Complete Dynamic Programming Problems & Resources Guide

Hey everyone! Dynamic Programming is often considered the hardest topic in DSA, but it doesn't have to be. I've spent months working through DP problems for both tech interviews and competitive programming, and I wanted to share a complete guide that demystifies this topic.

The key insight that changed everything for me, DP isn't one giant topic, it's actually 15-20 clear patterns. Once you recognize these patterns, most DP problems become straightforward. This guide includes curated problem lists, the best learning resources, and proven roadmaps for both interview prep and competitive programming.

Understanding Dynamic Programming

What makes DP hard? Unlike graphs or arrays where you can visualize the data structure, DP problems require you to identify overlapping subproblems and optimal substructure. The same problem can be solved with different state definitions, making it confusing for beginners.

For Tech Interviews: You need to recognize 10-12 core DP patterns and implement them cleanly. About 60-80 well-understood problems is enough. Most interview DP problems are medium difficulty. Focus on 1D DP, 2D DP, Knapsack variations, and LCS/LIS patterns.

For Competitive Programming: You need deep understanding of state transitions, optimization techniques, and the ability to come up with novel DP formulations. This requires 200+ problems. Everything from interviews plus advanced topics like Digit DP, DP on Trees, Bitmask DP, and optimization techniques.

Best Learning Resources

Video Resources for Interview Prep

NeetCode - Dynamic Programming Playlist The absolute best resource for learning interview DP patterns. Clear explanations showing how to identify DP problems and break them down systematically. Covers all important patterns with Python code. About 8-10 hours total, but worth every minute.

take U forward (Striver) - DP Series Comprehensive coverage with detailed explanations. His DP series is legendary in the Indian coding community. Covers everything from basics to advanced with C++/Java implementations. Around 40+ hours of content, very thorough.

Abdul Bari - Dynamic Programming The best for understanding the theory behind DP. His explanations of optimal substructure and overlapping subproblems are crystal clear. His videos on 0/1 Knapsack and LCS are must-watch. About 10-15 hours.

Back To Back SWE - DP Problems Great for understanding the thought process. He explains how to break down problems, identify states, and build recurrence relations. Particularly good for recursion to memoization to tabulation progression.

Video Resources for Competitive Programming

Errichto - DP Lectures Phenomenal series on DP for competitive programming. Covers everything from basics to advanced topics like SOS DP and Convex Hull Trick. His problem-solving streams show how top coders approach DP in contests.

SecondThread - DP Optimization Techniques Excellent coverage of advanced DP optimizations like Convex Hull Trick, Divide and Conquer DP, and Knuth's optimization. Essential for CF Div1 and Div2 D/E problems.

William Fiset - DP Algorithms Clear explanations of classic DP algorithms with good visualizations. Covers both interview and CP aspects well.

Written Resources

For Interviews:

- LeetCode Discuss DP tag - Read top solutions for different approaches
- GeeksforGeeks DP section - Good for understanding concepts with examples
- Tech Interview Handbook DP section - Quick revision before interviews

For Competitive Programming:

- Codeforces DP Catalog - Problems organized by technique
- CP-Algorithms DP section - In-depth explanations with complexity analysis
- USACO Guide DP modules - Structured learning from Bronze to Platinum
- AtCoder DP Contest - 26 educational DP problems covering all basics

Books Worth Reading

Cracking the Coding Interview - Chapter on DP covers interview essentials, good starting point

Dynamic Programming for Coding Interviews by Meenakshi Excellent book focused specifically on DP for tech interviews. Covers all important patterns with detailed explanations. Highly recommended for interview prep.

Competitive Programming 4 - Multiple chapters on DP covering everything from basics to advanced techniques. The section on DP optimization is gold.

Introduction to Algorithms (CLRS) - Chapter 15 covers DP theory rigorously with proofs. Heavy but comprehensive.

Interview Problems (70 Total)

Phase 1: 1D DP - Linear Problems

These are the foundations. If you struggle with these, everything else will be hard. Focus on understanding the transition from recursion to memoization to tabulation.

