FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION

OF HIGHER EDUCATION

ITMO UNIVERSITY

**Report**

**on the practical task No. 1**

**“Experimental time complexity analysis”**

Performed by

Chernobrovkin Timofey (412642)

Academic group J4133c

Accepted by

Dr Petr Chunaev

St. Petersburg

2023

# Goal

Experimental study of the time complexity of different algorithms

# Problem

For each n from 1 to 2000, measure the average computer execution time (using timestamps) of programs implementing the algorithms and functions below for five runs. Plot the data obtained showing the average execution time as a function of n. Conduct the theoretical analysis of the time complexity of the algorithms in question and compare the empirical and theoretical time complexities.

**I.** Generate an n-dimensional random vector with non-negative elements. For , implement the following calculations and algorithms:

1) (constant function);

2) (the sum of elements);

3) (the product of elements);

4) supposing that the elements of are the coefficients of a polynomial of degree , calculate the value by a direct calculation of (i.e. evaluating each term one by one) and by Horner’s method by representing the polynomial as ;

5) Bubble Sort of the elements of ;

6) Quick Sort of the elements of ;

7) Timsort of the elements of .

**II.** Generate random matrices and of size with non-negative elements. Find the usual matrix product for and.

**III.** Describe the data structures and design techniques used within the algorithms.

# Theory

Time complexity is a characteristic of an algorithm designed to give an idea of the amount of time required for the algorithm to work on a certain amount of data. The time complexity of an algorithm is estimated by calculating the number of elementary operations (additions, multiplications, etc.) performed by the algorithm for a given amount of data. It is assumed that the execution of each elementary operation requires a fixed amount of time. Due to the fact that the running time of the algorithm may be different for data of different volume, the time complexity is usually defined using the function T, reflecting the dependence of the running time of the algorithm T(n) on the volume n of input data. In this case, special attention is paid to the asymptotic behavior of T(n) when n → ∞. When analyzing the asymptotic behavior of T(n), only the highest-order summands are naturally considered, and without attention to their constant multipliers. Therefore, the time complexity of T(n) is most often written in O-big format. In such terms, the statement that the time complexity of some algorithm is equal to O(t(n)), where t(n) > 0 is some function, means that as the amount n of input data increases, the running time of the algorithm will asymptotically increase no faster than C-t(n) with some fixed constant C > 0.

To get an idea of the time complexity of the algorithm, an empirical approach is used. It consists in making a series of measurements of the algorithm's running time when the amount of input data changes. For example, for an algorithm that takes vectors of dimension n as input, we can measure its running time at n, say, from 1 to 105 in steps of 10. It is assumed that the algorithm is run under identical conditions, in particular, on the same computer that does not perform any additional computational processes that can significantly affect the running time of the algorithm under consideration. The presence of identical conditions in each measurement is fundamentally important for the quality of the obtained results. It should be noted that even when using the same computer, it is not always possible to achieve this - numerous factors, including background processes of the system, can influence the result. A reasonable compromise in such a situation is to average time measurements over several runs for the same amount of data.

# Materials and methods

In this task, all calculations were performed on the student's personal laptop. The work was performed in the Python programming language.

# Results

## Constant function

By definition, the constant function (1) has a time complexity equal to O(1). The data of measuring the time complexity of the constant function are presented in Figure 1.

(1)

Изображение выглядит как снимок экрана, линия, диаграмма, текст

Автоматически созданное описание

Figure 1 – Constant function running time

## The sum of elements

Since the summation (2) performs only 1 cycle on all elements of the list, the running time depends linearly on the number of elements in the list. Therefore, the time complexity of the summation function is O(n). The data of measuring the operation time of the summation function are presented in Figure 2.

(2)

Изображение выглядит как линия, снимок экрана, диаграмма, текст

Автоматически созданное описание

Figure 2 – Summing function running time

## The product of elements

Similarly to the summation function, when multiplying (3) all the elements of the list, one cycle is executed over all the elements of the list, but with the multiplication operation. The time complexity of the multiplication function is O(n). The data of measuring the operation time of the multiplication function are presented in Figure 3.

(3)

Изображение выглядит как снимок экрана, текст, График, линия

Автоматически созданное описание

Figure 3 – Production function running time

## A direct calculation polynomial function

When calculating a polynomial directly, a loop is performed over all elements of the list with the operation of multiplication of each element of this list by a constant raised to the degree of the previous element of the list. The time complexity of the polynomial calculation function by the direct method is O(n). The data of measuring the time of operation of the function of calculating the polynomial by the direct method are presented in Figure 4.

