CSE 531: Algorithm Analysis and Design Fall 2024

Homework 3

Instructor: Kelin Luo **Deadline: Nov/11/2024**

Your Name: _____ Your Student ID: _____

Problems	1	2	3	4	5	Total
Max. Score	5	10	15	15	10	55
Your Score						

Requirement:

- Save and submit your HW 3 submission to Brightspace as a single typed PDF file. Name your file: Your Last Name Your First Name YourStudents ID Number Assignment Number. Example: **Doe_John_55552222_HW3**
- You should view your submission after you upload it to make sure that it is not corrupted or malformed. Submissions that are rotated, upside down, or that do not load will not receive credit. Illegible submissions may also lose credit depending on what can be read. You are responsible for making sure your submission went through successfully.
- To create a figure, you have the option to use software such as "Ipe Drawing Editor" or other similar tools for digital illustrations. Alternatively, you may choose to draw the figure by hand and then insert it into your document.
- The HW 3 deadline is 11 Nov, 11:59PM EST. Late submissions (within 24 hours with 25 % penalty or 48 hours with a 50% penalty) should be submitted via Email to Instructor and Head TAs. Submissions will close 48 hours after the deadline.
- Only the most recent submission is kept on Brightspace.
- Note that your total points for HW3 will not exceed 55. Suppose you receive $P1$, $P2$, $P3$, $P4$, and $P5$ points for each of the questions. Your HW3 grade, as counted towards your final grade, will be calculated as $5*\min\{P1+P2+P3+P4+P5, 55\}/55$.

Problem 1 (5 points). For each of the following recurrences, use the master theorem to give the tight asymptotic upper bound (Please specify the variables a, b, c, along with the specific condition, and then provide the running time tight asymptotic upper bound.).

(a) **(1 points)** $T(n) = T(n/3) + O(1)$.

(b) **(1 points)** $T(n) = 2T(n/3) + O(n^3)$.

(c) **(1 points)** $T(n) = 4T(n/2) + O(n^{1.5})$.

(d) **(1 points)** $T(n) = 4T(n/2) + O(n^2)$.

(e) **(1 points)** $T(n) = 4T(n/4) + O(n)$.

Problem 2 (10 points). We consider the following problem of counting **ordered pairs**. Given an array A of n positive integers, a pair $i, j \in \{1, 2, 3, ..., n\}$ of indices is called a ordered pair if $i < j$ and $A[i] < A[j]$. The goal of the problem is to count the number of ordered pairs for a given array A. Please design an $O(n \log n)$-time divide-and-conquer algorithm to solve the problem and briefly argue the running time with recurrence relation.

Problem 3 (15 points). Background: Suppose we are working on an image processing project, with a specific focus on text detection within images. Our central goal is to precisely locate and extract text regions from these images, a capability that holds significant value for various applications, such as document analysis. We depict the pixel intensity distribution in the image using a variable-width histogram. Our objective is to identify either horizontal or vertical text lines within the image. By identifying the largest rectangle in this histogram, we can effectively detect and delineate the text regions present in the image.

A formal description of the largest rectangle in a histogram problem: The input is a two-dimensional array A of size n, where each entry represents a rectangle in a variable width histogram with height l_i and width w_i. Our problem is to find the largest rectangle that can be formed in this histogram, such that the total perimeter of this rectangle is maximized. For example, if the input is

$$A = ((4, 2), (6, 1), (13, 4), (17, 2), (16, 5), (20, 2), (15, 3), (14, 4), (7, 1), (11, 2)),$$

then the largest rectangle has perimeter $66 = 2 \times (13 + 20)$, with height 13 and width 20, covering rectangles 3 to rectangles 8 (See Figure 1). Please design an $O(n \log n)$-time divide-and-conquer algorithm to solve the problem, briefly argue the running time with recurrence relation, and briefly argue the correctness of your algorithm.

