Internal Assessment Test 2 – May 2025

Sub:	Analysis & Design of Algorithms					Sub Code:	BCS401	Branch:	AIDS & CSE(AIDS)	
Date:	26/5/2025	Duration:	90 minutes	Max Marks:	50	Sem/Sec:	IV -A, B & C		OBE	

		Answer any FIVE FULL Questions	MARKS	CO	RBT
1	a	What is backtracking? Apply backtracking to solve the below instance of sum of subset problem S={5,10,12,13,15,18} d=30	10	CO6	L3
2	a	Using LC Branch and Bound technique solve the below instance of knapsackProblem weight={2,1,3,2} profit={12,10,20,5} capacity 5	10	CO6	L3
3	a	Define Heap. Explain the bottom -up heap construction algorithm. apply heap sort to sort the list of numbers 2,9,7,6,5,8 in ascending order using array representation	10	CO3	L3
4	a	Define AVL tree with the example. Give the worst case efficiency of operations on AVL tree. construct an AVL tree of the list of the keys:5,6,8,3,2,4,7 indicating each step of key insertion and rotation.	10	CO3	L3
5	b	Explain the following with examples i) P problem ii) NP Problem iii) NP- Complete problem iv) NP – Hard Problems	10	CO5	L3
6	a	Differentiate between branch and bound technique and Backtracking .	5	CO6	L2
	b	Write a algorithm or C code to implement shift table in Horspool algorithm string matching.	5	CO3	L3

