CS 201, Fall 2018

Homework Assignment 2

Due: 23:59, December 12, 2018

In this homework, you will study the problem of finding the *n*th Fibonacci number. Two alternative algorithms that solve this problem are discussed below (also in class). Each algorithm has a different time complexity. The goal of this homework is to simulate the growth rates of both algorithms using different inputs.

Algorithm 1: An iterative algorithm which works in O(n) time.

Algorithm 2: A recursive algorithm which works in O(2^n) time.

It is possible to improve the run time of the recursive solution using data structures but for the scope of this homework, you are expected to use the version discussed in class.

ASSIGNMENT:

1. Study the algorithms and understand how the upper bounds are found for the running time of each solution.
2. Use the implementations given in the slides and write a driver (main) function that calls these functions. Then, run each solution and record the execution times when different *n* values are used. You are expected to try many different input sizes, both small inputs and very large inputs (as large as around 1,000,000,000 for the iterative solution and around 55 for the recursive solution), to observe the effects of different growth rates.
3. Report the results in a table. Then, use these results to generate a plot of running time (y-axis) versus the input size *n* (x-axis). Specifically, you are expected to produce a plot as in Figure 2.3 of the handout chapter on algorithm analysis.
4. Provide the specifications of the computer you used to obtain these execution times. You can use any computer with any operating system for this assignment.
5. Plot the expected growth rates obtained from the theoretical analysis (as given for each algorithm above) by using the same *n* values that you used in your simulations.
6. Compare the expected growth rates and the obtained results, and discuss your observations in a paragraph.

You can use the following code segment to compute the execution time of a code block. For these operations, you must include the ctime header file.

```
//Store the starting time
double duration;
clock_t startTime = clock();

//Code block
...

//Compute the number of seconds that passed since the starting time
duration = 1000 * double( clock() - startTime ) / CLOCKS_PER_SEC;
cout << "Execution took " << duration << " milliseconds." << endl;
```

An alternative code segment to compute the execution time is as follows. For these operations, you must include the chrono header file.

```
//Declare necessary variables
std::chrono::time_point< std::chrono::system_clock > startTime;
std::chrono::duration< double, milli > elapsedTime;

//Store the starting time
startTime = std::chrono::system_clock::now();

//Code block
...
//Compute the number of seconds that passed since the starting time
elapsedTime = std::chrono::system_clock::now() - startTime;
cout << "Execution took " << elapsedTime.count() << " milliseconds." << endl;
```

NOTES:

1. In this assignment, you must submit a report (as a pdf file) that contains all information requested above (plots, tables, computer specification, discussion) and a cpp file that contains the main function that you used as the driver in your simulations (you do not need to submit the solution functions) in a single zip file. The name of this zip file should conform to the following name convention: secX-Firstname-Lastname-StudentID.zip where X is your section number. Otherwise, you may lose points.
2. You should prepare your report (plots, tables, computer specification, discussion) using a word processor (in other words, do not submit images of handwritten answers) and convert it to a pdf file. If you have any images of handwritten answers in your report, you will lose a considerable number of points.
3. The recursive algorithm leads to a large range of running times. Thus, to obtain meaningful plots and better compare the running times of the iterative and recursive algorithms, we strongly suggest you to use the logarithmic scale for the y-axis of your plots. Please use a standard scale to report the running times of these algorithms in the table.
4. The code segments given above may display 0 milliseconds when the value of *n* is small. Of course, the running time cannot be 0 but it seems to be 0 because of the precision of the used clock. However, we will not accept 0 as an answer. Thus, to obtain a running time greater than 0, please consider running an algorithm *M* times using a loop, and then divide the running time (which will be greater than 0) to *M*.
5. Then, before 23:59 on December 12, you need to send an email with a subject line CS 201 HW2 to Gözde Nur Güneşli, by attaching this zipped file containing your report and your driver file.
6. No hardcopy submission is needed. The standard rules about late homework submissions apply.
7. This assignment will be graded by your TA Gözde Nur Güneşli (nur.gunesli@bilkent.edu.tr). Thus, you may ask your homework related questions directly to her.

VERY IMPORTANT: We expect all of you to comply with academic integrity. The honor code statement, which has been sent to you by email, was prepared to clarify our expectations about the academic integrity principles. Please study the part of this statement related with "individual assignments" very carefully.

If you submit this homework assignment, we will assume that you all read this honor code statement and comprehensively understood its details. Thus, please make sure that you understood it. If you have any questions (any confusions) about the honor code statement, consult with the course instructors.

We will check your submissions for plagiarism.