1. Climbing Stairs
2. Min Cost Climbing Stairs
3. House Robber
4. House Robber II
5. Delete and Earn
6. Maximum Product Subarray
7. Jump Game
8. Jump Game II
9. Decode Ways
10. Decode Ways II
11. Fibonacci Number
12. N-th Tribonacci Number
13. Count Number of Ways to Place Houses
14. Solving Questions With Brainpower
15. Maximum Alternating Subsequence Sum

Phase 2: 2D DP - Grid & Matrix

Grid DP is very common in interviews. Master the pattern of building solution bottom-up from dp[i][j].

16. Unique Paths
17. Unique Paths II
18. Minimum Path Sum
19. Triangle

20. Maximal Square
21. Dungeon Game
22. Cherry Pickup
23. Cherry Pickup II
24. Minimum Falling Path Sum
25. Minimum Falling Path Sum II
26. Maximal Rectangle
27. Count Square Submatrices with All Ones

Phase 3: Knapsack Pattern

Knapsack is THE most important DP pattern. It appears in countless variations. Understand both 0/1 and unbounded knapsack.

28. Partition Equal Subset Sum
29. Target Sum
30. Ones and Zeroes
31. Coin Change
32. Coin Change II
33. Perfect Squares
34. Minimum Cost For Tickets
35. Last Stone Weight II
36. Profitable Schemes
37. Tallest Billboard

Phase 4: String DP

String DP problems often involve LCS, LIS, or palindrome patterns. These are very common in interviews.

38. Longest Common Subsequence
39. Longest Palindromic Substring
40. Longest Palindromic Subsequence
41. Palindromic Substrings
42. Edit Distance
43. Distinct Subsequences
44. Minimum ASCII Delete Sum for Two Strings
45. Delete Operation for Two Strings
46. Shortest Common Supersequence
47. Interleaving String
48. Regular Expression Matching
49. Wildcard Matching

Phase 5: Longest Increasing Subsequence

LIS is special because it has both O(n²) and O(n log n) solutions. Know both approaches.

50. Longest Increasing Subsequence
51. Number of Longest Increasing Subsequence
52. Largest Divisible Subset
53. Russian Doll Envelopes
54. Maximum Length of Pair Chain
55. Longest String Chain
56. Minimum Number of Removals to Make Mountain Array
57. Delete Columns to Make Sorted III

Phase 6: Stock Problems

Stock problems are a special category that tests state machine DP. Learn the pattern once and all variations become easy.

58. Best Time to Buy and Sell Stock
59. Best Time to Buy and Sell Stock II
60. Best Time to Buy and Sell Stock III
61. Best Time to Buy and Sell Stock IV
62. Best Time to Buy and Sell Stock with Cooldown
63. Best Time to Buy and Sell Stock with Transaction Fee

Phase 7: Advanced Interview DP

These are harder problems that combine multiple patterns or require clever state definitions.

64. Word Break
65. Word Break II
66. Palindrome Partitioning II
67. Burst Balloons
68. Stone Game IV
69. Maximum Score from Performing Multiplication Operations
70. Minimum Difficulty of a Job Schedule

Competitive Programming Problems

AtCoder Educational DP Contest (26 problems - Must Solve)

This contest is specifically designed to teach DP. Solve all 26 problems in order. These are perfectly curated for learning.

1. A - Frog 1
2. B - Frog 2
3. C - Vacation
4. D - Knapsack 1
5. E - Knapsack 2
6. F - LCS
7. G - Longest Path
8. H - Grid 1
9. I - Coins
10. J - Sushi
11. K - Stones
12. L - Deque
13. M - Candies
14. N - Slimes
15. O - Matching
16. P - Independent Set
17. Q - Flowers
18. R - Walk
19. S - Digit Sum
20. T - Permutation
21. U - Grouping
22. V - Subtree
23. W - Intervals
24. X - Tower
25. Y - Grid 2
26. Z - Frog 3

CSES Dynamic Programming Section (19 problems)

Excellent problem quality, covers all important DP techniques for competitive programming.