CI CCI HOD

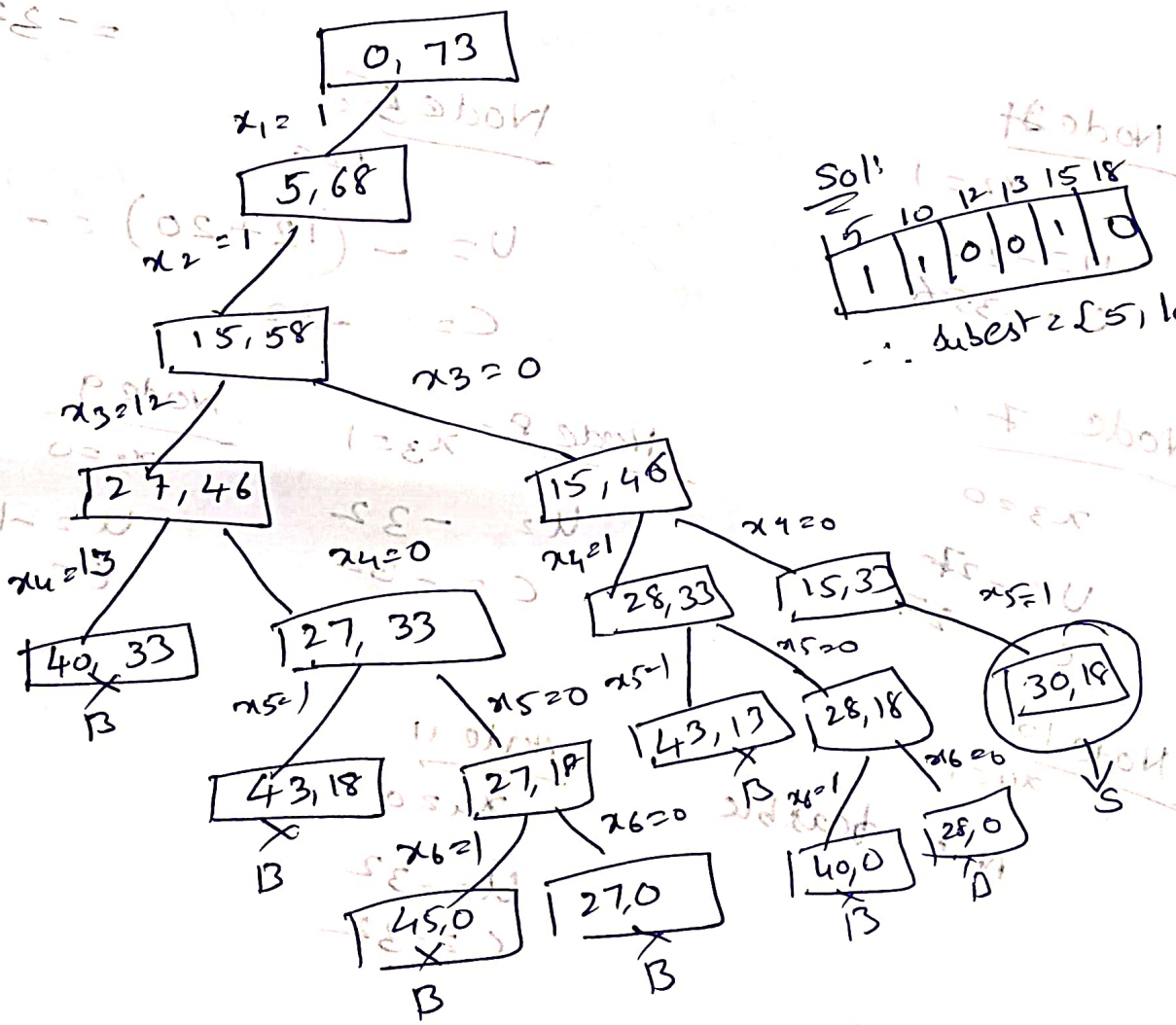
6.66
20
3,

(1)
(a)

Backtracking:

Backtracking is a problem-solving algorithmic technique that involves finding a solution incrementally by trying different options & undoing them if they lead to a dead end.

Given $S = \{5, 10, 12, 13, 15, 18\}$ $d = 30$



Soln:

5	10	12	13	15	18
1	1	0	0	1	0

$\therefore$ subset $= \{5, 10, 15\}$

(2) $w = \{2, 1, 3, 2\}$ $p = \{12, 10, 20, 5\}$ capacity = 5

Node 1:

$ub = -(12+10) = -22$

$c = -(12+10+\frac{6.6}{20} \times \frac{2}{3}) = 35.6$

Node 2: $x_1 = 1$

$ub = -22$

$c = -(35.6)$

Node 3: $x_1 = 0$

$ub = -(10+20) = -30$

$c = -(10+20+\frac{1}{2} \times 5)$

$= -32.5$

Node 4:

$x_2 = 1$

$U = -22$

$C = -35.6$

Node 5:

$x_2 =$

$U = -(12+20) = -32$

$C = -32$

Node 7:

$x_3 = 0$

$U = -27$

$C = -27$

Node 8: $x_3 = 1$

$U = -32$

$C = -32$

Node 9

$x_3 = 0$

$U = -17$

$C = -17$

Node 10:

