

Analysis of Algorithms
CS 375, Spring 2020

Homework 8

Due **AT THE BEGINNING OF CLASS** Monday, March 9

- From your textbook (CLRS), please read all of Chapters 2 and 3. Although not all of it will show up on CS375 exercises, it is all good material to know.
- Please recall the guidelines for algorithms given on previous HW assignment sheets. They continue to apply to all HWs in CS375.
- 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 for CS375:* 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 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. List the following functions of n according to their order of growth—that is, how fast each function grows as n gets big—from lowest to highest:

$$(n-2)!, \quad 5 \lg(n+100)^{10}, \quad 2^{2n}, \quad 0.001n^4 + 3n^3 + 1, \quad \ln^2 n, \quad \sqrt[3]{n}, \quad 3^n.$$

(As is conventional, the $\lg$ function is logarithm base 2; the $\ln$ function is the *natural logarithm*, logarithm base e ; and $\ln^2 n$ is common notation for $(\ln n)^2$.) Although you don't need to explain every part of the ordering for this exercise, please give short explanations (1–2 sentences) for the following:

- (a) how you know the second-smallest comes before the third-smallest; and
 - (b) how you know the second-largest comes after the third-largest.
2. For each of the following assertions, say if they are True or False, and explain your answers by showing in detail that the relevant functions satisfy (or don't satisfy, as appropriate) the formal definitions of asymptotic complexity, as we've done in the examples in class.

As part of this, please recall that to show a existentially quantified statement is true—e.g., “ $\exists c > 0$ such that ...”—you must give a concrete value for the relevant variable (here, c); just for terminology's sake, a concrete value that shows the truth of an existentially quantified statement is called a *witness*. Merely saying that such a c exists without giving a witness is not an complete explanation of how we know the statement is true.

- (a) $\frac{n(n+1)}{2} \in O(n^3)$
- (b) $\frac{n(n+1)}{2} \in O(n^2)$
- (c) $\frac{n(n+1)}{2} \in \Theta(n^3)$
- (d) $\frac{n(n+1)}{2} \in \Omega(n)$

3. CLRS Exercise 3-4, parts b and f (page 62).

For each part, if the assertion is true, say so, and if not, provide a *counterexample* (i.e., a specific, concrete example that demonstrates the assertion is not true). There is no need to give full proofs of your answers, but please give short explanations (2–3 sentences or so) for each answer, explaining your reasoning (e.g., how you know the assertion is true, or what makes your counterexample a counterexample).

As always in CS375, for full credit, please make sure your explanations demonstrate command of the relevant concepts—merely saying that a statement is true because it is given as true in the textbook is not a sufficient explanation for CS375, because it does not demonstrate understanding of the reasons *why* the statement is true, based on the relevant definitions. Please feel free to ask your Prof. any questions about this, or about explanations in CS375 more generally!

4. Prof. E. Nigma of the Portland Institute of Technology hired you to analyze the algorithm given here in pseudocode, but as usual, Prof. Nigma neglected to explain what the algorithm does.

```
// Input: A matrix A[0..n-1, 0..n-1] of integers
for i = 0 to n-2 do
    for j = i+1 to n-1 do
        if A[i,j] != A[j,i]
            return False
return True
```

In the above, recall that a matrix is essentially just a two-dimensional array, so $A[i, j]$ might in some languages be written as $A[i][j]$.

- (a) What does this algorithm do? Give an English description of what inputs lead to it returning True and what inputs lead to it returning False. (You do not need to give examples as part of your answer, but you are welcome to include example 2D arrays along with the English description, if it would make your answer clearer.)
- (b) Give an *exact* count of the number of array accesses (count $A[x, y]$ as a single array access) done by this algorithm in the worst case on input of size n , and based on that, give the Θ complexity class for this algorithm. Be sure to explain the details behind your answer: show all work that you did to count those operations and arrive at a summation that expresses the running time; and explain how you got from that summation to the Θ complexity bound, giving the relevant threshold n_0 and leading constants c_1, c_2 used to establish that Θ bound.