

Analysis of Algorithms
CS 375, Fall 2019

Homework 3

Due **AT THE BEGINNING OF CLASS** Wednesday, September 18

- In general, there may be multiple correct ways of presenting an algorithm, although excessively inefficient or inelegant solutions may not receive full credit. If you have questions about whether your proposed solution is excessively inefficient or inelegant, please ask your Prof.!
- *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. 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)!

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).
2. Using the *LList* data structure from class, write a recursive algorithm for the *reverse* problem on lists:

```
# Input: List  $L = [a_0, a_1, \dots, a_n]$   
# Output: List  $L' = [a_n, \dots, a_1, a_0]$  with the same elements as in  $L$   
#           but in reverse order
```

As usual, give a short English explanation of correctness; because the algorithm is recursive, make sure it's an inductive explanation.

3. **(Same problem, new syntax!)** Using Python-like list-slicing and indexing syntax in pseudocode, along with Python-like list functions **append** and **extend** (if needed), write a recursive algorithm for the *reverse* problem on lists.

As usual, give a short English explanation of correctness. (You can certainly refer to your explanation from the previous exercise when doing this one! Because the operations used here are different, though, the explanation here will not be identical to the previous one.)