$x_4 = 1$

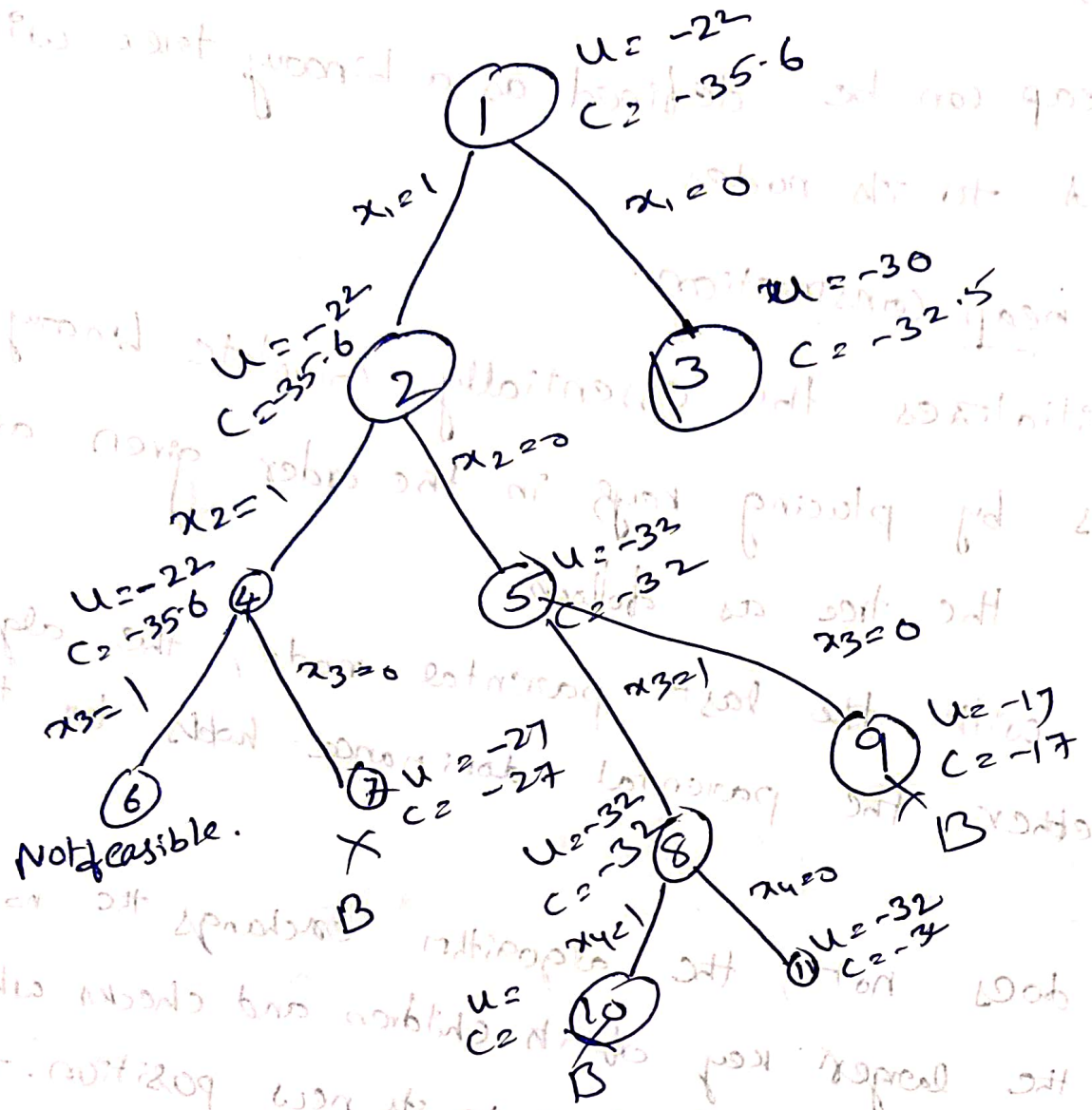
Not feasible

Node 11

$x_4 = 0$

$U = -32$

$C = -32$



profit = 32

$\{x_1, x_2, x_3, x_4\} = \{1, 0, 1, 0\}$

12 + 20 = 32.

(3)

Define heap:

A heap can be defined as a binary tree with keys assigned to its nodes

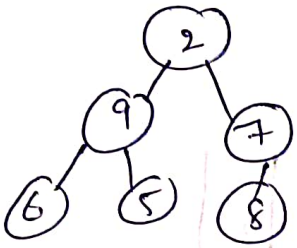
Bottom-up heap construction:

It initializes the essentially complete binary tree with n nodes by placing keys in the order given and then heapifies the tree as follows:

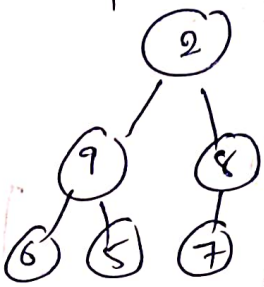
* Starting with the last parental node, the algorithm checks whether the parental dominance holds for the key at this node.
* If it does not, the algorithm exchanges the nodes key k with the larger key of its children and checks whether the parental dominance hold for k in its new position. This process continues until the parental dominance requirement for k is satisfied
* After completing the heapification of the subtree rooted at the current parental node, the algorithm proceeds to do the same for the node's immediate predecessor.

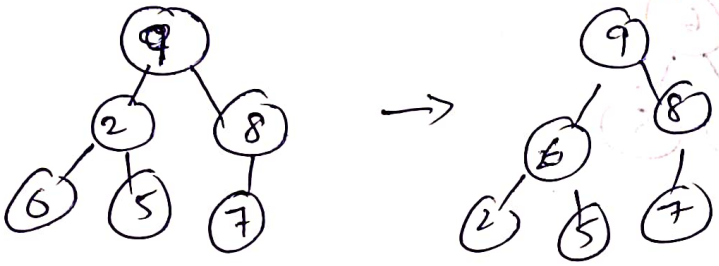
Given list : 2, 9, 7, 6, 5, 8

1



Comparison starts



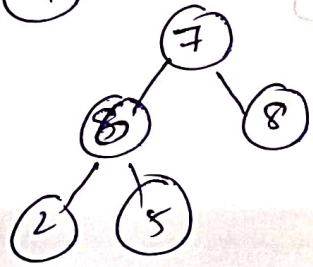


Array:

9	6	8	2	5	7

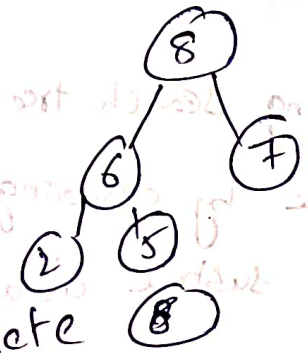
For heap sort:

Delete 9



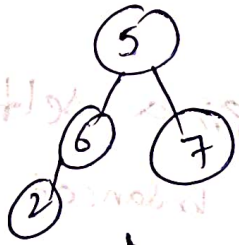
7	6	8	2	5	9

deleted node

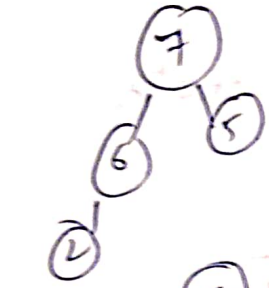


Delete 8

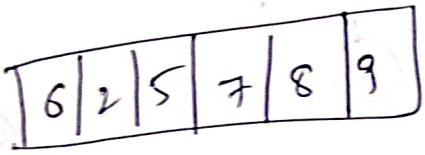
(2)

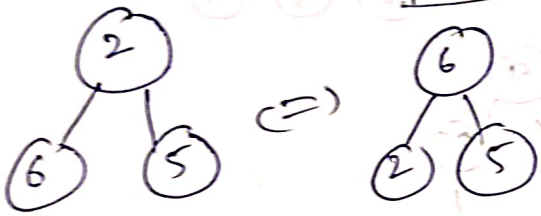


5	6	7	2	8	9

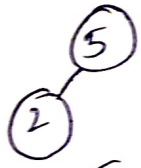


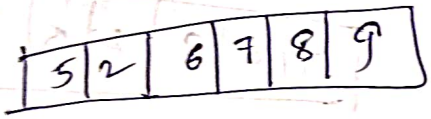
Delete (7)





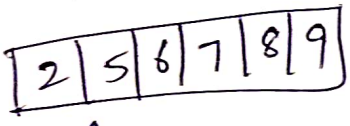
Delete 6





Delete 5.

delete (2)



sorted list

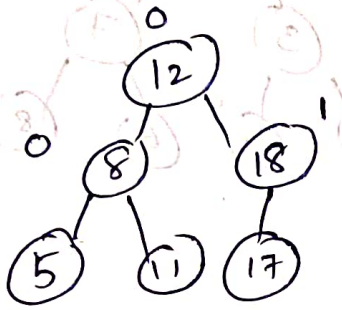
Worst case : (nlogn)

(4) AVL tree:

An AVL tree is a self balancing search tree (BST) that maintains a balanced structure by ensuring the height difference b/w the left & right subtree of any node is no more than one.

* Balance factor: is difference b/w the height of left subtree & right subtree. Should be 0, 1, -1.

Ex:



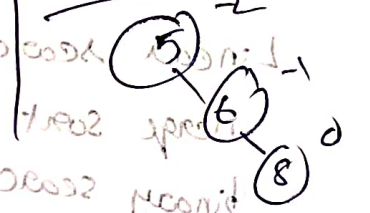
Worst case efficiency is of $O(\log n)$

Given: 5, 6, 8, 3, 2, 4, 7

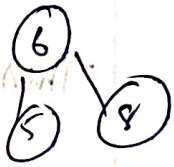
Insert 5

Insert 6

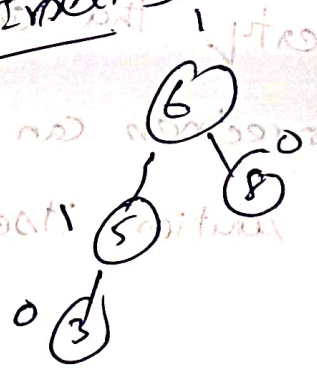
Insert 8



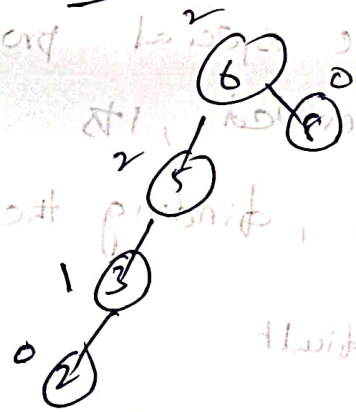
(=) L(5)

