Basic Algorithms, Summer 2023
CSCI-UA.0310-001

Homework 2

Name:

Due Friday, June 9 (11:59 p.m.)

Instructions

- Answer each question on a separate page.
- Submit your homework as a *single* pdf file to Gradescope. It can be either a scanned copy of your written solutions or photos joined into a pdf; your solution should be readable. You are also encouraged to typeset the solutions in LaTeX or MS Word. Gradescope will ask you to mark where each answer is - please make sure to do this - points will be deducted otherwise.
- For each question where you are required to design an algorithm, you need to clearly explain the algorithm and prove the running time complexity as asked in the question.
- You are encouraged to discuss with your peers/consult supplementary material if necessary, but the final written solutions must be your own.
  - For each solution that you discussed or looked at extra materials for hints, you are required to mention it in your solution. This will not incur any penalty, but if noticed otherwise by the graders it may bright into question your academic integrity and there will be more serious repercussions. Writing down solutions or answers from some source without citing the source or without understanding how what you have written is correct is cheating and will be treated as such.

Question 0: List all your collaborators and sources: ($-\infty$ points if left blank)

You must enter the names of your collaborators or other sources as a response to Question 0. Do NOT leave this blank; if you worked on the homework entirely on your own, please write “None” here. Even though collaborations in groups of up to 3 people are encouraged, you are required to write your own solution.

Homework 2

Name:

Question 1: (6+4=10 points)

Recall that a sorting algorithm is *stable* if objects with equal keys appear in the same order in the sorted output as in the unsorted input. For example, consider a list of Student Objects, with two fields, Student.name and Student.age. If sorting according to age, given a list of Student objects, a stable sort will order the list in increasing order of the student ages. If two students have the same age, they will appear in the same order as in the unsorted list.

1. State and prove an appropriate loop invariant for the Merge subroutine (as seen in class) needed to prove the stability of MergeSort.
2. Using the result of part 1, prove that MergeSort (as seen in class) is stable for any input size $n \in \mathbb{N}$ by induction on n.

Question 2: (1+2+4+3=10 points)

Let $A[1, 2, \ldots, n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called a *transposition* of A. Answer the following questions:

1. List all the transpositions of the array $(7, 5, 2, 6, 9)$
2. Which arrays with distinct elements from the set $\{1, 2, \ldots, n\}$ have the smallest and the largest number of transpositions and why? State the expressions exactly in terms of n.
3. Give an algorithm that determines the number of transpositions in an array consisting of n numbers in $\Theta(n \log n)$ worst-case time. (Hint: Modify MergeSort.)
4. Prove the correctness and run time bounds for your algorithm.

Question 3: (5+5=10 points)

Recall MergeSort, in which a list is sorted by first sorting the left and right halves, and then merging the two lists. We define the *3-MergeSort* algorithm, in which the input list is split into 3 equal length parts (or as equal as possible), each is sorted recursively, and then the three lists are merged to create a final sorted list.

1. Write a recurrence for T(n), the worst-case run time for 3-MergeSort on any input containing n elements.

2. Solve this recurrence for T(n). Prove your result formally using the substitution method (i.e., induction). For full credit, you'll need to show both upper and lower bound.

Question 4: (5+5+5=15 points)

Solve each of the following recurrences by using the recursion tree method. For each level $0, 1, \ldots, d-1, d$ in the recursion tree (where d represents the last level or *depth* of the recursion tree), include

(i) the size of the problem (begins with n at level 0),

(ii) non-recursive cost of one problem (obtained by substituting the size of the problem in the nonrecursive term of the recurrence),

(iii) number of problems (corresponds to the number of nodes at the level),

(iv) the total cost at that level (non-recursive cost of one problem $\times$ number of problems).

Then, determine d and write a summation representing the total cost across all levels and get an answer f(n) such that $T(n) = \Theta(f(n))$.

1. $T(n) = 2T(n-1) + 1$, $T(0) = 1$
2. $T(n) = T(n/3) + n$, $T(1) = 1$ (for simplicity's sake, you may assume that n is a power of 3)
3. $T(n) = 3T(n/2) + n$, $T(1) = 1$ (for simplicity's sake, you may assume that n is a power of 2)

Question 5: (3+7=10 points)

Consider the following recurrence:

$$T(1) = 0 \tag{1}$$
$$T(2) = 1 \tag{2}$$
$$T(n) = T(\lceil n/3 \rceil) + T(\lceil 2n/3 \rceil) + 1 \quad \text{if } n > 2 \tag{3}$$

You will use the substitution method (i.e., induction) to find a Θ bound for $T(n)$.

1. *Warmup.* For this part, ignore the base cases and also do not worry about the ceilings in (3).

   To get a better feel for the recurrence and have a guess for a solution, try plugging in $\sqrt{n}$, n, and n^2 into (3). Compare the left and right hand sides of (3) for each of these substitutions, and determine whether the left hand side or right hand side of (3) is greater for each.

2. Use your work in part (a) to help you determine a guess for $p > 0$ such that $T(n) = \Theta(n^p)$. Formally prove your result using the substitution method. For full credit, you'll need to show both upper and lower bound. (Hint: consider $cn^p + d$ when proving the bounds, with your value of p, and determine appropriate values of c and d.)

Homework 2

Name:

Question 6: (5 points)

Consider the following inductive step in an attempted proof by induction showing $n^2 = \mathcal{O}(n)$.

Suppose for the strong inductive hypothesis that $n^2 = \mathcal{O}(n)$ for all $n \leq k$. Then

$$\begin{aligned}(k+1)^2 &= k^2 + 2k + 1 \\ &= \mathcal{O}(k) + 2k + 1 \text{ by the inductive hypothesis} \\ &= \mathcal{O}(k)\end{aligned}$$

Explain precisely why this is incorrect.