An Empirical Comparison of Implementation Efficiency of Iterative and Recursive Algorithms of Fast Fourier Transform

Lin Lin, Zeng Xu, He Huan, Zhao Jian, and Liang Li-Xin(✉)

College of Big Data and Internet,
Shenzhen Technology University, Shenzhen 518118, China
lianglixin@sztu.edu.cn

Abstract. Fourier Transform is the basis of modern signal processing technology, and Fast Fourier Transform reduces the time complexity of Fourier Transform. Iterative algorithm and recursive algorithm are two basic methods to implement Fast Fourier Transform on the computer. This paper uses Java, C#, C++, and Python to implement iterative algorithm and recursive algorithms of Fast Fourier Transform, and tests the runtime of these two algorithms on Windows, Mac, and Linux platforms. The experimental results show that program written in Python has the longest runtime no matter iterative or recursive algorithm, and iterative algorithm written in C++ runs most efficiently; iterative program written in C++, C# and Python is more efficient than recursive algorithm, and Java's recursive algorithm is more efficient than iterative algorithm.

Keywords: Fast Fourier Transform · Recursive algorithm · Iterative algorithm · Runtime efficiency

1 Introduction

In 1807, French scientist Fourier took the lead in using sine curves to carry out research on thermal theory. In 1822, he published his research results in the landmark work "Analytical Theory of Heat", which detailed the Fourier series and Fourier integral theory [1]. Fourier Transform (FT) occupies an extremely important position in the field of basic mathematical physics, and also lays the foundation for modern signal analysis. In the field of signal processing, Fourier Transform (FT) is a method of signal analysis, which can realize the conversion of signals between time domain and frequency domain. However, for a long time since the establishment of Fourier Transform, it made sense for the theoretical analysis, but it was difficult to apply in engineering practice because of the limitation in its huge computational complexity. A great breakthrough was made in 1965, when Cooley and Turkey proposed the famous Fast Fourier Transform (FFT) [2] ,which significantly reduced the time complexity of the algorithm,leading to a great advance on the application of FT in engineering fields, such as communication system, data compression, signal processing, nonlinear science and laser engineering etc. [3].

M. Guan and Z. Na (Eds.): MLICOM 2020, LNICST 342, pp. 73–81, 2021.
https://doi.org/10.1007/978-3-030-66785-6_9

There are two ways to implement Fast Fourier Transform on the computer: iterative algorithm and recursive algorithm [4]. It is generally believed that the iterative algorithm has a large amount of code, and the algorithm implementation process is complicated and difficult to understand, but the program can run at high efficiency. On the contrary, the recursive algorithm has a clear code structure, and is easy to understand because of its better readability. The disadvantage is that the recursive program takes more time to run, and the code efficiency will drop sharply as the scale of the problem increases [5]. However, in real programming environment, there is no more in-depth research on the actual effect of iterative and recursive algorithms of FFT.

Regarding the above issue, in this paper we attempt to implement iterative and recursive algorithms of FFT using four mainstream object-oriented languages: Java, C#, C++ and Python, test the runtime of these two algorithms on Windows, Mac and Linux operating systems, study the performance of these two algorithms in different platforms and different compilation environments using statistical methods, and conduct a reasonable analysis and discussion.

2 The Implementation Method of Fast Fourier Transform

The Fourier Transform of discrete sequences is called Discrete Fourier transform (DFT). If a N-point discrete signal sequence (x[n], n = 0, 1, …, N – 1) is given, then the kth Fourier coefficient can be calculated as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \qquad n, k \in [0, N-1] \tag{1}$$

where twiddle factor $W_N^{nk} = e^{-i\frac{2\pi}{N}nk}$ where $i = \sqrt{-1}$. The algorithm has a time complexity of O(N2).

The Fast Fourier Transform in the case of radix 2 is split into odd and even subsequences, and the DFT of *N*-point sequence can be synthesized by the DFT of the two *N*/2-point subsequences [6]. In the same way, the DFT of each subsequence can be calculated by the DFT of the smaller subsequences. The procedure is so called butterfly operation as its butterfly-shaped. The Fast Fourier Transform algorithm can reduce the time complexity to O($Nlog_2N$).

