EAST WEST UNIVERSITY

TITLE : "TIME COMPLEXITY ANALYSIS OF DIRECTED GRAPH GENERATION AND DEGREE COMPUTATION"

Group Members:
Shafin Rahman (2025-1-60-186)
Tawsif Islam (2025-1-60-194)
Ayman Rahman(2025-1-60-195)

Presented to:
Ahmed Abdal Shafi Rasel Sir

Course name: Discrete Mathematics
Course code: CSE-106
Section: 7
Group : 03
Group name : Code Line

Objectives

1.**To develop a C program** that randomly generates a **directed graph** represented by an **adjacency matrix** for different graph sizes.

2.**To calculate in-degrees and out-degrees** of all vertices and verify that the **sum of in-degrees equals the sum of out-degrees**.

3.**To measure computational time** for degree calculation for different values of **n (1000–5000 vertices)**.

4.**To visualize computational performance** by plotting a **graph of time vs. n** in Microsoft Excel and fitting a **polynomial trendline**.

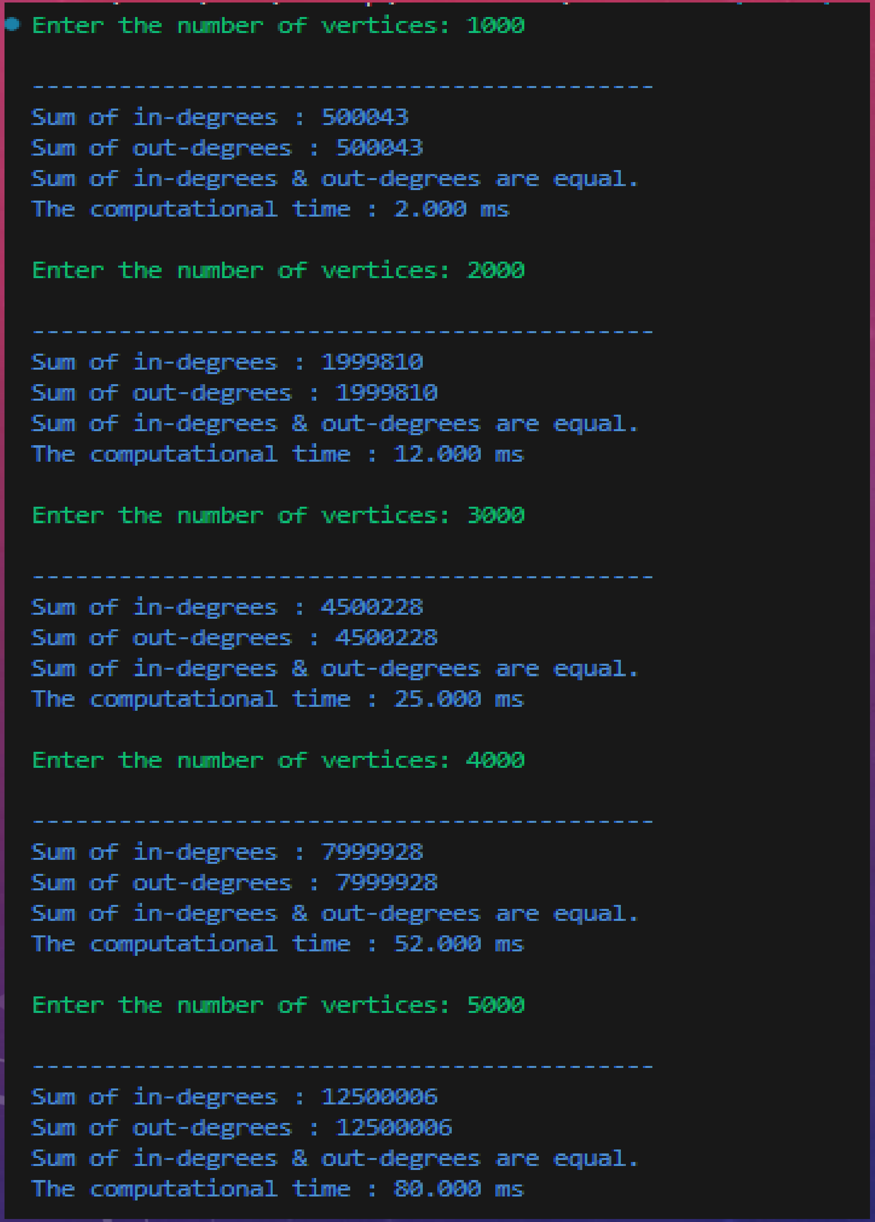
5.**To determine experimental time complexity** of the algorithm and compare it with **theoretical time complexity**.

6.**To summarize findings** in a **report and presentation** for clear understanding of performance trends.

