This homework has a total of 4 problems on 2 pages. Solutions should be submitted to T-square before 12:05pm on Friday Sep 2.

1. (2 points total over 2 parts) In each of the following situations, indicate whether \( f = O(g) \), \( f = \Omega(g) \), or \( f = \Theta(g) \). Justify your answer.

\[
\begin{array}{c|c|c}
  f(n) & g(n) \\
  \hline
  (a) n^2 & n \log n \\
  (b) n^2 & 2^n \\
\end{array}
\]

2. (3 points total over 3 parts) You are trying to choose between the following three algorithms:

(a) Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

(b) Algorithm B solves problems of size \( n \) by recursively solving two subproblems of size \( n - 1 \) and then combining the solutions in constant time.

(c) Algorithm C solves problems of size \( n \) by dividing them into nine subproblems of size \( n/3 \), recursively solving each subproblem, and then combining the solutions in \( O(n^2) \) time.

What are the asymptotic running times (in big-O notation) of each of these algorithms? Which one would you choose?

3. (5 points total over 2 parts) The \( i^{th} \) order statistic of an array is defined as the \( i \) smallest element in the array. One way to find the \( i^{th} \) order statistic is to sort the array in increasing order and return the element in the \( i^{th} \) position. However, this takes \( \Theta(n \log n) \) time. In this problem we will develop a faster algorithm that finds the \( i^{th} \) order statistic in \( O(n) \) time.

Suppose you have access to an algorithm that finds the median of a list in \( O(n) \) time. Note that the median is the \( \lfloor \frac{n}{2} \rfloor^{th} \) order statistic.

(a) (2 points) Consider an \( i^{th} \)-order statistic on an array of size \( n \). Show that given the median of this array, this statistic can be answered using the \( i'^{th} \)-order statistic on an array of size \( n/2 \) for some appropriately chosen \( i' \).
(b) (3 points) Using this median algorithm as a subroutine, design and analyze an \( O(n) \) time algorithm that returns the \( i \)th order statistic of an array \( A \) of \( n \) integers. Explain why your algorithm runs in \( O(n) \) time and why it is correct.

4. (5 points total over 4 parts) The goal of this problem is to devise a fast algorithm that reports the number of mismatches of all ways of fitting a pattern string into a text. The text and pattern are both binary strings, with text as \( T[1 \ldots n] \) and pattern as \( P[1 \ldots m] \). The goal is to compute for each shift value \( i \), the number of matches if we place the pattern into text with shift \( i \). That is, the number of positions \( j \in [1 \ldots m] \) such that

\[
P[j] = T[i + j - 1].
\]

For instance, suppose \( T = 101011 \) and \( P = 101 \), the shifts \( i = 1 \) and \( i = 3 \) both yield 3 matches, while the shift \( i = 4 \) only gives 1 match.

All possible combinations can be tried in \( O(nm) \) time. Below we will see that a faster algorithm is possible by reducing it to a polynomial multiplication problem, which will be discussed in lectures on Aug 29 and 31. The parts below breaks down this reduction.

(a) (1 point) A match can happen with when \( P[j] = 1 \) and \( T[i + j - 1] = 1 \). What does the quantity

\[
M[i] = \sum_{j=1}^{m} P[j]T[i + j - 1]
\]

return? The case of \( M[4] \) in the example above may be helpful.

(b) (1 point) The other type of match can be found by using a similar approach, but after transforming \( T \) and \( P \). Explain how this can be done using an example.

(c) (2 points) Show that all the quantities \( M[i] \) (for \( 1 \leq i \leq n - m + 1 \)) can be extracted from product of two suitably chosen polynomials based on \( T \) and \( P \).

Note that for two polynomials in \( x \), \( P(x) \) and \( Q(x) \), with coefficients of \( x^i \) being \( p_i \) and \( q_i \) respectively, the coefficient of \( x^i \) in their product, \( PQ \) is

\[
\sum_{k=0}^{i} p_k q_{i-k}.
\]

(d) (1 point) Using the answers you have obtained so far, devise and analyze an algorithm that computes all the number of matches for all shifts in \( O(n^{1.9}) \) time. You may assume the existence of an \( O(n^{1.8}) \) time algorithm for multiplying two polynomials of degree \( n \).