Recursion Quickstart Guide

In this mini-workbook, we'll get you up and running with recursion. Recursion is a fundamental concept in computer science and a precursor to more involved algorithms such as dynamic programming so it is important to deeply understand the inner workings of recursion.

In this guide, we will quickly highlight the key concepts that you need to know for your interview. If you're unfamiliar with any of these, I highly recommend taking some time to review them.

Then, I've provided you with a series of daily exercises to help you get up and running quickly. I recommend using these similarly to how you use your daily workbook. Do one set of exercises each day and be sure to log them in your tracker. This will help you ensure that you've reviewed all the most important points.

For more information on recursive patterns, see this post.

Key Recursion Terminology

- **Basic Recursion Terms**
  - Base case
  - Recursive step
  - Tail recursion
  - Backtracking

Key Recursion Patterns

- **Recursion Fundamentals**
  The foundations of recursion.
- **Iteration**
  iterate over a variety of data structures using recursion, both in one and multiple dimensions.
- **Subproblems**
  We'll go over the most fundamental pattern in all of recursion: Subproblems.
- **Selection**
  The selection pattern is one of the most commonly occurring patterns that you will see in your interview and is fundamental to understanding dynamic programming.
- **Ordering**
  Learn how to use permutations to solve many common recursive problems.
- **Divide & Conquer**
  These are two common subpatterns that will be critical to your interview success.

Daily Exercises

Day 1

Iteration pattern: Iterate over an array or list using recursion. This is rarely useful except in the circumstances of simplifying code.

- `factorial(int n)`
  Calculates the factorial of a given integer "n".
- `find_uppercase_recursive(string s)`
  Finds the uppercase letters in a given input string "s". (LucidProgramming Video)
- `recursive_str_len(string s)`
  Finds the length of a given input string "s". (LucidProgramming Video)
- `recursive_count_consonants()`
  Counts the number of consonants in a given input string "s". (LucidProgramming Video)

- `recursive_multiply(int x, int y)`
  Compute the product of two integers “x” and “y” (LucidProgramming Video)

Day 2

Breaking into Subproblems pattern: This includes all of the “classic” recursive problems. Technically all recursive-problems, but many are not obvious.

- Towers of Hanoi
- Fibonacci
- All recursive problems

Day 3

Selection (Combinations) Pattern: Fundamentally, problems that can be solved by finding all valid combinations. Optimize by validating as we go/backtracking

- Knapsack problem
- Word break
- Phonespell
- N Queens

Day 4

Ordering (Permutations) Pattern: The Ordering Pattern is similar to the Selection Pattern except now, order matters.

- Find all permutations of inputs
- Find all N-digit numbers whose digits sum to a specific value
- Word squares

Day 5

Divide and Conquer Pattern: This pattern is common with searching, sorting, and tree data structures. In this pattern, we ask whether we can solve the problem for each half of the input and easily combine the results.

- Mergesort
- Generate all BSTs for a set of items
- Checking whether a BST is valid (LucidProgramming Video)
- Calculate Height of BT (LucidProgramming Video)
- Calculate Size of BT (LucidProgramming Video)
- Find all valid parentheses

Exercises Solutions

Coming soon!