CS 170 Homework 1

Due **09/07/2020, at 10:00 pm (grace period until 10:30pm)**

1 Three Part Solution

For each of the algorithm-design questions, please provide a three part solution which includes:

1. The main idea **or** pseudocode underlying your algorithm
2. A proof of correctness
3. A runtime analysis

Further explanation of this format is on the website (`https://cs170.org/resources/homework-guidelines/`).

2 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

3 Course Policies

(a) What dates and times are the exams for CS170 this semester?

Will we be offering alternate exams?

(b) Homework is due at 10:00pm, with a late deadline at 10:30pm. At what time do we recommend you have your homework finished?

(c) We provide 2 homework drops for cases of emergency or technical issues that may arise due to homework submission. If you miss the Gradescope late deadline (even by a few minutes) and need to submit the homework, what should you do?

(d) What is the primary source of communication for CS170 to reach students? We will email out all important deadlines through this medium, and you are responsible for checking your emails and reading each announcement fully.

(e) Please read all of the following:

(i) **Syllabus and Policies:** `https://cs170.org/syllabus/`
(ii) **Homework Guidelines:** `https://cs170.org/resources/homework-guidelines/`
(iii) **Regrade Etiquette:** `https://cs170.org/resources/regrade-etiquette/`
(iv) **Piazza Etiquette:** `https://cs170.org/resources/piazza-etiquette/`

Once you have read them, copy and sign the following sentence on your homework submission.

"I have read and understood the course syllabus and policies."

4 Understanding Academic Dishonesty

Before you answer any of the following questions, make sure you have read over the syllabus and course policies (`https://cs170.org/syllabus/`) carefully. For each statement below, write *OK* if it is allowed by the course policies and *Not OK* otherwise.

(a) You ask a friend who took CS 170 previously for their homework solutions, some of which overlap with this semester's problem sets. You look at their solutions, then later write them down in your own words.

(b) You had 5 midterms on the same day and are behind on your homework. You decide to ask your classmate, who's already done the homework, for help. They tell you how to do the first three problems.

(c) You look up a homework problem online and find the exact solution. You then write it in your words and cite the source.

(d) You were looking up Dijkstra's on the internet, and run into a website with a problem very similar to one on your homework. You read it, including the solution, and then you close the website, write up your solution, and cite the website URL in your homework writeup.

5 Asymptotic Complexity Comparisons

(a) Order the following functions so that for all i, j, if f_i comes before f_j in the order then $f_i = O(f_j)$. Do not justify your answers.

- $f_1(n) = 3^n$
- $f_2(n) = n^{\frac{1}{3}}$
- $f_3(n) = 12$
- $f_4(n) = 2^{\log_2 n}$
- $f_5(n) = \sqrt{n}$
- $f_6(n) = 2^n$
- $f_7(n) = \log_2 n$
- $f_8(n) = 2^{\sqrt{n}}$
- $f_9(n) = n^3$

As an answer you may just write the functions as a list, e.g. $f_8, f_9, f_1, \ldots$

(b) In each of the following, indicate whether $f = O(g)$, $f = \Omega(g)$, or both (in which case $f = \Theta(g)$). ***Briefly*** justify each of your answers. Recall that in terms of asymptotic growth rate, logarithmic $<$ polynomial $<$ exponential.

	$f(n)$	$g(n)$
(i)	$\log_3 n$	$\log_4(n)$
(ii)	$n \log(n^4)$	$n^2 \log(n^3)$
(iii)	$\sqrt{n}$	$(\log n)^3$
(iv)	$n + \log n$	$n + (\log n)^2$

6 Computing Factorials

Consider the problem of computing $N! = 1 \times 2 \times \cdots \times N$.

(a) N is $\log N$ bits long (this is how many bits are needed to store a number the size of N). Find an $f(N)$ so that $N!$ is $\Theta(f(N))$ bits long. Simplify your answer as much as possible, and give an argument for why it is true.

Note: You may not use Stirling's formula.

(b) Give a simple (naive) algorithm to compute $N!$. Use $O(n^2)$ as the runtime for multiplying two n bit numbers.

For this problem, you don't need to write a proof of correctness (that is, just state your algorithm and analyze its runtime).

7 Decimal to Binary

Given the n-digit decimal representation of a number, converting it into binary in the obvious way takes $O(n^2)$ steps. Give a divide and conquer algorithm to do the conversion and show that it does not take much more time than Karatsuba's algorithm for integer multiplication.

Just state the main idea behind your algorithm and its runtime analysis; no proof of correctness is needed as long as your main idea is clear.

8 Maximum Subarray Sum

Given an array A of n integers, the maximum subarray sum is the largest sum of any contiguous subarray of A (including the empty subarray). In other words, the maximum subarray sum is:

$$\max_{i \le j} \sum_{k=i}^{j} A[k]$$

For example, the maximum subarray sum of $[-2, 1, -3, 4, -1, 2, 1, -5, 4]$ is 6, the sum of the contiguous subarray $[4, -1, 2, 1]$.

Design an $O(n \log n)$-time algorithm that finds the maximum subarray sum.

Give a 3-part solution.

Hint: Split the array into two equally-sized pieces. What are the possibilities for the subarray, and how does this apply if we want to use divide and conquer?

9 Monotone matrices

A m-by-n matrix A is *monotone* if $n \ge m$, each row of A has no duplicate entries, and it has the following property: if the minimum of row i is located at column j_i, then $j_1 < j_2 < j_3 \ldots j_m$. For example, the following matrix is monotone (the minimum of each row is bolded):

$$\begin{bmatrix} \mathbf{1} & 3 & 4 & 6 & 5 & 2 \\ 7 & 3 & \mathbf{2} & 5 & 6 & 4 \\ 7 & 9 & 6 & 3 & 10 & \mathbf{0} \end{bmatrix}$$

Give an efficient (i.e., better than $O(mn)$-time) algorithm that finds the minimum in each row of an m-by-n monotone matrix A.

Give a 3-part solution. You do not need to write a formal recurrence relation in your runtime analysis; an informal summary of the runtime analysis such as "proof by picture" is fine.