Radix 2 FFT has a strong regularity. The DFT of *N*-point sequence depends only on the *N*/2-point subsequences, so the FFT of the entire sequence can be obtained by iterative calculation. That is so-called iterative algorithm of FFT. In computer programming, iteration can be realized by loop nesting. The flow chart is shown in Fig. 1 (a). From another prospective, each sequence in FFT butterfly operation calculates and divides subsequences in the same way. If this process is encapsulated into a function, the operations at each layer can be implemented by this function. When calling a function, there is a direct call to the function itself until the sequence becomes only one data. That is a typical recursive problem, and this calculation method is so-called recursive algorithm of FFT which is showed in Fig. 1(b).

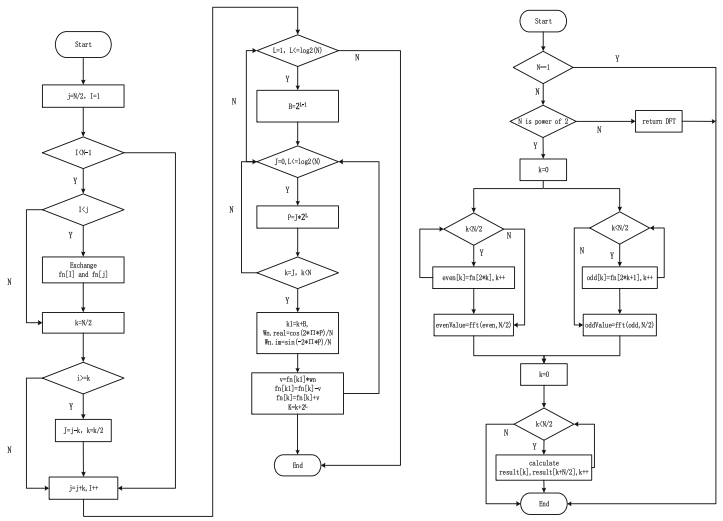

Fig. 1. Schematic of iterative (a) and recursive (b) algorithms of FFT

3 The Experiment Study

3.1 The Test Platform

In this paper, the performances of iterative and recursive algorithms of FFT with C++, C#, Java and Python languages on Windows, Mac and Linux operating system are studied. For the better accuracy of the experiment, computers with similar hardware configuration are used. Table 1 shows the information about the hardware and operating system of experimental computers, and Table 2 shows the compilers of four programming languages in three platforms.

Table 1. Hardware and operating system of experimental computers

	CPU	RAM	OS
Windows	Intel Core i7 @3.6 GHz	8G	Windows 10
Mac	Intel Core i7 @2.7 GHz	16G	MacOs Mojave
Linux	Intel Core i7 @3.6 GHz	8G	Ubuntu 18.04.2

Table 2. Compilers used in the experiment

	Java	C#	C++	Python
Windows	JRE 8u211	CSC15.0	CL 14.16	Python 3.7
Mac	JRE 8u211	CSC15.0	LLVM 10.0.1	Python 3.7
Linux	JRE 8u211	Net Core 2.2.203	GNU g++7.3.0	Python 3.7

3.2 Experiment Description

The purpose of this study is to analyze the actual performances of iterative and recursive algorithms of FFT through experiments. Considering that the hardware platform, operating system and compiler may have an impact on the performance of the algorithm, the experimental design is as follows: use Java, Python, C++, and C# to implement iterative and recursive algorithms of Fast Fourier Transform on Windows, Linux, and Mac platforms respectively, run each program ten times with an input data of a random sequence of length N = 2 M (where M = 1, 2, …, 14), and record the experiment results in milliseconds for data analysis.

In order to ensure the consistency of the code complexity as much as possible, all programming languages rewrite the complex class and the four operations in the same way to avoid using the library functions provided by the platform or third parties to implement complex operations. Similar operations, such as the definition of data structures, use consistent processing in programming to minimize the impact of different code quality on runtime efficiency.

3.3 Data Processing Methods

1. Statistical method: Paired t-test is performed on the data measured by each set of the iterative algorithm and the recursive algorithm to check whether the runtime of the two algorithms of FFT is statistically different.
2. Analysis method: If there is a statistical difference between the runtime of the iterative algorithm and the recursive algorithm, then the difference between the efficiency of these two algorithms is further studied. In this paper, the Rate of Runtime (RORT) of the two algorithms is used as an indicator to measure the time efficiency, defined as RORT = T_i/T_r, where T_i represents the runtime of the iterative algorithm and T_r represents the runtime of the recursive algorithm.

