ESO207A: Data Structures and Algorithms

Theoretical Assignment 2

Due Date: 26th September 2025, 11:59 PM

Total Number of Pages: 3 Total Points 100

Instructions-

1. For submission typeset the solution to each problem and compile them in a single pdf file. Hand-written solutions will not be accepted. You can use LaTeX or Word for typesetting.
2. Start each problem from a new page. Write down your Name, Roll number and problem number clearly for each problem.
3. For each question, give the pseudo-code of the algorithm with a clear description of the algorithm. Unclear description will receive less marks. Less optimal solutions will receive only partial marks.
4. Assume that sorting would have $O(n \log(n))$ complexity.

Question 1. The Archive Sorting Ritual (30 points)

In the Great Archive of Chronos, priceless scrolls containing the kingdom's history are stored on long shelves. The librarians must keep these scrolls in increasing chronological order (from year 1 to year n), but after festivals, the scrolls are often returned in jumbled order.

The Archive's enchanted shelves perform a special sorting ritual. If a segment of scrolls is already in strictly increasing order, the ritual halts there — no energy is spent sorting it further. Otherwise, the segment is divided into two halves, each half is sorted recursively, and finally the two halves are merged together.

The ritual is described below:

Algorithm 1 ArchiveSort($A[l..r]$)

```
1: if A[l..r] is in strictly increasing order then
2:     return
3: end if
4: if l < r then
5:     mid ← ⌊(l + r)/2⌋
6:     ArchiveSort(A[l..mid])
7:     ArchiveSort(A[mid + 1..r])
8:     Merge(A[l..mid], A[mid + 1..r])
9: end if
```

Each invocation of `ArchiveSort` consumes one unit of magical energy. The Council of Mages has decreed that exactly k units of energy must be consumed as:

If fewer than k units are used, the ritual is incomplete, and the scrolls remain unstable.

If more than k units are used, the excess energy may backfire, damaging the shelves.

Thus, the Head Librarian must carefully arrange the scrolls *before* sorting so that the ritual consumes exactly k units.

Formally:

- You are given two integers n and k.
- Construct a permutation of $1, 2, \ldots, n$ such that the above ritual is invoked exactly k times.

(a) (15 points) Design an algorithm that, given n and k, constructs a permutation of $1, 2, \ldots, n$ such that `ArchiveSort` is invoked exactly k times. Provide pseudocode and explain the construction clearly.

(b) (10 points) Prove the correctness of your construction. Why does your permutation guarantee exactly k recursive calls?

(c) (5 points) Analyze the time complexity of your construction algorithm.

Question 2. Royal Guard Deployment (30 points)

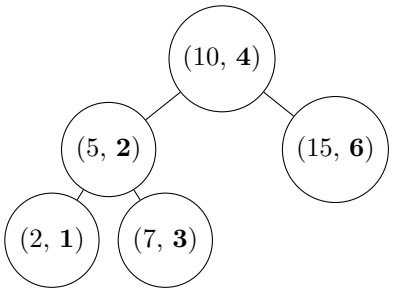
In the kingdom of Algoria, commanders are organized in a binary search tree (BST) based on their rank. Each commander has:

- a **key** representing their rank,
- a **strength value** representing their combat power.

The king wants to select a group of commanders as royal guards. There is one rule for selection:

> **Parent-Child Rule:** If a commander is selected as a royal guard, none of their **children in the BST** can be selected. This is because each parent and child commander are friends and will spend time talking instead of guarding.

Example:



Here, the optimal selection is commanders with keys $\{2, 7, 15\}$, giving a total strength of $1 + 3 + 6 = 10$. Parent-child pairs are not selected together, respecting their friendship rule.

You have to select a subset of commanders maximizing the total strength of the royal guard while respecting the Parent-Child Rule.

(a) (20 points) Design an algorithm in $O(n)$ time and write a clear pseudocode.

(b) (10 points) Give time complexity analysis of your approach for part (a).

Question 3. As Pretty As It Gets! (40 points)

IIT Kanpur is setting up a brand-new row of halls of residence along the road to the Nankari gate. Each plot i has an engineering limit m_i, which represents the tallest possible hall that can be built there.

But here's the catch: the *Hall Executive Committee* insists that the row of hostels must look **pretty in the skyline photos** (especially for the next IITK brochure). To ensure this:

- The height a_i of the hall at plot i must satisfy $1 \leq a_i \leq m_i$.
- No hall should look like a "dip" between two taller halls. Formally, there must not exist indices $j < i < k$ such that $a_j > a_i$ and $a_k > a_i$.

The Institute wants to maximize the **total number of floors across all halls**, i.e., $\sum a_i$, while keeping the skyline picturesque.

Your task is to figure out the optimal distribution of heights $a_1, a_2, \ldots, a_n$.

(a) (10 points) Describe a brute-force algorithm to assign hall heights while satisfying the rules. Explain how it works and analyze its time complexity.

(b) (20 points) Propose an efficient algorithm that computes the hall heights in $O(n)$ time.
Provide pseudocode for your method.
Hint: Try using a *stack-based approach* to get the final height in two passes.

(c) (10 points) Analyze the time complexity of your optimized algorithm for part (b).