using the Huffman coding algorithm.
Keywords: lossless data compression; Huffman algorithm average time analysis; Huffman coding; data protection with data compression; efficiency of lossless data compression.
For citation: Kostikov V.A., Maslova M.A. Compression of encrypted and hidden data. Fixing the average time when performing the Huffman coding algorithm // Research result. Information technologies. – T.6, №4, 2021. – P. 27-33. DOI: 10.18413/2518-1092-2021-6-4-0-4

Костиков В.А.
Маслова М.А.

СЖАТИЕ ЗАШИФРОВАННЫХ И СКРЫТЫХ ДАННЫХ. ФИКСИРОВАНИЕ СРЕДНЕГО ВРЕМЕНИ ПРИ ПРОВЕДЕНИИ АЛГОРИТМА КОДИРОВАНИЯ ХАФФМАНА

Севастопольский государственный университет, ул. Университетская, д. 33, г. Севастополь, 299053, Россия

e-mail: my.virus.kaspersky@gmail.com, mashechka-81@mail.ru

Аннотация
В данной статье представлен алгоритм сжатия данных, которые были зашифрованы или скрыты, а также фиксирование производительности – среднего времени, за которое происходит кодирование и декодирование (compress / decompress) файла без потерь до 30%, с помощью алгоритма кодирования Хаффмана.
Ключевые слова: сжатие без потерь данных; анализ среднего времени алгоритма Хаффмана; кодирование Хаффмана; защита данных с помощью сжатия данных; эффективность сжатия данных без потерь.
Для цитирования: Костиков В.А., Маслова М.А. Сжатие зашифрованных и скрытых данных. Фиксирование среднего времени при проведении алгоритма кодирования Хаффмана // Научный результат. Информационные технологии. – Т.6, №4, 2021 – С. 27-33. DOI: 10.18413/2518-1092-2021-6-4-0-4

INTRODUCTION

The current problem is the space occupied on the storage medium. Strange as it may seem, it is the fact that the data that is on the media can be reduced by up to 30% without data loss. Therefore, by implementing this compression algorithm, it is possible to save your resources. The only question is: "how much time is needed to compress the files?". Of course, a lot depends on the processing power of the machine you're working on and the amount of data to be compressed. Nevertheless, based on the files to be stored on the server, you can calculate the average time of compressing files.

This solution is necessary because compression allows and performs the function of hiding data on the server. Such a solution has already been considered as a stegochannel or audio file hiding. Therefore, this measure is important as well as hiding the image or audio file on the server [1, 2].

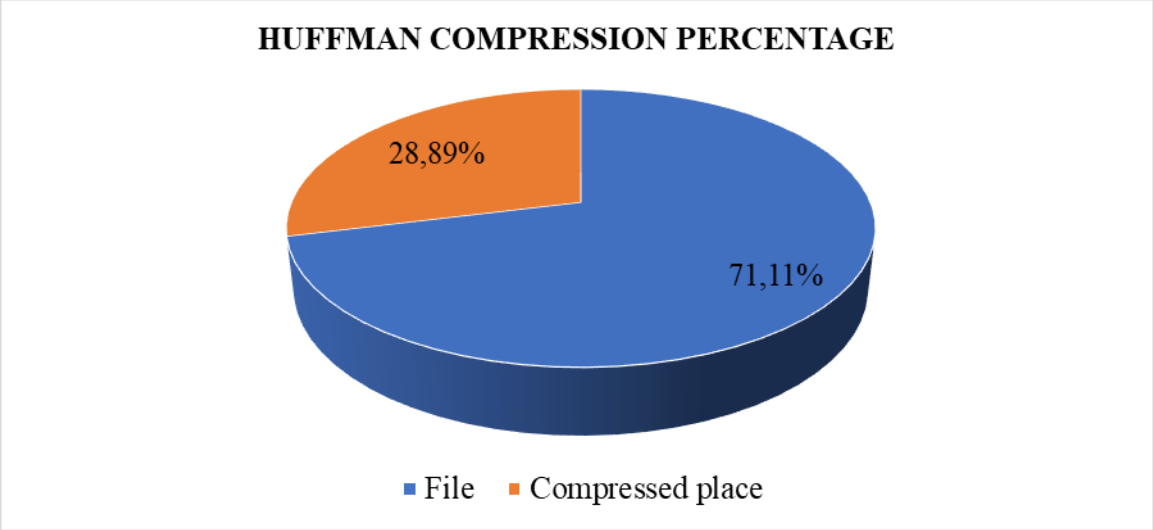

Fig. 1. Data compression using the Huffman coding method (%)
Рис. 1. Сжатие данных с помощью метода кодирования Хаффмана (%)

Therefore, it is necessary to understand what can be compressed and in what volumes. Within the framework of this article such factors of data compression algorithm will be considered as: efficiency, speed and practicality. Of course, the efficiency and practicality considered, which proves how much information can be compressed without data loss. At this point, it is necessary to consider and fix the average operation time.

