CSC 545 **Homework 1 (version 1)** **Spring 2022**

This homework is due Thursday, February 10, at 3:30pm. Please upload a single `PDF` file containing your submission (ensuring scans of handwritten work are legible) on `Gradescope` by that time.

The questions are drawn from Chapters 2, 3, and 4 of the text and from the material given in class on *asymptotics*, *recurrences*, and *divide-and-conquer*.

The homework is worth a total of 100 points. In problems with several parts where point breakdowns are not given, each part has equal weight.

For questions that ask you to design an algorithm, you should:

(i) describe an algorithm for the problem using *prose* and *pictures*,
(ii) argue that your algorithm is correct, and
(iii) analyze the running time of your algorithm.

Pseudocode is *not* required, but do clearly explain the *ideas* behind your algorithm. As a general rule, only give high-level pseudocode if it aids the time analysis of your algorithm, or if it is needed to clarify the steps.

Please remember to (a) start each problem on a *new page*, and (b) put your answers in the *correct order*. If you can't solve a problem, state this, and write only what you know to be correct. Conciseness counts!

(1) **(Ordering functions asymptotically)** (20 points) Order the following functions according to their rate of growth. Specifically, group the functions into equivalence classes such that functions f and g are in the same class iff $f = \Theta(g)$, and then order the equivalence classes from slowest to fastest growing.

For each successive pair of functions (f, g) in your order, prove the relationship between f and g (either $f = \Theta(g)$ or $f = o(g)$) using the list of asymptotic properties given in class or possibly by taking a limit. Each proof should be no more than one to three lines, and should give the name of the property used in each step.

Organize your write-up so that you (1) first give the ordered list of functions, with a footnote labeling each successive pair in the list, and (2) then separately give the short proof for each pair in a list of footnotes.

$$\begin{array}{cccc} (\frac{3}{2})^n & n^2 & n! & \lfloor \ln n \rfloor ! \\ n & n \ln n & \ln(n!) & 2^{2^n} \\ n2^n & n^{\ln \ln n} & \ln n & 2^{2^{n+1}} \\ 2^n & (\ln n)^{\ln n} & (n+1)! & \ln \ln n \end{array}$$

(Hint: First form a rough initial order of the functions, and then use insertion sort on this list to obtain the final order.)

(2) **(bonus) (Superpolynomial growth)** (10 points) Prove or disprove the following.

> **Conjecture** *Let $f(n)$ be an asymptotically positive function. Suppose $f(n) = \omega(n^c)$ for every c. Then $f(n) = \Omega(b^n)$ for some $b > 1$.*

(Note: The above conjecture says that if f is growing faster than a polynomial, then it is growing exponentially fast.)

(3) **(Solving recurrences)** (20 points) Derive a Θ-bound on the solution to each of the following recurrences. Use the Master Theorem (including its specialization and extension given in class). Do not worry about taking floors or ceilings of arguments.

(a) $T(n) = 4T(n/2) + n^2\sqrt{n}$.
(b) $T(n) = 3T(n/2) + n \lg n$.
(c) $T(n) = 2T(n/2) + n$.
(d) $T(n) = 2T(n/2) + n \lg n$.

(4) **(Minimum positive-sum subarray)** (30 points) Given a one-dimensional array $A[1:n]$ of positive and negative real numbers, a *minimum positive-sum subarray* is a subarray $A[i:j]$ with $1 \le i \le j \le n$ such that the sum of its elements $\sum_{i \le k \le j} A[k]$ is: (1) strictly greater than zero, and is (2) minimum. In other words, it is a subarray of smallest positive sum.

(a) (30 points) Using the *divide-and-conquer* strategy, design an algorithm that finds a minimum positive-sum subarray in an input array of length n in $\Theta(n \log^2 n)$ worst-case time.

(Hint: Remember that n real numbers can be sorted in $\Theta(n \log n)$ worst-case time. To analyze the time for your algorithm, you may need to use the extended form of the Master Theorem for divide-and-conquer recurrences that was given in class.)

(b) **(bonus)** (10 points) Using an *incremental strategy*, now design an algorithm that finds a minimum positive-sum subarray in $O(n \log n)$ time. The incremental strategy proceeds by solving a subproblem of size k, adding one element, and updating the solution to solve a problem of size $k+1$.

(Hint: You may find a balanced search tree useful.)

(5) **(Identifying good chips by pairwise tests)** (30 points) Suppose we have n computer chips, some of which are good and some of which are faulty. We want to identify which chips are good by performing pairwise tests. In a pairwise test, we choose a pair of chips, and have each chip test the other. (So in a pairwise test of chips A and B, chip A tests B and reports the result, and chip B also tests A and reports the result.) Good chips always report correctly whether the tested chip is good or faulty, but faulty chips can report anything.

(a) (5 points) Show that if at most $n/2$ chips are good, then no algorithm can correctly identify the good chips by pairwise tests.

(Hint: Show how to construct a set of test results that will fool any procedure that claims to solve the problem.)

(b) (20 points) Suppose more than $n/2$ chips are good. Show how to reduce the problem of finding 1 good chip from among the n chips to a problem of half the size, using $\Theta(n)$ pairwise tests.

(c) (5 points) Using Part (b), design an algorithm that correctly identifies all the good chips using $\Theta(n)$ pairwise tests, assuming that more than $n/2$ chips are good.

Note that Problems (2) and (4)(b) are *bonus* questions. They are *not* required, and their points are not added to regular points.