4 Experimental Results

Figure 2 shows the runtime of the Fast Fourier Transform algorithm written in four mainstream languages on three platforms. The three subgraphs all use a base-2 logarithmic coordinate system, where the abscissa is the sequence number of the FFT, the ordinate is the runtime of algorithm in ms, and the data points in the graph represent the average time of running 10 times. Figure 2(a) (b) (c) shows experimental data on Windows, Linux and Mac platforms, respectively, where the black, blue, green and red color represent C#, C++, Java and Python languages, and the solid and dashed lines represent the runtime of iterative and recursive algorithms of FFT, respectively.

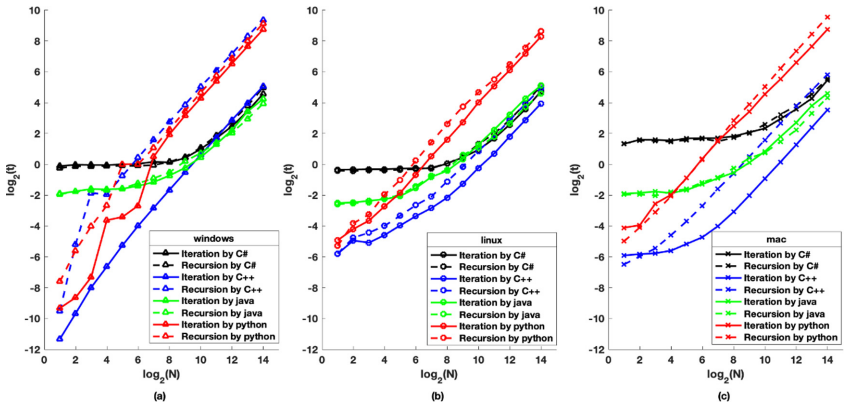

Fig. 2. Runtime of FFT algorithm

It can be seen from Fig. 2 that the runtime of algorithm has the following characteristics:

1. The curves under all platforms show that as the length of the sequence increases, whether it is the iterative algorithm or the recursive algorithm, the runtime increases accordingly. Such relationship approaches to strictly linear for adequate long sequence. When the sequence length is less than 256, C++ and Python generally maintain linear increment, while C# and Java show nonlinear characteristics. This shows that the increase in sequence length does not result in a significant increase in the runtime of the FFT transform when the sequence is short.
2. In overall, when the sequence is short, the runtime of C++ programs is the shortest, followed by Python and Java, while that of C# is the longest; when the sequence is long, the runtime of C++, C#, and Java is close while that of Python is much longer. This phenomenon is related to the mechanism of the four programming languages. C++ is a compiled language. After compiling, C++ code is a binary file that can be recognized by the machine and run directly on the operating system. Therefore, C++ is the most efficient programming language. However, C++ is not cross-platform. C++ programs that use the API under Windows cannot generally be compiled on Unix and Mac,and different compilers have different interpretations of language specifications, therefore C++ is not as compatible as the other three languages. Although C++ has advantages in terms of efficiency, it cannot be applied to various application scenarios. This study also noticed that on Windows platform, the recursive algorithm written in C++ is quite special, and its runtime even exceeds Python. This phenomenon is probably related to the C++ compiler on Windows system which will perform automatic optimization of the recursive code. Java is a compiled and interpreted language, the source code is first compiled into a bytecode file by javac, and then interpreted and run by a Java virtual machine installed on different operating systems [7]. This mechanism is also the reason why Java can run across platforms and explains why Java's runtime under three operating system platforms is almost the same. C# is a compiled and recompiled language, the code is

first compiled into a low-level intermediate language by the C# compiler, then compiled into machine code by the C# virtual machine, and finally run by the CPU. This is the mechanism for two-time compilation [8]. Python is an interpreted language that dynamically changes its structure at runtime. In each process of running the code, the Python parser first converts it into bytecode, which is then converted to machine language by the virtual machine [9]. Compared with Java and C# languages, the process is more complicated. In the environment with small data size, the difference between the three is not big, but when the amount of data being processed is large, the performance of Python will be significantly affected, and the runtime is significantly larger than Java and C#.

