|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE200L: Data Structures & Algorithms Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 06/09/2022** |
| **Teaching Assistant: Muhammad Sufyan Ashraf** | **Semester: Fall 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 2A. Performance Analysis of a Program in terms of Time & Space Complexity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| NIMRA MAQBOOL | BSCE21012 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to provide the knowledge of basic data structures and their implementations.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

**Time Complexity** is a type of computational complexity that describes the time required to execute an algorithm. The time complexity of an algorithm is the amount of time it takes for each statement to complete. As a result, it is highly dependent on the size of the processed data. How much time does mean:

How fast is your computer? Are we running other programs simultaneously? Which programming languages you are using etc.

It is basically how the time taken to execute the program increases with the increase in input size. There are three types of measurements such as: Best Case, Average Case, Worst Case. Rules for calculating complexity:

1. Find the fastest growing term (dominating term) & ignore low order terms.

2. Take out the coefficient (ignore constant).

**Space Complexity** of a program is the amount of memory it needs to run to completion.

**Lab Task**

**Task A**

Find the time complexities of the following expression & codes:

a) x = 15 – (35/7);

|  |
| --- |
| FUNCTION cost repitition total |
| x = 15 – (35/7); 1 1 1 |
| BIG -O is 1=O(1). |

b) x = 15 – (35/7);

cout<<x;

y = 5\*9;

cout<<y;

|  |
| --- |
| FUNCTION cost repitition total |
| x = 15 – (35/7); 1 1 1 |
| cout<<x; 1 1 1 |
| y = 5\*9; 1 1 1 |
| cout<<y; 1 1 1 |
| BIG -O is 1=O(1). |

c) for(i=1; i<n; i++)

cout<<i;

|  |
| --- |
| FUNCTION cost repetition total |
| for(i=1; i<n; i++) 1 1 for (i=1) 1  1 n for (i<n) n  1 n-1 for (i++) n-1 |
| cout<<i; 1 n n |
| BIG – O is n = O(n) |

d) cout<<”Enter value of n”;

cin>>n;

for(i=1; i<=n; i++)

cout<<i;

|  |
| --- |
| FUNCTION cost repetition total |
| cout<<”Enter value of n”; 1 1 1 |
| cin>>n; 1 1 1 |
| for(i=1; i<=n; i++) 1 1 for (i=1) 1  1 n+1 for (i<=n) n+1  1 n for (i++) n |
| cout<<i; 1 n n |
| BIG – O is n = O(n) |

e) for(x=1; x<=n; x++)

for(y=1; y<=n; y++;)

cout<<x+y;

|  |
| --- |
| FUNCTION cost repetition total |
| for(x=1; x<=n; x++) 1 1 for (i=1) 1  1 n+1 for (i<=n) n+1  1 n for (i++) n |
| for(y=1; y<=n; y++;) 1 n for (i=1) 1  1 n(n+1) for (i<=n) (n+1)n  1 n\*n for (i++) n\* n |
| cout<<x+y; 1 n\*n n\*n |
| BIG-O is n^2= o(n^2) |

f) x=5\*9;

for(i=1; i<=n; i++)

cout<<i;

for(x=1; x<=n; x++)

for(y=1; y<=n; y++)

cout<<x\*y;

|  |
| --- |
| FUNCTION cost repetition total |
| x=5\*9; 1 1 1 |
| for(i=1; i<=n; i++) 1 1 for (i=1) 1  1 n+1 for (i<=n) n+1  1 n for (i++) n |
| cout<<i; 1 n n |
| for(x=1; x<=n; x++) 1 n for (i=1) 1  1 (n+1) for (i<=n) (n+1)  1 n for (i++) n |
| for(y=1; y<=n; y++) 1 n for (i=1) n  1 n(n+1) for (i<=n) (n+1)n  1 n\*n for (i++) n\*n |
| cout<<x\*y; 1 n\*n n\*n. |
| BID-O is n^2 =O(n^2). |

