DSA ASSIGNMENT 1

Asymptotic Notations, Algorithm Design,
Sorting/Searching, Recursion
Deadline: 11:59 PM, 7/2/2025

Total Marks : 60 Marks

Instructions:

1. Assignments are to be attempted individually.
2. Submit the assignment as a single zipped folder (**A1_⟨RollNumber⟩.zip**) containing a pdf file for all the theory questions (**must contain the explanation of programming questions and screenshots of relevant test cases**) and "C" files (code_⟨QuesNum⟩_⟨RollNumber⟩.c) for programming questions, for example "code_3_1_⟨RollNumber⟩.c" for programming question 3.1.
3. Please read the instructions given in the questions carefully. In case of any ambiguity, post your queries on Google Classroom at least a week before the deadline. **No TA will be responsible for responding to the queries after this.**
4. All the TAs will strictly follow the rubric provided to them. **No requests will be entertained related to the scoring strategy.**
5. **The use of generative tools (such as ChatGPT, Gemini, etc.) is strictly prohibited.** Failure to comply may result in severe consequences related to plagiarism.
6. **Extension and Penalty clause:**
   - Even a 1-minute late submission on Google Classroom will be considered late. Please turn in your submissions at least 5 minutes before the deadline to avoid any hassle.
   - Not explaining the answers properly will lead to zero marks.

Theory [30 Marks]

1 Algorithm Complexity [10 marks]

You are designing three different algorithms for a distributed computing system. Each algorithm is designed to solve a problem in a different way, and their performance can be described by recurrences. For each algorithm, you are tasked with analyzing its time complexity using asymptotic notations.

1.1 Master Theorem

Consider an algorithm for processing large data sets, where the system recursively divides the data into smaller chunks. The recurrence for the runtime of the algorithm is given by $T(n) = 2T(n/8) + \sqrt[3]{n}$.

1. Using the Master Theorem, determine the tight asymptotic upper bound for the time complexity of this algorithm. **[3 marks]**

2. Interpret the result in terms of how the system would scale as the input size n increases. **[2 marks]**

1.2 Recursion Tree Method

The second algorithm divides the workload into two specifically sized subproblems: one with a size of n/3 and the other with n/4. Additionally, the system's task manager incurs an overhead cost of 5n for coordinating the subproblem.

1. Using the recursion tree method, derive the cost at the kth level of the recursive tree, and the time complexity of this algorithm. Explain your answer. **[3 marks]**

2. Use the substitution method to validate your findings. **[2 marks]**

2 Algorithm Design [20 marks]

2.1 Sort and Search

With its digitization system, the IIIT-D library is able to pull up records of books as and when required. However, given their vast collection, it sometimes takes quite some time to go through the long list of titles.

Perhaps it would be quicker to sort the titles by title length?

IIIT-D has enlisted you for this project!

1. Write a pseudocode to implement an insertion sort that sorts an array of strings (book titles) by their lengths in ascending order. Your code must include a method to calculate the length of each title. Also, derive the time complexity for this code in Big-O notation. **[3 + 1 marks]**

2. Write a pseudocode to implement a binary search to find a specific book that takes the sorted array of book titles and the book to be searched as inputs. Assume there are no duplicate titles in the library. Also, analyze the time complexity of the above code. **[3 + 1 marks]**

3. Assume you have an unsorted array with you (with no duplicate elements). Which process would take longer:

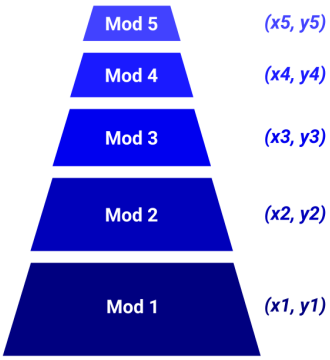
   (a) Sorting once and doing multiple binary searches, or

   (b) Doing multiple linear searches.

   Analyze and compare these two scenarios with an example.**[2 marks]**

2.2 Problem Solving

A space research team is designing a vertical colony structure on an alien planet. Each module in the colony has **specific width (x) and weight (y) constraints**. For stability, each module must be narrower and lighter than the module below it.



1. Given a list of modules, each defined by its width and weight (x, y), write a pseudocode to compute the tallest possible stable structure of modules. *(Hint: You will only need to use the concepts discussed in class. And it is not necessary for every module to be included in the most stable structure.)* [8 marks]

2. Shed light on the efficiency of your algorithm based on its time complexity. [2 marks]

Programming (in "C" only) [30 Marks]

3 Sorting/Searching [15 Marks]

3.1 The Quest for the Twin Swords of Xylon [15 Marks]

In the ancient kingdom of Xylon, there was a legend of two powerful twin swords that could only be wielded by a true heir to the throne. The swords were said to be hidden deep within the royal archives, guarded by a series of scrolls containing records of every heir to the throne who ever lived. The scrolls were stored in a large sorted archive in the Royal Library, where each scroll had the name of an heir and their unique royal identifier.

