Report of Project 2

Group 4,Section-1

Fima Faria Lisa(2020-1-60-026)

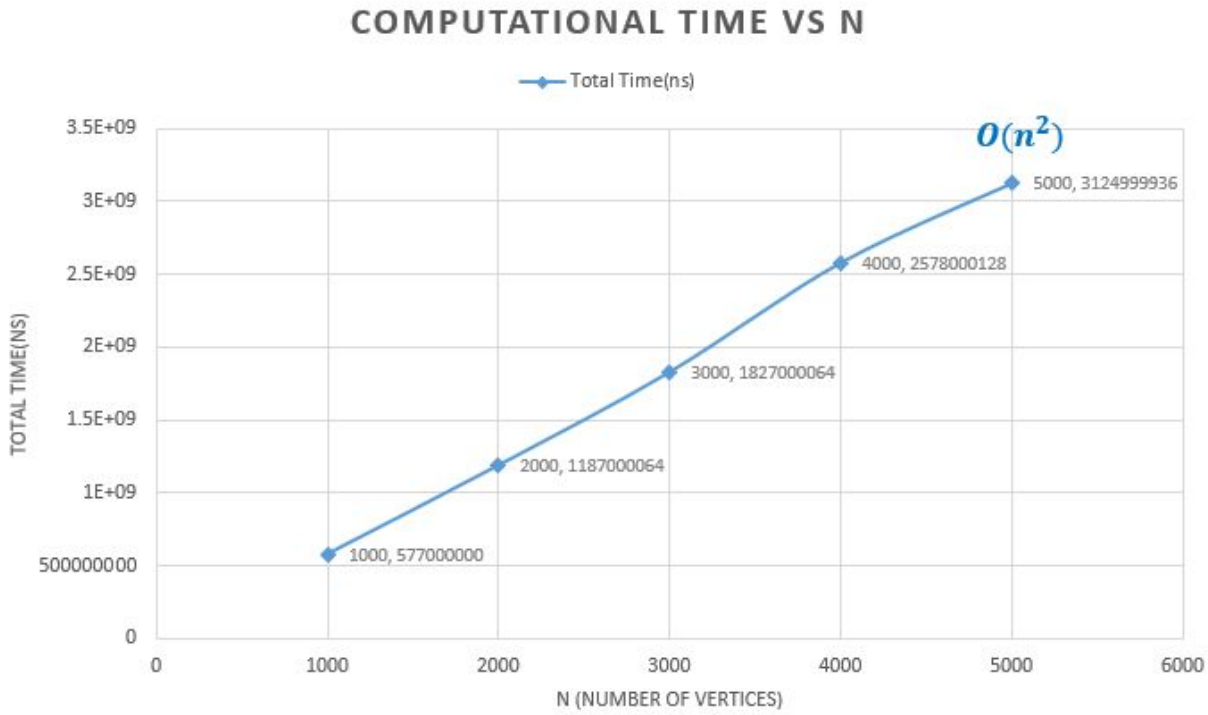
Md.Abid Sarkar(2020-1-60-196)

S.M.Sabbir Sobhani(2020-1-60-088)

Sanjida Akter(2020-1-60-222)

Ans of the question no.4

	A	B
1	n(Number of Ver	n(Number of Vertices)
2	1000	577000000
3	2000	1187000064
4	3000	1827000064
5	4000	2578000128
6	5000	3124999936
7		
8		



From the graph,the approximate time complexity of the program as a function of n is O(n^2).

Ans to the question no.5

In the code,line 1-17,25-26,54-58,63-72 will not have an effect on the final time complexity as they are all linear O(1).

Line 18-24:

```
for (initial_Vertex = 0; initial_Vertex < n; initial_Vertex++)
   {
      for (Terminal_Vertex = 0; Terminal_Vertex < n; Terminal_Vertex++)
      {
         a[initial_Vertex][Terminal_Vertex] = (rand() % 2); //use rand function to fill the
matrix randomly where %2 is use to put only 0 and 1
      }
   }
```

From line 18 to 24,The outer loop executes n times.Every time the outer loop executes,the inner loop also executes n times.As a result,the statement in the inner loop executes a total of n*n times.
Thus the time complexity is O(n^2).

