Mini Project on Directed Graph Using Adjacency Matrix

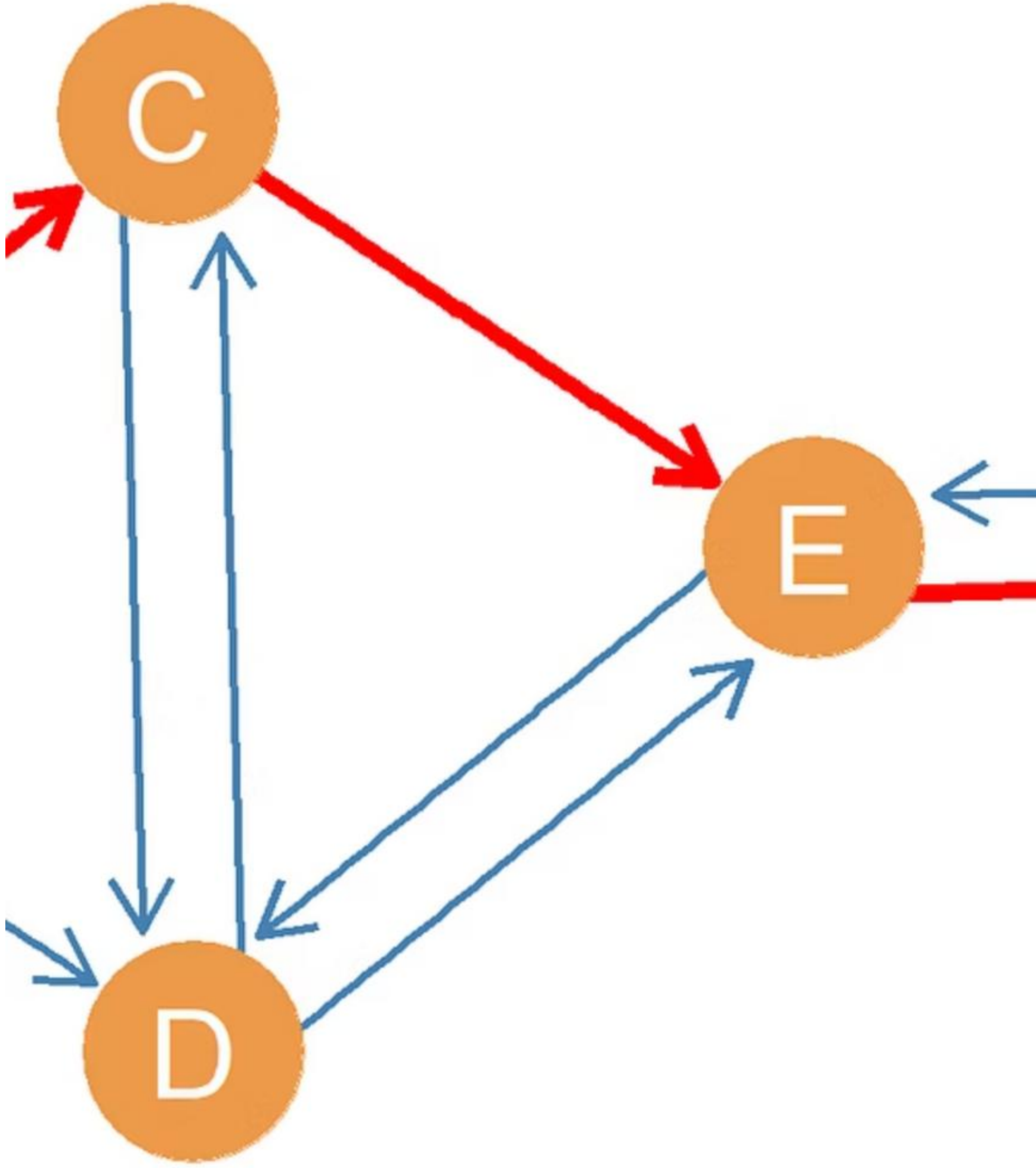
Course: CSE 106 – Discrete Mathematics Mini Project 1 (Odd Group)

Team Name: AlgoAvengers

Team Members:

- **Arup Bhowmik Pritom (2025-2-60-330)**
- **Mahin Mahjabin (2025-2-60-385)**
- **Lakibul Hasan (2025-2-60-294)**