The kingdom had long lost track of which heir was the true rightful ruler, but there was a single clue that could help: The twins' swords were last wielded by two kings who shared the same royal identifier. The clue: **Find the range of the royal identifier** of the twin kings, buried deep in the archive.

One day, the young heir, Princess Elara, approached the wise Librarian with a quest. She needed to uncover the range of a royal identifier from the sorted list of all royal records. Princess Elara knew that the identifier she was looking for appeared multiple times in the archives, but she was unsure of its exact range—whether it was found in one record or many.

Princess Elara: "Librarian, I have a challenge for you. I seek to find the range of a specific royal identifier hidden among these ancient scrolls. I need to find both the first and last position of this identifier in the archive. Can you help me find it?"
The Librarian: "Ah, Princess, your task is a difficult one. The archives are sorted in non-decreasing order, but time is of the essence, and we must work swiftly. You must search for this identifier efficiently, and if it appears in multiple records, you must locate both the first and last appearances."

The Librarian handed Princess Elara the scrolls and explained the rules:

- Each record in the scrolls was sorted in **non-decreasing order** by royal identifier.
- The identifier she sought would appear at least once in the sorted list, but could also appear multiple times.

Example Scenarios from the Archives:

- Scrolls Example 1 The royal archives contain the following identifiers: Scrolls = [5, 7, 7, 8, 8, 10] The identifier Elara sought was 8. After using the Librarian's method, she discovered that 8 appeared first at index 3 and last at index 4. Hence, it is [3, 4].
- Scrolls Example 2 The royal archives contain the following identifiers: Scrolls = [1, 2, 3, 4, 5] The identifier Elara sought was 6. Alas, the Librarian's method revealed that 6 was not present in the archives. She returned the result [-1, -1].

- Scrolls Example 2 The royal archives contain the following identifiers: Scrolls = [2, 4, 4, 4, 6, 8, 9] The identifier Elara sought was 4. After using the Librarian's method, she discovered that 4 appeared first at index 1 and last at index 3. Hence, it is [1, 3].

You must find the range of the royal identifier from scrolls. The quest is fraught with complexities: you must balance time efficiency with accuracy in identifying both the first and last occurrences. [**10 Marks**]

Apart from the above algorithm implementation, answer the questions below.

1. What is the time (number of comparisons) and space (additional memory) complexity of your algorithm, and how does it compare to a brute-force approach to identifying? Also, write the equation of time complexity for representing the relation (in the form of T, n, O and constants) [**3 Marks**]

2. Can you prove that your efficient identifying is the most efficient way to find the range in this non-decreasing order (in this particular instance)? [**2 Marks**]

4 Recursion [15 marks]

4.1 The Guardian Selection Ritual [15 Marks]

In an ancient kingdom, every year, the elders hold a grand selection ritual to determine the kingdom's new guardian. The kingdom has a vast population, and each citizen is assigned a position in a circle. The ritual proceeds in a sequence of eliminations, with the goal of determining the final person who will be entrusted with the kingdom's most sacred duties.

The Process:

1. **Starting Sequence:** The citizens stand in a circle, numbered sequentially from 1 to n. The number n represents the total population.

2. **Elimination Pattern:**

   - At the beginning of the ritual, every second citizen is removed from the circle, starting from the first person in line.
   - In the next round, the remaining citizens are counted again, and the elimination continues from the opposite direction, starting from the last remaining individual in the circle.
   - The process continues, alternating between every ritual, until only one citizen remains.

3. **Objective:** Given the number of citizens, n, determine the position of the last remaining citizen who will become the guardian of the kingdom.

Input:

- A single integer n represents the number of citizens in the kingdom.

Output:

- A single integer representing the position of the last remaining citizen.

Example Scenarios:

- **Example 1** n = 7 Output = 4
- **Example 2** n = 1 Output = 1

Constraints:

- The elimination process involves removing every second individual alternating the order during each ritual.
- The problem must be solved efficiently, especially for large n (up to 1 million citizens).

Task

- Write a code that calculates the position of the last remaining citizen based on the number n. **[7 Marks]**
- Analyze the time (number of iterations and comparison operations) and space (additional memory) complexity of the solution. How does the algorithm behave as n grows larger (somewhere around 10^6), and what steps can be taken to ensure the program runs efficiently for large inputs? **[4 Marks]**
- Suppose instead of eliminating every second citizen, the rule is to eliminate every k^{th} citizen, where k is an input integer. How would it impact the code to calculate the position of the last remaining citizen. **[4 Marks]**