Lecture 1:

1. Insertion Sort Algorithm (code, example, dry run)
2. Insertion Sort time complexity analysis
3. Best case, worst case analysis

Lecture 2:

1. Asymptotic Notations
2. Big-Oh, Big-Omega, Big-theta

Lecture 3:

1. Divide and Conquer
2. Merge Sort Algorithm (example, code, dry run to understand recursive calls)
3. Merge Sort time complexity analysis
4. Solving Recursions (Tree Method)

Lectures 4 &5:

1. Merge Sort space complexity analysis
2. Insertion Sort space complexity analysis (In-place algorithm)
3. Loop invariants (proof of correctness):
    a. Sum of array
    b. Insertion Sort
    c. Merge Sort
4. Solving Recursions:
    a. Tree method
    b. Back Substitution/Iterative method

Lecture 6:

1. More Recursions Practice:
    a. Tree method
    b. Back Substitution/Iterative method
2. Maximum subarray sum problem:
    a. Brute force algo 1 — O (n^3)
    b. Brute force algo 2 — O (n^2)
    c. Divide and conquer approach (crossing sub-array case)

Lecture 7:

1. Maximum subarray sum problem:
    a. Complete Divide and Conquer solution
    b. Complexity analysis
2. Quick Sort
    a. Algorithm (using Divide and conquer)
    b. Space Complexity Analysis (in-place algorithm)
    c. Time Complexity Analysis

    d. Best Case
    e. Worst Case
    f. Average case
    g. Tight bounds of average case of Quick Sort
    h. Considering first element as pivot
    i. Loop Invariant

Lecture 8:

1. Quiz 1
2. Solving Recursions:
    a. Difference between Substitution and Back Substitution (Iterative) Method
    b. Substitution Method (with Mathematical Induction)
    c. Master Theorem
    (LEARN TO SOLVE EQUATIONS!!!)

Lecture 9:

1. Extended Master Theorem
2. Counting Inversions:
    a. Brute Force Solution
    b. Divide and Conquer Solution
    c. Time Complexity Analysis

Lecture 10:

1. Heap Sort:
    a. Heapify (algo + analysis)
    b. Build_Heap (algo + loose upper bound + tight upper bound)

Lecture 11 & 12:

1. Lower bound of worst case of comparison based sorting ($h = \Omega(n\lg n)$), Decision Tree for Insertion Sort with 3 elements.
2. Comparison based sorting Vs Linear Sorting (count based)
3. Concept of stable vs unstable sorting algorithms
4. Counting Sort (algo + analysis)
5. Radix Sort (HW: complete the code of radix sort)

Lecture 13:

1. Fibonacci numbers - Recursive Solution, time complexity analysis)
2. Fibonacci numbers - Dynamic Programming, overlapping sub-problems, time complexity
3. Steps of Dynamic Programming
4. Maximum subarray sum - brute force solution, time complexity

5. Maximum subarray sum - Dynamic Programming (Kadane's Algorithm - gives optimal value).
6. Time complexity Analysis
7. HW:
    a. Dry run
    b. modify the code to find optimal solution

Lecture 14:

1. 0/1 Knapsack Problem - brute force solution
2. Overlapping sub-problems, optimal substructure, recursive formula/definition
3. Dynamic Programming solution (gives optimal value)
4. Time complexity analysis
5. HW:
    a. Dry Run
    b. modify the code to find optimal solution

Lecture 15:

1. Rod cutting problem - recursive solution, time complexity
2. Overlapping sub-problems
3. Optimal substructure (Proof by Contradiction)
4. Recursive Definition
5. Dry Run
6. Dynamic Programming Solution (gives optimal value)
7. Time complexity Analysis
8. HW:
    a. Complete Dry Run
    b. Modify code to find optimal solution

Lecture 16:

1. Longest Common Subsequence
2. Optimal Substructure and over lapping sub-problems
3. Smaller Sub-problems
4. Recursive definition
5. DP Algorithm of LCS
6. Time complexity
7. Dry run

Lecture 17:

1. Quiz 2
2. What are Greedy Algorithms?
3. Greedy Algorithms vs. Divide and Conquer
4. Greedy Algorithms vs. Dynamic Programming

5. Minimum Coin Change Problem
6. Activity Selection problem (3 greedy strategies)
7. How to construct a counter example for a greedy strategy
8. HW: Give counter example for the 2nd greedy strategy