1. Determine the cost and structure of an optimal binary search tree for a set of N=6 keys with the following probabilities:

Integers	1	2	3	4	5	6
Probabilities	0.01	0.02	0.04	0.08	0.16	0.69

2. Knuth has shown that here are always roots of optimal sub trees such that
   root[i , j-1] ≤ root[i , j] ≤ root[i+1 , j] for all 1 ≤ i < j ≤ n
   Use this fact to modify the OPTIMAL-BST procedure to run in $\Theta(n^2)$ time.

3. Determine an LCS of ⟨1,0,0,1,0,1,0,1⟩ and ⟨0,1,0,1,1,0,1,1,0⟩.

4. Give an $O(n^2)$ time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers.

5. Give an O(nlogn) time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers.

6. Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible with all previously selected activities.
   Describe how this approach is a greedy algorithm and prove that it yields an optimal solution.

7. Suppose that we have a set of activities to schedule among a large number of lecture halls.
   We wish to schedule all the activities using as few lecture halls as possible.
   Give an efficient greedy algorithm to determine which activity should use which lecture hall.

8. Suppose that in a **0/1 knapsack problem**, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value.
   Give an efficient algorithm to find an optimal solution to this variant of the knapsack problem , and argue that your algorithm is correct.

9. Describe an efficient algorithm that, given a set $\{x_1, x_2, \cdots, x_n\}$ of points on the real line, determines the smallest set of unit length closed intervals that contains all of the given points.
   Argue that your algorithm is correct.

10. Suppose you are given two sets A and B , each containing n positive integers. You can choose to reorder each set how ever you like. After reordering ,let **a_i** be the "**i**"th element of setA , and let **b_i** be the "**i**" th element of setB . You then receive a payoff of $\prod_{i=1}^{n} a_i^{b_i}$. Give An algorithm that will maximize your payoff. Prove that your algorithm maximizes the payoff, and state its running time.

11. Implement the Huffman's algorithm for compressing and decompressing a file.

12. Generalize Huffman's algorithm to ternary codewords (i.e., codewords using the symbols 0 , 1 , and 2) , and prove that it yields optimal prefix-free ternary codes.

13. Prove that a binary tree that is not full can not correspond to an optimal prefix code.

14. What is an optimal Huffman code for the following set of frequencies , based on the first 8 Fibonacci numbers?

Character	A	B	C	D	E	F	G	H
Frequent	1	1	2	3	5	8	13	21

Can you generalize your answer to find the optimal code when the frequencies are the first n Fibonacci numbers?

15. Suppose we have an optimal prefix code on a set **C={0,1,⋯,n−1}** of characters and we wish to transmit this code using as few bits as possible. Show how to represent any optimal prefix code on **C** using only **2n-1+n[logn]** bits.

Good Luck 😉