

Analysis of Algorithms
CS 376, Fall 2020

Problem Set 1

Due **AT THE BEGINNING OF CLASS** Wednesday, September 9

Naming conventions: Please submit this assignment in a file named

CS376_PS1_<YourName>.pdf.

- From your textbook (CLRS), please read:
 - Appendices A, B.1–B.2 (that’s Appendix B, sections 1 through 2), C.1, D.1. (The rest of Appendix B will be helpful later in the course, too!)
 - Chapters 1 and 2.1–3.
- Unless otherwise specified, exercises will be from the CLRS textbook and will be named on HW assignments by exercise number used in the book.
- A few essential style guidelines for writing algorithms in CS376:
 - Unless explicitly instructed otherwise, avoid using **break** (or **continue** or **pass**) statements in algorithms for CS376—employing them can interfere with common approaches to reasoning about code correctness (e.g., using loop invariants). Instead, please find other approaches, perhaps using boolean variables to indicate loop termination conditions.
Answers employing **break** (**continue**, **pass**) may not receive full credit. Please feel free to talk with me about any particular situations that might come up—I’m happy to talk with you about how to avoid **breaks** in your code!
 - When presenting an algorithm, avoid using language-specific commands/routines (without comments) in pseudocode. Pseudocode, by definition, is supposed to be language-independent, at a level of abstraction higher-level than particular programming languages.
 - Please be sure to initialize variables or declare inputs/outputs, and be sure the purpose of **every variable**—including the names of functions or algorithms—can be quickly and fully understood by a reader; typically, this is done either by using descriptive variable names or by describing variables in English. Algorithms in which variables cannot be promptly understood may not receive full credit, so please feel free to ask me about variable names and descriptions—I’m happy to talk with you about particular situations that might come up!
 - Often, the clearest way to present an algorithm is with both pseudocode and English description. Pseudocode must be accompanied by some kind of clarifying English description to count as fully explained.

Please note that, unless explicitly instructed otherwise, these guidelines apply to all exercises in CS376, whether or not they explicitly appear as part of the assignment itself. (This is similar to how all answers should be accompanied by explanations unless explicitly specified otherwise, whether or not that instruction is explicitly given as part of the assignment.)

- *A general note for CS376:* When writing up your homework, please write neatly and **explain your answers clearly**, giving all details needed to make your answers easy to understand. Clear communication *is* the point, on every assignment.

In general in CS376, unless explicitly specified otherwise, answers should be accompanied by explanations; as part of this, in general, when you present an algorithm, give an accompanying correctness argument. Answers without explanations may not receive full credit. Please feel free to ask me any questions about explanations that might come up!

Exercises

1. Give proofs for the following properties of logarithms, assuming that a, b, c, n are positive real numbers. (These properties will be useful for complexity analysis later in CS376!) Write your proofs out carefully.

(a) $\log_b a^n = n \log_b a$.

(b) $\log_b a = \frac{\log_c a}{\log_c b}$.

(c) $a^{\log_b n} = n^{\log_b a}$.

2. CLRS B.2-3 (page 1165).
3. Using the LList data structure, write a recursive algorithm for the *LLRemoveAll* problem on lists:

```
# Input: Item  $x$  (assume  $x$  is not a list) and LList  $L$ 
# Output: List  $L'$  containing exactly the elements of  $L$ 
#         not equal to  $x$ , in the order in which they occur in  $L$ 
#         (see examples below)
```

This removes all—and only!—top-level occurrences of x in L , i.e., those that are not nested inside other lists. For examples,

- `LLRemoveAll(42, [55, 77, 42, 11, 42, 88])` returns `[55, 77, 11, 88]`
- `LLRemoveAll(42, [55, [77, 42], [11, 42], 88])` returns `[55, [77, 42], [11, 42], 88]`. (Note that 42 is not top-level.)
- `LLRemoveAll(42, [42, 67, 42, [42, 42, 43], 47])` returns `[67, [42, 42, 43], 47]`. (Only the top-level 42's are removed.)

As usual, explain your answer and give a correctness argument.

4. CLRS 2.3-5 (page 39), modified a little: Write a recursive binary search, not iterative; and you do not have to give a complexity argument.
5. Use induction on n to show that for all integers $n \geq 0$

$$1 + a + a^2 + a^3 + \dots + a^n = \frac{a^{n+1} - 1}{a - 1}$$

where a is an arbitrary real number other than 1. Where does your proof rely on the fact that $a \neq 1$?

6. Write a recursive algorithm that takes a natural number n as input, returns nothing, and prints number n one line at a time on the screen. For example, on input 376, the output on the screen would be

3
7
6

In your algorithm, **for full credit, you may not use any string functions (other than print)**—the input is a number and should be treated that way. In addition, **for full credit, your algorithm should use only the basic arithmetic functions $+$, $-$, $*$, $/$, and the modulo operator $\%$** when performing number-valued arithmetic operations, it should not use auxiliary functions, and it should not use other arithmetic functions (e.g., power or log) even if you write those functions yourself.