EC330 Applied Algorithms and Data Structures for Engineers
Spring 2024

Homework 2

Out: February 8, 2024
Due: February 22, 2024

This homework has a written part and a programming part. Both are due at 11:59 pm on February 22. You should submit both parts on Gradescope.

You should complete the written part (Q1) and the first programming problem (Q2.a) on your own. For the second coding problem (Q2.b), you are allowed to partner with a classmate and submit a single solution for both of you on Gradescope. You are also welcomed to complete this part on your own. However, you cannot be a partner in more than one team. See the course syllabus for our policy on collaboration.

1. Asymptotic Comparison [30 pt]

In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (i.e. $f = \Theta(g)$). *Justify your choice*. **[5 pt each]**

	$f(n)$	$g(n)$
a)	$n + 330 \log n$	$(\log n)^3$
b)	$330 \log 330n$	$\log (n^2)$
c)	$n^{1.01}$	$n \log^2 n$
d)	$n^2 / \log n$	$(\log n)^2$
e)	$(\log n)^{\log n}$	$n / \log n$
f)	$n^2 2^n$	3^n

2. Programming [70 pt]

Make sure you acknowledge any source you consult at the top of your program. Do not include a `main` *in your submitted files. Do not modify the header files. You can make a group submission (for both you and your partner) on Gradescope. If you choose to make separate submissions, make sure you write your partner's name as a comment at the top of the submitted .cpp file.*

a) **[40 pt]** Consider the following sequence of numbers A_i such that $A_0 = 1$ and $A_{n+1} = \sum_{i=0}^{n} A_i A_{n-i}$ for $n \geq 0$. For example, $A_1 = \sum_{i=0}^{n=0} A_i A_{n-i} = A_0 A_0 = 1$, and $A_2 = \sum_{i=0}^{n=1} A_i A_{n-i} = A_0 A_1 + A_1 A_0 = 2$, and $A_3 = 5$ (try to work this out yourself on paper).

Write both a recursive program (which can contain loops) and an iterative program (which must not make recursive calls) in C++ that generates A_n in this sequence. You can assume n is a non-negative number. For the recursive implementation, it must not make redundant recursive calls.

Example:
Input: 3 (explanation: 3rd number in the sequence, i.e. A_2)
Output: 2

Implement the function *genA_recur and genA_iter* in *genA.cpp*. Submit your completed *genA.cpp* file on Gradescope.

In addition, create a *test_genA.cpp* file that compares the runtimes of genA_recur and genA_iter for a sufficiently large range of n. In the written part of your solution, include a figure that plots the runtime (in milliseconds) of both programs (on the same figure) over this range. In addition, answer the following questions.

- What is the time complexity in ($\boldsymbol{\Theta}(\cdot)$) of the iterative solution? Justify your answer.
- Does the recursive solution have the same time complexity?
- Do the two solutions have roughly the same runtime in practice (when they are run on your machine)? If not, which one runs faster and why (one reason is enough for the 'why' part).

You will also need to submit *test_genA.cpp* on Gradescope. This file will not be tested but will be checked for code plagiarism. For this part, you are allowed to use ChatGPT (or similar technologies) to figure out how to time the execution of a C++ program and/or to plot its runtime.

b) **[30 pt]** Consider the numerical part of your BU ID and treat that as an integer. Wenchao wants to find the two students in the class such that the sum of their BU IDs modulo 330 is the smallest. You can assume that $x \ mod \ 330$ is in the range of [0, 329] for any integer x.

For instance, say we have students Alice, Bob and Carol and their BU IDs (shorter than the real ones for simplicity) are 7305, 2204 and 1573 respectively. The smallest (BU ID #1 + BU ID #2) mod 330 is (2204 + 1573) mod 330 = 147, and the corresponding students are Bob and Carol.

Implement the *findStudents* function in *findStudents.cpp*. Your code must run asymptotically faster than $\boldsymbol{O}(n^2)$ where n is the number of students in the class. If there are more than one possible result, you can return any of those.