Data Structures and Algorithms
ISI-Delhi
Instructor: Krishnendu Mukhopadhyaya

Problems Session 1

Asymptotic Notations

Problem 1

For each row i, indicate whether A_i is in $\mathcal{O}$, Ω, or Θ of B_i. Mark all that apply. The first two rows are examples.

A	**B**	$\mathcal{O}$	Ω	Θ
$5n$	n	✓	✓	✓
5	n	✓		
n^2	$2^{10}n$			
$(7/8)^n$	$(8/7)^n$			
8^n	4^n			
n^5	$32^{\log n}$			
n^{2025}	2025^n			
$n^{0.8}$	$(0.8)^n$			
$n^{\log 3}$	$3^{\log n}$			
$\log(n!)$	$\log(n^n)$			
$n^{1/\log n}$	1			
$\log^7 n$	$n^{0.7}$			
$\log \sqrt{n}$	$4^{\sqrt{n}}$			
$\log(\sqrt{n})$	$\sqrt{\log n}$			

Problem 2

Prove the following:

a) $3n^3 + 4n^2 = \mathcal{O}(n^3)$

b) 16^n is **not** $\mathcal{O}(2^n)$

Problem 3

Use a recursion tree to determine a good asymptotic upper bound on the recurrence

$$T(n) = 3\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n.$$

Then use the substitution method to verify your bound. Clearly state your final asymptotic result in Big-Theta notation.

Problem 4

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures.

a) $f(n) = O(g(n))$ implies $g(n) = O(f(n))$.

b) $f(n) + g(n) = \Theta(\min(f(n), g(n)))$.

c) $f(n) = O(g(n))$ implies $\lg(f(n)) = O(\lg(g(n)))$, where $\lg(g(n)) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n.

d) $f(n) = O(g(n))$ implies $2^{f(n)} = O(2^{g(n)})$.

e) $f(n) = O((f(n))^2)$.

f) $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$.

g) $f(n) = \Theta(f(n/2))$.

h) $f(n) + o(f(n)) = \Theta(f(n))$.

Problem 5

Let

$$p(n) = \sum_{i=0}^{d} a_i n^i,$$

where $a_d > 0$, be a degree-d polynomial in n, and let k be a constant. Use the definitions of the asymptotic notations to prove the following properties:

a) If $k \geq d$, then $p(n) = O(n^k)$.

b) If $k \leq d$, then $p(n) = \Omega(n^k)$.

c) If $k = d$, then $p(n) = \Theta(n^k)$.

d) If $k > d$, then $p(n) = o(n^k)$.

e) If $k < d$, then $p(n) = \omega(n^k)$.

Data Structures and Algorithms
ISI-Delhi
Instructor: Krishnendu Mukhopadhyaya

Problems Session 2

Introduction to Data Structures

Problem 1

Given a singly linked list, the task is to find the middle node of the linked list.

- If the number of nodes is **odd**, return the middle node.
- If the number of nodes is **even**, there are two middle nodes, so return the **second middle node**.

Problem 2

Show how to implement a queue using two stacks. Analyze the running time of the queue operations.

Problem 3

For the set $\{1, 4, 5, 10, 16, 17, 21\}$ of keys, draw binary search trees of heights 2, 3, 4, 5, and 6.

Problem 4

Give a nonrecursive algorithm that performs an inorder tree walk using stacks.

Problem 5

For each of the four types of lists in the following table, what is the asymptotic worst-case running time for each dynamic-set operation listed?

Operation	Unsorted singly linked	Sorted singly linked	Unsorted doubly linked	Sorted doubly linked
SEARCH(L, k)				
INSERT(L, x)				
DELETE(L, x)				
SUCCESSOR(L, x)				
PREDECESSOR(L, x)				
MINIMUM(L)				
MAXIMUM(L)				

Supplementary Excercise

1. Argue that since sorting n elements takes $\Omega(n \lg n)$ time in the worst case in the comparison model, any comparison-based algorithm for constructing a binary search tree from an arbitrary list of n elements takes $\Omega(n \lg n)$ time in the worst case.

Data Structures and Algorithms
ISI-Delhi
Instructor: Krishnendu Mukhopadhyaya

Problems Session 3

Heaps and Hash Functions

Problem 1

Is the array with values $\langle 23, 17, 14, 6, 13, 10, 1, 5, 7, 12 \rangle$ a max-heap?

Problem 2

What is the running time of HEAPSORT on an array A of length n that is already sorted in increasing order? What about decreasing order?

Problem 3

Demonstrate what happens when we insert the keys $5, 28, 19, 15, 20, 33, 12, 17, 10$ into a hash table with collisions resolved by chaining. Let the table have 9 slots, and let the hash function be $h(k) = k \bmod 9$.

Problem 4

Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 88, 59$ into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k$. Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \bmod (m - 1))$.