SOURCE CODE :

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAXN 5000
int graph[MAXN][MAXN];

int main() {
    int sizes[] = {1000, 2000, 3000, 4000, 5000};
    int num_sizes = 5;

    srand(time(NULL));

    for (int s = 0; s < num_sizes; s++) {
        int n = sizes[s];

        printf("\033[0;32mEnter the number of vertices: %d\n\033[0m", n);

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                graph[i][j] = rand() % 2;
            }
        }
 clock_t start, end;
        start = clock();

        long long indeg_sum = 0, outdeg_sum = 0;
    for (int i = 0; i < n; i++) {
      int indeg = 0, outdeg = 0;

      for (int j = 0; j < n; j++) {
        outdeg += graph[i][j];
        indeg += graph[j][i];
      }

      indeg_sum += indeg;
      outdeg_sum += outdeg;
    }

    end = clock();

    double cpu_time_used = ((double)(end - start))

    printf("\033[0;34m");
printf("\n---------------------------------------------\n");
printf("Sum of in-degrees : %lld\n", indeg_sum);
printf("Sum of out-degrees : %lld\n", outdeg_sum);

if (indeg_sum == outdeg_sum)
  printf("Sum of in-degrees & out-degrees are equal.\n");
else
  printf("Sum of in-degrees & out-degrees are NOT equal.\n");

printf("The computational time : %.3f ms\n\n", cpu_time_used);
printf("\033[0m");
    }
    return 0;
}
```

OUTPUT :



```
Enter the number of vertices: 1000

-------------------------------------------
Sum of in-degrees : 500043
Sum of out-degrees : 500043
Sum of in-degrees & out-degrees are equal.
The computational time : 2.000 ms

Enter the number of vertices: 2000

-------------------------------------------
Sum of in-degrees : 1999810
Sum of out-degrees : 1999810
Sum of in-degrees & out-degrees are equal.
The computational time : 12.000 ms

Enter the number of vertices: 3000

-------------------------------------------
Sum of in-degrees : 4500228
Sum of out-degrees : 4500228
Sum of in-degrees & out-degrees are equal.
The computational time : 25.000 ms

Enter the number of vertices: 4000

-------------------------------------------
Sum of in-degrees : 7999928
Sum of out-degrees : 7999928
Sum of in-degrees & out-degrees are equal.
The computational time : 52.000 ms

