University of Houston

Homework 1

# COSC 3320
# Algorithms and Data Structures

**Gopal Pandurangan**

NAME

Due: Sunday, February 21, 2021
11:59 PM

Read the University of Houston Academic Honesty policy.

**Academic Honesty Policy**

All submitted work should be your own. Copying or using other people's work (including from the Web) will result in $-$`MAX` points, where `MAX` is the maximum possible number of points for that assignment. Repeat offenses will result in a failing grade for the course and will be reported to the Chair. If you have any questions, please reach out to the professor and the TAs. The best way to ask is on Piazza.

By submitting this assignment, you affirm that you have followed the Academic Honesty Policy.

Your submission **must be typed**. We prefer you use LaTeX to type your solutions — LaTeX is the standard way to type works in mathematical sciences, like computer science, and is highly recommended; for more information on using LaTeX, please see this post on Piazza — but any method of typing your solutions (e.g., MS Word, Google Docs, Markdown) is acceptable. **Your submission must be in pdf format.** The assignment can be submitted **up to two days late for a penalty of 10% per day.** A submission more than **two days late** will receive a **zero**.

**Reading**

Chapters 4 and 5. In particular, several worked exercises with solutions are provided at the end of each chapter. Attempting to solve the worked exercises **before** seeing their solutions is a good learning technique.

The exercises below are from the book. The book is updated periodically, so be sure to use the latest version.

**Exercises**

4.1(a), 4.3(6), 4.4, 5.1, 5.9

**Justify your answers. Show appropriate work.**

This page intentionally left blank.

# 1 Class Questions

## 1.1 January 26

**Question 1**

Show that any degree $d$ polynomial $p(n) = a_0 n^d + a_1 n^{d-1} + \cdots + a_{n-1} n + a_n$, with $a_0 > 0$, is $\mathcal{O}(n^d)$.

**Solution**. TYPE SOLUTION HERE. □

**Question 2**

Show that, for any $a > 1$, $n^b = \mathcal{O}(a^n)$.

**Solution**. TYPE SOLUTION HERE. □

**Question 3**

Show that $a^{\log_b n} = n^{\log_b a}$.

**Solution**. TYPE SOLUTION HERE. □

## 1.2 January 28

**Question 1**

Prove by mathematical induction that the gcd algorithm given in class is correct. You may assume $\gcd(a, b) = \gcd(b, a \bmod b)$.

**Solution**. TYPE SOLUTION HERE. □

## 1.3 February 4

**Question 1**

Solve $T(n) = 3T(n/2) + n^2$.

**Solution**. TYPE SOLUTION HERE. □

**Question 2**

Show the correctness of `MergeSort` using Induction. You may assume the `Merge` subroutine is correct.

**Solution**. TYPE SOLUTION HERE. □

**Question 3**

Argue that QuickSort cannot take more than $\binom{n}{2} = n(n-1)/2$ comparisons.

**Solution**. TYPE SOLUTION HERE. □

## 2 Textbook Exercises

**Exercise 4.1**

Prove the asymptotic bound for the following recurrences by using induction. Assume that base cases of all the recurrences are constants i.e., $T(n) = \Theta(1)$, for $n < c$ where $c$ is some constant.

(a) $T(n) \leq 2T(n/2) + n^2$. Then, $T(n) = \mathcal{O}\big(n^2 \log n\big)$.

**Solution**. TYPE SOLUTION HERE. □

**Exercise 4.3(6)**

Solve the following recurrences. Give the answer in terms of *Big-Theta* notation. Solve up to constant factors, i.e., your answer must give the correct function for $T(n)$, up to constant factors. You can assume constant base cases, i.e., $T(1) = T(0) = c$, where $c$ is a positive constant. You can ignore floors and ceilings. You can use the DC Recurrence Theorem if it applies.

6. $T(n) = 4T(n/2) + n^3$

**Solution**. TYPE SOLUTION HERE. □

**Exercise 4.4**

You are given an array consisting of $n$ numbers. A popular element is an element that occurs (strictly) **more than** $n/2$ times in the array. Give an algorithm that finds the popular element in the array if it exists, otherwise it should output "NO". Your algorithm should take no more than $2n$ comparisons. (As usual, we only count the comparisons between array elements.) Give pseudocode, argue its correctness, and show that your algorithm indeed takes no more than $2n$ comparisons. (Hint: Use a decrease and conquer strategy, similar to the celebrity problem.)

**Solution**. TYPE SOLUTION HERE □

**Exercise 5.1**

Determine the total number of comparisons that each of the following algorithms takes on $S = [8, 2, 6, 7, 5, 1, 4, 3]$.

- SimpleSort
- MergeSort
- QuickSort

Show the steps of the algorithm when calculating the number of comparisons.

**Solution**. TYPE SOLUTION HERE □

**Exercise 5.9**

Given three SORTED (in ascending order) arrays $A[1..n], B[1..n]$, and $C[1..n]$, each containing $n$ numbers, give an $\mathcal{O}(\log n)$-time algorithm (again, counting the number of comparisons) to find the $n^{\text{th}}$ smallest number of all $3n$ elements in arrays $A, B$, and $C$.

**Solution**. TYPE SOLUTION HERE □