Group – 03

Noushin Pervez

Id: 2020-1-60-189

Moni Analin

Id: 2020-1-60-182

Project Report

Project Description: Using a C-program, randomly generate a directed graph represented by adjacency matrix with n = 1000 vertices. Determine in-degrees and out-degrees of all vertices and show that sum of the in-degree and sum of out-degree are equal. Determine computational time in this step in ns. Repeat the whole process for n = 2000, n = 3000, n = 4000 and n = 5000.

Discussion:

1. **Header files:** In our C-program, we have used three header files and they are <stdio.h>,

   <stdlib.h> and <time.h>.

   - <stdio.h> is the header file for input and output.
   - <stdlib.h> is the header file for standard library of C-programing language which includes random function to generate a directed graph represented by adjacency matrix.
   - <time.h> is the header file to get the exact program execution time. From the header file, we get three functions. They are, clock_t, CLOCKS_PER_SECOND and clock().

In this section, we defined the adjacency matrix array with a size limit of 5000. As we need to run the program to the limit of 5000. The reason to define the array before main function is to run the program without any error. As the stack has a limited size and consequently can only hold a limited amount of data. If the program tries to put too much data on the stack, the program cannot do anything and return to zero without any result. To avoid this error, we have defined the array here.

2. **Main Function**: Now we procced for the next steps to show that the sum of in-degrees and out-degrees are equal.

- At first, we took float time = 0.0 to store the execution time of code. Also, we took log(vertex), initial_vertex, terminal_vertex which are integers.
- Then we took scanf and printf to scan the vertex and take input.

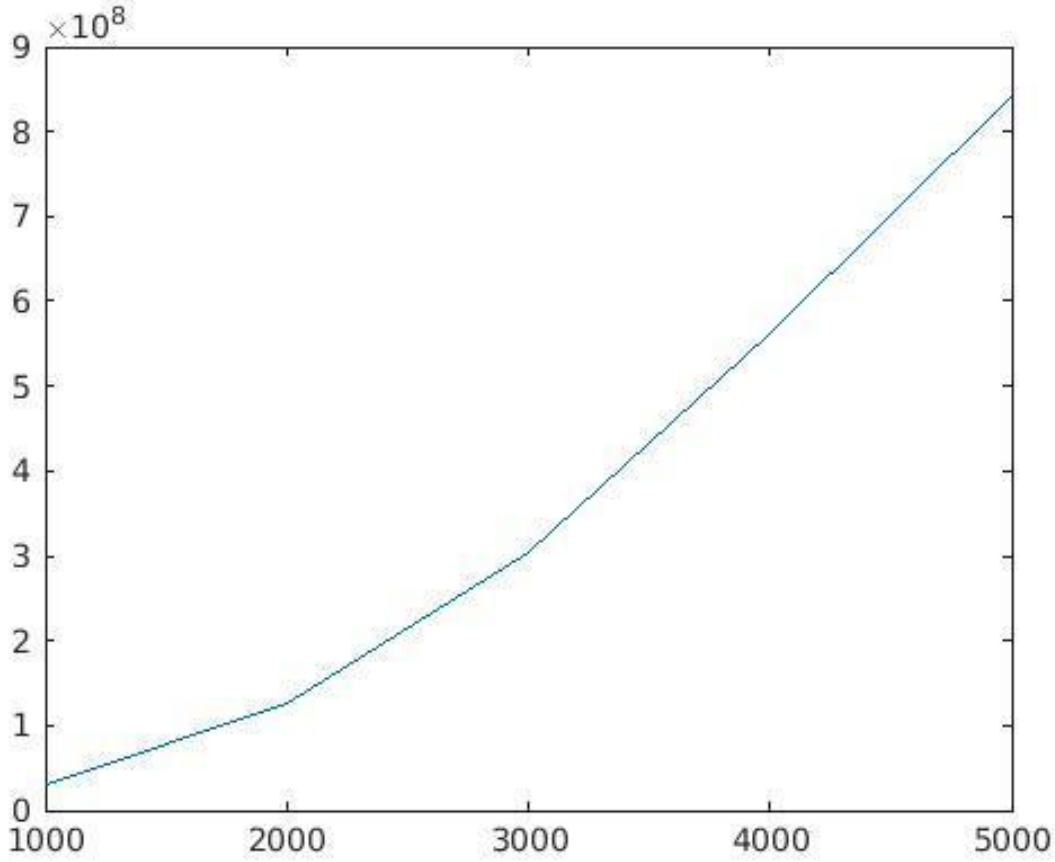
- Before for loop, we took the clock-t to begin the clock time so that we can start counting the exact runtime of the code.
- The outer loop is to iterate the vertex for rows and the inner loop is to iterate the vertex for columns.
- We fill the adjacency matrix randomly with (rand () %2) where %2 is used to put only 0 and 1.
- Then, initially we took in_degree = 0 and out_degree = 0. Then in the first for loop, we counted the summation of row and named as out_degree. After that, in the second for loop, we counted the summation of column and named it as in_degree.
- After getting the summation of in-degrees and out-degrees of all vertices, we end the clock_t function.
- We calculate the elapsed time by finding difference between the two clock() functions.
- Now, from the <time.h> header file, we use the other functions to get the exact run time of the program. We divide the difference by CLOCKS_PER_SECOND to get the time in seconds.
- Then we printed the time elapse.

- As we had to calculate time in nanoseconds, we multiplied the calculated time with 10^9.

- We check if the summation of in-degrees and summation of out-degrees are equal or not using if-else.
- Lastly, if the summations are equal, we print that the “Sum of in-degrees and outdegrees are equal.” If the summations are not equal, we print that the “Sum of indegrees and out-degrees are not equal.”

Therefore, we have shown that the summation of in-degrees and out-degrees are equal along with the execution time of the program. Using MATLAB, we have drawn a graph showing computational time vs. vertex.

Graph: Computational Time (nanoseconds) vs Vertex



The graph drawn from the computational time is a ***Quadratic graph***.
The time grows linearly to the square of the number of input elements.
Therefore, the approximate time complexity of the program as a function of n is $O(n^2)$.

Theoretical Time Complexity:

```
for(initial_vertex = 0; initial_vertex < log; initial_vertex++)
    for(terminal_vertex = 0; terminal_vertex < log; terminal_vertex++)
    {
        in_degree += adjacency_matrix[terminal_vertex][initial_vertex];
        out_degree += adjacency_matrix[initial_vertex][terminal_vertex];
    }
}
```

The calculation of in-degrees and out-degrees is done by nested loop.
Outer loop runs ***n*** times. Inner also runs ***n*** times. So, $\mathbf{f(n)} = n^2$.
Therefore, the time complexity of the program is, $\mathbf{f(n)} = \theta(n^2)$

Therefore, Approximate time complexity = Theoretical time complexity