

**CSE106 Discrete Mathematics**

**Section: 06**

**Report of Mini Project**

Representing a relation with n-dimensional relational matrix

by using C program

**Submitted by:** Group 06

|  |  |
| --- | --- |
| Student ID | Student Name |
| 2022-3-60-109 | Sheikh Sarafat Hossain |
| 2022-3-60-192 | Rijia Parveen Raya |
| 2022-3-60-128 | Syeda Sayma Sultana |
| 2022-3-60-154 | MD. Shahmul Islam |

**Submitted To:**

**Dr. Mohammad Salah Uddin**

Associate Professor  
 Department of Computer Science & Engineering

East West University

**Date of Submission:**

22.05.2023

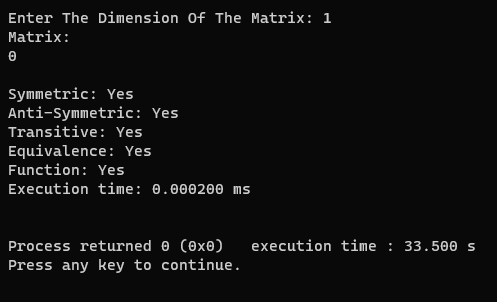
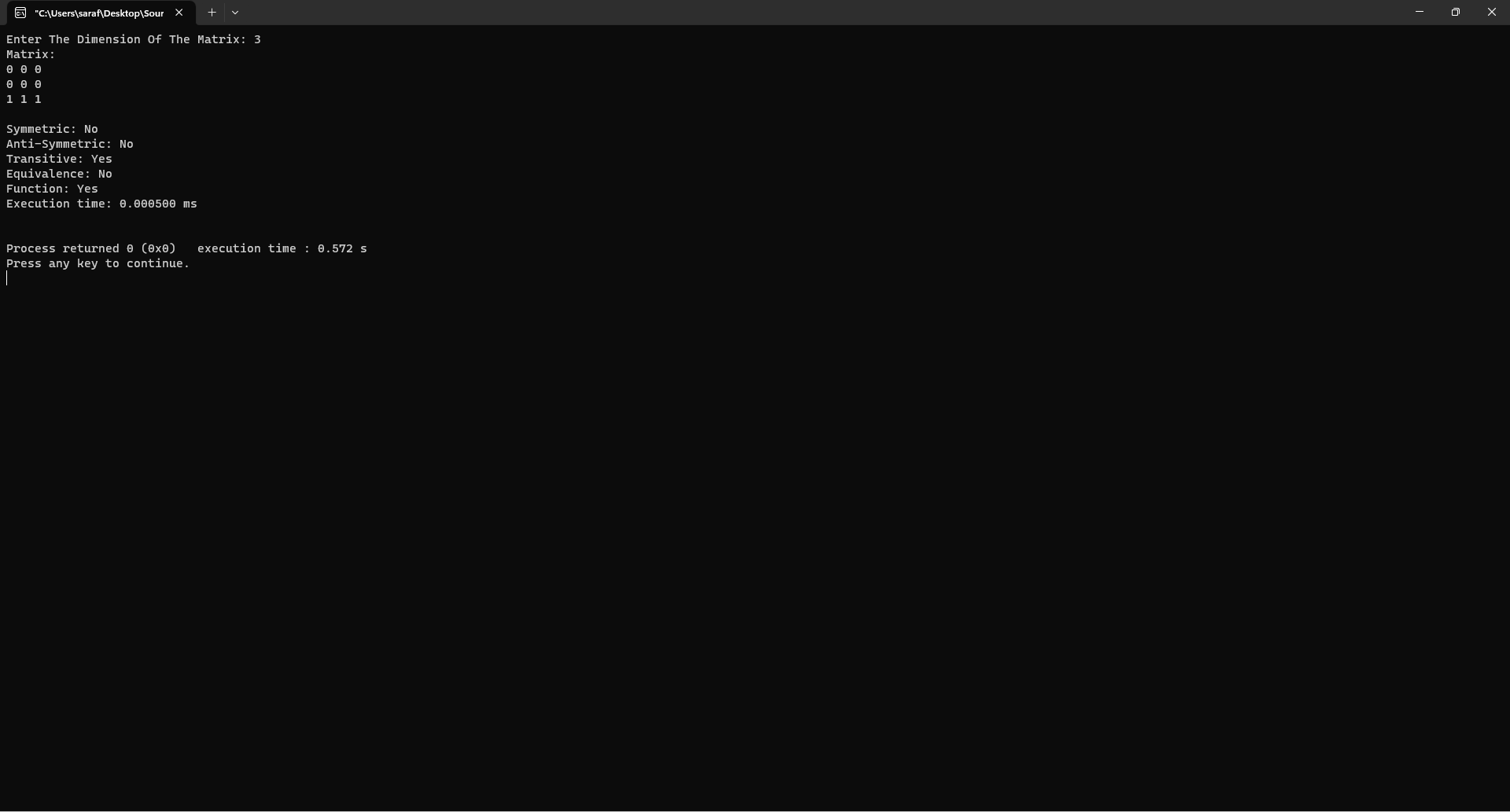
**Introduction:**