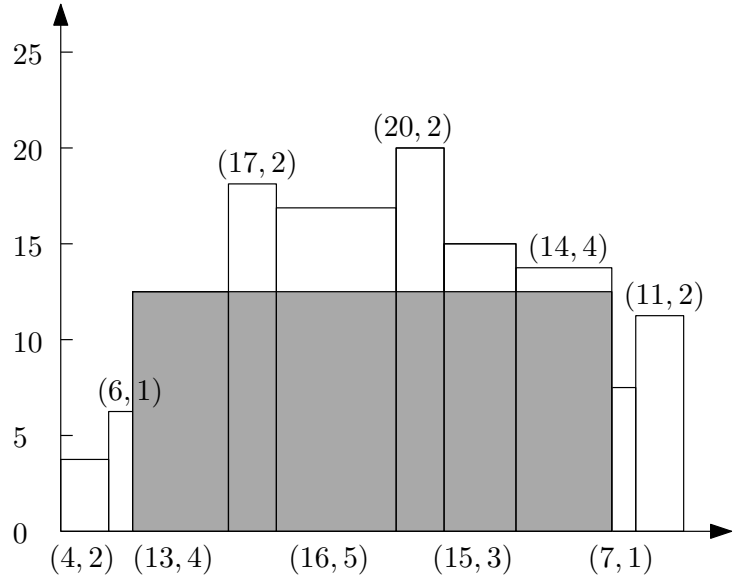

Figure 1: An example of Problem 3. The largest rectangle of in the histogram is given by the shaded area.

Problem 4 (15 points). Background: Knapsack problems are encountered in real-world decision-making processes across a wide range of fields, like selecting investments and portfolios. In the following section, we will explore generalized knapsack problem with an unlimited supply for some items.

Given n items indexed by $[n] = \{1, 2, 3, ..., n\}$. Each item $i \in [n]$ has an integer value $v_i \geq 1$ and an integer price $p_i \geq 1$. Each item $i \in \{1, 2, ..., \lceil n/2 \rceil\}$ has unlimited number of copies, and each item $i \in \{\lceil n/2 \rceil + 1, \lceil n/2 \rceil + 2, ..., n\}$ has exactly one copy. You have a budget of P. The goal of the problem is to buy a set of items with the budget P so as to maximize the value. Formally you need to find a vector $(c_1, c_2, ..., c_n)$, where each c_i is a non-negative integer if item $i \leq \lceil n/2 \rceil$, each $c_i = 0$ or 1 if item $i \geq \lceil n/2 \rceil + 1$, and $\sum_{i=1}^{n} c_i \cdot p_i \leq P$, so as to maximize $\sum_{i=1}^{n} c_i \cdot v_i$.

Describe a Dynamic Programming algorithm (Define sub-problems, establish base case and recurrence relations between sub-problems, write the dynamic programming algorithm pseudo-code and the recover optimal solution algorithm pseudo-code.) for solving this problem in $O(nP)$-time. Note that in the algorithm pseudo code, you need to output the maximum value and the c_i's that achieve the value.

Problem 5 (10 points). Given strings s_1, s_2, and s_3, find whether s_3 is formed by an **interleaving** of s_1 and s_2. An interleaving of two strings a and b is a configuration where a and b are divided into n and m substrings respectively, such that: $a = a_1 + a_2 + ... + a_n$, $b = b_1 + b_2 + ... + b_m$ and $|n - m| \leq 1$. The interleaving is $a_1 + b_1 + a_2 + b_2 + a_3 + b_3 + ...$ or $b_1 + a_1 + b_2 + a_2 + b_3 + a_3 + ...$ Note: $a_i + b_j$ is the concatenation of strings a_i and b_j.

Describe a Dynamic Programming algorithm (Define sub-problems, establish base case and recurrence relations between sub-problems, write the dynamic programming algorithm pseudo-code.) for solving this problem in $O(|s_1| \cdot |s_2|)$-time. For convenience, you only need to output the solution "true" or "false".

For example, Input: s_1 = "aabcc", s_2= "dbbca", s_3 = "aadbbcbcac" Output: "true". Explanation: One way to obtain s_3 is: Split s_1 into s_1 = "aa" + "bc" + "c", and s_2 into s_2 = "dbbc" + "a". Interleaving the two splits, we get "aa" + "dbbc" + "bc" + "a" + "c" = "aadbbcbcac". Since s_3 can be obtained by interleaving s_1 and s_2, we return "true".