In order to further study the similarities and differences between the iterative algorithm and the recursive algorithm, this paper uses statistical method to perform paired t-test on the measured data to verify whether the runtime of these two algorithms is statistically different. The results are shown in Fig. 3. The logarithmic coordinates of the abscissa with the base of 2 show the number of points of the FFT sequence. The thick black line in the figure indicates that there is a statistically significant difference between the runtime of the iterative algorithm and the recursive algorithm, while the blank interval indicates that there is no significant difference.

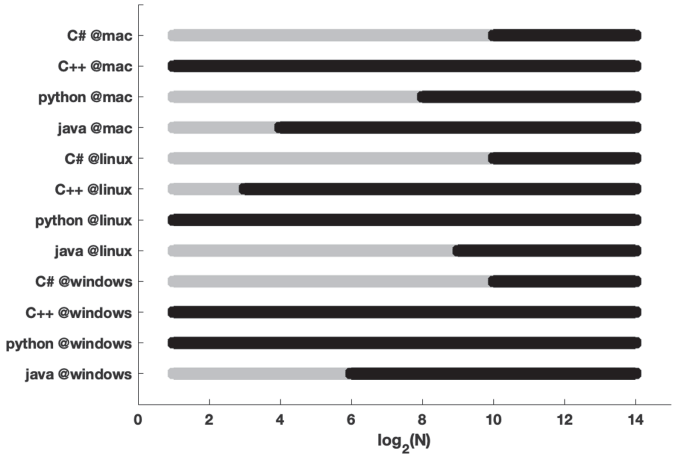

Fig. 3. Difference analysis of the runtime of the iterative algorithm and the recursive algorithm

As can be seen from Fig. 3, when the sequence length is greater than 1024, the runtime between iterative and recursive algorithms of FFT written in the four languages of Java, C#, C++ and Python is significant different on Windows, Linux or Mac operating systems. When the sequence length is small, the efficiency of the iterative algorithm and the recursive algorithm is related to the programming language and operating system. Among them, for C# and Python on Mac operating system, C# and Java on Linux operating system, C# and Java on Windows operating system, the runtime of the iterative algorithm and the recursive algorithm is not significantly

different. This phenomenon suggests that when the sequence is short, the execution efficiency of the two algorithms is not much different. Considering the clear and easy-to-understand structure of the recursive algorithm, the use of the recursive algorithm in engineering practice is beneficial to improve the efficiency of writing code.

Figure 4 shows the rate of runtime of the iterative algorithm and the recursive algorithm over different sequence lengths. When the RORT is less than 1, the runtime of the iterative algorithm is smaller than that of the recursive algorithm. The running efficiency of the iterative algorithm is better than that of the recursive algorithm. On the contrary, when the RORT is greater than 1, the runtime of the iterative algorithm is greater than that of the recursive algorithm. The running efficiency of the recursive algorithm is better than that of the iterative algorithm.

