Analysis of Algorithms
CS 375, Spring 2019
Homework 12
Due **AT THE BEGINNING OF CLASS** Monday, April 15

- From your textbook (CLRS), please read Chapter 15, pages 359–360, 378–396. These are not the topic of this HW, but we will be discussing this material in the next class meetings.
- Some exercises on this problem set are adapted from exercises in our recommended Levitin textbook.
- *A general note:* When writing up your homework, please write neatly and **explain your answers clearly**, giving all details needed to make your answers easy to understand. Graders may not award credit to incomplete or illegible solutions. Clear communication *is* the point, on every assignment.

Exercises

1. Find the number of different ways to climb an n-stair staircase if each step is either one or two stairs. (For example, a 3-stair staircase can be climbed three ways: 1-1-1, 1-2, or 2-1.)

   As always, be sure to include a short explanation (a few sentences) of your answer.

2. Consider a rectangle whose side lengths are two consecutive Fibonacci numbers. (Of course, neither of them is 0.) Such a rectangle could be, for example, 3 by 5, or 8 by 13, or 21 by 34, etc.

   (a) Give a recursive algorithm to dissect such a rectangle into squares such that no more than two of the resulting squares are the same size. (For example, if you had two 3 by 3 squares, you could have at most one 4 by 4 square.) As always, be sure to give an English description of the algorithm, explaining the main points of its design—pseudocode is very helpful for algorithms, but a pseudocode presentation without an English explanation will not receive full credit.

   (b) What is the time complexity of your algorithm in part 2a? As always, fully explain your answer, and be sure to explicitly say what each variable in your complexity class stands for (e.g., if you're presenting a $\Theta(n^3)$ algorithm, be sure to say what n refers to).

   (c) **Short Answer:** Please reflect on the following: What would the design ideas be for an iterative algorithm for this problem? If you can't readily come up with an iterative algorithm, why do you think that is?

   The purpose of this part of the exercise is not to ask you to fully develop an iterative algorithm, but to ask you to reflect on the differences between recursive design and iterative design. Please address that in your answer! (A few sentences, a short paragraph at most, should be sufficient.)