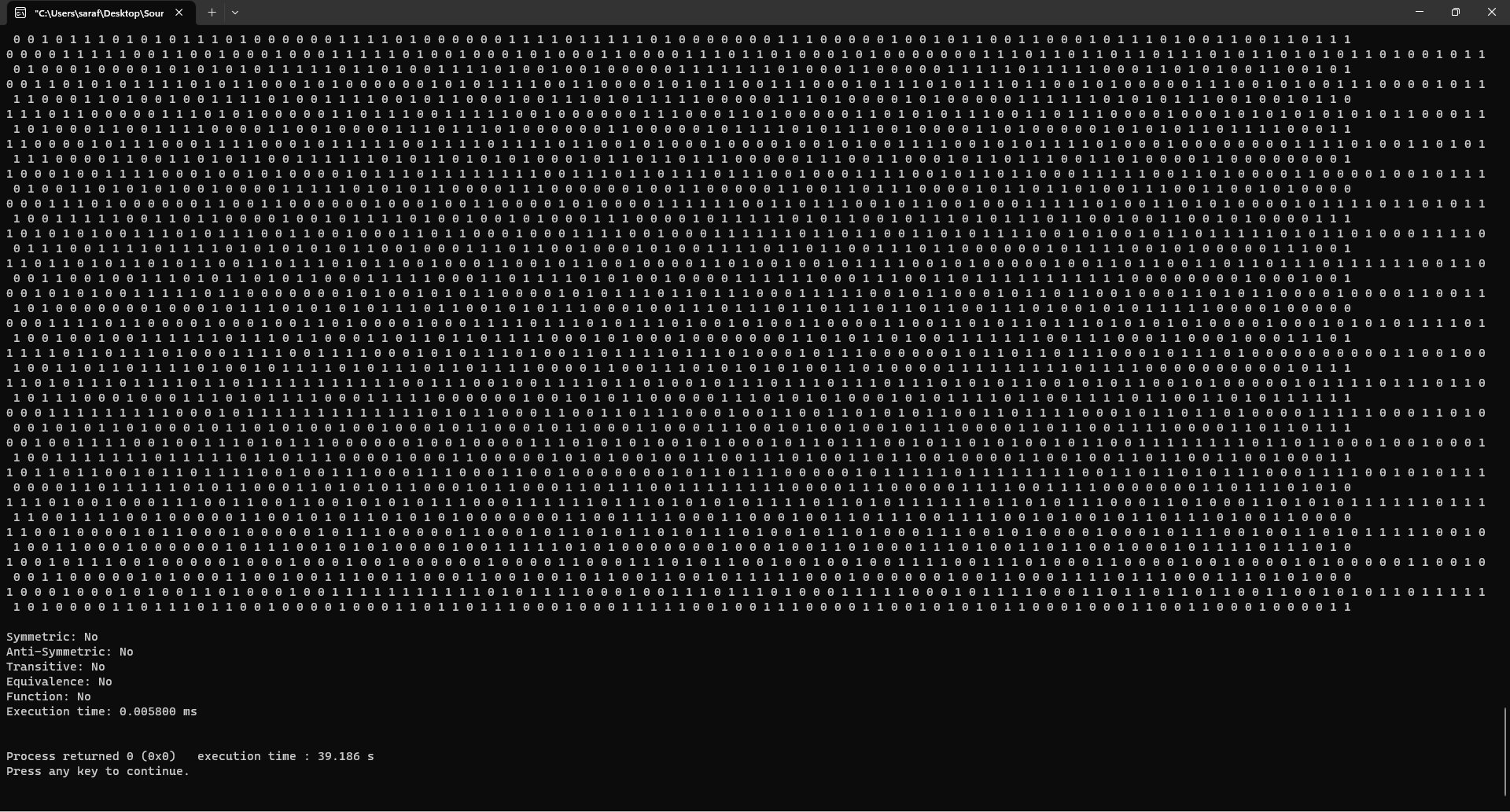
A relational matrix is a mathematical representation of a relation between elements, and it can provide valuable insights into the properties and behaviors of the relation it represents. We focus on matrices with dimensions ranging from n=1 to n=300, allowing us to observe how the time complexity scales with the increasing size of the input. In this project, we aim to analyze the time complexity of a C program that generates and examines relational matrices.

