School of Computer Science, University of Windsor

COMP 2540: Data Structures and Algorithms

Term: Summer 2024

Instructor: Dr. Asish Mukhopadhyay

Lab 4
Posted: 03 June, 2024
Due: 11:59 pm, 11th June, 2024

Instructions:

- You are expected to finish the lab by the end of the posted date. Submissions beyond the due date will earn a penalty of $n * 25\%$, where $n = submissionDay - dueDay$. Thus if the lab is due Tuesday and you submit on Wednesday, this will be considered a day late.
- Whether or not you finish your work during the lab hour you will have to upload your work on Brightspace for record-keeping and grading before the beginning of the next lab. Create a script file as follows:

```
1. script LabName.txt
2. cat LabName.c (this includes the c-file)
3. cat input.txt (ignore this line if no input.txt file has been created)
4. cc labName.c  (or gcc -Wall labName.c -lm if using the library function log and ceil)
5. ./a.out < input.txt (./a.out, if there is no input.txt file)
6. ls -l
7. exit (DO NOT FORGET THIS STEP!!)
```

- There will be no make-up for missed labs. If you have missed a lab for truly extenuating circumstances (like illness or family emergency) I will consider allowing you to make a late submission. However, I need to be informed by email about this on the day of the missed lab. The email should include your name and SID.

Problem:

In this lab you will have to write three different functions to compute the n-th Fibonacci number for a given non-negative integer n. The idea is to emphasize the fact that if you write a recursive program for computing a Fibonacci number, based directly on its recursive definition, it can be agonizingly slow. All implementations are to be done in C.

Part 1: Implement the recursive algorithm for computing the n-th Fibonacci number, based on the following slightly modified recursive definition of the n-th Fibonacci number than in the courseware or the lecture slides:

$$F_n = F_{n-1} + F_{n-2} \text{ for } n > 1, \text{ and } F_0 = 0, F_1 = 1.$$

Part 2: Implement an iterative algorithm for computing the n-th Fibonacci number. You have to store the two previous Fibonacci numbers, compute the current Fibonacci number and update the previous two in a loop till you reach the n-th. Once again, this is also based on the above definition.

Part 3: The following code segment in C (the part within braces is the same in C, only the declarations need to be modified) uses the above recursive definition, except that it stores in a global array $knownF[maxN]$, the previously computed values of the Fibonacci numbers. This method is technically known as dynamic programming and I shall have more to say about it in my lectures. For now, figure out how it works, why it takes $O(n)$ time to compute the n-th Fibonacci number, complete all the coding details and then run it.

```
// declarations

{
  if (knownF[n] != 0) return knownF[n];
  unsigned long t = n ;
  if (n < 0) return 0;
  if (n > 1) t = F(n-1) + F(n-2);
  return knownF[n] = t;
}
```

You have to compare the running times for all the three implementations for computing the Fibonacci number of a given rank $n = 0, 1, 2, 3, ..,$ as input; make your program interactive. You will face two problems: integer overflow and the slowness of the recursive method as n gets large.

Note that for the purpose of this lab we have defined $F_0 = 0$ and $F_1 = 1$, unlike in the course notes and lecture slides where we have defined $F_0 = 1$ and $F_1 = 1$.

(10 points)

Evaluation scheme: 5 marks for effort, 4 marks for correctness, 1 mark for comments and readability of your program.