Assignment - Day 6: 2-D Dynamic Programming Problems

Assignment Problems:

1. 1. Unique Paths II (LeetCode 63)

Link: https://leetcode.com/problems/unique-paths-ii/
Objective: Find the total number of unique paths from the top-left corner to the bottom-right corner of a grid with obstacles.
Hint:
- Define a recursive function f(i, j) that returns the number of ways to reach cell (i, j).
- Recursive Step: f(i, j) = f(i-1, j) + f(i, j-1)
- Store results in a dp[i][j] cache before returning.

2. 2. Dungeon Game (LeetCode 174)

Link: https://leetcode.com/problems/dungeon-game/
Objective: Find the knight's minimum initial health needed to rescue the princess starting from (0, 0).
Hint:
- Define a recursive function f(i, j) that returns the minimum health required from cell (i, j) to reach the destination.

3. 3. Minimum Distance to Type a Word Using Two Fingers (LeetCode 1320)

Link: https://leetcode.com/problems/minimum-distance-to-type-a-word-using-two-fingers/
Objective: Find the minimum total distance needed to type a word using two fingers.
Hint :
- Define a recursive function f(index, left, right) → minimum distance to type substring starting from index given left and right finger positions.

4. 4. Minimum Falling Path Sum II (LeetCode 1289)

Link: https://leetcode.com/problems/minimum-falling-path-sum-ii/
Objective: Find the minimum sum of a falling path through a grid such that no two consecutive elements come from the same column.
Hint (Memoization Approach):
- Define f(i, j) = minimum path sum starting from cell (i, j) to the bottom.
- Base Case: If i is the last row → return grid[i][j].
- Recursive Step: f(i, j) = grid[i][j] + min(f(i+1, k)) for all k != j.
- Store results in dp[i][j] to avoid recomputation.

5. 5. Optimal Division (LeetCode 553)

Link: https://leetcode.com/problems/optimal-division/
Objective: Find an expression that yields the maximum result by placing parentheses optimally.

Key Concepts to Revise:

Before attempting these problems, ensure you understand:
1. How recursion breaks problems into smaller overlapping subproblems.
2. Defining recursive states with multiple parameters.
3. Using a memo dictionary or 2D array to cache results.

Submission Instructions:

- Solve and submit all five required problems using recursion + memoization.
- Avoid direct tabulation unless explicitly discussed.
- Submit your code via the Google Form.
- Deadline: 13 November 2025 (02:00 am).

Note:

Memoization allows you to visualize recursion with caching — focus on understanding state parameters, base cases, and recursive transitions. Tabulation is simply the next step of converting your memoized recursion into iteration.

Best regards,
Training Team