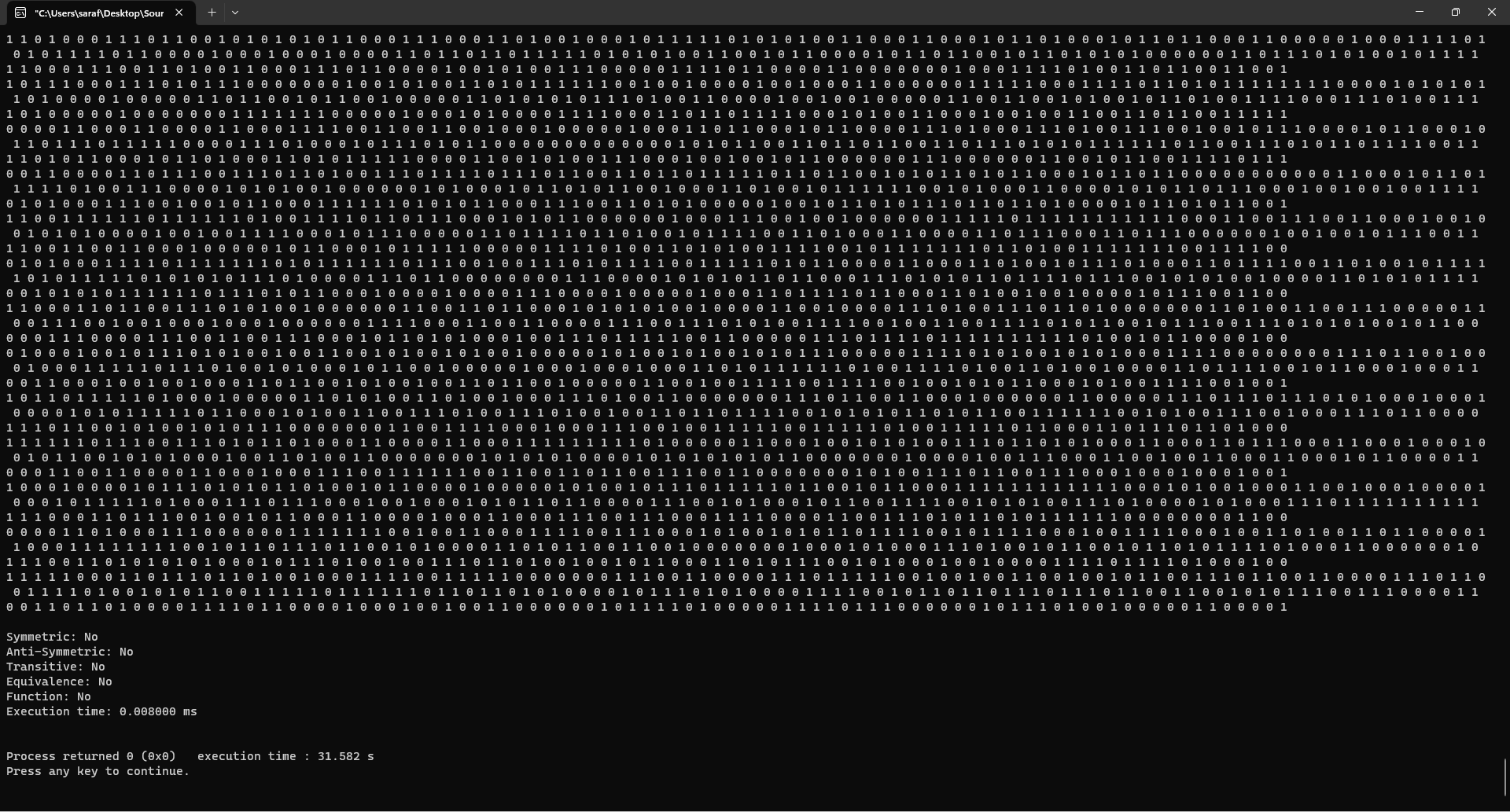
**Approach:**

1. Firstly, we made a C program to randomly generate a relational matrix represented by relation with dimensions n=1,3,50,100,150,200, 300.
2. Secondly, the output will show the properties of the relation such as symmetric, Anti-Symmetric, transitive, and equivalence. The output will also show computational time in milliseconds (ms).
3. The output will show whether the relation represents any function or not. The output will also show computational time in milliseconds (ms).
4. After that, from the computational time of n=1,3,50,100,150,200, 300-dimensional matrixes, we used Microsoft Excel Worksheet to draw a line graph showing computational time vs. n dimension.
5. Finally, we theoretically determine the computational time complexity of this program as a function of n. And compared it with the time complexity with steps 2 and 3.

**The Output Of The Program:**



**Time Complexity**

**Graph:**



**Theoretically:**

* The nested loops for generating random values have a time complexity of O(n^2), where n is the dimension of the matrix.
* **Checking symmetry:** The nested loops for comparing matrix elements have a time complexity of O(n^2), as we compare each element with its corresponding element in the transpose.
* **Checking anti-symmetry:** The time complexity is O(n^2), similar to checking symmetry.
* **Checking transitivity:** The nested loops have a time complexity of O(n^3), as we check each element for the transitive property by comparing it with other elements in the matrix.
* **Checking equivalence:** It depends on the time complexities of symmetry and transitivity, so the time complexity is O(n^2 + n^3), which simplifies to O(n^3).
* **Checking function:** The nested loops have a time complexity of O(n^2), as we count the number of "1" entries in each row.

Overall, the code's time complexity is dominated by the operations for checking transitivity, which is O(n^3)