DSA Roadmap (Kunal Kushwaha)

i) chose programming language

ii) control flow statements

iii) Data type
- → i/p o/p
- → syntax
- → language standard functions
- → Basic problems
- → arrays
- → vectors
- → Linear and binary search
- → sorting
  selection sort, Bubble sort, Insertion sort
  count sort
- → Do question on leetcode
- → rotating arrays
- → Interview problems on array & array list.
- → number theory and bit masking
- → eucalydian algorithm
- → c of eratthemesis
- → bit masking operator.
- → Interview problem of bit masking.
- → Steings (Introduction)
- → memory management (stack and heap).
- → Steing manipulations
- → questions on steing manipulation on leetcode

→ complexity analysis - space & time complexity analysis
  → best case
  → worst case
  → average case
  →

→ sandwitch theorem
→ recurrence relation
→ big O notation, omega notation, theta notation
→ understanding complexity of various loops
→ understand auxillary space vs total space
→ understand static and dynamic memory allocation
→ Learn about Recursion & control flow
→ Understand Recursive tree
  → meege sort
  → quick sort
→ Do interview problems on this topics
  → maze path
  → backtracking problem
  → sudoku solver
  → Lexo permute
  → permutation
  → Rat in maze
  → strings & recursion
  → backtracking problems
  →

→ Learn OOP

→ Learn about classes and objects
→ constructors
→ Destructors
→ behaviours
→ Learn about object reference keywords like "this".
→ Learn core OOP concepts
    → abstraction
    → Inheritence
    → polymorphism
    → encapsulation
    → access modifiers
    → static and non-static fields

→ make projects in this (functions) like → railway reservation
    → Ticket booking

DSA
→ Stacks and queus
    → stacks with arrays
    → stacks with linked lists
    → operation in stacks
    → push efficient and pop-efficient stacks

→ ~~Ques~~ queus
    → generic collection
    → abstract data types
    → Operations in Queue
    → queue using two stacks
    → Interview problems on stacks & queus on leetcode

- Linked list
  - Introduction to linked list
  - Implementation
  - same topics as linked list
  - cycle detection algorithms
  - fast and slow pointers
  - solve problems on leetcode
- Binary tree
  - Implementation of binary tree
  - tree traversals
  - preorder | Inorder | post-order
- searching algorithm
  - Binary search tree
  - time complexity and space complexity of this
- avl trees → self balancing binary tree
  - rotations
  - problems on trees and binary search tree
- Heaps and Hash Maps
  - Implementation of heap
  - time complexity of heap
  - Learn about priority
  - heap sort (sorting algo)
  - concept of hashing
  - various ways of implementing hashing

→ hashmap implementation using linked list, array list
→ collision detection
→ open chain addressing
→ types of maps in that particular language
→ problems on hash maps and heaps
→ time interval problems

→ Graphs
→ Basics of Graphs
→ Terminologies of Graphs
→ edges and vertices
→ implement graphs like edge list implementation like
→ make list of all the edges
→ adjacency list implementation
→ adjacency map implementation
→ searching algorithms in Graphs dfs and vfs
→ table implementation
→ connected components, cycle detection, path finding algorithm like cruskal and prims for minimum spanning tree
→ minimum spanning tree
→ path finding
→ dijkstra algorithm
→ bellman ford
→ negative weights

→ Do problems on leetcode
→ topological sort

→ * Dynamic Programming (Important)

You have to be clear with the recursion and functions concept clear in dynamic programming.

→ Basic concepts of DP
→ Overlapping subproblems
→ memoization approach v/s tabulation approach
→ top down approach and bottom up approach
→ zero-one knapsack problem
→ wildcard pattern problem
→ egg drop problem
→ matrix chain multiplication
→ confidence :- Solve problems

Approach to solve leetcode problems mostly DP problem

→ Don't give more than 40 min to one problems
→ chose one problem, try to solve it for 15 mins, If difficult to solve then look for hints if you are not able to solve them yet look at the final solution then come back to that problem after one week

→ Dynamic Programming Question
→ Do recursion solution
→ Do recursion + dp solution
→ also calculate space and Time complexity
→ convert it into iterative solution (tabular form) → write it in words
→ Iteration + space optimised

If Good + truly dedicated = 3/4 Months

Note: Regularly and daily practice problem.