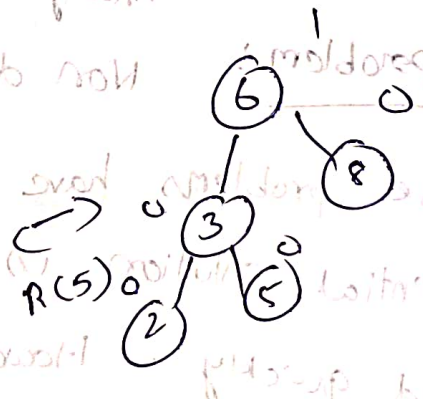


Insert 3

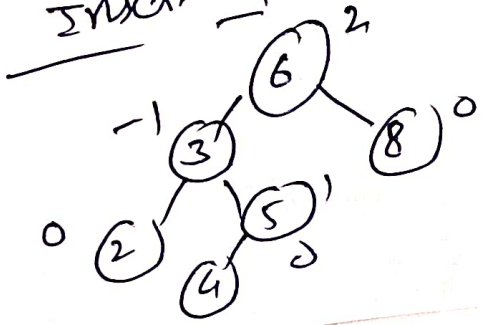


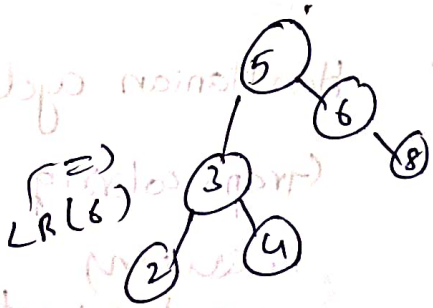
Insert 2



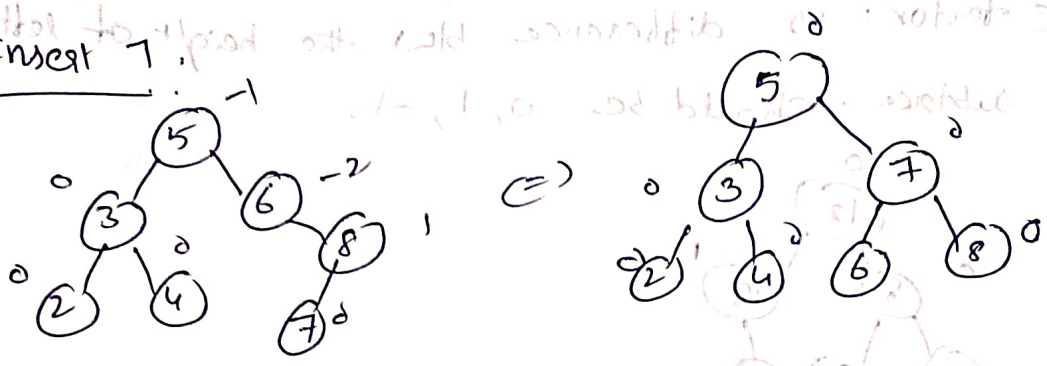


Insert 4





Insert 7.



(5)

P problem:

Class P is a class of decision problem that can be solved in polynomial time by deterministic algorithms. This class of problem is called polynomial.

Ex: Linear search.
Merge sort
binary search

NP problem: Non deterministic polynomial time

These problems have the special property that once a potential solution is provided, its correctness can be verified quickly. However, finding the solution itself may be computationally difficult

Ex: Hamiltonian cycle
Graph coloring
N Queens
sum of subsets

NP Complete:

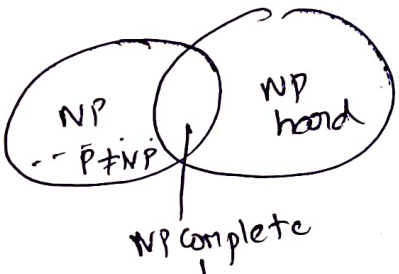
A problem is NP complete if it both NP & NP-hard. NP complete problems are the hard problems in NP.

Ex: Hamiltonian cycle
Satisfiability

NP hard:

An NP-hard problem is at least as hard as the hardest pblm in NP and it is a class of problems such that every problem in NP reduces to NP-hard.

Ex: Halting problem.



6 @

Backtracking	Branch and Bound
It is used to find all possible solution available to a problem. When it realises that it has made a bad choice, it undoes the last choice by backing it up.	Branch-and-Bound is used to solve optimisation problems. When it realises that it already has a better optimal solution.

* Solution is represented by state space tree.	* state space tree to get Optimal solution
* Backtracking traverses the DFS Manner	* Branch and Bound traverse the DFS & BFS
* used for solving Decision problem Ex: N queen Sum of subset	* used for solving Optimization problem Ex: TSP knapsack problem

(6)(b) Algorithm to implement Shift table in horspool algorithm string matching.

```
Shift value (P[0..m-1])
for i = 0 to size-1 do
    Table[i] = m
for j = 0 to m-2 do
    Table[P[j]] = m-1-j
return Table.
```