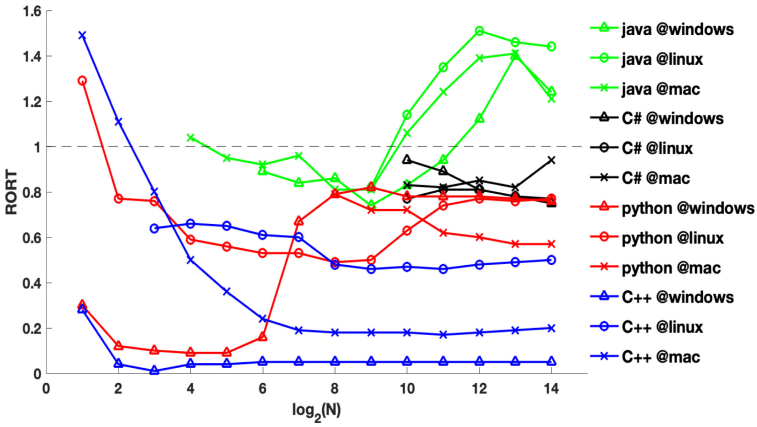

Fig. 4. Rate of runtime of the iterative algorithm and the recursive algorithm

As can be seen from Fig. 4, in the FFT calculation of the long sequence, the programming language has a greater influence on the efficiency of the algorithm than the operating system. The iterative algorithm written in C#, Python and C++ is more efficient than the recursive algorithm. C++ on Window is the most efficient, and the runtime of the iterative algorithm is about 5% of the recursive algorithm. This phenomenon is related to the recursive mechanism. In the process of program execution, recursive calculation includes two phases. The first is the extension phase, which breaks down the problem into smaller problems. In this process, the compiler constructs a recursive work stack to hold the deferred operation. The second is the shrinking phase, which is executed back and forth in the recursive work stack. Throughout this process, it is necessary to complete the repeated calls to itself. As the sequence increases, the number of steps increases dramatically, resulting in greater additional time overhead. The measured data in C#, Python, and C++ supports this view. It also suggests that when using C#, Python, and C++ to implement Fast Fourier Transform in engineering practice, the iterative algorithm should be preferred.

Figure 4 also shows that Java is inconsistent with the measured performance of the other three languages. When the sequence length is less than 1024, the runtime of the iterative algorithm is smaller than that of the recursive algorithm; when the sequence length is greater than 1024, the running efficiency of the recursive algorithm is better than that of the iterative algorithm. This phenomenon is related to Java's JIT compiler. When the same function is called repeatedly in a program, the JIT compiler dynamically optimizes the recursive code [10]. The measured data of the Java code suggests that the running efficiency of the program is related to the length of the sequence. When the sequence is long, the running efficiency of the recursive algorithm is higher than that of the iterative algorithm.

5 Conclusions

In engineering practice, it is often necessary to use the Fast Fourier Transform to perform time-frequency conversion on the signal, and perform FFT calculations on the computer. There are two ways to implement Fast Fourier Transform: the iterative algorithm and the recursive algorithm. These two algorithms have their own advantages and disadvantages. It is generally believed that the iterative algorithm is more efficient than the recursive algorithm. This study uses iterative and recursive algorithms of FFT in Java, C#, C++ and Python, and tests the runtime of the iterative algorithm and the recursive algorithm on Windows, Mac and Linux platforms. The experimental results show that the program written in Python has the longest runtime, whether it is the iterative or recursive algorithm. The iterative algorithm written in C++ has the highest running efficiency, and this phenomenon has little relationship with the operating system. The paired t-test is performed on the experimental data of iterative and recursive algorithms. The results show that the sequence length has an impact on the runtime. In the case of a long sequence, iterative and recursive algorithms have statistically significant differences. Further calculating the rate of runtime of iterative and recursive algorithms for different sequence lengths shows that the iterative program written by C++, C# and Python is better than the recursive algorithm, while Java is the opposite where the efficiency of the recursive algorithm is better than that of the iterative algorithm. The research conclusions of this paper have certain guiding significance for engineering practice, and provide a reference basis for engineers to select programming language and implementation algorithm for FFT.

References

1. Fourier, J.: The Analytical Theory of Heat. Translated with notes by Alexander Freeman. Dover Publication, New York 1-209, pp. 333–464 (1955)
2. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. **19**(90), 297–301 (1965)
3. Kovács, M., Kollár, Z.: Software implementation of the recursive discrete Fourier transform. IEEE Radioelektronika (2017)

4. Hammes, J., Sur, S., Böhm, W.: On the effectiveness of functional language features: NAS benchmark FT. J. Funct. Programm. **7**(1), 103–123 (1997)
5. Zhenyuan, Z., Cheng, Z.: Non-recursive implementation of recursive algorithm. Small Comput. Syst. **3**, 567–570 (2003)
6. Houjin, C., Jian, X., Jian, H.: Digital Signal Processing. Higher Education Press, Beijing (2004)
7. Lindholm, T., Yellin, F.: The Java virtual machine specification. Pearson Schweiz Ag **15**(2), 27–59 (1996)
8. Xu, P.: Analysis on the Features and Functions of C# Programming Language. Digital World, p. 12 (2017)
9. Downey, A.: Think Python.: An Introduction to Software Design. CreateSpace Independent Publishing Platform, Scotts Valley (2011)
10. Nikishkov, G.P., Nikishkov, Y.G., Savchenko, V.V.: Comparison of C and Java performance in finite element computations. Comput. Struct. **81**(24), 2401–2408 (2003)