CS515, Practice Midterm

Name: ______________________

- This exam is worth 40 points. It consists of five problems, each worth 10 points. The sum of the four highest scored problems defines the final grade.

- If you want to ask a question, write it on the provided sheet of paper and raise your hand. I will collect it, write down the answer and return it back to you.

- If you need more scrap paper, raise your hand. Use only the scrap paper provided.

- Please turn off your cell-phones and other electronic devices.
Problem 1

Rank the following functions by order of growth; that is, find an arrangement $g_1(n), g_2(n), \ldots, g_{10}(n)$ of functions satisfying $g_i(n) = \Omega(g_{i+1}(n))$ for every $i \in \{1, \ldots, 9\}$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. You do not have to prove your answers.

\[
\begin{array}{cccc}
  n \log_3 n & n^{5.3} + 2n^7 & n \log(n^2) & 3^n & 5\sqrt{n} \\
  \frac{1}{10000}n^7 & 1 & n^{-3} & 2^{7 \log n} & 1 + 4n^{-3}
\end{array}
\]

Recall that in this class we use $\log n$ to denote the logarithm base 2.
Problem 2
Consider the following divide-and-conquer algorithm that takes as input an array \( A = [a_1, a_2, \ldots, a_k] \), and two indices \( \ell \) and \( r \) such that \( 1 \leq \ell \leq r \leq n \):

\[
\text{WhatAmIDoing}(A, \ell, r)
\]

1. If \( \ell = r \) then return \( a_\ell \).
2. Else
3. \hspace{1em} Let \( m = \lfloor (\ell + r)/2 \rfloor \).
4. \hspace{1em} Let \( x = \text{WhatAmIDoing}(A, \ell, m) \).
5. \hspace{1em} Let \( y = \text{WhatAmIDoing}(A, m+1, r) \).
6. \hspace{1em} Return \( \min \{x, y\} \).

• Trace the algorithm for the following input: \( A = [2, 5, 7, 3, 4, 1, 6, 8] \), and \( \ell = 1, r = 8 \). More precisely, list every recursive call (state the values of its parameters \( \ell \) and \( r \)), and for each recursive call state its return value:

• What is this algorithm doing? Describe in words the value returned by the algorithm:

• Analyze the running time of the algorithm. More precisely, let \( T(n) \) be the running time of the algorithm, where \( n = r - \ell + 1 \). For simplicity, suppose that \( n \) is a power of two.

   (a) State the recurrence for \( T(n) \) and explain where each of the terms in the recurrence comes from:

   (b) Solve the recurrence for \( T(n) \), i.e., find a function \( g(n) \) such that \( T(n) = O(g(n)) \) (ideally, \( T(n) = \Theta(g(n)) \)).
Problem 3

Given is a sequence of numbers $a_1, a_2, \ldots, a_n$ and a number $b$. We want to know whether there are two numbers $a_i$ and $a_j$, $i \neq j$, such that $a_i + a_j = b$. Give an $O(n \log n)$ algorithm that determines whether such $i$ and $j$ exist. Reason the running time of your algorithm.
Problem 4

Recall the knapsack problem, where we are given \( n \) items, the \( i \)-th item is specified by its cost \( c_i \) and weight \( w_i \). We are also given a weight limit \( W \) and we want to find a subset of the items with total weight at most \( W \) and the largest possible total cost.

We have the following helper function:

\[
\text{ALGO-HELPER}((c_1, w_1), (c_2, w_2), \ldots, (c_n, w_n), W)
\]

1. Let \( w_{\text{sum}} = 0 \).
2. Let \( \ell \) be an empty list.
3. For \( i = 1 \) to \( n \) do
4. \quad If \( w_{\text{sum}} + w_i \leq W \) then
5. \quad \quad Add the \( i \)-th item to \( \ell \).
6. \quad Let \( w_{\text{sum}} = w_{\text{sum}} + w_i \).
7. Return \( \ell \).

Now consider the four greedy algorithms below and for each algorithm say whether it works or not. If you say that an algorithm works, reason its correctness. If you say that an algorithm does not work, provide an input on which the algorithm fails.

- **ALGO1((c_1, w_1), (c_2, w_2), \ldots, (c_n, w_n), W)**

  1. Sort the items decreasingly by their cost to weight ratios, i.e., \( c_i/w_i > c_j/w_j \) for \( i < j \).
  2. Take items in the list computed by ALGO-HELPER((c_1, w_1), (c_2, w_2), \ldots, (c_n, w_n), W).

Does the algorithm work? If yes, reason its correctness. If no, provide a counterexample.

- **ALGO2((c_1, w_1), (c_2, w_2), \ldots, (c_n, w_n), W)**

  1. Sort the items increasingly by their weights, i.e., \( w_i < w_j \) for \( i < j \).
  2. Take items in the list computed by ALGO-HELPER((c_1, w_1), (c_2, w_2), \ldots, (c_n, w_n), W).

