# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BESE 13AB**

# Lab 6: Computational time of algorithms

**Date: 27th October, 2023**

**Time: 10 am - 1 pm & 2 pm - 5 pm**

**Name:** Aimen Munawar

**Class:** BESE-13-A

**CMS ID:** 415867

**Lab Engineer: Anum Asif**

# 

# Lab 6: 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 2012 or gcc or g++

**Helping Material**

Lecture slides, text book

**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.

**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 the 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.

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

(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.)

How to 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;

**Important Note:** Practice your knowledge of OOP with C++ when creating a solution. Remember to comment your code properly. Inappropriate or no comment may result in deduction of marks.

**Solution:**

|  |
| --- |
| Solution |
| Task 1 Code (recursive, iterative):  #include<iostream>  #include<chrono>  using namespace std::chrono;  using namespace std;  // Task 1: Fibonacci Series  // Recursive function to calculate Fibonacci  long long fibonacci\_recursive(long long i) {  if ((i == 0) || (i == 1)){  return i;  } else {  return (fibonacci\_recursive(i - 1) + fibonacci\_recursive(i - 2));  }  }  // Iterative function to calculate Fibonacci  void fibonacci\_iterative(long long size) {  long long firstNumber = 0, secondNumber = 1, temp;  cout << firstNumber << ", "; // Print first Fibonacci number  cout << secondNumber << ", "; // Print second Fibonacci number  for (long long i = 2; i < size; i++){  temp = firstNumber + secondNumber; // Calculate next Fibonacci number  cout << temp << ", "; // Print next Fibonacci number  firstNumber = secondNumber; // Update first number  secondNumber = temp; // Update second number  }  }  int main(){  int choice;  long long size;  cout << "Enter the number of elements in the series:";  cin >> size;  cout << "Choose the approach (1 for Recursive, 2 for Iterative): ";  cin >> choice;  switch (choice) {  case 1: {  cout << "The fibonacci series of numbers is:" << endl;  auto start = high\_resolution\_clock::now(); // Start timing  for (int i = 0; i < size; i++){  cout << fibonacci\_recursive(i) << ", "; // Print Fibonacci series  }  auto stop = high\_resolution\_clock::now(); // Stop timing  auto duration = duration\_cast<microseconds>(stop - start); // Calculate duration  cout <<endl<< "Computational time is " << duration.count() << " µs" << endl; // Print duration  break;  }  case 2: {  cout << "The fibonacci series of numbers is:" << endl;  auto start = high\_resolution\_clock::now(); // Start timing  fibonacci\_iterative(size); // Calculate Fibonacci series  auto stop = high\_resolution\_clock::now(); // Stop timing  auto duration = duration\_cast<microseconds>(stop - start); // Calculate duration  cout <<endl<< "Computational time is " << duration.count() << " µs" << endl; // Print duration  break;  }  default: {  cout << "Invalid choice!" << endl; // Print error message for invalid choice  break;  }  }  return 0;  }  Task 1 Output (recursive, iterative):                      Task 1 Graphs (recursive, iterative):  Table:   |  |  |  | | --- | --- | --- | | **Input** | **Iterative Method (Time)**  **(µs)** | **Recursive Method (Time)**  **(µs)** | | **10** | 5 | 6 | | **15** | 6 | 16 | | **20** | 6 | 101 | | **25** | 7 | 1483 | | **30** | 7 | 19557 |   Graph:  Task 2 Code (recursive, iterative):  #include<iostream>  #include<time.h>  #include <chrono>  using namespace std::chrono;  using namespace std;  // Task 2: Factorial Functions  // Iterative Factorial  long long factorial\_iterative(long long num){  long long fact = 1;  for (long long i = 1; i <= num; i++){  fact = fact\*i;  }  return fact;  }  // Recursive Factorial  long long factorial\_recursive(long long i){  if ((i == 0) || (i == 1)){  return 1;  }  else{  return i \* factorial\_recursive(i - 1);  }  }  int main(){  int choice;  long long number;  // User input  cout << "Enter the number for which the factorial is to be calculated:";  cin >> number;  // User choice  cout << "Choose the approach (1 for Recursive, 2 for Iterative): ";  cin >> choice;  switch (choice) {  case 1: {  // Recursive calculation  cout << "The factorial of this number is:" << endl;  auto start = high\_resolution\_clock::now(); // Start timing  cout << factorial\_recursive(number); // Calculate factorial  auto stop = high\_resolution\_clock::now(); // Stop timing  auto duration = duration\_cast<microseconds>(stop - start); // Calculate duration  cout << endl << "Computational time is " << duration.count() << " µs" << endl; // Print duration  break;  }  case 2: {  // Iterative calculation  cout << "The factorial of this number is:" << endl;  auto start = high\_resolution\_clock::now(); // Start timing  cout << factorial\_iterative(number); // Calculate factorial  auto stop = high\_resolution\_clock::now(); // Stop timing  auto duration = duration\_cast<microseconds>(stop - start); // Calculate duration  cout << endl << "Computational time is " << duration.count() << " µs" << endl; // Print duration  break;  }  default: {  cout << "Invalid choice!" << endl; // Print error message for invalid choice  break;  }  }  return 0;  }  Task 2 Output (recursive, iterative):                      Task 2 Graphs (recursive, iterative):  Table:   |  |  |  | | --- | --- | --- | | **Input** | **Iterative Method (Time)**  **(µs)** | **Recursive Method (Time)**  **(µs)** | | **5** | 4 | 4 | | **10** | 4 | 5 | | **15** | 4 | 5 | | **20** | 5 | 5 | | **25** | 5 | 5 |   Graph: |

### Deliverables

Compile a single word document by filling in the solution part and submit this Word file on LMS. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS.