National University of Computer and Emerging Sciences

Data Structures (CS2001)

Final Exam

Date: Fri, 20 Dec 2024

Course Instructor(s)

ZA, SK, FA, MN, SF, AK, MM, UN, UH, AK

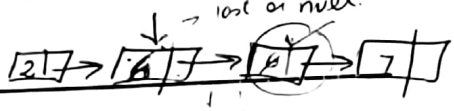
Total Time (Hrs): 3hr

Total Marks: 70

Total Questions: 10

Roll No _____ Section _____ Student Signature _____

Note: Solve Q1-Q8 on the Question Paper and only Q9, Q10 on the answer sheet.



CLO # 1: Demonstrate basic concepts of data structure and algorithms

Q1: **(Marks: 4)**

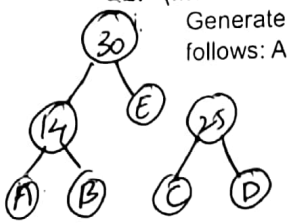
Given a singly linked list and a pointer to a node that needs to be deleted (the node is not the head of the list), design an efficient method to delete the specified node without traversing the list from the head. Note that you only have access to the pointer of the node to be deleted.

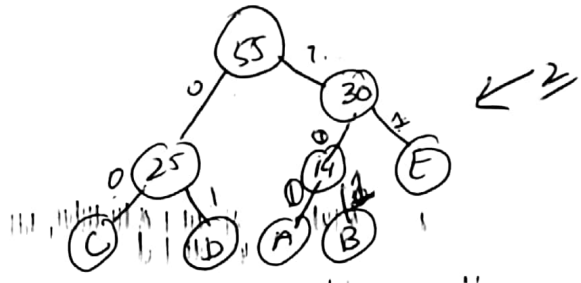
Check if node is null or last node.
↳ if last node then copy its next it will create a dangling pointer to previous or
↳ if not last node
↳ copy (i+1) value to i and connect i to i+1 → next - store i+1 in a temp variable and then delete

CLO # 1: Demonstrate basic concepts of data structure and algorithms

Q2: **(Marks: 2+2)**

Generate the Huffman encoding tree for the message, with each character's frequency as follows: A (5), B(9), C(12), D(13), and E(16).





i) Using the Huffman encoding tree generated above, assign binary codes to each character

A= 100	B= 101	C= 00	D= 01	E= 11

Binary

CLO # 1: Demonstrate basic concepts of data structure and algorithms

Q3: (Marks: 6)

You are given a hash table of size 15 (Indices 0-14). Insert the following keys using **Coalesced Hashing**. For each collision, link the indices appropriately

How Coalesced Hashing Works:

1. Coalesced hashing combines **linear probing** and **linked iists** to handle collisions.
2. Each slot in the table has two fields:
   - **Key:** Stores the value or is empty.
   - **Pointer:** Points to the next slot in case of a collision.
3. When a collision occurs:
   - The key is placed in a free slot next in the table.
   - A **pointer** from the original index points to that new slot.

Example:

- **Hash Function:** h(k)=k mod 10
- Insert **12**: place it at index 2.
- Insert **22**: h(22) → index 2 is occupied → place **22** in the next available slot, e.g., index 3, and link index 2 → index 3.

Keys to Insert (k): 15, 26, 38, 45, 55, 30

Hash Function: h(k)=(3· k+7) mod 15

Fill the table:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Key		26						15	45	55	26	30			
Pointer (From)		Null						Null	7	7		7			

Total number of collisions: 3

CLO # 1: Demonstrate basic concepts of data structure and algorithms

Q4: (Marks: 10)

For each of the following scenarios, suggest an appropriate data structure (time and space efficient). Also, provide the **time complexity** for best and worst case.

1. **Scenario:** We want to design a spell checker for a text editor. Each word typed by the user is checked to determine whether it is spelled correctly or not.
Task: Identify a suitable data structure required for the spell checker. Additionally, calculate the time complexity of checking the spelling of one word using the chosen data structure.

BST → best O(logn)
→ worst O(n) if unbalanced tree otherwise O(logn).
✓AVL →

Hashing can also be used but worst case will be O(1).

2. **Scenario**: In a software project, tasks need to be assigned to multiple team members. Clients can add new tasks, each consisting of a task description and the time required to complete it. The tasks are assigned to team members so that the tasks requiring the least amount of time are assigned first.
Task: Based on the given requirements, identify the most suitable data structure for managing and assigning tasks. Additionally, determine the time complexity for adding a new task and assigning an existing task to a team member.

Min heap.
Adding = O(logn) heap insertion.
Assigning = O(logn) heap deletion.

3. **Scenario**: The government of Pakistan has implemented fake news detection software to monitor Twitter posts. The software must check each post uploaded on Twitter before it becomes accessible in Pakistan. However, the rate at which posts are created on Twitter is faster than the rate at which the fake news detector can process them. As a result, the posts need to be temporarily stored in a buffer.
Task: Identify the most suitable data structure for storing the buffered posts until the fake news detector can process them. Also, determine the time complexity for adding new posts to the buffer and retrieving posts for processing.

1- Queue.
add → O(1)
deque → O(1)
Answer.

2- ~~Doubly~~ link list (with both front and tail).

4. **Scenario**: In an e-commerce website, customers store the items they wish to purchase in a shopping cart. These items are then processed during checkout.
Task: Identify the most suitable data structure for storing items in the shopping cart. Also, determine the time complexity of adding a new item to the cart.

