

Analysis of Algorithms
CS 375, Fall 2019

Homework 10

Due **AT THE BEGINNING OF CLASS** Monday, October 28

- From your textbook (CLRS), please read all of Chapters 2 and 3, and pages 65–67 and 88–93 from Chapter 4. Although not all of it will show up on CS375 exercises, it is all good material to know.

Exercises

1. Solve the following recurrence **showing your work, using the recursion tree method** and give the Θ class of the solution.

$$T(n) = 3T(n - 1) \text{ for } n > 1; T(1) = 4.$$

2. An algorithm to compute 2^n would be defined by the following specifications:

```
// Input: n, a non-negative integer (i.e., 0 or greater)
// Output: 2^n (i.e., 2 to the n'th power)
```

- (a) Write a recursive algorithm for computing 2^n for any non-negative integer n **that is based on the formula** $2^n = 2^{n-1} + 2^{n-1}$. (This may not be the most natural way to think of an algorithm to compute 2^n , but please be sure to use it for this exercise!)
 - (b) Set up a recurrence for the number of additions made by the algorithm on input n and solve it **using the recursion tree method**.
 - (c) What is the Θ complexity class of this algorithm? If the above recurrence is helpful in this, explain why; if not, set up a different recurrence for the runtime of this algorithm and solve it **using a method of your choice** to get the complexity class.
 - (d) Is this a good algorithm for computing 2^n ? As always, be sure to explain your answer!
3. Prof. Nigma at the Portland Institute of Technology (motto: “We don’t think about acronyms!”) hired you to analyze the algorithm given here in pseudocode, but as usual, 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 ≥ 1
  if n = 1 return A[0]
  else temp = Q(A[0..n - 2])
    if temp ≤ 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. (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) Set up a recurrence for an exact count of the number of comparison operations $\leq$ performed by the algorithm, and solve it using the recursion tree method, showing your work.
 - (c) What is the Θ complexity class of this algorithm? If the above recurrence is helpful in this, explain why; if not, set up a different recurrence for the runtime of this algorithm and solve it **using a method of your choice** to get the complexity class.
4. Give Θ bounds for the following recurrences. *Be sure to use the Master Theorem for these exercises.* As always, be sure to give a brief explanation of your answers—here, that will include the values for each relevant variable in the Master Theorem, what case of the Theorem you are applying, and a brief explanation of how you know what case to apply.
- (a) $T(n) = 4T(n/2) + n^2, T(1) = 1.$
 - (b) $T(n) = 4T(n/2) + n^3, T(1) = 1.$
5. Give Θ bounds for the following recurrences, using any of the three methods introduced in class (unwinding, recursion tree, Master method). As always, be sure to give a brief explanation (with appropriate details) of your answers.
- (a) $T(n) = 3T(n/2) + n \lg n, T(1) = 1.$
 - (b) $T(n) = 4T(n/2) + n^2\sqrt{n}, T(1) = 1.$

OPTIONAL Exercise: Practice with Loop Invariants

In case you're looking for more practice with loop invariants, here's the exercise I mentioned previously: The maintenance step for Selection Sort. This exercise will not be graded, so please do not submit it with HW10! If you'd like to write it up and have me give you feedback on it, however, I'll be happy to do so—just turn it in on paper to me directly.

- Consider this pseudocode algorithm for the sorting method *Selection Sort*:

```
SELECTIONSORT( $A[1 \dots n]$ )
  for  $i = 1$  to  $\text{length}[A] - 1$ 
    min =  $i$ 
    for  $j = i + 1$  to  $\text{length}[A]$ 
      if  $A[j] < A[\text{min}]$ 
        min =  $j$ 
    // the next 3 lines swap  $A[i]$  and  $A[\text{min}]$ , using a temporary variable
    temp =  $A[i]$ 
     $A[i] = A[\text{min}]$ 
     $A[\text{min}] = \text{temp}$ 
```

Given the following proposed loop invariant for the outer for loop of SELECTIONSORT, show the *maintenance* part of a correctness argument using the invariant.

To do this, consider the loop invariant, presented here for convenience:

Subarray $A[1..i - 1]$ contains the $i - 1$ smallest elements of A in sorted order,
and $A[i..n]$ consists of the remaining values of A (no constraint on order).

Then, to show the maintenance step, show that for the algorithm as given here, if the invariant is true at the beginning of an iteration, then it's true at the end of the iteration / the beginning of the next iteration. That is, for any iteration m , if it's true when $i = m$, then it's true when the iteration is over and i becomes $m + 1$. Use the definition of the algorithm (i.e., the pseudocode) in the explanation. (Diagrams or specific examples are not sufficient for an explanation, but if you'd like to include them along with a textual explanation, feel free to do so.)