MAIN PART

In order to determine how fast the file data can be compressed and decompressed - a series of tests will be conducted in the form of a cycle of repetitions and measurement of the average operation execution time of this Huffman algorithm. This test is subjective and may have different values for different PCs, as the measurement is related to the computing power of the machine. When programming it is necessary to consider first of all the components of computer: central processing unit (CPU), main memory (RAM), storage device - hard disk drive (HDD) or more modern storage device - flash drive (SSD) [3].

When the program is started (in our case everything happens through the PyCharm IDE), the program execution data is copied from the permanent secondary (additional) memory to the main memory of the PC. After booting up, the CPU will start executing the program. Let us consider this step from the technical side - the CPU follows the "fetch-execute cycle" operation, which translates as "fetch-execute cycle" [4]. Initially the first rule is fetched from memory, then it is decoded to define the essence of execution of this program fragment, then it is fetched and decoded completely and then the next rule is executed. I.e. the main principle of calculating PC is the execution of the cycle. Therefore, the power (speed) of calculation of these or those files, mainly depends on the constituent parts of the PC.

You should also take into account the fact of your PC workload - free disk space, other executable files that can influence the speed of calculations (compress/decompress). Also you should take into consideration the state of the operating system on which the computation tests are performed.

Before testing for average data compression rate using the Huffman method, it is necessary to imagine what kind of PC composition is being acted upon:

1. Processor (CPU): Intel® Core™ i5-8400 CPU @ 2.80GHz
2. Installed memory (RAM): 8.00 GB,
3. System type: 64-bit operating system, x64 processor,
4. Operating system (OS): Windows 10 Home.

There is no more information about testing. The PyCharm IDE (Python 3.9) is also used. This information will be used further to denote computing performance.

We use the Huffman code which is based on the condition that some file data can be used more often than others. This principle allows compressing file data considerably (up to 30%) (considered as *.txt format). The main issue besides efficiency is its execution time [5].

Thus, we have a number of factors that directly influence the computation of the machine and, consequently, the compression time is a variable, but nevertheless, having the initial configuration, as well as the estimated values and possible changes during the power conversion both in the whole VM and OS, we can determine the values that we will get at the output, depending on the use of a particular means of VM and OS. Statistical data will be obtained, which will allow us to focus on the average execution time of the program performing the coding function of the Huffman algorithm.

RESEARCH RESULTS AND THEIR DISCUSSION

The following results were obtained as a result of the test, which was created using the "while" loop:

- average encoding time (comress) is - 18.020284133 seconds for 100 (one hundred) passes,
- average decoding time (decompress) is - 20.72587044 seconds for 100 (one hundred) passes.

```
==================================
The average time of the compress process - 18.020284133000015 seconds, for 100 passes
The average time of the decompress process - 20.725870447000005 seconds, for 100 passes
==================================
```

Fig. 2. The result of the average time spent on the implementation of coding by the Huffman method (compress / decompress)
Рис. 2. Результат среднего времени, затраченного на выполнение кодирования методом Хаффмана (compress / decompress)

For a more objective analysis, it is also worth mentioning that the program was executed 100 times to get the average time for the Huffman coding process (compress/decompress).

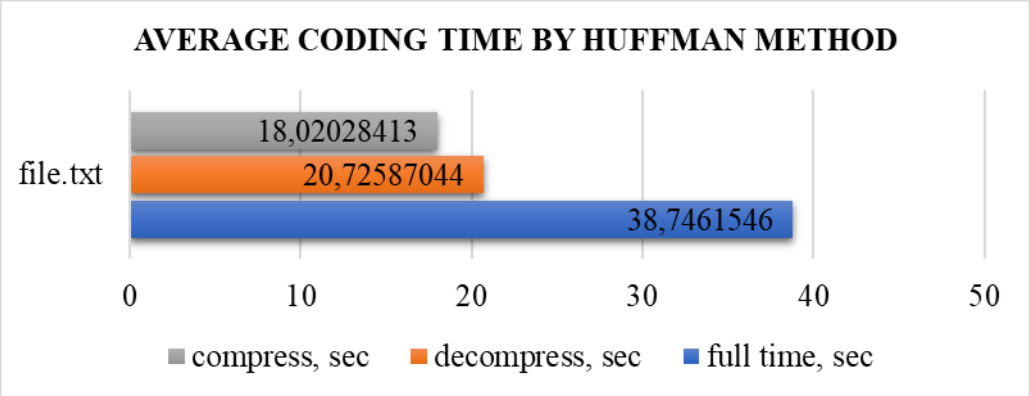

Fig. 3. Average encoding time (compress / decompress) by Huffman method
Рис. 3. Среднее время кодирования (compress / decompress) методом Хаффмана

