Fall 17: CSci 5421—Advanced Algorithms and Data Structures

Out 9/6 Homework 1 Due 9/20

Instructions:

• Please be sure to read and understand the sections on assignment submission and academic integrity in the syllabus (under “Class Policies”). They apply to this and all future coursework.

• Please do all problems. However, due to TA resource limitations, we will grade only a subset of the assigned problems (same subset for everyone).

• Any exercise/problem number specified refers to the 3rd edition of the text.

• Remember that we will be looking at a very large number of assignments, so it is important that you do your best to facilitate the grading process (and keep us in good humor :-)). Towards this end, please keep the following in mind:

  Write legibly or, better still, type your answers (using LATEX if possible). Staple your answer sheets and include your name and student ID#. Communicate your ideas clearly and precisely, especially where proofs and/or algorithms are involved. Don’t leave it to us to guess what you mean; we can give points only for what we see on the answer sheet, not for what you may have intended for us to see. At the same time, don’t be verbose; keep your answers concise and to-the-point. The points you earn will be determined by the content and clarity of your answers, not their length (see below).

• As a guideline, answers will be graded according to the following general criteria:

  – Answer demonstrates complete understanding of the problem. It has no errors and is crystal clear. It belongs in the textbook. Answers in this category will receive 100%.

  – Answer demonstrates substantial understanding of the problem. It has only one or two minor errors and is explained clearly. Answers in this category will receive between 66% and 100%.

  – Answer demonstrates only partial understanding of the problem. It has several minor errors and/or a major error. Answers in this category will receive between 33% and 66%.

  – Answer demonstrates little or no understanding of the problem. It has several major errors. Answers in this category will receive between 0% and 33%, depending on the severity of the errors.

Where algorithms are requested, you must generally give three things: (a) a brief description of the main ideas behind your algorithms, including data structures used, from which the correctness of your approach should be evident; (b) pseudocode (along the lines of the text); and (c) an analysis of the running time. (Bear in mind that these are general guidelines—some problems may have more specific requirements.)

To give you a better idea for what is meant, the class web page has a sample solution that you may use as a model.

• Above all, have fun! Algorithm design is similar to solving a recreational puzzle—and should be just as enjoyable.

Over ☺️
1. (9 points) Let \( y \) and \( z \) be \( n \)-bit integers, where \( n \) is a power of 3. Consider the following divide-and-conquer algorithm to compute the product \( yz \).

Break \( y \) into three \( \frac{2n}{3} \)-bit pieces, \( a \), \( b \), and \( c \); thus \( y = a2^{2n/3} + b2^{n/3} + c \), where the powers of 2 denote appropriate bit-shifting. Similarly, break \( z \) into pieces \( d \), \( e \), and \( f \). Now compute \( yz \) recursively as:

\[
yz = ad2^{4n/3} + (ae + bd)2^{n} + (af + be + cd)2^{2n/3} + (bf + ce)2^{n/3} + cf.
\]

You may ignore the issue of “carries” throughout.

(a) What is the running time of this algorithm as a function of \( n \)? Justify your answer by writing down and analyzing the recurrence.

With a view towards improving the running time in part (a), consider the following approach, where we first compute certain intermediate products \( (r_1, \ldots, r_6) \) and use these along with additions and bit-shifts to compute \( yz \).

\[
r_1 = ?, \quad r_2 = (a + b)(d + e), \quad r_3 = be, \quad r_4 = ?, \quad r_5 = cf, \quad r_6 = ?, \quad \text{and} \quad yz = ?.
\]

(b) Fill in the missing information above for \( r_1, r_4, \) and \( r_6 \) and show how to compute \( yz \).

(c) What is the running time of this algorithm and how does it compare with the one we designed in class? Justify your answer by writing down and analyzing the recurrence.

2. (12 points) In each case below, use the Master Theorem (MT) to provide tight asymptotic bounds for the indicated recurrence. Show your work. (For each case, assume that \( T(1) = \Theta(1) \).)

(a) \( T(n) = 2T(n/4) + n \log n \), if \( n > 1 \).

(b) \( T(n) = 2T(n^{1/4}) + 1 \), if \( n > 1 \). (Consider doing a change of variables, as in Sec. 4.3.)

(c) \( T(n) = aT(n/b) + dn \), if \( n > 1 \), where \( a \geq 1 \) and \( b > 1 \) are integer constants and \( d > 0 \) is a real constant. Recall that this is the recurrence solved by the Little Master Theorem (LMT) seen in class. There we derived the solution by iterating the recurrence from first principles; here you should derive it directly from the MT. Specifically, show that

\[
T(n) = \begin{cases} 
\Theta(n^{\log_b a}) & \text{if } a > b \\
\Theta(n \log_2 n) & \text{if } a = b \\
\Theta(n) & \text{if } a < b 
\end{cases}
\]

3. (10 points) Let \( X \) be a \( kn \times n \) matrix and \( Y \) be an \( n \times kn \) matrix, for some integer \( k \).

(a) Describe an algorithm which computes the product \( XY \) using Strassen’s algorithm as a subroutine, i.e., using it as a black-box without modifying it. A careful answer, in words, suffices; pseudocode is not required. Justify your answer briefly, i.e., argue that your algorithm does compute \( XY \). Establish its running time.

(b) Repeat the problem for computing the product \( YX \).
4. (12 points) Let $A$ and $B$ be sets of points in the plane, where the points of $A$ are in the first quadrant (i.e., all coordinates are positive) and those of $B$ are in the third quadrant (i.e., all coordinates are negative). Assume that $A$ and $B$ each have at least 5 points. Let $A'$ consist of the 5 points of $A$ that are closest to $O$ under the $L_\infty$ distance metric (defined below), with ties broken arbitrarily. Define $B'$ similarly with respect to $B$. Let $(p, q)$ be a closest pair in $A \cup B$ under the Euclidean distance metric, where $p \in A$ and $q \in B$. Prove that $p \in A'$ and $q \in B'$; in other words, $(p, q)$ is one of the 25 pairs in $A' \times B'$.

Suppose that $a = (a_x, a_y)$ and $b = (b_x, b_y)$ are points in the plane. The $L_\infty$ distance between $a$ and $b$ is defined as $d_\infty(a, b) = \max\{|a_x - b_x|, |a_y - b_y|\}$. If (say) $b$ is the origin $O = (0, 0)$, then $d_\infty(a, O) = \max\{|a_x|, |a_y|\}$. Note that $d(a, b) \geq d_\infty(a, b)$, where $d(a, b)$ is the Euclidean distance between $a$ and $b$.

5. (13 points) Let $S = \{x_1, x_2, \ldots, x_n\}$ be a set of $n$ distinct reals. Associated with each $x_i$ is a positive weight $w_i$ such that $\sum_{i=1}^{n} w_i = 1$. The weighted median of $S$ is the element $x_k$ such that $\sum_{i<x_k} w_i \leq 1/2$ and $\sum_{i>x_k} w_i \leq 1/2$. Argue briefly that such an $x_k$ always exists and give an $O(n)$-time divide-and-conquer algorithm to compute it, using the linear-time algorithm for the unweighted median as a subroutine, i.e., using it as a black-box without modifying it. (Note that $S$ is not given in sorted order nor can your algorithm afford to sort it.)

Your answer should include (a) a clear description of the main ideas, from which the correctness of your algorithm should be self-evident, (b) pseudocode, and (c) an analysis of the running time.

6. (16 points) Problem 4-6, p. 110, on Monge matrices. For part (d), give a clear explanation in words; pseudocode is not needed.