algorithmica
re-engineering it talent

Top-20 Training Program
(DP Thinking)

Apply the recursive thinking discussed in class and check whether we need memory(dynamic programming) technique to solve the following problems.

Problem1: Implementing Recurrence Equation

Consider the following recurrence relation whose base cases are T(0) = T(1) = 1 and for n>1,

$$T(n) = \sum_{i=1}^{n-1} T(i)T(i-1)$$

a) Do you think direct recursive implementation of above equation is efficient in terms of time? If not, why?

b) Write a bottom-up dynamic programming solution to compute T(n) efficiently. What are the time and space complexities of your solution?

Problem2: Implementing Recurrence Equation

For n>0, consider the following recurrence relation to compute M(i, j) for $1 \le i \le j \le n$:

$$M(i,j) = \begin{cases} 1 & i = j, \\ 2 & j = i+1 \\ M(i+1,j-1) \cdot M(i+1,j) \cdot M(i,j-1) & j > i+1. \end{cases}$$

a) Do you think direct recursive implementation of above equation is efficient in terms of time? If not, why?

b) Write a bottom-up dynamic programming solution to compute M(1,n) efficiently. What are the time and space complexities of your solution?

Problem3: Number of Legal Paths

Given an n by n array whose entries are either `.' or `+'. A legal path from position (1,1) (which is the upper left corner) to position (n,n) (lower right corner) is any path that always goes right from the current array point or down from the current point but avoids entries which contain a `+'. If we denote a path using (x,y) coordinates then going along a legal path must always either increase the x or the y coordinate by one. Write an efficient function to compute the number of legal paths from (1,1) to (n,n). What are the time and space complexities of your solution?

Top-20 Training Program
(DP Thinking)

Problem4: Maximal Sum without Consecutive Numbers

Given an array of n positive numbers, write an efficient function to select a subset of the numbers with maximal sum and such that no two consecutive numbers are selected. Your function should return the maximal sum only. What are the time and space complexities of your solution?