Masterclass in Data Structure and Algorithms
The course contents

- **Part 0: Fundamental Techniques:**
  - Dynamic memory allocation: malloc(), calloc(), realloc(), free(), Dynamic array of built in data types, Dynamic array of structures.
  - Recursion: Theory of recursive functions, designing recursion, single level recursion, multi-level recursion.

- **Part 1: Searching and Sorting Algorithms:**
  - **Sorting algorithms**:
  - Insertion sort: non-recursive O(n^2) algorithm
  - Merge sort: recursive O(n.lg(n)) algorithm
  - Quick sort: recursive O(n.lg(n)) algorithm
  - Heap sort: recursive O(n.lg(n)) algorithm
  - Bucket sort: O(n) algorithm used to sort uniformly distributed input over a range
  - Shell sort: A variation of insertion sort. O(n^2) algorithm
  - Counting sort: Sort values between specific range O(n+k) algorithm.
  - Radix sort
  - **Searching algorithms:** Linear Search
  - Binary Search
  - More advanced search techniques will be covered with the help of data structures

- **Part 2: Fundamental Data Structures**
  - **Basic Data Structures:** Array as a data structure.
  - Array Based Data Structures:
    - Stack,
    - Queue,
    - Dequeue,
    - Priority Queue.
  - Dynamically resizable array: vector.

  - Dynamic multi-dimensional arrays of any magnitude (m x n 2D dynamic array), (m x n x p 3D dynamic array)
  - Matrix Operations: Addition, subtraction, multiplication and inverse of m x n matrices.
  - Linked Lists:
    - Singly Linked List,
    - Singly Circular Linked List,
    - Doubly Linked List,
    - Doubly Circular Linked List
  - List Based
    - Stack,
    - Queue,
    - Dequeue,
    - Priority Queue.
  - Dynamic disjoint sets
  - Hash tables
  - Basic Indexing Techniques

- **Part 3: The World of Trees**:
  - **Binary Search Tree**:
    - Create(),
    - Insert(),
    - Search(),
    - Remove(),
    - Destroy(),
    - recursive preorder recursive,
    - recursive postorder recursive,
    - recursive inorder traversal,
    - nonrecursive preorder traversal,
    - nonrecursive postorder traversal,
    - nonrecursive inorder traversal,
    - predecessor,
    - successor

  - **Height & Weight Balanced Trees:**
    - **Red Black Tree:**
      - Height balanced tree based on coloring

        - tree rotations-left rotate, right rotate, insert (), delete()
        - Rest routines are that of BST.
      - **AVL Tree:**
        - Height balanced tree based on node height
        - Tree rotations – left rotate, right rotate
        - Rest routines are that of BST.
      - **Radix Tree:**
        - A space optimized prefix tree (also known as radix trie)
        - Insertion (), deletion(), lookup()
      - **B Tree:**
        - A data structure for efficient secondary storage
        - Insertion(), deletion(), search(), split_child()
      - **Leftist tree:** Efficient priority queue implementation, variant of binomial heap.

  - **Tree Structures for the sets of intervals**:
    - **Interval Trees:** augmented height balance binary search tree. Insert(), overlap(), overlap_search(), remove()
    - **Segment Trees:** Construction of segment trees, update(), query()
  - **Heaps:**
    - **Binomial Heap:** min heap property, max heap property, extract_min(), extract_max(), insert(), decrease_key(), remove()
    - **Fibonacci Heap:** make-heap(), Insert(), inimum(), extract_min(), union(), decrease_key(), delete()

- **Part 4: The World of Graph:**
  - **Graph management algorithms:**
    - Create_graph()
    - Add_vertex()
    - Add_edge()
    - Remove_vertex()

        - Remove_edge()
        - Print_graph()
        - Destroy_graph()
    - **Graph Traversal Algorithms:**
        - Depth First Search
        - Breadth First Search
    - **Shortest Path Algorithms:**
        - Dijikstra's shortest path algorithm
        - Bellman Ford Algorithm
    - **Minimum Spanning Tree:**
        - Prim's Algorithm
        - Kruskal's Algorithm

- **Part 5: Design of Algorithms: Algorithm Design Strategies:**
    - **Divide and Conquer:** General solution, case studies with merge sort, quick sort.
    - **Greedy:** General Solution, case study with Prim's and Kruskal
    - **Dynamic Programming:**
        - Introduction through ROD cutting problem.
        - recursive top-down implementation
        - Bottom up cut rod
        - memoized cut rod
        - **Elements of dynamic programming:**
            - Optimal substructure
            - Overlapping subproblems
            - Reconstructing optimal solution
            - Longest common subsequence
            - Optimal Binary Search Tree
        - Applying dynamic programming:
            - Matrix chain multiplication using
            - Longest common subsequence
    - **Backtracking:** General strategy and case study with 8 queens' problem.
    - **Introduction and solution to some well-known algorithmic problems:**

        - 0/1 knapsack,
        - Hamiltonian graph problem,
        - Satisfiability problem.

- **Part 6: Analysis of algorithms:**
    - Time complexity.
    - Asymptotic notation to measure time complexity
        - Big Theta
        - Big O
        - Big omega
        - Small O
        - Small omega
- Time Complexity of Non-recursive algorithms
    - Computing Complexity of non-recursive algorithms via step counting
    - Converting step counting output into asymptotic notation
    - Time Complexity of Recursive algorithms
- Computing Complexity of recursive algorithms
    - Step counting
    - Generate RECURSIVE EQUATION
    - Solving RECURSIVE EQUATION
    - Converting solution to recursive equation into asymptotic notation
- Theory of recursive equations
    - Master Theorem
    - Other forms of recurrence relations & solving techniques