**Task B**

For the following codes carry out the analytical analysis to evaluate time complexity and express it in terms of Big Oh:

|  |  |
| --- | --- |
| int sum, i;  sum=0;  for(i=0;i<n;++i)  sum++; | int sum,i,j;  sum=0;  for(i=1;i<n;i=i\*2)  sum++; |
| int sum,i,j;  sum=0;  for(i=1;i<=n;++i)  {  for(j=0;j<i;++j)  {  sum++;  }  } | int sum,i,j;  sum=0;  for(i=0;i<n;++i)  {  for(j=0;j<n;++j)  {  sum++;  }  } |
| int sum,i,j;  sum=0;  for(i=1;i<=n;i=i\*2)  {  for(j=0;j<n;++j)  {  sum++;  }  } | int sum,i,j;  sum=0;  for(i=1;i<=n;i=i\*2)  {  for(j=0;j<i;j++)  {  sum++;  }  } |

|  |
| --- |
| 1) int sum, i;  sum=0;  for(i=0;i<n;++i)  sum++;    FUNCTION cost repetition total  int sum, i; 1 1 1  sum=0; 1 1 1  for(i=0;i<n;++i) 1 n+1 n+1  1 n n  sum++; 1 n n  BIG-O is n = O(n).  2) int sum,i,j;  sum=0;  for(i=1;i<n;i=i\*2)  sum++;  FUNCTION cost repetition total  int sum, i,j; 1 1 1  sum=0; 1 1 1  for(i=1;i<n;i=i\*2) 1 n n  1 log(n) +1 log(n)+1  sum++; 1 log(n) log(n)  BIG-O is log(n) =O(log(n)).  3) int sum,i,j;  sum=0;  for(i=1;i<=n;++i)  {  for(j=0;j<i;++j)  {  sum++;  }  }  FUNCTION cost repetition total  int sum, i,j; 1 1 1  sum=0; 1 1 1  for(i=1;i<n;++i) 1 n+1 for (i<n) n+1  1 n+1 for (++i) n+1  for(j=0;j<i;++j) 1 n for (j<i) n  1 n for (++j) n  sum++; 1 n(n+1)/2 n(n+1)/2  BIG-O is n^2 =O(n^2).  4) int sum,i,j;  sum=0;  for(i=0;i<n;++i)  {  for(j=0;j<n;++j)  {  sum++;  }  }  FUNCTION cost repetition total  int sum, i,j; 1 1 1  sum=0; 1 1 1  for(i=0;i<n;++i) 1 n+1 for (i<n) n-1  1 n+1 for (++i) n-1  for(j=0;j<n;++j) 1 n n  1 n\*n n\*n  sum++; 1 n\*n n\*n    BIG-O is n^2 =O(n^2).  5) int sum,i,j;  sum=0;  for(i=1;i<=n;i=i\*2)  {  for(j=0;j<n;++j)  {  sum++;  }  }  FUNCTION cost repetition total  int sum, i,j; 1 1 1  sum=0; 1 1 1  for(i=1;i<=n;i=i\*2) 1 log(n) +1 log(n)+1  1 log(n) log(n)  for(j=0;j<n;++j) 1 ( logn) for (j<n) log(n)  1 n n  Sum++; 1 nlogn nlogn  BIG-O is nlogn =O(nlog(n)).  6) int sum,i,j;  sum=0;  for(i=1;i<=n;i=i\*2)  {  for(j=0;j<i;j++)  {  sum++;  }  }    FUNCTION cost repetition total  int sum, i,j; 1 1 1  sum=0; 1 1 1  for(i=1;i<=n;i=i\*2) 1 log(n) log(n)  1 log(n)+1 log(n)+1  for(j=0;j<i;j++) 1 logn(log(n)) forj<i logn(log(n))  BIG-O is logn^2=O(logn)^2 |
|  |

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & Github Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_