FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION

OF HIGHER EDUCATION

ITMO UNIVERSITY

Report

on the practical task No. 1

“Experimental time complexity analysis”

Performed by

Ekaterina Deviatiarova,

Valeria Sakovskaya

J4134c

Accepted by

Dr Petr Chunaev

St. Petersburg

2022

**Goal**

Experimental study of the time complexity of different algorithms.

**Formulation of the 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.

**Brief theoretical part**

Provide with brief theoretical information related to the task, for example, definitions, descriptions of algorithms, methodological approaches to solving the assigned tasks (at most 2 pages)

*Time complexity* is defined as the amount of time taken by an algorithm to run, as a function of the length of the input. It measures the time taken to execute each statement of code in an algorithm. It is not going to examine the total execution time of an algorithm. Rather, it is going to give information about the variation (increase or decrease) in execution time when the number of operations (increase or decrease) in an algorithm.

In this report, experimental study of the time complexity is presented. We will observe execution time of algorithms and functions below.

**Big-O Notation**

Big-O notation, sometimes called “asymptotic notation”, is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.

In computer science, Big-O notation is used to classify algorithms according to how their running time or space requirements grow as the input size (n) grows. This notation characterizes functions according to their growth rates: different functions with the same growth rate may be represented using the same O notation.

**Results**

**Constant Time – O(1)**

In first case, we implemented a function which returns the first value of a given array of positive integers. Independently of array size, it will always have the same running time since it only gets first value of the list. And since this is the first value, it will be found on the first iteration.

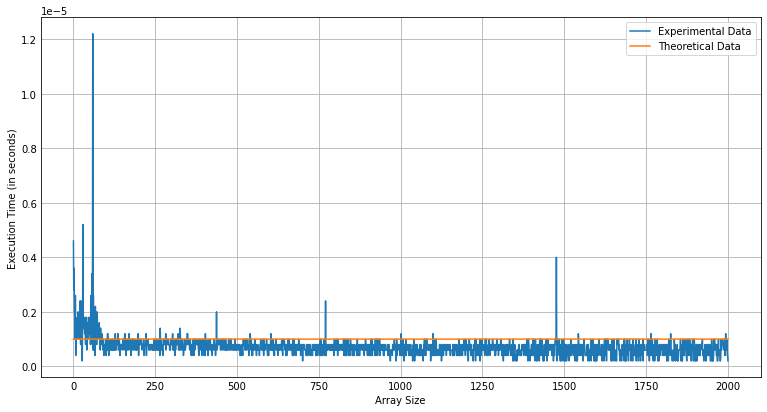


Figure 1 – Empirical and theoretical time complexities of program   
implementing the function

**Linear Time – O(n)**

An algorithm is said to have a linear time complexity when the running time increases at most linearly with the size of the input data. In this case, we measured execution time of the function of the sum of elements. Time complexity grows linearly as we successively iterate through all values in the list.

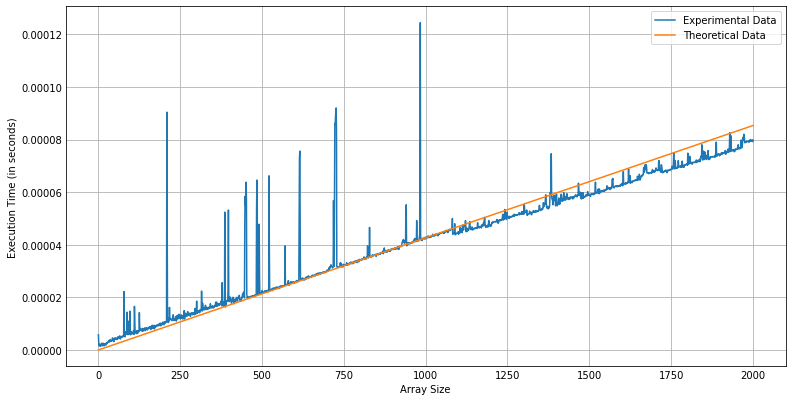


Figure 2 – Empirical and theoretical time complexities of program   
implementing the function

We get same results for function of the product of elements as we must examine all values in the input data.

Figure 3 – Empirical and theoretical time complexities of program   
implementing the function

In general, for any polynomial , where are constants and , we have .

Since any constant is a degree-0 polynomial, we can express any constant function as , or This latter notation is a minor abuse, however, because the expression does not indicate what variable is tending to infinity. We shall often use the notation to mean either a constant or a constant function with respect to some variable.

