MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF THE REPUBLIC OF MOLDOVA

Technical University of Moldova Faculty of Computers, Informatics and Microelectronics Department of Software and Automation Engineering

Postoronca Dumitru FAF-233

Report

Laboratory Work №2

of AA

Checked by:
Fistic Cristofor, *university assistant*
DISA, FCIM, UTM

Chișinău – 2024

Conditions of the Task

Study and empirical analysis of sorting algorithms. Analysis of quickSort, mergeSort, heapSort, one of your choice (Counting Sort)

1. Implement the algorithms listed above in a programming language
2. Establish the properties of the input data against which the analysis is performed
3. Choose metrics for comparing algorithms
4. Perform empirical analysis of the proposed algorithms
5. Make a graphical presentation of the data obtained
6. Make a conclusion on the work done.

Algorithm Implementation

Implementation of the algorithm will be done in Javascript and running in Node environment. In order to reduce the possible fluctuations, all the algorithms will be run in bytecode mode without any optimizations. In this way I ensure that all algorithms are runned in same conditions and are not influences by the particular optimizations applyed by runtime.[1]

Input data

To analyze and compare the sorting algorithms, we define the following input data conditions:

1. Size of the Input (n):
    - Small datasets ($n = 100$)
    - Medium datasets ($n =$ 10,000)
    - Large datasets ($n =$ 1,000,000)
2. Order of the input Elements:
    - Random ordered elements
    - Elements are already sorted in **ascending order**
    - Elements are already sorted in **descending order**
    - Array contains small number of unique values, leading to high redundancy
3. Range of Values:
    - Small Range [1, 100]
    - Large Range [1, 1 000 000]
    - Negative and Positive Values
4. Edge Cases:
    - Single Element Array
    - Empty array
    - All elements identical

Metrics in Algorithm analysis

Graphical Representation

Conclusions

References

[1] Franceska Hinkelman (2017) - Understanding bytecode *Medium*