1. Dice Combinations
2. Minimizing Coins
3. Coin Combinations I
4. Coin Combinations II
5. Removing Digits
6. Grid Paths
7. Book Shop
8. Array Description
9. Counting Towers
10. Edit Distance
11. Rectangle Cutting

12. Money Sums
13. Removal Game
14. Two Sets II
15. Increasing Subsequence
16. Projects
17. Elevator Rides
18. Counting Tilings
19. Counting Numbers

Codeforces DP Problems by Rating

Rating 1200-1400 (Beginner CP):

1. Frog 1
2. Array Shrinking
3. Yet Another Array Restoration
4. Block Towers
5. Permutation Transformation

Rating 1400-1600 (Intermediate):

6. Lucky Numbers

7. Balanced Tunnel

8. Wilbur and Array

9. Greg and Array

10. Present

Rating 1600-1800 (Advanced):

11. Kyoya and Colored Balls

12. Dreamoon and WiFi

13. Make Them Equal

14. Integer Sequence Dividing

15. Vasya and Magic Matrix

Rating 1800-2000 (Expert):

16. Subsequences

17. Xenia and Bit Operations

18. Beautiful Numbers

19. Coloring Trees

20. Karen and Cards

Rating 2000-2200 (Candidate Master):

21. Tree Painting

22. Count Good Numbers

23. Ehab's REAL Number Theory Problem

24. Weights Distribution

25. Special Permutation

Advanced CP Topics

Digit DP Problems:

1. CSES - Counting Numbers
2. AtCoder DP S - Digit Sum
3. Codeforces - Magic Numbers
4. SPOJ - NUMTSN
5. SPOJ - PR003004

Bitmask DP Problems:

1. AtCoder DP O - Matching
2. AtCoder DP U - Grouping
3. CSES - Elevator Rides
4. CSES - Hamiltonian Flights
5. Codeforces - Fish

DP on Trees:

1. AtCoder DP P - Independent Set
2. AtCoder DP V - Subtree
3. CSES - Tree Matching
4. CSES - Tree Distances I
5. Codeforces - Tree Painting

SOS DP (Sum Over Subsets):

1. Codeforces - Compatible Numbers
2. Codeforces - Special Permutation
3. Codechef - COVERING

DP Optimization Techniques:

Convex Hull Trick:

1. AtCoder DP Z - Frog 3
2. Codeforces - Levels and Regions

Divide and Conquer DP:

1. AtCoder DP N - Slimes
2. Codeforces - Ciel and Gondolas

Knuth Optimization:

1. AtCoder DP N - Slimes
2. SPOJ - BRKSTRNG

Learning Timeline

For Interview Preparation (7-8 weeks)

Week 1: 1D DP Foundation Solve all 15 problems from Phase 1. Start with Climbing Stairs and understand the progression from recursion to memoization to tabulation. By the end of this week, you should clearly understand what "overlapping subproblems" means. Don't rush this week - it's the foundation for everything else.

Week 2: 2D DP and Grid Problems Solve 12 problems from Phase 2. Grid DP is very visual and intuitive once you get it. Practice drawing the DP table and understanding how each cell depends on previous cells. Unique Paths is the classic starting problem.

Week 3: Knapsack Pattern Solve all 10 problems from Phase 3. This is THE most important DP pattern for interviews. Understand both 0/1 knapsack and unbounded knapsack thoroughly. Many seemingly unrelated problems are actually knapsack in disguise.

Week 4: String DP Solve 12 problems from Phase 4. String DP is common in interviews. Master LCS first, then move to palindrome problems, then edit distance. These three patterns cover most string DP questions.

Week 5: LIS Pattern Solve 8 problems from Phase 5. Understand both the $O(n^2)$ DP solution and the $O(n \log n)$ binary search optimization. This pattern appears in many forms.

Week 6: Stock Problems and State Machines Solve 6 stock problems from Phase 6. These teach you state machine DP which is useful for many other problems. Learn the pattern once and all variations become straightforward.

Week 7: Advanced Problems Solve 7 problems from Phase 7. These combine multiple patterns or require creative state definitions. They're closer to real interview difficulty.

Week 8: Review and Practice Solve 15-20 mixed problems. Do mock interviews. Time yourself. Review problems where you struggled. Make sure you can identify patterns quickly.

For Competitive Programming (6-8 months)

Month 1: Strong Foundation Complete AtCoder Educational DP Contest problems A-M (13 problems). Solve CSES problems 1-8. Practice both top-down and bottom-up approaches. Master basic knapsack, grid DP, and LCS. Target 40-50 problems this month including Codeforces 1200-1400.

