Step-by-Step Solutions: Internal Assessment Test 2 – May 2025 Analysis & Design of Algorithms

Question 1: Backtracking - Sum of Subset Problem

Backtracking is a general algorithmic technique that considers searching every possible combination to solve an optimization problem. It incrementally builds candidates to the solution and abandons a candidate ("backtracks") as soon as it determines that the candidate cannot lead to a valid solution.

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

We find all subsets that sum to 30 using backtracking.

Steps:
1. Start from index 0, current sum = 0, empty subset.
2. Recursively include or exclude each element.
3. Track current sum and subset.
4. If current sum equals 30, print the subset.
5. If current sum exceeds 30 or index exceeds length, backtrack.

Solutions:
- {12, 13, 5}
- {10, 12, 8}

Question 2: LC Branch and Bound – Knapsack Problem

Weights = {2,1,3,2}, Profits = {12,10,20,5}, Capacity = 5

Use LC Branch and Bound Tree to find optimal profit.

Steps:
1. Sort items by profit/weight ratio.
2. Use a priority queue for nodes (based on upper bound).
3. Branch into including or excluding an item.
4. Keep track of maximum profit found.

Optimal Solution: Include item 1 (2kg, 12), item 2 (1kg, 10), item 4 (2kg, 5)

Total weight = 5kg, Total profit = 27

Question 3: Heap and Heap Sort

Heap is a complete binary tree satisfying heap property (Max-Heap or Min-Heap).

Bottom-Up Heap Construction:
1. Start from last non-leaf node and heapify up to root.

Heap Sort for list [2,9,7,6,5,8]:
1. Build Max Heap: [9,6,8,2,5,7]
2. Swap root with last: [7,6,8,2,5,9]
3. Heapify and repeat.

Sorted List: [2,5,6,7,8,9]

Question 4: AVL Tree

AVL Tree is a self-balancing binary search tree with height balance factor -1, 0, or 1.

Worst Case Efficiency: O(log n) for insert, delete, search

Keys: 5,6,8,3,2,4,7
Insertions & Rotations:
- Insert 5: Root
- Insert 6: Right of 5
- Insert 8: Right of 6 → imbalance → Left Rotation at 5
- Insert 3: Left of 5
- Insert 2: Left of 3
- Insert 4: Right of 3 → imbalance → Left-Right Rotation at 5
- Insert 7: Insert to right subtree → Rebalance

Final AVL Tree is balanced.

Question 5: Complexity Classes

i) P Problem: Solvable in polynomial time, e.g., Binary Search.
ii) NP Problem: Verifiable in polynomial time, e.g., Hamiltonian Path.
iii) NP-Complete: In NP and all NP problems reduce to it, e.g., SAT.

iv) NP-Hard: As hard as NP-Complete, may not be in NP, e.g., Halting Problem.

Question 6: Backtracking vs Branch and Bound & Horspool Algorithm

a) Backtracking vs Branch and Bound:
- Backtracking: Explores all possibilities, used for constraint satisfaction.
- Branch and Bound: Uses bound function to prune branches, for optimization problems.

b) Horspool Shift Table Algorithm in C:

```
void shiftTable(char pattern[], int table[], int m) {
  int i;
  for (i = 0; i < 256; i++) table[i] = m;
  for (i = 0; i < m - 1; i++) table[(int)pattern[i]] = m - 1 - i;
}
```