# Dynamic programming

Lesson in recycling

## Quote

Those who can’t remember the past are condemned to repeat it!

- Writes down "1+1+1+1+1+1+1+1 =" on a sheet of paper.
- "What's that equal to?"
- Counting "Eight!"
- Writes down another "1+" on the left.
- "What about that?"
- “Nine!" " How'd you know it was nine so fast?"
- "So you didn't need to recount because you remembered there were eight! Dynamic Programming is just a fancy way to say remembering stuff to save time later!"

The intuition behind dynamic programming is that we trade space for time, i.e. to say that instead of calculating all the states taking a lot of time but no space, we take up space to store the results of all the subproblems to save time later.

## Wikipedia Definition

“In computer science, mathematics, management science, economics and bioinformatics, dynamic programming (also known as dynamic optimization) is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions.”

## Classic use cases

Problems that:

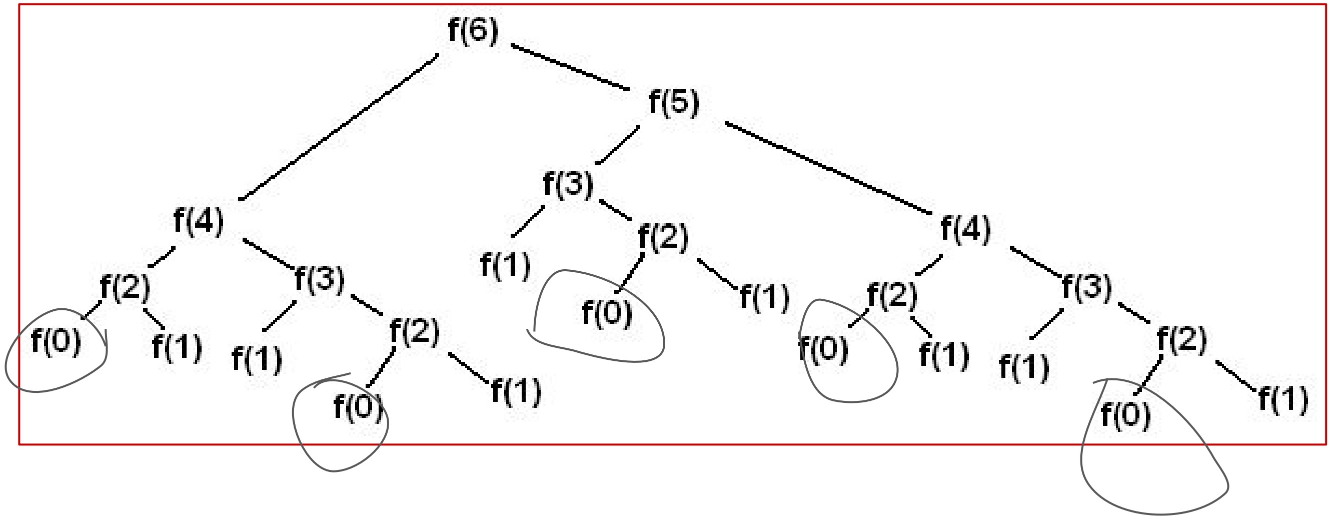
- Can split a problem into smaller sub problems
- And the subproblems can be reused

## Methods to solve

- Intuitive way of pattern recognition
  - Maintain the pattern structure in a list of lists (a data structure)
- Recursion
  - A function calling itself
  - Split a problem into subproblems and recognise the pattern
  - How do you identify which row you are sitting in a stadium?
  - rownumber = RowNumber(seating position) is the function, You pass the news that you want the row number to all the rows below you and the person in the first row returns saying row number “1” and all others return the row number by adding “1” to it, all the way up to you.
  - https://youtu.be/TSjKVb8yL8E
- Memoization where we store the values calculated
  - Some cases in recursion where values can be stored
- Tabulation
  - Calculate from bottom up

## Fibonacci series

- 0 1 1 2 3 5 8
-



## Fibonacci

- Problem can be split into sub problems
- Memoization
  - Cache the F(0) and F(1) - do not calculate it every time
- Tabulation
  - Calculate F(0) and F(1) and store it and calculate it upwards

## Pascal’s triangle

- Code for all the approaches
- Discuss

## Now solve Fibonacci!

- Try all the approaches

## Other classic problems

- Longest common subsequence “ABCDGH” and “AEDFHR” is “ADH”
    - Used in finding the difference between two files
- Knapsack problem: Consider maximum capacity for a bag and you have weights for N items and their corresponding Values. The values need to be maximized
    - Used by download managers. Data is broken into chunks and value of the download is maximized
- Coin change problem: Get 5 Rs with 1/2/3/4 Rs coins. A common problem
- Many more! Read up!

## Classic ones being used

- Unix diff for comparing two files
- Bellman-Ford for shortest path routing in networks
- TeX the ancestor of LaTeX
- WASP - Winning and Score Predicton