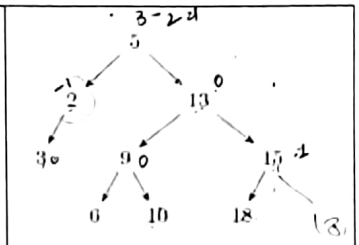
array or LinkList.
add = O(1)

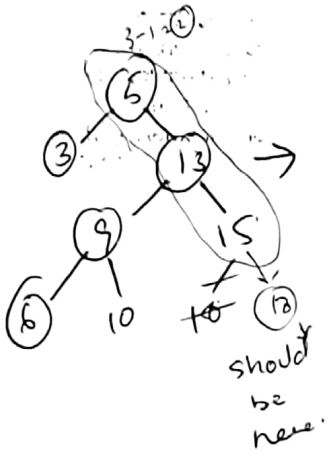
CLO # 4: Determine bugs in programs and recognize required operations with data structures.

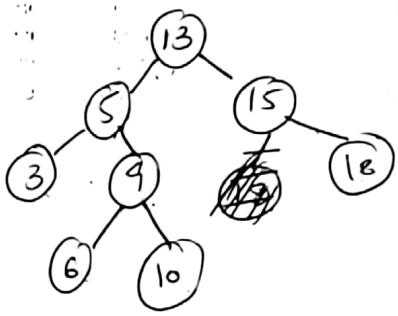
Q5: (Marks:5)

Consider the given AVL tree. Perform the following operations:

a) Calculate and indicate the balance factor for each node on the given AVL tree.
b) Delete Node 2. Identify any imbalanced node(s).
c) Perform the necessary rotation(s) to restore the AVL tree property. Clearly mention each rotation performed and show the final balanced tree after all rotations.







National University of Computer and Emerging Sciences

CLO # 2: Evaluate different data structures in terms of memory complexity and time

Q6: (Marks: 3+1+1)

Consider a directed graph in which the vertices represent different courses and edges represent the prerequisite relation between these courses. **Answer the following questions and Explain your answer briefly.**

i. How can we determine all the courses that have no prerequisite courses? Explain your answer briefly.

Identify vertices with no incoming edges. 3/

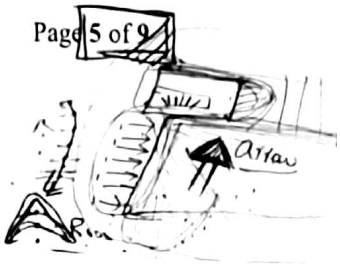
ii. Give time complexity for the above problem when the graph is represented as an Adjacency Matrix.

$O(V^2)$ 1/

iii. Give time complexity for the above problem when the graph is represented as an Adjacency List.

$O(V+E)$ 1/

for dense graph $O(V^2)$.



National University of Computer and Emerging Sciences

CLO # 2: Evaluate different data structures in terms of memory complexity and time

Q7: (Marks: 5)

Compute the time complexity for the following function. First, compute each statement's big-oh frequency and then give the function's overall (big-oh)time complexity. Give the tight bounds.

```
int func(int arr[], int n) {
  int ** dp = new int*[n];
  for(int i=0;i<n;i++)
    dp[i] = new int[n];

  for (int len = 2; len < n; len++) {
    for (int i = 0; i < n - len; i++) {
      int j = i + len;
      dp[i][j] = INT_MAX;

      for (int k = i + 1; k < j; k++) {
        int cost = dp[i][k] + dp[k][j] + arr[i] * arr[k] * arr[j];
        dp[i][j] = min(dp[i][j], cost);
      }
    }
  }

  return dp[0][n - 1];
}
```

T(N) for each line

T(N) estimate and Big O

CLO # 4: Determine bugs in programs and recognize required operations with data structures.

Q8: (Marks: 6)

We need to design a data structure that can efficiently handle dynamic data, insert new numbers, and return the median of all numbers seen so far in O(log n) time. To solve this problem efficiently, we decided to use **two heaps**: a **max-heap** to store the smaller half of the numbers and a **min-heap** to store the larger half of the numbers.

We ensure that the **root** of the max-heap will always be less than or equal to the **root** of the min-heap. Furthermore, to maintain the balance, the size of the max-heap can either be equal to or one greater than the size of the min- heap.

i) Explain how a new number is inserted into the appropriate heap:

⇒ If the new number is smaller than or equal to the root of the Max-heap, it goes to max heap. Otherwise Min.

⇒ Compare the new no with root of Max heap and follow properties of heap.

ii) Explain how you will rebalance the heaps so that the size property above is maintained.

→ If Max heap large → remove root and insert in min heap.

→ If Min heap is too large → remove root ↓ insert to max heap.

iii) How can we get the median in O(1) time using this combo of min-max heap? Explain both cases when total numbers are even or odd.

odd → Max heap root is median as it will have 1 extra element

even → avg of roots of max and min.

CLO # 3: Design appropriate data structures to solve real world problems related to the program.

Q9: (Marks: 10)

Given a BST, find the maximum balance factor in **O(n) time.** The balance factor of a node is defined as the difference between the height of the left and right subtree of the node. Your task is to write a recursive function **MaxFactor** in C++ that calculates and returns the *maximum balance factor* **in one traversal of the given BST.** *Note: Code should be efficient (space and time), well structured, and properly commented. The comments carry two marks.*

```
int maxFactor(node *root, int &maxb){
    if (root==null) return 0;
    int lefth = maxFactor(root->left, maxb)
    in righth = maxFactor(root->right, maxb);
    int balance = abs(lefth - righth);
    maxb = max(maxb, balance);
    return max(lefth, righth)+1;
}
```