Week 12 Summary: Dynamic Programming (Memoization, Knapsack, LCS, Edit Distance)

Z5007 Programming and Data Structures · MTech DSAI · Odd Semester 2026-27 · Dr. Innocent Nyalala

1. Optimal Substructure and Overlapping Subproblems

Dynamic programming (DP) solves problems whose optimal solution combines optimal subproblem solutions (optimal substructure) and whose naive recursion recomputes identical subproblems exponentially often (overlapping subproblems).

- **Top-Down Memoization:** Keep the natural recursive formulation and cache each solved state in a dictionary or table.
- **Bottom-Up Tabulation:** Fill a DP table in topological dependency order, eliminating recursion overhead and often compressing space.

2. 0/1 Knapsack ($O(nW)$ Pseudo-Polynomial Time)

Given n indivisible items (v_i, w_i) and capacity W, greedy value-per-weight ranking can fail. Define $\mathrm{dp}[i][w]$ as the maximum value using the first i items within capacity w:

$$\mathrm{dp}[i][w] = \begin{cases} \mathrm{dp}[i-1][w] & \text{if } w_i > w, \\ \max\big(\mathrm{dp}[i-1][w],\ v_i + \mathrm{dp}[i-1][w-w_i]\big) & \text{if } w_i \le w. \end{cases}$$

Backtracking from $\mathrm{dp}[n][W]$ checks whether $\mathrm{dp}[i][w] \neq \mathrm{dp}[i-1][w]$ to recover the exact chosen subset.

3. Sequence DP: Longest Common Subsequence (LCS) and Edit Distance

For strings $A[0..m-1]$ and $B[0..n-1]$, both LCS and Levenshtein edit distance fill an $(m+1)\times(n+1)$ table in $\Theta(mn)$ time:

$$\mathrm{LCS}[i][j] = \begin{cases} 1 + \mathrm{LCS}[i-1][j-1] & A[i-1] = B[j-1], \\ \max(\mathrm{LCS}[i-1][j], \mathrm{LCS}[i][j-1]) & A[i-1] \neq B[j-1], \end{cases}$$

$$\mathrm{ED}[i][j] = \min\big(\mathrm{ED}[i-1][j]+1,\ \mathrm{ED}[i][j-1]+1,\ \mathrm{ED}[i-1][j-1] + \mathbf{1}[A[i-1] \neq B[j-1]]\big).$$

Worked Comparison: Fractional vs. 0/1 Knapsack

With capacity $W = 50$ and items $(60, 10), (100, 20), (120, 30)$, value-to-weight densities are $6.0, 5.0, 4.0$. Greedy picks the first two whole items for value 160 (leaving 20 kg unused), whereas 0/1 DP picks $(100, 20)$ and $(120, 30)$ for optimal value 220.