Data Structures and Algorithms
ISI-Delhi
Instructor: Krishnendu Mukhopadhyaya

Problems Session 4

Sorting, divide and conquer algorithms

Problem 1

Given two arrays `arr[]` and `dep[]`, that represent the arrival and departure time of i-th train respectively. Find an algorithm to determine the minimum number of platforms required so that no train has to wait. If the departure time of one train is the same as the arrival time of another train, both trains cannot use the same platform at that time.

Note: Time intervals are in the 24-hour format (HHMM), where the first two characters represent hour (between 00 to 23) and the last two characters represent minutes (this will be ≤ 59 and ≥ 0). Leading zeros for hours less than 10 are optional (e.g., 0900 is the same as 900).

Input: `arr[] = [1000, 935, 1100]`, `dep[] = [1200, 1240, 1130]`
Output: 3
Explanation: *We need 3 platforms for the arrival of these trains because all three trains have overlapping time.*

Input: `arr[] = [900, 1235, 1100]`, `dep[] = [1000, 1240, 1200]`
Output: 1
Explanation: *Only 1 platform is required for all the three trains because the departure time of each train is less than arrival time of next train.*

Problem 2

Use a recursion tree to determine a good asymptotic upper bound on the recurrence

$$T(n) = 3\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n.$$

Problem 3

Given a set S of n numbers and a number x, describe a $\Theta(n \log n)$-time algorithm to check whether there are two numbers in S whose sum equals x.

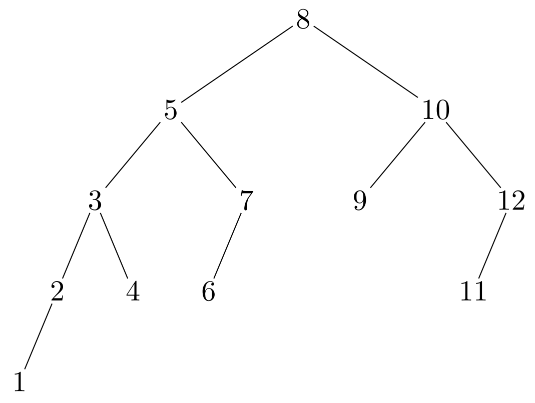
Data Structures and Algorithms
ISI-Delhi
Instructor: Krishnendu Mukhopadhyaya

Problems Session 5

Sorting, AVL Trees

Problem 1: Sequence Rotations

Below is a Sequence AVL Tree T. Perform operation `T.delete_at(8)` and draw the tree after each rotation operation performed during the operation.



Problem 2: Quicksort on Doubly Linked List

Given a **doubly linked list**, the task is to sort the doubly linked list in **non-decreasing** order using the **quicksort**.
Examples:

- **Input:** head: $5 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 1 \leftrightarrow 2$

- **Input:** head: $1 \leftrightarrow 5 \leftrightarrow 2 \leftrightarrow 3$

Problem 3

Given an array arr[] of integers. In one operation you can choose an index i, and increment the element arr[i] by 1. The task is to return the minimum number of operations needed to make every value in the array arr[] unique.

Input: `arr[] = [3, 2, 1, 2, 1, 7]`

Input: `arr[] = [1, 2, 2]`

Input: `arr[] = [5, 4, 3, 2, 1]`

Data Structures and Algorithms
ISI-Delhi
Instructor: Krishnendu Mukhopadhyaya

Problems Session 6

Practice problems

Problem 1

Given **in-order** and **pre-order** traversals of a Binary Tree, the task is to **construct** the Binary Tree and return its **root**.
Example:

- *Input*: `inorder[]` = [3, 1, 4, 0, 5, 2], `preorder[]` = [0, 1, 3, 4, 2, 5]
- *Output*: [0, 1, 2, 3, 4, 5]

Problem 2

Given an array `arr[]` of rope lengths, connect all ropes into a single rope with the minimum total cost. The cost to connect two ropes is the sum of their lengths.
Examples:

- *Input*: `arr[]` = [4, 3, 2, 6]
- *Output*: 29

Problem 3

Which of the following is TRUE?

(A) The cost of searching an AVL tree is $\theta(\log n)$ but that of a binary search tree is $O(n)$

(B) The cost of searching an AVL tree is $\theta(\log n)$ but that of a complete binary tree is $\theta(n \log n)$

(C) The cost of searching a binary search tree is $O(\log n)$ but that of an AVL tree is $\theta(n)$

(D) The cost of searching an AVL tree is $\theta(n \log n)$ but that of a binary search tree is $O(n)$

Problem 4

Given an array arr[] of integers of length n, in one operation you can choose an index i, and increment the element arr[i] by 1. Give an asymptotic upper bound on the minimum number of operations needed to make every value in the array arr[] unique, and provide an example where the bound is attained.