DAA Tutorial-7
Backtracking and Branch & Bound

Problem 1: N-Queens

Implement a single backtracking program that places n queens on an $n \times n$ board by filling rows left-to-right and pruning any placement that conflicts by column or diagonal; for an input n (test with $n = 4$) print every solution as an n-tuple of column indices (row $1 \dots n$), and also report the total nodes generated and pruned.

Problem 2: Tug of War Problem

Given a set of integers, divide the set into two subsets such that the difference between their sums is minimized. Use backtracking (or recursion) by exploring the choice of placing each element into one of the two groups and tracking the partition with minimal difference.

Problem 3: Hamiltonian Cycle

Develop a single backtracking routine that grows a path from vertex 1 in the undirected graph $V = \{1, 2, 3, 4, 5, 6\}$, $E = \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1), (1, 3), (2, 5)\}$, never revisiting vertices and pruning dead ends; on success print a Hamiltonian cycle starting and ending at 1, otherwise print "none".

Problem 4: Subset Sum

Create one backtracking solver that explores include/exclude choices over $S = \{3, 4, 5, 6, 7\}$ in the given order to reach target $T = 12$, pruning whenever the current sum exceeds T or the current sum plus the sum of all remaining elements cannot reach T; print all subsets that sum to 12.

Problem 5: 0/1 Knapsack (Branch & Bound)

Implement a best-first Branch & Bound solver for capacity $C = 9$ and items $(p_i, w_i) \in \{(20, 2), (30, 5), (35, 7), (12, 3), (3, 1)\}$ using the fractional-knapsack upper bound at each node; output the optimal item set with total profit and, for transparency, the number of nodes expanded and pruned.

Problem 6: Traveling Salesperson (Branch & Bound)

Implement a Branch & Bound TSP using cost-matrix reduction as a lower bound and best-first expansion on the symmetric matrix for cities A, B, C, D, E:

	A	B	C	D	E
A	0	10	15	20	10
B	10	0	35	25	17
C	15	35	0	30	28
D	20	25	30	0	22
E	10	17	28	22	0

Run the algorithm and print the optimal tour and its total cost, along with the number of nodes expanded.