(4)

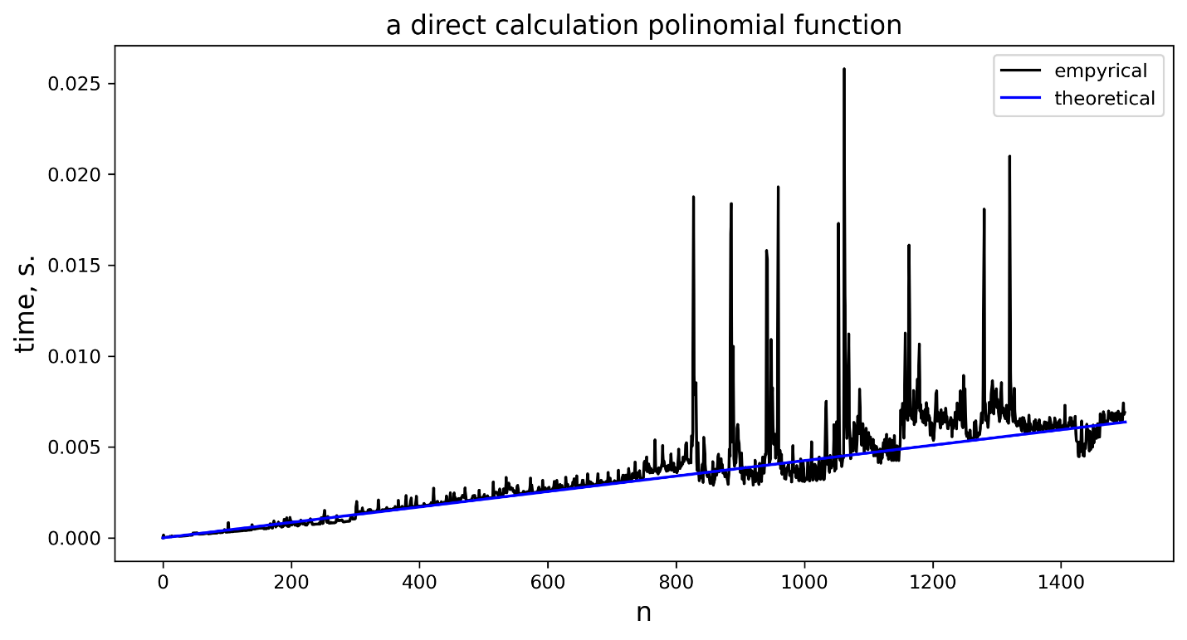


Figure 4 – Polynimial normal function running time

## Horner’s polynomial function

Similar to the direct computation method for Horner's method (5), the time complexity is O(n). The data of measuring the running time of the polynomial calculation function by Horner's method are presented in Figure 5.

(5)

Изображение выглядит как снимок экрана, линия, График, диаграмма

Автоматически созданное описание

Figure 5 – Polinimial Horner’s function running time

## Bubble Sort

Bubble sorting uses a loop nested in another loop, so the complexity of this algorithm is O(n2). The data of measuring the running time of the bubble sorting algorithm are presented in Figure 6.

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Figure 6 – Bubble sort function running time

## Quick Sort

The complexity of this sorting algorithm is O(nlogn), which is confirmed by the data shown in Figure 7.

Изображение выглядит как снимок экрана, линия, График, диаграмма

Автоматически созданное описание

Figure 7 – Quick sort function running time

## Timsort

The complexity of this sorting algorithm is O(nlogn), which is confirmed by the data shown in Figure 8.

Изображение выглядит как снимок экрана, диаграмма, линия, График

Автоматически созданное описание

Figure 8 – Timsort function running time

## Matrix production

Since the multiplication of n x n matrices can be reduced to a nested loop, the complexity of this algorithm is O(n2), which is shown in Figure 9.

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Figure 9 – Matrix production function running time

# Conclusion

In the course of this task, I got acquainted with the concept of time complexity of an algorithm, calculated the execution time of these algorithms: constant function, sum of elements, product of elements, polynomial calculation by direct and Horner's methods, Bubble sort, Quick sort, Timsort, n×n matrix production and compared them with theoretical values. In all cases, the experimental data exceeded the theoretical data due to parallel processes in the laptop. Nevertheless, in all cases the graph of experimental data was always close in shape to the theoretical value.

In all algorithms, lists with integer values were used.

# Appendix

GitHub link:

<https://github.com/LesostepnoyGnom/Homework/blob/main/Task_1_05.09.23/Task_1_Chernobrovkin_J4133c.py>