Line 27-53:

```
for (initial_Vertex = 0; initial_Vertex < n; initial_Vertex++)
   {
      count_degree = 0;
      for (Terminal_Vertex = 0; Terminal_Vertex < n; Terminal_Vertex++)
      {
         if (a[initial_Vertex][Terminal_Vertex] == a[Terminal_Vertex][initial_Vertex]) //here
the condition follow the rules
         {
            Adjacency_Matrix[initial_Vertex][Terminal_Vertex] =
a[initial_Vertex][Terminal_Vertex];

            if (initial_Vertex == Terminal_Vertex &&
Adjacency_Matrix[initial_Vertex][Terminal_Vertex] == 1) //this is for the loops
            {
```

```
                count_degree = count_degree + 2;
            }
            else if (Adjacency_Matrix[initial_Vertex][Terminal_Vertex] == 1)
            {
                count_degree++;
            }
        }
        else if (a[initial_Vertex][Terminal_Vertex] != a[Terminal_Vertex][initial_Vertex])
//here did not follow
        {
            Adjacency_Matrix[initial_Vertex][Terminal_Vertex] = 0; //replies the matrix with
0
        }
    }
    total_degree = total_degree + count_degree; //summation of all degrees
    //degree[initial_Vertex] = count_degree; //storage of degree of every vertex
    printf("degree of vertex %d = %d \n", initial_Vertex, count_degree);
  }
```

From line 27-53, Outer for loop executes n times.Inner for loop also executes n times.

In the inner for loop,there are an outer and inner if then else.In the inner if then else,
From line 36-43,

```
if (initial_Vertex == Terminal_Vertex &&
Adjacency_Matrix[initial_Vertex][Terminal_Vertex] == 1) //this is for the loops
            {
                count_degree = count_degree + 2;
            }
            else if (Adjacency_Matrix[initial_Vertex][Terminal_Vertex] == 1)
            {
                count_degree++;
            }
```

Here either the statement of line 38 executes or the statement of line 42 executes.Both statements are simple and have time complexity O(1).
So,the time complexity of line 36-48=O(max(1,1))=O(1).

```
if (a[initial_Vertex][Terminal_Vertex] == a[Terminal_Vertex][initial_Vertex]) //here the
condition follow the rules
        {
            Adjacency_Matrix[initial_Vertex][Terminal_Vertex] =
a[initial_Vertex][Terminal_Vertex];

            if (initial_Vertex == Terminal_Vertex &&
Adjacency_Matrix[initial_Vertex][Terminal_Vertex] == 1) //this is for the loops
            {
                count_degree = count_degree + 2;
            }
            else if (Adjacency_Matrix[initial_Vertex][Terminal_Vertex] == 1)
            {
                count_degree++;
            }
        }
        else if (a[initial_Vertex][Terminal_Vertex] != a[Terminal_Vertex][initial_Vertex])
//here did not follow
        {
            Adjacency_Matrix[initial_Vertex][Terminal_Vertex] = 0; //replies the matrix with
0
        }
```

Here,Also,In the outer if then else,both the if and else if statement holds simple statement.So,again the time complexity of line 32-48 =O(max(1,1)=O(1).

As the inner for loop executes n times and also the outer for loop executes n times.
Thus the time complexity of line 27-53= n*n*O(1)=O(n^2).

Line 59-62,

```
   if (total_degree == 2 * edges)
   {
      printf("Handshaking theorem holds\n");
   }
```

From line 59-62,

“If” has a simple statement(basic operations).so the time complexity= O(1).

Thus,The total time complexity of the full code
= O(1)+O(n^2)+O(n^2)+O(1)
=O(max(1,n^2,n^2,1)
=O(n^2).

So,theoretically the computational time complexity of the program as a function of n is O(n^2).Also,From the graph in step 4,We have found time complexity of the program as a function of n is O(n^2).Both are the same.