Does the algorithm work? If yes, reason its correctness. If no, provide a counterexample.
• ALGO3((c_1, w_1), (c_2, w_2), \ldots, (c_n, w_n), W)

1. Sort the items decreasingly by their costs, i.e., c_i > c_j for i < j.
2. Take items in the list computed by ALGO-HELPER((c_1, w_1), (c_2, w_2), \ldots, (c_n, w_n), W).

Does the algorithm work? If yes, reason its correctness. If no, provide a counterexample.

• ALGO4((c_1, w_1), (c_2, w_2), \ldots, (c_n, w_n), W)

1. Sort the items decreasingly by their cost to weight ratios, i.e., c_i/w_i > c_j/w_j for i < j.
2. Let c_{max} = 0 and let \ell_{max} be an empty list.
3. For i = 1 to n do
   4. Run ALGO-HELPER((c_i, w_i), (c_{i+1}, w_{i+1}), (c_{i+2}, w_{i+2}), \ldots, (c_n, w_n), W) and let \ell be the list returned by this function.
   5. Let c_{sum} be the sum of the costs of the items in the list \ell.
   6. If c_{sum} > c_{max} then
      7. Let c_{max} = c_{sum} and \ell_{max} = \ell.
   8. Take items specified by \ell_{max}.

Does the algorithm work? If yes, reason its correctness. If no, provide a counterexample.

• For each of the four algorithms, estimate its running time.
Problem 5

Consider the following “yellow-green” weighted interval scheduling variant. Given are \( n \) intervals, the \( i \)-th interval is specified by \((s_i, f_i, c_i, d_i)\), where the interval runs from the starting time \( s_i \) to the finishing time \( f_i \), its cost is \( c_i > 0 \), and its color is \( d_i \in \{\text{yellow, green}\} \), for \( i \in \{1, 2, \ldots, n\} \). We call a non-overlapping subset of intervals a yellow-green interval sequence, if the intervals sorted by their finishing times alternate between yellow and green (they might start and end with either a yellow or a green interval; thus, patterns “yellow-green-yellow-green-...” or “green-yellow-green-yellow-...” are allowed). We want to find the maximum possible cost of a yellow-green interval sequence. (Check next page to see an example input.)

Suppose that the input intervals are sorted by their finishing times and we are interested in computing the maximum cost of a yellow-green interval sequence (we do not need to find the corresponding intervals).

Let the first part of the “heart of the solution” (the verbal description of \( S[j] \)) be defined as follows:

\[
S[j] = \text{the maximum cost of a yellow-green interval sequence ending with interval } j
\]

Your task is to design the second and the third part of the heart, i.e., the mathematical expression to compute \( S[j] \) from already pre-computed values in \( S \) (write it inside the box in the pseudocode below), and the return value of the algorithm (the last line of the pseudocode).

- Fill in the blanks in the following pseudocode that finds the maximum cost of a yellow-green interval sequence. Make sure your algorithm runs in time \( O(n^2) \).

**YELLOW-GREEN**(Input: \((s_1, f_1, c_1, d_1), (s_2, f_2, c_2, d_2), \ldots, (s_n, f_n, c_n, d_n)\))

For \( j = 1 \) to \( n \) do

\[
\text{\text{\textbf{}}}
\]

Return ________.

- Give the contents of the \( S \)-array (as defined by the heart of the solution) for input \((1, 3, 5, \text{yellow}), (2, 5, 8, \text{green}), (4, 7, 9, \text{yellow}), (6, 8, 2, \text{green}), (9, 11, 3, \text{yellow}), \) and \((10, 12, 1, \text{yellow})\).

\[
S = \boxed{\text{\text{\textbf{}}}}
\]

For example, if the intervals are \((1, 3, 5, \text{yellow}), (2, 5, 8, \text{green}), (4, 7, 9, \text{yellow}), (6, 8, 2, \text{green}), (9, 11, 3, \text{yellow}), \) and \((10, 12, 1, \text{yellow})\), then the yellow-green interval sequence with the largest cost contains the second and the fifth interval for a total cost of
\[ 8 + 3 = 11. \] Notice that there are many other yellow-green interval sequences, for example, the first, the fourth, and the fifth intervals, with a total cost of \( 5 + 2 + 3 = 10 \), or just the sequence containing a single interval, the third interval, with cost 9. Also notice that the subset of intervals containing the first, the third, and the fifth interval has total cost \( 5 + 9 + 3 = 17 \) but this subset is not yellow-green and therefore we are not considering it. Another subset that does not work (because the intervals do not alternate between yellow and green) is the set containing the second, the fourth and the fifth interval for a total cost \( 8 + 2 + 3 = 13 \). Therefore, the maximum possible cost of a yellow-green interval sequence for the above input is 11.