Syllabus: Practical Data Structures and Algorithms

Choose data structures, write Python solutions to LeetCode problems, and evaluate correctness and computational cost.

Course design and notes: Sunwoo Kim · Introductory · no CS background required

Course purpose

Define the inputs and required operations of a problem, choose a data structure, and write a Python solution. Trace the execution on small examples, check boundary cases, and evaluate how time and memory use change with input size.

LeetCode problems provide practice in searching, grouping, processing sequences, and following dependencies. Correctness, computational cost, and the conditions required to reuse a method in a work task are assessed separately.

Sunwoo Kim designed this eight-week, self-paced course for learners without a computer science background. Original explanations, worked readings, and Python solutions accompany official problem statements and documentation.

Audience and prerequisites

The course is for researchers, engineers, and analysts who need to process data or automate tasks. Basic Python variables, conditionals, loops, and functions are expected. Node references, recursion, and complexity notation are introduced when needed; prior algorithms study and advanced mathematics are not required.

Learning outcomes

By the end of the course, learners should be able to:

1. Define a problem's inputs, outputs, allowed operations, and constraints.
2. Choose lists, dictionaries, sets, stacks, queues, or heaps for the required operations.
3. Write and trace Python solutions using search, traversal, greedy rules, and one- and two-dimensional dynamic programming.
4. Explain time and space costs, accounting for sorting, copying, and recursive calls where relevant.
5. Check boundary cases and compare results with a simple reference solution on small inputs.
6. Identify the assumptions of a solution and revise its input handling and tests before applying it to work data.

Eight-week sequence

Week	Topic and scope	Required output
1	**Hashing and two pointers.** Two Sum; Contains Duplicate II; Two Sum II - Input Array Is Sorted; 3Sum	Python solutions, state traces, reasoning, cost analysis, and boundary checks.
2	**Sliding windows and stacks.** Longest Substring Without Repeating Characters; Minimum Size Subarray Sum; Min Stack; Evaluate Reverse Polish Notation	Python solutions, state traces, reasoning, cost analysis, and boundary checks.
3	**Binary search and linked lists.** Search a 2D Matrix; Search in Rotated Sorted Array; Reverse Linked List; Merge Two Sorted Lists	Python solutions, state traces, reasoning, cost analysis, and boundary checks.
4	**Tree recursion, BFS, and BSTs.** Maximum Depth of Binary Tree; Binary Tree Level Order Traversal; Validate Binary Search Tree; Kth Smallest Element in a BST	Python solutions, state traces, reasoning, cost analysis, and boundary checks.
5	**Heaps, intervals, and graph traversal.** Kth Largest Element in an Array; Merge Intervals; Number of Islands; Rotting Oranges	Python solutions, state traces, reasoning, cost analysis, and boundary checks.
6	**Dependencies and backtracking.** Course Schedule; Course Schedule II; Subsets; Combination Sum	Python solutions, state traces, reasoning, cost analysis, and boundary checks.
7	**One- and two-dimensional DP.** House Robber; Coin Change; Unique Paths; Longest Common Subsequence	Python solutions, state traces, reasoning, cost analysis, and boundary checks.
8	**Greedy reasoning and mixed assessment.** Jump Game; Jump Game II; Gas Station; Number of Islands	Python solutions, state traces, reasoning, cost analysis, and boundary checks.

Core sources

- Erik Demaine and Srini Devadas, **Introduction to Algorithms, MIT 6.006** (Fall 2011).