

Analysis of Algorithms
CS 375, Spring 2021

Problem Set: Divide & Conquer I

Due **BY THE BEGINNING OF CLASS** Monday, March 1

- To submit this Problem Set, the standard file naming and HW submission conventions for CS375 apply: Please put your answers in a PDF file named

CS375_PSDivConqI_<userid>.pdf

where <userid> is replaced by your full Colby userid (including your class year)—for example, my file would be called `CS375_PSDivConqI_eaaron.pdf`—and submit it to your **SubmittedWork** folder (as discussed in our Feb. 17 class meeting). Please reach out to me right away with any questions or concerns about this!

- From your textbook (CLRS), please read Chapters 2.1, 2.2, and 2.3.1.
- **IMPORTANT:** Some of these exercises may build upon topics covered in our Feb. 24 class meeting; they are included here “early” so you can see all of the exercises on this assignment.
- 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 CS375:
 - 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. (And see below about the importance of explanations on CS375 HW!)
- *A general note for CS375:* As always, please present answers cleanly and **explain them clearly and thoroughly**, giving all details needed to make your answers easy to understand; typed-up (rather than handwritten) answers are especially appreciated. (Feel free to talk with your Prof. or TA’s about using L^AT_EX to typeset your answers!) Graders may not award full credit to incomplete or illegible 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. Using our *IntBinTree* data structure from class (*IBT*, for short), come up with a recursive algorithm that returns the sum of the elements in a tree.

```
# Input: IntBinTree T
# Output: The sum of all of the integers in tree T
```

What did you decide that the algorithm should return on an empty tree as input? Explain your reasoning for that decision (a sentence or so could be sufficient), and give an inductive explanation of the algorithm's correctness.

2. Write a straightforward *recursive* algorithm to compute the exponential function 2^n for natural numbers n :

```
# Input: Natural number  $n$  (i.e., integer  $\geq 0$ )
# Output: Value  $2^n$ 
```

It is expected that your answer will be straightforward and uncomplicated, but you may not use an exponentiation operator in your code (other than the one you're defining)! As always, give a correctness argument for your algorithm.

Also, answer the following questions:

- (a) How many multiplication operations does it do, over all recursive calls, before it terminates?
 - (b) Consider the related problem where input n is restricted to be a large power of 2 (i.e., n itself would be 2^k for some natural number $k > 10$), and come up with an algorithm to compute 2^n using fewer multiplications than your straightforward algorithm would. Describe the algorithm in English (you may also use pseudocode, if you'd like) and say how many multiplication operations this algorithm would do to compute 2^n (for these restricted values of n).
3. (Yet another classic problem solving puzzle. No bridge this time!) There are n hikers who need to cross a river. The river is too wide and deep to cross on foot, and there's no bridge in sight. As is common in these kinds of puzzles, however, there's a twist!

They notice two boys playing in a small rowboat by the river bank. The boat is so tiny, however, that it can only hold two boys or one hiker. How can all n of the hikers get across the river and leave the boys in joint possession of the boat? In your solution, how many times does the boat need to pass from one side of the river to the other?