Month 2: Intermediate Patterns Complete AtCoder DP Contest problems N-Z (13 problems). Solve CSES problems 9-15. Learn interval DP, game theory DP, and probability DP basics. Solve Codeforces 1400-1600 rated problems. Target 40-50 problems.

Month 3: Bitmask DP Master bitmask DP thoroughly. This is crucial for many CP problems. Solve all bitmask DP problems from the advanced section. Understand traveling salesman, assignment problems, and subset sum using bitmasks. Target 30-40 problems at 1400-1700 rating.

Month 4: DP on Trees Learn tree DP techniques. Solve all tree DP problems listed. Understand rerooting technique. This opens up a whole new category of problems. Target 30-40 problems at 1500-1800 rating.

Month 5: Digit DP Master digit DP completely. This technique is specific but appears regularly. Solve all digit DP problems in the advanced section. Practice forming digit DP states quickly. Target 25-30 problems at 1600-1900 rating.

Month 6: Advanced Optimization Learn Convex Hull Trick, Divide and Conquer DP, and Knuth Optimization. These techniques optimize $O(n^3)$ to $O(n^2)$ or $O(n^2)$ to $O(n \log n)$. Solve problems requiring these optimizations. Target 20-25 problems at 1700-2000 rating.

Months 7-8: SOS DP and Mastery Master SOS DP (Sum Over Subsets). Practice on mixed DP problems from contests. Participate in virtual contests regularly. Focus on solving DP problems quickly. Target 1800+ rated problems.

Study Approach

For Interviews

The key to mastering DP for interviews is recognizing patterns. When you see a problem, ask yourself:

- Can this be broken into smaller subproblems?
- Do subproblems overlap?
- What state do I need to track?
- What's the recurrence relation?

Start with recursion. Write a recursive solution first, even if it's inefficient. Then identify what you're computing repeatedly and add memoization. Finally, convert to bottom-up tabulation if needed.

Practice the three-step approach:

1. Identify the pattern (1D DP, knapsack, LCS, etc)
2. Define state and write recurrence relation
3. Implement (preferably bottom-up for interviews)

Don't just solve problems - after solving, think about:

- Why this state definition works
- Could you define state differently?
- What's the time and space complexity?
- How would you explain this to an interviewer?

For Competitive Programming

For CP, you need speed and the ability to handle novel DP formulations. Build a strong template library. Practice identifying DP in problems that don't obviously look like DP.

Key skills to develop:

- Fast pattern recognition from problem statements
- Ability to formulate DP states for new problems
- Quick implementation of both top-down and bottom-up
- Understanding when optimization techniques are needed

For each new DP technique, solve at least 10-15 problems until it becomes second nature. Participate in virtual contests and upsolve all DP problems you couldn't solve during the contest.

Read editorials even for problems you solved - you'll often find better state definitions or optimizations you didn't think of.

Advanced CP tips:

- Learn to spot when $O(n^2)$ needs optimization
- Master space optimization techniques
- Understand amortized complexity analysis
- Practice formulating DP for game theory problems

Common DP Patterns Summary

1D DP - Single parameter state, usually index. Climbing stairs, house robber.

2D DP - Two parameter state. Grid problems, string matching.

Knapsack - Optimization with capacity constraint. Subset sum, partition problems.

LCS/LIS - Subsequence problems. Edit distance, longest common subsequence.

Interval DP - Breaking interval into smaller intervals. Matrix chain multiplication, burst balloons.

State Machine DP - Tracking current state. Stock problems, state transitions.

Tree DP - DP on tree structure. Independent set on trees, tree distances.

Bitmask DP - Using bits to represent state. Traveling salesman, assignment problems.

Digit DP - Counting numbers with certain properties. Count numbers in range satisfying conditions.

Probability DP - Computing probabilities. Expected value problems.

Game Theory DP - Optimal play in games. Nim game variations.

DP on DAG - DP on directed acyclic graphs. Longest path, counting paths.

Identifying DP Problems

A problem is likely DP if:

- It asks for optimal value (minimum/maximum)
- It asks for number of ways to do something
- It involves making choices at each step
- Current decision affects future decisions
- Problem has overlapping subproblems

Keywords that often indicate DP:

- "Minimum/Maximum cost/sum/length"
- "Count number of ways"
- "Is it possible to..."
- "Longest/Shortest subsequence/substring"
- "Optimize" or "best strategy"

Not every optimization problem is DP though. If the problem has a greedy choice property without needing to consider all subproblems, greedy might be better.