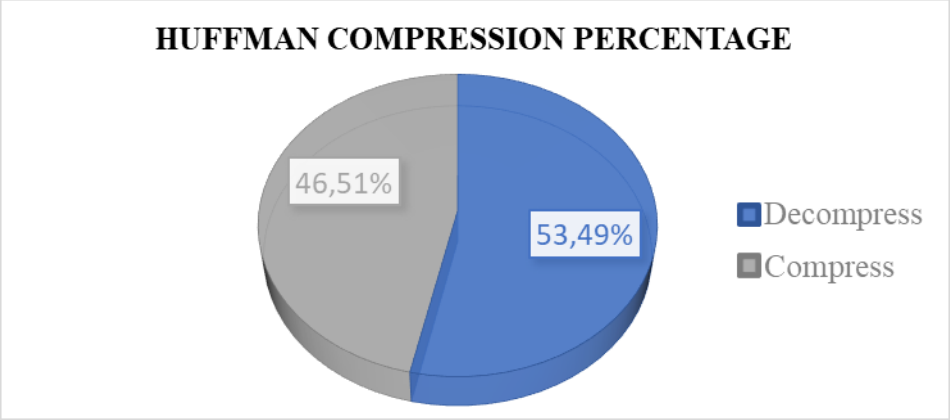

Fig. 4. Average encoding time (compress / decompress) by Huffman method
Рис. 4. Среднее время кодирования (compress / decompress) методом Хаффмана

Let's calculate how much time it takes to compress (compress / decompress) 1 KB of information using Huffman's method. Based on the data obtained in the study of data compression efficiency, you can determine how much time it takes to compress 1 GB of information. To do this, we obtain the file compression rate, using the following formula to find the value of A (Speed Compress File):

$$A = \frac{size\ file\ before\ Compression}{time\ for\ after\ Compress\ file} \quad (1)$$

$$A = \frac{3530752}{18.020284133} = 195932.094\ bytes/s = 191.34\ kB/s = 0.186855\ MB/s \quad (2)$$

According to the data obtained, we can say that the compression rate of the file is $A = 0.186855$ MB/s, which is a good indicator. Knowing this parameter, we can calculate how much time it takes to compress a file with the required weight. Then determine how much time it takes to convert a 1 GB file, knowing the compression rate of the file (A):

$$V = \frac{size\ file\ are\ we\ want}{\text{speed compress file}} \quad (3)$$

$$V = \frac{1024\ MB}{0.186855} = 5480.18517\ s = 91.3364195\ \text{min} = 1.52227366\ hours \quad (4)$$

It is assumed that given the need for compression, for example, a text file (with *. txt extension) – a file with a speed of $A = 0.186855$ MB/s it is possible to convert in about $V = 1.52227366$ hours, which is, of course, a rather average value. At use of the received data it is necessary to be defined in necessity of compression of the given format with the given volume of the information. Next, let's look at the next very important parameter – the ability to determine how many data files can be converted, knowing the preliminary weight as well as the time:

$$S = \frac{speed\ converting\ file * 3600}{size\ file\ before\ Compression} \quad (5)$$

$$S = \frac{A*3600}{3.3672} = \frac{0.186855*3600}{3.3672} = 199.773699 \quad (6)$$

Thus, it is possible to estimate the amount of data to be compressed, as well as to calculate the possible actions, knowing the capabilities of your PC and OS. At the same time it is necessary to remember that this factor is fundamental.

A – measurement of quantitative data per second, how much data can be converted per second (Speed Compress File, data/sec);

V – in what time it is possible to convert a file with information with a certain weight, knowing the data conversion speed (Time For Wanted Data, time);

S – how many units of data files can be converted, knowing the file weight and measurement time of quantitative data (Number Of Files, units).

Due to the fact that mostly compression time is considered both encoding and decoding separately. In some cases exactly decoding time can decide a huge role, although in other cases, may not play a role at all [6]. In our case, however, the difference between encoding and decoding is small, about 13%, which is 2.70558631 seconds. Due to the fact that the data are supposed to be stored and "pulled" if necessary, these figures are insignificant.

