Analysis of Algorithms
CS 375, Fall 2022

Problem Set 4

Due **AT THE BEGINNING OF CLASS** Wednesday, November 30

- For this assignment, standard file naming conventions apply: Please submit your typewritten answers in a PDF file named `CS375_PS4_<userid>.pdf` where `<userid>` is replaced by your full Colby userid, and submit it to your `SubmittedWork` folder. Please reach out to me right away with any questions or concerns about this!

- *A general note for CS375:* When writing up your homework, please present your answers neatly and **explain your answers clearly**, giving all details needed to make your answers easy to understand. Graders may not award full credit to incomplete or hard to understand solutions. Clear communication *is* the point, on every assignment.

  In general in CS375, unless explicitly specified otherwise, answers should be accompanied by explanations. Answers without explanations may not receive full credit. Please feel free to ask me any questions about explanations that might come up!

Exercises

1. Prof. Nigma at the Portland Institute of Technology (which still doesn't exist) was thrilled with your previous work and hired you again to analyze the algorithm given here in pseudocode! As usual, however, Prof. Nigma neglected to explain what the algorithm does.

```
def Q($A[0..n-1]$)
// Input: Array $A[0..n-1]$ of $n$ real numbers, for $n \geq 1$
    if $n = 1$ return $A[0]$
    else $temp = Q(A[0..n-2])$
        if $temp \leq A[n-1]$ return $temp$
        else return $A[n-1]$
```

(a) What does this algorithm compute? Give an English description of what value it returns on a given input array, along with a convincing explanation of your answer. (You do not need to give examples as part of your answer, but as always, you are welcome to include examples along with the English description, if it would make your answer clearer.)

(b) What is the Θ complexity class of this algorithm? To find it, first state a recurrence for the time complexity of the algorithm, then solve it using one of the methods presented in class for solving recurrences. Be sure to give a thorough explanation of what makes your recurrence correct, and show your work for solving the recurrence, including showing a k'th step in your method, if you use unwinding or recursion trees. (If you use recursion trees, give a table as on Smaller Assignment SA4 to show your work.)

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.) Here's a specification for your algorithm:

   ```
   // Input: Two consecutive Fibonacci numbers f0, f1,
        representing an f0 by f1 rectangle, such that f0 <= f1.
        (Neither f0 nor f1 will be 0.)
   // Output: A list of integers representing side lengths of squares,
        such that the input rectangle can be dissected into squares
        of those sizes. No more than two of the squares can be the
        same size.
   ```

   Please be sure to give an English description of the algorithm along with pseudocode, explaining the main points of its design, and a concise inductive argument for its correctness (i.e., say what makes the base case correct, what makes the recursive cases correct, and how you know the algorithm terminates).

   (b) What is the time complexity of your algorithm in part 2a? Give a recurrence for your algorithm, and solve it to get a Θ complexity bound. As always, fully explain your answer, show your work in solving the recurrence, 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).