Algorithms and Data Structures Week 13

Georg Hasebe, 9th December.

Dynamic Programming Summary

Dynamic Programming (DP) is a problem-solving approach that involves breaking a problem into smaller, overlapping subproblems and building up solutions systematically.

- **Understand the Problem**: Begin by deeply understanding the problem. Work through several concrete examples to get a sense of the subproblems, their dependencies, and how they combine to form the overall solution.
- **Don’t Jump to Bottom-Up**: Avoid starting directly with a bottom-up tabulation approach. Instead, focus on grasping the problem’s structure before choosing an implementation strategy.
- **Define Subproblems and Recurrence**: Identify the subproblems and formulate the recurrence relationships that connect them. These are the core of your solution and must be clear before proceeding.
- **Stress-Test Your Recurrence**: Validate your recurrence by stress-testing it with different examples to ensure its correctness for all input scenarios.
- **Don’t Force a Structure**: Avoid trying to fit the problem into a framework or schema that isn’t suitable (recall the star into square analogy). Let the problem naturally guide the structure of your DP approach.
- **Choose an Implementation**: Depending on the problem and recurrence, decide whether to solve it top-down with memoization or bottom-up with tabulation. If using tabulation, think carefully about how to translate subproblem dependencies into a table and the order of filling it.
- **Practice Regularly**: Solving DP problems requires practice. Work through a variety of problems (1D, 2D, and 3D DP) to gain flexibility and confidence. This practice will prepare you to experiment and adapt during exams.
- **Stay Flexible**: If one approach isn’t working, don’t get frustrated. Be open to revisiting the problem and exploring alternative strategies.