**Department of Computing**

**School of Electrical Engineering and Computer Science**

**CS250 – Data Structures and Algorithms**



**Lab 10: Computational Time of Algorithms**

**Submission Details**

|  |  |
| --- | --- |
| Name | CMS ID |
| Muhammad Umer | 345834 |
| Group | GP – 1 |
| Lab Engineer | Anum Asif |
| Faculty Member | Bostan Khan |
| Class | BEE12 |
| Date | 15/04/2024 |
| Time | 10:00 am – 12:50 pm |

**Table of Contents**

[2 Computational Time of Algorithms 3](#_Toc164070166)

[2.1 Introduction 3](#_Toc164070167)

[2.2 Objectives 3](#_Toc164070168)

[2.3 Tools/Software Requirement 3](#_Toc164070169)

[2.4 Deliverables 3](#_Toc164070170)

[3 Lab Tasks 4](#_Toc164070171)

[3.1 Task 1 4](#_Toc164070172)

[3.2 Task 2 6](#_Toc164070173)

# Computational Time of Algorithms

## Introduction

This lab is based on the analysis of the computational time acquired by an algorithm.

## Objectives

Objective of this lab is to practice of finding the computational time of an algorithm.

## Tools/Software Requirement

* Visual Studio C++

## Deliverables

Compile a single word document by filling in the solution parts and submit this file on LMS. The name of word document should follow this format. i.e., YourFullName(reg)\_Lab#. You must show the implementation of the tasks in a complete manner to get your work graded.

# Lab Tasks

Consider two different algorithms listed in lab tasks. You will implement two different approaches for solving the same problem and will figure out the time difference between their successful executions.

*Hint: input at least 5 different values of “n” and make their entry in excel sheet. “n” can be of any data type. Try to use larger values of “n” for getting the useful data.)*

**Calculate time in seconds in C+**

#include<time.h>

clock\_t startTime = clock();

//YOUR CODE HERE E.G. RECURSIVE CALLS

cout << double(clock()-startTime)/CLOCKS\_PER\_SEC;

## Task 1

Calculate the Fibonacci series of a number “n” given by user both *recursively and iteratively*. Figure out the difference in computational time by using both approaches.

**Recursive Iterative**

E.g. for **N=10000** 20ms 2ms

* Run some timing experiments with your program while trying different values of *n*. Make sure to time only the computation and not the user entering input etc.
* Draw a graph in Excel showing two functions, one for recursive function and other for iterative function.

Code

#include <time.h>

#include <iostream>

using *namespace* std;

*long* *long* *unsigned* *int* fib\_recursive(*int* *n*) {

    if (*n* == 0) return 0;

    if (*n* == 1) return 1;

    return fib\_recursive(*n* - 1) + fib\_recursive(*n* - 2);

}

*long* *long* *unsigned* *int* fib\_iterative(*int* *n*) {

    if (*n* == 0) return 0;

    if (*n* == 1) return 1;

*long* *long* *unsigned* *int* a = 0, b = 1, c;

    for (*int* i = 2; i <= *n*; i++) {

        c = a + b;

        a = b;

        b = c;

    }

    return c;

}

*int* main() {

*long* *long* *unsigned* *int* n;

*clock\_t* start = clock();

    cout << "Enter a number: ";

    cin >> n;

*// Recursive*

    cout << "Fibonacci series of " << n << " using recursive function: "

         << fib\_recursive(n) << endl;

    cout << "Time taken by recursive function: "

         << (*double*)(clock() - start) / CLOCKS\_PER\_SEC << "s" << endl;

*// Iterative*

    start = clock();

    cout << "Fibonacci series of " << n << " using iterative function: "

         << fib\_iterative(n) << endl;

    cout << "Time taken by iterative function: "

         << (*double*)(clock() - start) / CLOCKS\_PER\_SEC << "s" << endl << endl;

    return 0;

}

Output

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_1

Enter a number: 5

Fibonacci series of 5 using recursive function: 5

Time taken by recursive function: 0.00011s

Fibonacci series of 5 using iterative function: 5

Time taken by iterative function: 2e-06s

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_1

Enter a number: 10

Fibonacci series of 10 using recursive function: 55

Time taken by recursive function: 0.000117s

Fibonacci series of 10 using iterative function: 55

Time taken by iterative function: 3e-06s

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_1

Enter a number: 20

Fibonacci series of 20 using recursive function: 6765

Time taken by recursive function: 0.00015s

Fibonacci series of 20 using iterative function: 6765

Time taken by iterative function: 4e-06s

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_1

Enter a number: 27

Fibonacci series of 27 using recursive function: 196418

Time taken by recursive function: 0.001319s

Fibonacci series of 27 using iterative function: 196418

Time taken by iterative function: 2.5e-05s

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_1

Enter a number: 33

Fibonacci series of 33 using recursive function: 3524578

Time taken by recursive function: 0.020327s

Fibonacci series of 33 using iterative function: 3524578

Time taken by iterative function: 3e-06s

Graph

## Task 2

Calculate the factorial of a number “n” given by user both *recursively and iteratively*. Figure out the difference in computational time by using both approaches.

* Draw a graph in Excel showing two functions, one for recursive function and other for iterative function.

Code

#include <time.h>

#include <iostream>

using *namespace* std;

*long* *long* *unsigned* *int* factorial\_recursive(*int* *n*) {

    if (*n* == 0) return 1;

    if (*n* == 1) return 1;

    return *n* \* factorial\_recursive(*n* - 1);

}

*long* *long* *unsigned* *int* factorial\_iterative(*int* *n*) {

*long* *long* *unsigned* *int* fact = 1;

    for (*int* i = *n*; i > 0; i--) {

        fact \*= i;

    }

    return fact;

}

*int* main() {

*long* *long* *unsigned* *int* n;

*clock\_t* start = clock();

    cout << "Enter a number: ";

    cin >> n;

*// Recursive*

    cout << "Factorial of " << n << " using recursive function: "

         << factorial\_recursive(n) << endl;

    cout << "Time taken by recursive function: "

         << (*double*)(clock() - start) / CLOCKS\_PER\_SEC << "s" << endl;

*// Iterative*

    start = clock();

    cout << "Factorial of " << n << " using iterative function: "

         << factorial\_iterative(n) << endl;

    cout << "Time taken by iterative function: "

         << (*double*)(clock() - start) / CLOCKS\_PER\_SEC << "s" << endl << endl;

    return 0;

}

Output

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_2

Enter a number: 4

Factorial of 4 using recursive function: 24

Time taken by recursive function: 0.000118s

Factorial of 4 using iterative function: 24

Time taken by iterative function: 3e-06s

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_2

Enter a number: 8

Factorial of 8 using recursive function: 40320

Time taken by recursive function: 0.000124s

Factorial of 8 using iterative function: 40320

Time taken by iterative function: 3e-06s

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_2

Enter a number: 12

Factorial of 12 using recursive function: 479001600

Time taken by recursive function: 0.000126s

Factorial of 12 using iterative function: 479001600

Time taken by iterative function: 4e-06s

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_2

Enter a number: 16

Factorial of 16 using recursive function: 20922789888000

Time taken by recursive function: 0.000145s

Factorial of 16 using iterative function: 20922789888000

Time taken by iterative function: 3e-06s

root@Zonularity:/home/zonularity/dsa/lab\_10# ./task\_2

Enter a number: 20

Factorial of 20 using recursive function: 2432902008176640000

Time taken by recursive function: 0.000204s

Factorial of 20 using iterative function: 2432902008176640000

Time taken by iterative function: 4e-06s

Graph