Enter the number of vertices: 5000

-------------------------------------------
Sum of in-degrees : 12500006
Sum of out-degrees : 12500006
Sum of in-degrees & out-degrees are equal.
The computational time : 80.000 ms
```

GRAPH :

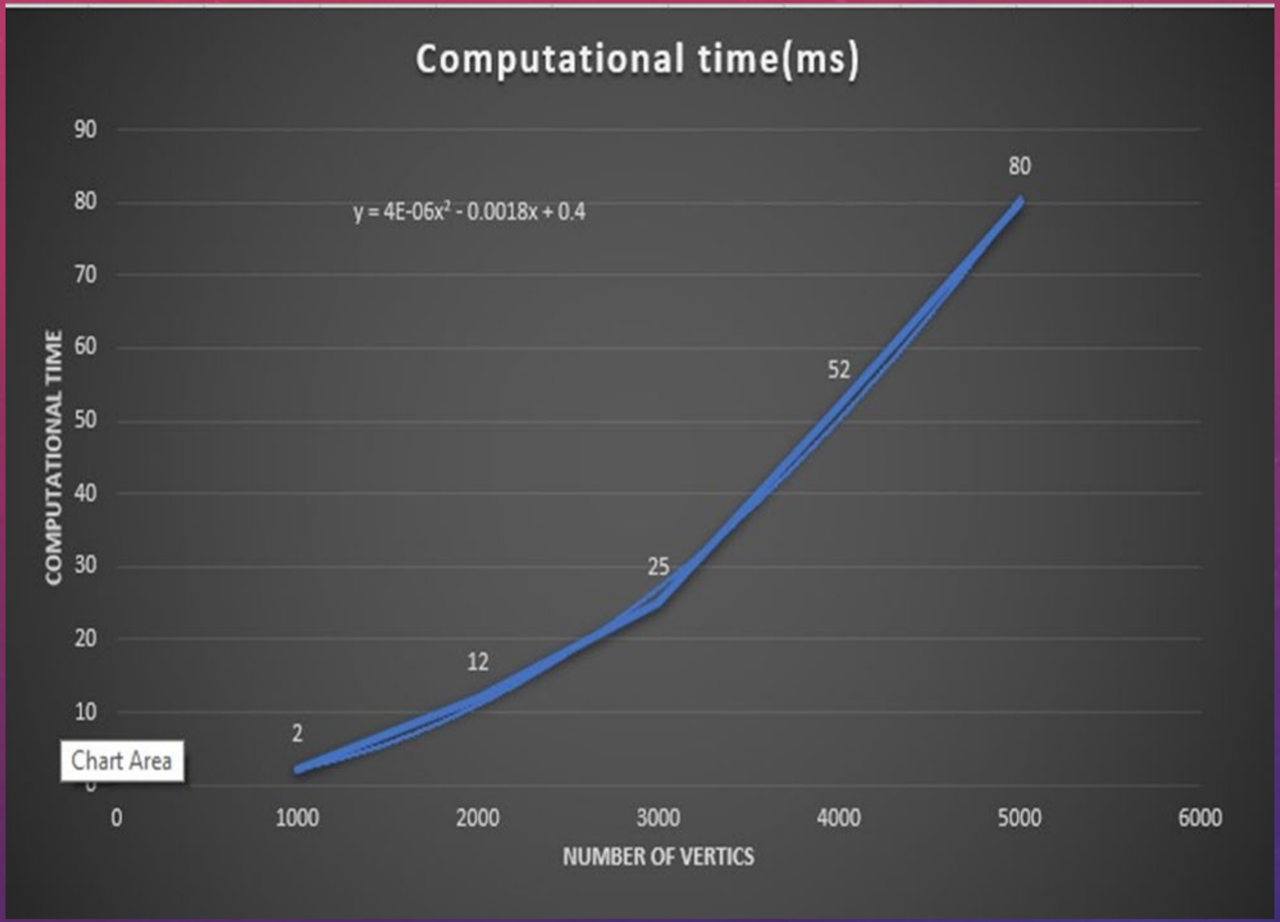
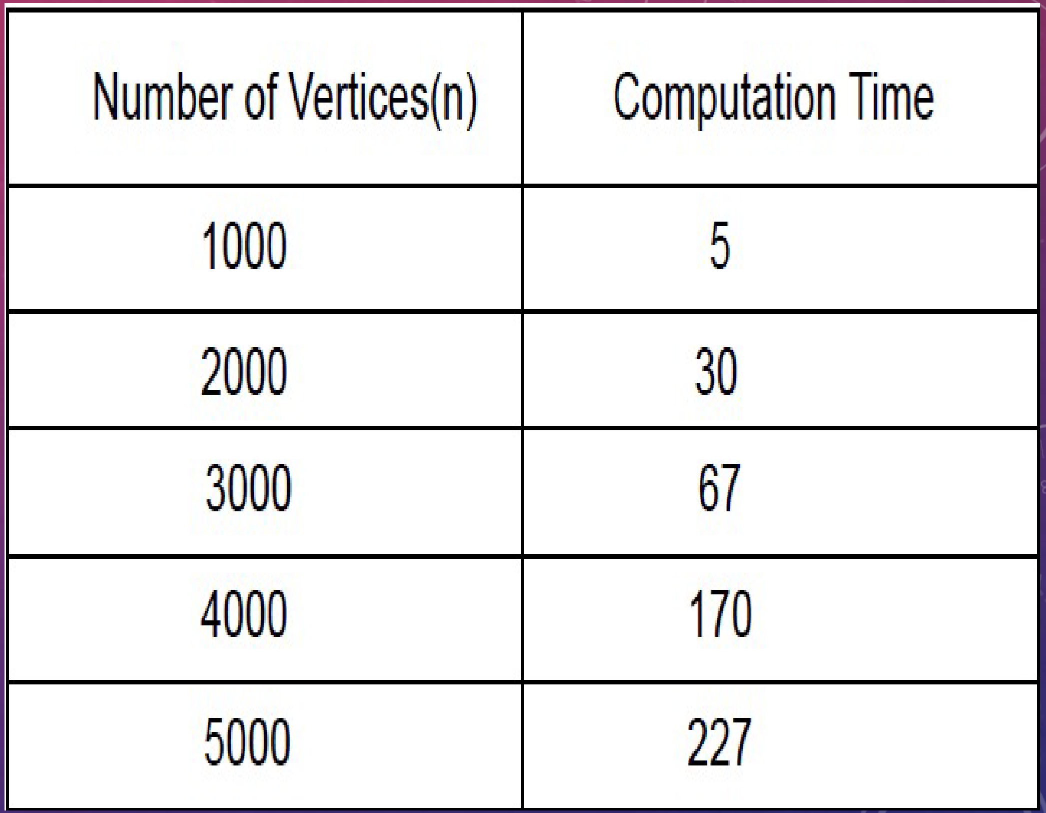


CHART :



Number of Vertices(n)	Computation Time
1000	5
2000	30
3000	67
4000	170
5000	227

POLYNOMIAL EQUATION DERIVED FROM THE GRAPH:

F(N)=4E-06x^2+0.0018x+0.4

THANK YOU!