# Dynamic Programming

|  |  |  |
| --- | --- | --- |
| S. no. | Type of Question | # Alternatives |
| 1 | 0-1 Knapsack | 6 |
| 2 | Unbounded Knapsack | 5 |
| 3 | Fibonacci | 7 |
| 4 | Longest common subsequence | 15 |
| 5 | Longest increasing subsequence | 10 |
| 6 | Kadane’s Algorithm | 6 |
| 7 | Matrix chain multiplication | 7 |
| 8 | DP on trees | 4 |
| 9 | DP on grid | 14 |
| 10 | Others | 5 |
|  |  | **79** |

## 0-1 Knapsack

|  |  |
| --- | --- |
| S.no. | Question |
| 1 | Subset Sum |
| 2 | Equal sum partition |
| 3 | Count of subset sum |
| 4 | Minimum subset sum difference |
| 5 | Target sum |
| 6 | No of subset with given difference |

# Longest Common Subsequence

|  |  |
| --- | --- |
| S.no. | Question |
| 1 | Longest common substring |
| 2 | Print LCS |
| 3 | Shortest common subsequence |
| 4 | Print SCS |
| 5 | Min no of insertion and deletion a -> b |
| 6 | Longest repeating subsequence |
| 7 | Length of largest subsequence of ‘a’ which is a substring in ‘b’ |
| 8 | Subsequence pattern matching |
| 9 | Count how many times ‘a’ appear as subsequence in ‘b’ |
| 10 | Longest palindromic subsequence |
| 11 | Longest palindromic substring |
| 12 | Count of palindromic substring |
| 13 | Min no of deletion in a string to make it a palindrome |
| 14 | Min no of insertion in a string to make it a palindrome |

3 things to see while thinking recursion solution:

1. Smaller input
2. Base case
3. Choice diagram

How to check if a question is of DP or not?

1. Choice – options to choose from
2. Optimal -> Max, Min, Longest, shortest , etc.

Always First think of recursive solution

Then, convert that recursive solution to top down

How to check if 2 questions are similar. Use **Pattern Matching Algo:**

1. Compare I/P, Ques, O/P
2. If u get 2/3 or 3/3, then same ques.

Memoization -> Recursion + matrix

Top - Down -> Matrix only

TO Check->

1. Min no of Insert, delete and replace to make string a-> string b
2. Print shortest common super sequence