So, for comparison, to understand how much information is compressed for 18.020284133 seconds, which is as much as 3446 KB, then complete work of Leo Tolstoy "War and Peace" in English and Russian is - 2011 KB, and only English text of this work is - 755 KB data [7]. It can be noted that the data is sufficient for such a file extension.

These results and indicators are not bad, but they could be improved with more powerful computing equipment. For this purpose it is necessary to compare the estimation of processor performance.

For our processor Intel Core i5-8400 - the figure is 5467.65. Taking into account the fact that new processors have been released, the compression result of 18.020284133 seconds is satisfactory for this figure. Let's consider what features processors of next generations have. Let's take the average and high (benchmark) in comparison, orienting, with the help of rating [8].

Table

Comparative characteristics of central processors

Таблица

Сравнительная характеристика центральных процессоров

№	CPU	Overwall rating	Frequency	Year of release	Cores / threads
1	AMD Ryzen Threadripper 3990X	51363.66	2900 MHz	2020	64 / 128
2	Intel Core i9-11900KB	15217.57	3300 MHz	2021	8 / 16
3	Intel Core i5-8400	5696.89	2800 MHz	2017	6 / 6

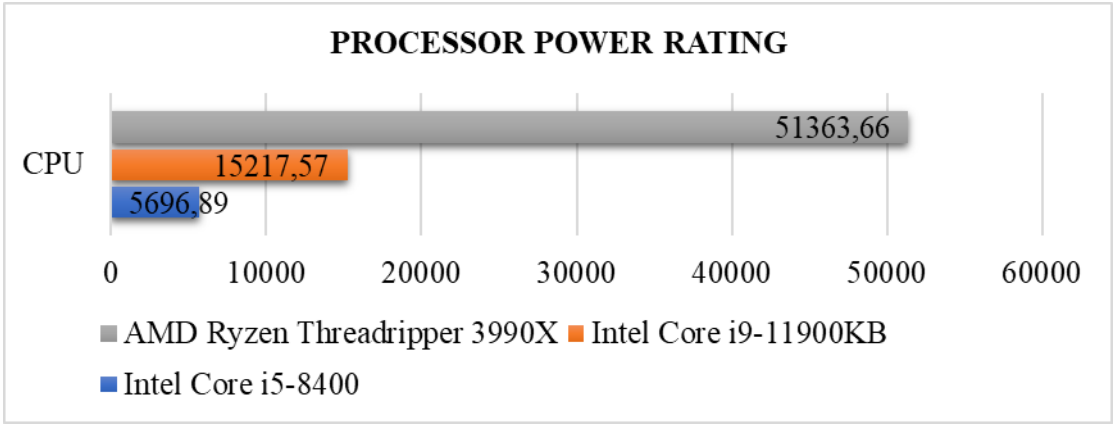

Fig. 5. Processor power rating

Рис. 5. Рейтинг оценки можности процессоров

As can be seen from Fig. 5 – the power of processors is different, but as practice shows, despite the high score, the statistics cannot be calculated with the help of the evaluation rating, and, accordingly, it cannot be estimated. It is assumed that from the data presented in Fig. 5, it is possible to speed up the

code, namely on average from 5% to 40%, with the help of processor (CPU) computing technique, but also RAM and other components.

Thus, it is offered to check in practical way and fix average speed of conversion (compress / decompres) of files by means of Huffman method. Also there is a possibility to optimize code, using built-in functions, as well as Numba and Cython code, which speeds up at least 4 times the code, which will significantly increase productivity [9, 10].

CONCLUSIONS

The given indicators of speed of work of the program of conversion (compress / decompress) of files by means of the method of Huffman have shown, that average time of coding (comress) makes - 18.020284133 seconds for 100 (hundred) passes, and average time of decoding (decompress) makes - 20.72587044 seconds for 100 (hundred) passes; the values allowing assuming possible labour input, time and speed of work of a code, and also efficiency and practicality of compression of Huffman code itself were calculated: 1. $\text{Speed Compress File (A)} = 0.186855\ \text{MB/s}$, 2. $\text{Time For Want Data (V)} = 1.52227366\ \text{hours}$, 3. $\text{Number Of Files (S)} = 199.773699\ \text{units}$.

Also, the work identified the main factors on which the outcome of the average speed of computation: the central processing unit (CPU), random access memory (RAM), the hard disk drive (HDD) or flash drive (SSD), as well as parameters such as the OS. It was considered the prospect of improving this coding Huffman – code optimization, which can eliminate the additional processes that also affect the speed of calculation, as there is a possibility of using the already built-in functions of calculation and conversion and use of the following possible solutions – Numba and Cypthon. These solutions will give better results in the long run, with the same resource costs.