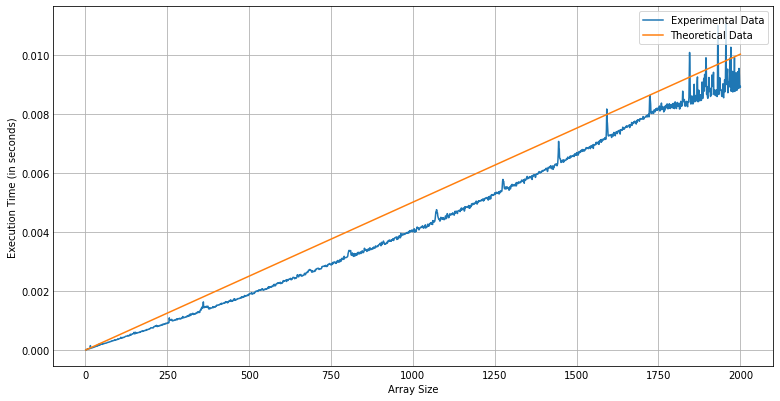


Figure 4 – Empirical and theoretical time complexities  
of a direct calculation of

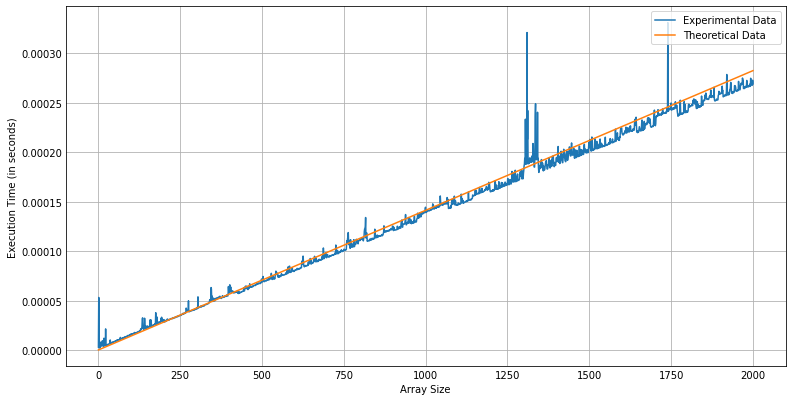


Figure 5 – Empirical and theoretical time complexities  
of Horner’s rule

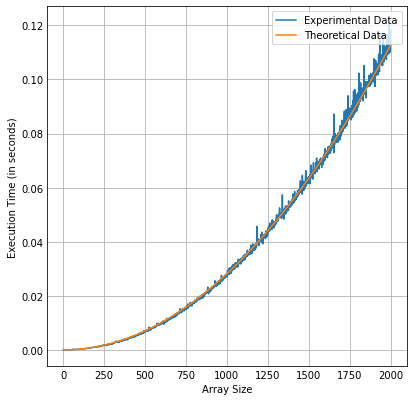


Figure 5 – Empirical and theoretical time complexities  
of Bubble Sort algorithm

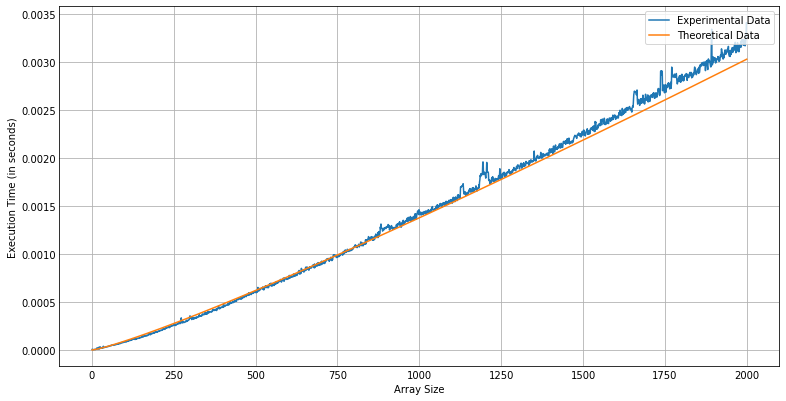


Figure 7 – Empirical and theoretical time complexities  
of Timsort algorithm

**Conclusions**

Make conclusions on the results obtained and on the achievement of the goal of your work

**Appendix**

GitHub link here