Department of Computer Science & Engineering



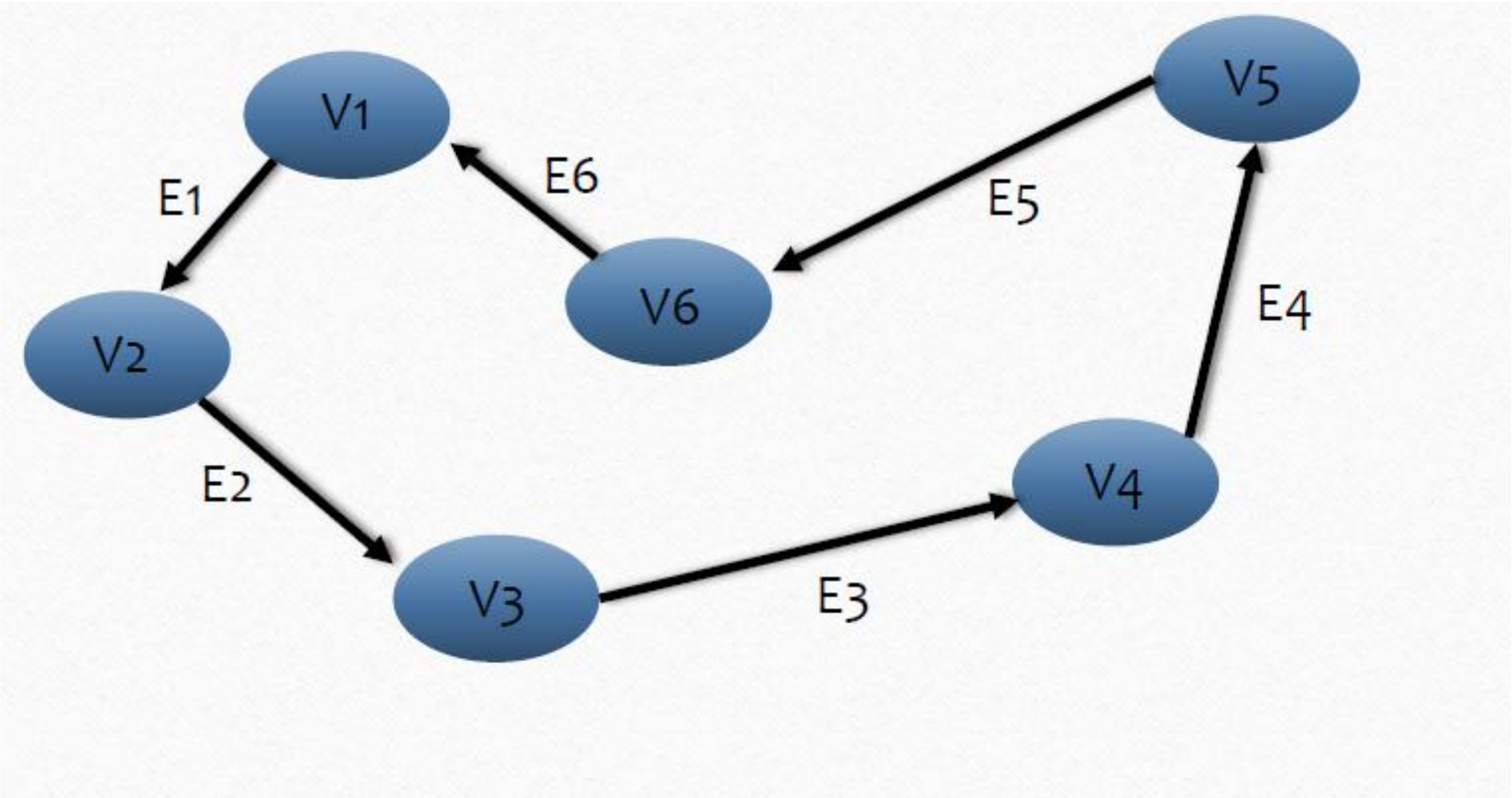
Objective of the Mini Project

- To randomly generate a directed graph using C programming
- To represent the graph using an adjacency matrix
- To compute the in-degree and out-degree of all vertices
- To verify that the sum of in-degrees equals the sum of out-degrees
- To analyze the computational time complexity of the program

Directed Graph and Adjacency Matrix

Directed Graph

- A directed graph consists of vertices and directed edges
- Each edge has a specific direction



Adjacency Matrix Representation:

- A 2D matrix of size n × n
- matrix[i][j] = 1 if there is an edge from vertex i to vertex j
- matrix[i][j] = 0 otherwise

Methodology

The program follows these steps:

01

Take the number of vertices n as input

02

Generate a random n × n adjacency matrix

03

Calculate in-degrees by summing each column

04

Calculate out-degrees by summing each row

05

Measure execution time using the clock() function

06

Repeat the process for different values of n

In-degree and Out-degree (Basic Property of Directed Graph)

Out-degree of a vertex:

Number of outgoing edges

Calculated by summing the elements of a row

In-degree of a vertex:

Number of incoming edges

Calculated by summing the elements of a column

Property: Sum of in-degrees = Sum of out-degrees

Experimental Setup

We executed the program for different input sizes:

n = 1000

n = 2000

n = 3000

n = 4000

n = 5000

For each value of n:

- We measured time execution.
- We measured Printing time.
- We recorded the results.

Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAXN 8000 // Max number of vertices
int matrix[MAXN][MAXN];
int row, column, n;

void fill(void);   // fill the adjacency matrix randomly
void result(void); // calculate In-Degree and Out-Degree
int main()
{
        clock_t start, end;
        double elapsed;
        start = clock();
        printf("Enter the amount of Vertices: ");
        scanf("%d", &n);

        start = clock();
        fill();
        result();
        end = clock();
        elapsed = (double)(end - start) * 1000 / CLOCKS_PER_SEC;
        printf("Time taken: %.2f ms.\n", elapsed);
        return 0;
}

void fill()
{
        srand((unsigned)time(NULL));
        for (row = 0; row < n; row++)
        {
                for (column = 0; column < n; column++)
                {
                        matrix[row][column] = rand() % 2;
                }
        }
}
```

```c
void result()
{
        int in_degree = 0, out_degree = 0;
        for (row = 0; row < n; row++)
        {
                for (column = 0; column < n; column++)
                {
                        in_degree += matrix[column][row];
                }
        }

        for (row = 0; row < n; row++)
        {
                for (column = 0; column < n; column++)
                {
                        out_degree += matrix[row][column];
                }
        }

        printf("Total In-Degree: %d\n", in_degree);
        printf("Total Out-Degree: %d\n", out_degree);

        if (in_degree != out_degree)
        {
                printf("Sum of In Degree and Out Degree are not Equal\n");
        }
        else
        {
                printf("Sum of In Degree and Out Degree are Equal\n");
        }
}
```

Output

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS F:\New folder (2)> cd "f:\New folder (2)\" ; if
Enter the amount of Vertices: 1000
Total In-Degree: 500024
Total Out-Degree: 500024
Sum of In Degree and Out Degree are Equal
Time taken: 16.00 ms.
PS F:\New folder (2)>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS F:\New folder (2)> cd "f:\New folder (2)\" ; if
Enter the amount of Vertices: 4000
Total In-Degree: 7999946
Total Out-Degree: 7999946
Sum of In Degree and Out Degree are Equal
Time taken: 258.00 ms.
PS F:\New folder (2)>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS F:\New folder (2)> cd "f:\New folder (2)\" ; if
Enter the amount of Vertices: 2000
Total In-Degree: 2000203
Total Out-Degree: 2000203
Sum of In Degree and Out Degree are Equal
Time taken: 59.00 ms.
PS F:\New folder (2)>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS F:\New folder (2)> cd "f:\New folder (2)\" ; if (
Enter the amount of Vertices: 3000
Total In-Degree: 4500117
Total Out-Degree: 4500117
Sum of In Degree and Out Degree are Equal
Time taken: 135.00 ms.
PS F:\New folder (2)>
```

