DAA Tutorial-5
Dynamic Programming

Note: For each problem, you are required to implement the solution in C++, present the complete code, and demonstrate its execution to the evaluator.

Problem 1: Matrix Chain Multiplication

Given a sequence of matrices $A_1, A_2, \ldots, A_n$ where matrix A_i has dimensions $p_{i-1} \times p_i$, the goal is to find the most efficient way to multiply these matrices by minimizing the number of scalar multiplications.

(a) Define the recursive formula for the Matrix Chain Multiplication problem.

(b) Create a dynamic programming solution that calculates the minimum cost to multiply the matrices.

(c) Describe the complexity of the solution in terms of time and space.

Input: Dimensions $p = [10, 30, 5, 60]$ (matrices $A_1 : 10 \times 30,\ A_2 : 30 \times 5,\ A_3 : 5 \times 60$).
Output: Minimum scalar multiplications = 4500.
Explanation: Optimal parenthesization is $((A_1A_2)A_3)$: cost $A_1A_2 = 10 \cdot 30 \cdot 5 = 1500$, then $(A_1A_2)A_3 = 10 \cdot 5 \cdot 60 = 3000$, total $1500 + 3000 = 4500$.

Problem 2: Edit Distance

The Edit Distance between two strings is the minimum number of operations required to convert one string into the other. Allowed operations are insertion, deletion, and substitution.

(a) Define the recursive formula for calculating the Edit Distance between two strings.

(b) Describe a dynamic programming approach to compute the Edit Distance.

(c) Analyze the time and space complexity of this approach.

Input: Strings “kitten” and “sitting”.
Output: Edit distance = 3.
Explanation: Convert “kitten” → “sitten” (substitute k→s), “sitten” → “sittin” (substitute e→i), “sittin” → “sitting” (insert g) — total 3 operations.

Problem 3: Palindrome Partitioning

Given a string S, partition it such that every substring in the partition is a palindrome. Minimize the number of cuts needed to partition the string.

(a) Define the recursive formula for finding the minimum cuts.

(b) Implement a dynamic programming approach to calculate the minimum number of cuts.

(c) Discuss the complexity of your solution.

Input: String $S =$“aab”.
Output: Minimum cuts = 1.
Explanation: Partition as “aa” — “b” (only one cut). Any other partition needs more cuts.

Problem 4: Egg Drop Problem

Given k eggs and n floors, find the minimum number of attempts needed to determine the highest floor from which an egg can be dropped without breaking.

(a) Define the recursive relation for the Egg Drop problem.

(b) Formulate a dynamic programming solution to minimize the number of attempts.

(c) Analyze the time complexity of this solution.

Input: $k = 2$ eggs, $n = 10$ floors.
Output: Minimum number of attempts in worst case = 4.
Explanation: Minimum t such that $t(t + 1)/2 \geq 10$; $t = 4$ gives triangular number 10, so 4 moves suffice.

Problem 5: Rod Cutting Problem

Given a rod of length n and a list of prices for each possible length, determine the maximum profit that can be obtained by cutting the rod into pieces.

(a) Define the recursive relation for the Rod Cutting problem.

(b) Create a dynamic programming solution using a 1D array to store the maximum profit for each length.

(c) Discuss the time complexity of the solution.

Input: Rod length $n = 4$, prices $p[1 \dots 4] = [1, 5, 8, 9]$ (price for length i is $p[i]$).
Output: Maximum profit = 10.
Explanation: Best cut is two pieces of length 2: $5 + 5 = 10 > 9$ (no cut), so maximum is 10.

Problem 6: Optimal Binary Search Tree (OBST)

Given a set of keys $K = \{k_1, k_2, \dots, k_n\}$ with associated probabilities $p_1, p_2, \dots, p_n$, construct a binary search tree that minimizes the expected search cost.

(a) Define the dynamic programming approach to build an OBST.

(b) Describe the recursive formula and use it to develop a DP table for calculating the minimum cost.

(c) Discuss the complexity of constructing an OBST using dynamic programming.

Input: Keys k_1, k_2, k_3 with access probabilities $p = [0.2, 0.5, 0.3]$.
Output: Minimum expected search cost = 1.5.
Explanation: Select k_2 as root: depths $\{k_2 : 1,\ k_1 : 2,\ k_3 : 2\}$. Expected cost $= 0.5 \cdot 1 + 0.2 \cdot 2 + 0.3 \cdot 2 = 1.5$, which is optimal among arrangements.