Thus, summarizing all told above, using all possibilities – program and technical, it is possible to accelerate the code in times with its use everywhere without restrictions. At this stage it is reasonable to use the code for data up to 10,000 KB to save time and wasted resources.

References

1. Bukhantsov A.D., Druzhkova I.V., Kuleshov S.I., Kiselyov Y.I. Research of algorithms concealed introduction of data in grayscale images spatial domain // Research result. Information technologies. – T. 2, № 4, 2017. URL: http://rrinformation.ru/media/information/2017/4/IT_7_pyOhQIz.pdf

2. Lykholob P.G., Medvedeva A.A., Likhogodina E.C., Mishina O.O. Research of sensitivity of some measures of quality assessment of hidden information in the audio content // Research result. Information technologies. – T. 1, № 4, 2016. URL: http://rrinformation.ru/media/information/2016/4/3_it.pdf

3. Effective Computation in Physics by Anthony Scopatz and Kathryn D.Huff, June 2015, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, p. 235-236 – URL: https://clck.ru/Yzrp8.

4. Python Programming: An Introduction To Computer Science, Second edition by John M. Zelle, Ph.D. Franklin, Beddle & Associates Inc, p. 4-5. URL: https://jntukucen.ac.in/ebook_files/93.pdf.

5. Lossless data compression and decompression algorithm and its hardware architecture a thesis submitted in partial fulfillment of the requirements for the degree of Master of Technology In VLSI Design and Embedded System By V.V.V. Sagar, p. 12-26. URL: https://core.ac.uk/download/pdf/53188756.pdf.

6. Lossless Compression by ScienceDirect [Elektronnyi resurs]. – Rezhim dostupa: https://clck.ru/Yggwf.

7. «Voina i mir» Lev Nikolaevich Tolstoi na angliiskom iazyke s perevodom. URL: https://clck.ru/ZBAjN.

8. Table of CPU performance. URL: https://clck.ru/ZBAiD.

9. How Numba and Cython speed up Python code by Artem Golubin, February 10, 2018. URL: https://clck.ru/ZBAiZ.

10.Essential Python Code Optimization Tips and Tricks by Meenakshi Agarwal, October 2, 2016. URL: https://clck.ru/ZBAiq.

Список литературы

1. Буханцов А.Д., Дружкова И.В., Кулешов С.И., Киселёв Ю.И. Исследование алгоритмов скрытного внедрения информации в пространственные компоненты монохромного изображения // Научный результат. Информационные технологии. – Т. 2, №4, 2017. URL: http://rrinformation.ru/media/information/2017/4/IT_7_pyOhQIz.pdf
2. Lykholob P.G., Medvedeva A.A., Likhogodina E.C., Mishina O.O. Research of sensitivity of some measures of quality assessment of hidden information in the audio content // Научный результат. Информационные технологии. – Т.1, №4, 2016. URL: http://rrinformation.ru/media/information/2016/4/3_it.pdf
3. Effective Computation in Physics by Anthony Scopatz and Kathryn D. Huff, June 2015, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, с. 235. URL: https://clck.ru/Yzrp8.
4. Python Programming: An Introduction To Computer Science, Second edition by John M. Zelle, Ph.D. Franklin, Beddle & Associates Inc, с. 4-5 URL: https://jntukucen.ac.in/ebook_files/93.pdf.
5. Lossless data compression and decompression algorithm and its hardware architecture a thesis submitted in partial fulfillment of the requirements for the degree of Master of Technology In VLSI Design and Embedded System By V.V.V. Sagar, с. 12-26. URL: https://core.ac.uk/download/pdf/53188756.pdf.
6. Lossless Compression by ScienceDirect. URL: https://clck.ru/Yggwf.
7. «Война и мир» Лев Николаевич Толстой на английском языке с переводом. URL: https://clck.ru/ZBAjN.
8. Table of CPU performance URL: https://clck.ru/ZBAiD.
9. How Numba and Cython speed up Python code by Artem Golubin, February 10, 2018: URL: https://clck.ru/ZBAiZ.
10. Essential Python Code Optimization Tips and Tricks by Meenakshi Agarwal, October 2, 2016. – URL: https://clck.ru/ZBAiq.

Kostikov Victor Aleksandrovich, fourth-year student of the Department Information security, Institute of Radioelectronics and Information security
Maslova Maria Alexandrovna, senior lecturer of the Department Information security, Institute of Radioelectronics and Information security

Костиков Виктор Александрович, студент четвертого курса кафедры Информационная безопасность Института радиоэлектроники и информационной безопасности
Маслова Мария Александровна, старший преподаватель кафедры Информационная безопасность Института радиоэлектроники и информационной безопасности