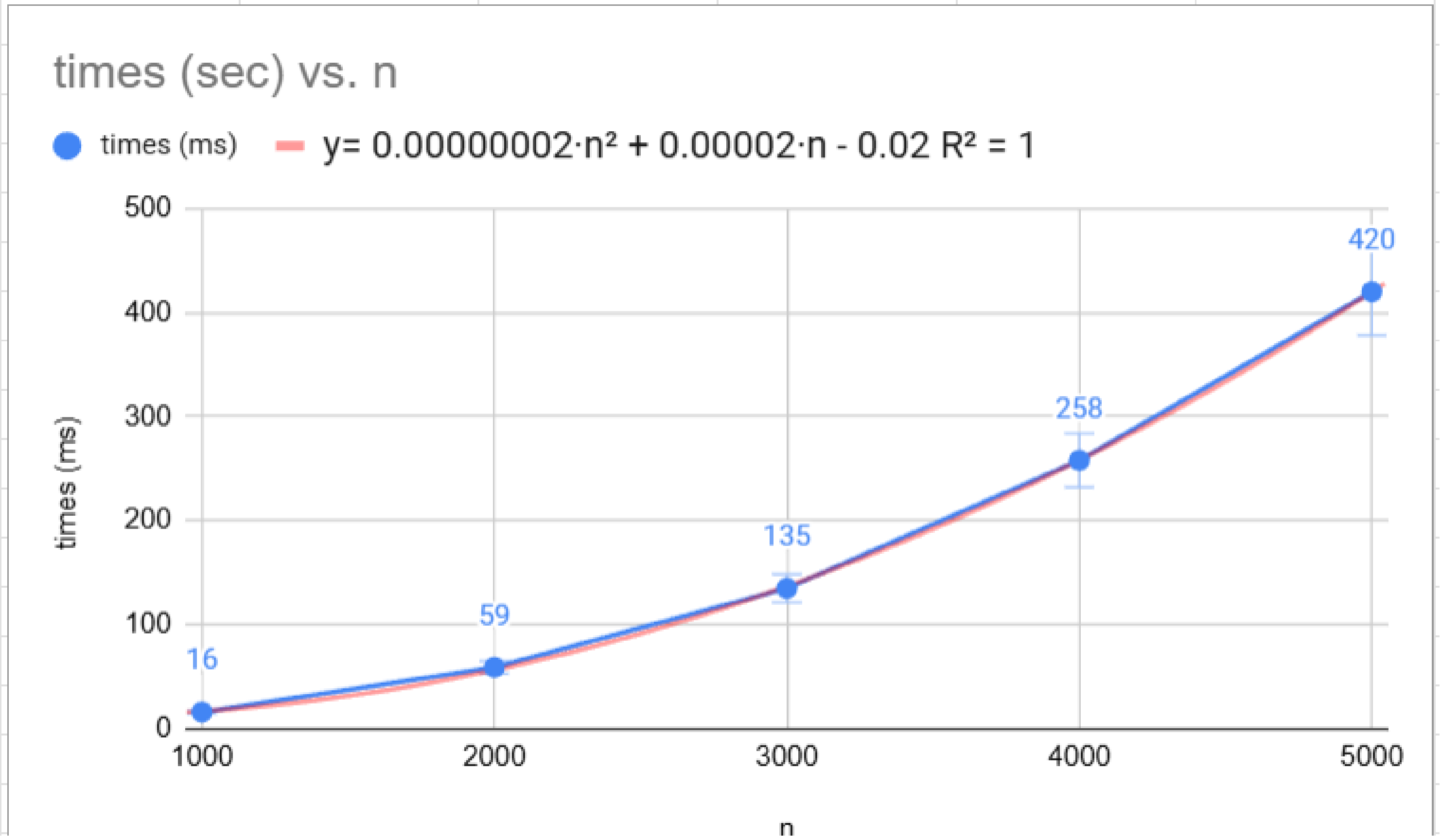
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS F:\New folder (2)> cd "f:\New folder (2)\" ; if
Enter the amount of Vertices: 5000
Total In-Degree: 12499939
Total Out-Degree: 12499939
Sum of In Degree and Out Degree are Equal
Time taken: 420.00 ms.
PS F:\New folder (2)>
```

Execution Time vs Number of Vertices

- We Plotted line graph using Google Sheets
- X-axis represents number of vertices (n)
- Y-axis represents execution time (seconds)
- A polynomial trendline was added
- Trendline equation and R^2 value were displayed

n	times (ms)
1000	16
2000	59
3000	135
4000	258
5000	420



Theoretical Time Complexity: $O(n^2)$

To calculate all in-degrees and out-degrees, we scan the entire n X n adjacency matrix once. Therefore, the total number of operations is proportional to n^2, making the complexity O(n^2).

Total operations = O(n)× O(n) = O(n^2)

Total time complexity = O(n^2) + O(n^2) = O(n^2)

Experimental time complexity (from graph) :

The equation of the trendline was of the form :

$$y = 0.00000002n2 + 0.00002n - 0.02$$

The highest degree term in the equation is n^2 which confirms that the time complexity grows quadratically with n.

The trendline shows a quadratic relationship between the computational time and n

The experimental results also suggest that the complexity of the program is O(n^ 2).

Comparison Theoretical vs Experimental

From analyzing the code, we determined that the time complexity is O(n^ 2).

From the excel graph and the polynomial trendline, confirming that the experimental time complexity also O(n^ 2)

Both the theoretical and experimental time complexity of the program are O(n^ 2).

Experimental and theoretical results match.

Conclusion

- Directed graph generation was successfully implemented
- In-degree and out-degree were correctly computed
- The property sum(in-degree) = sum(out-degree) was verified
- Execution time analysis shows $O(n^2)$ complexity

Our project demonstrates graph representation and algorithm efficiency.

Thank You