Final Thoughts

Dynamic Programming seems intimidating because problems look different on the surface. But underneath, they follow clear patterns. The key breakthrough for me was realizing that I didn't need to memorize solutions - I needed to recognize patterns.

For interviews, focus on the core patterns: 1D DP, 2D DP, Knapsack, LCS, and LIS. These five patterns cover 80% of interview DP problems. Solve 60-80 problems across these patterns and you'll be well-prepared for any tech interview.

For competitive programming, you need depth. Master the basics first, then move to advanced topics like bitmask DP, digit DP, and DP optimizations. Consistent practice over 6-8 months will get you to a strong level. Participate in contests regularly - you learn the most from upsolving problems you couldn't solve during the contest.

The progression from struggling with DP to mastering it takes time. Don't get discouraged if problems feel hard initially. Every expert was once a beginner. The difference is they kept solving problems consistently.

Some practical advice from my journey:

- Solve problems in pattern groups, not randomly
- After solving, try to solve the same problem again after 3 days
- Read multiple solutions for the same problem
- Explain your solution out loud as if teaching someone
- Don't look at solutions too quickly - struggle for 30-40 minutes first
- Keep a notebook of state definitions for different patterns

Remember: DP is a skill that compounds. Each problem you solve makes the next one easier. The first 20 problems are hardest. After 50 problems, you'll start seeing patterns clearly. After 100 problems, DP becomes one of your strong topics.

One more thing - don't compare your progress with others. Some people click with DP quickly, others take longer. What matters is consistent practice and genuine understanding, not how fast you get there.

Good luck with your DP journey! The struggle is worth it. Once you master DP, you unlock a whole new level of problem-solving ability.

Quick Reference

Interview DP Checklist:

- Can code basic recursion to DP conversion
- Understand memoization vs tabulation tradeoffs
- Master 1D and 2D DP patterns
- Comfortable with knapsack variations
- Know LCS and LIS cold
- Can explain time and space complexity
- Solved 60-80 problems across all patterns
- Can identify DP pattern in 2-3 minutes
- Comfortable implementing in interview setting

CP DP Checklist:

- All AtCoder Educational DP problems solved
- All CSES DP problems solved
- Comfortable with bitmask DP
- Understand digit DP formulation
- Know tree DP techniques
- Learned at least one DP optimization (CHT or D&C)
- Solved 150+ DP problems
- Can handle 1600-1800 rated DP problems
- Fast implementation with templates
- Regular contest participation

When You're Stuck:

1. Can you break the problem into smaller subproblems?
2. Do subproblems overlap?
3. What state do you need to track?
4. Write recursive solution first
5. Add memoization
6. Convert to bottom-up if needed
7. Check for space optimization opportunities

Common Mistakes to Avoid:

- Jumping to hard problems too quickly
- Not understanding recursion properly first
- Memorizing solutions instead of patterns
- Skipping the "thinking" phase
- Not considering base cases carefully
- Incorrect state transitions
- Not optimizing space when possible
- Giving up too soon on hard problems

Additional Resources

Practice Platforms:

- LeetCode - Best for interview prep, excellent discuss section
- Codeforces - Best for CP, active community
- AtCoder - Highest quality educational problems
- CSES - Clean, well-tested problems
- SPOJ - Classic problems, good for practice

Communities:

- LeetCode Discuss - Great for interview discussions
- Codeforces Blogs - Advanced DP techniques
- r/leetcode - Reddit community for interview prep
- r/competitiveprogramming - Reddit for CP

YouTube Channels to Follow:

- NeetCode for interview patterns
- Striver for comprehensive coverage
- Errichto for advanced CP techniques
- William Fiset for algorithm theory
- Abdul Bari for conceptual clarity

When to Use DP vs Other Approaches:

- DP: Optimization with overlapping subproblems
- Greedy: Optimization with greedy choice property
- Divide and Conquer: Independent subproblems
- Backtracking: Exploring all possibilities with constraints
- Graph Algorithms: Path/connectivity problems

What's your current DP level? Are you just starting or already comfortable with the basics? Share your struggles or wins in the comments. I'd love to help or learn from your experience!

Happy coding and keep grinding those DP problems!