CCU Undergraduate Algorithm 2025
Homework #1

1. (10%) Suppose you are asked to implement the n-th Fibanacci number using *recursion* only (i.e., no loop implementation). How to avoid stack overflow for large n? Briefly explain your solution.
2. (20%) What is the Big O complexity of following recurrence? Justify your answer.

   (a) (10%) $T(n) = 2T\left(\frac{n}{3}\right) + T\left(\frac{n}{2}\right) + cn$, where c is a constant.

   (b) (10%) $T(n) = 2T(\sqrt{n}) + \lg n$ (hint: you may replace n with other form)
3. (10%) Prove that an n-element heap has at most $\lceil \boldsymbol{n/2^{h+1}} \rceil$ nodes at height h.
4. (5%) The radix sort can achieve time complexity better than all the comparison-based sorting algorithms (e.g., merge sort, quick sort). Why do all the programming languages still implement comparison-based sorting algorithms as the build-in function (e.g., qsort in C)?
5. (10%) Consider a stick of length m meters that has to be cut at s_1, s_2, …, and s_n meters from left end, where m and s_i are all integers and the cut order of s_i can be any permutation. The cost of each cut s_i is the length of stick prior to the cut, and thus different permutation of cut order leads to different cost. Illustrate a dynamic programming algorithm for finding the minimum cut cost of a stick of length 30, which will be cut at 3, 6, 12, 17, 22, and 28 meters from left end. You should write down the recurrence, bottom-up DP computation, and backtrack one optimal solution.
6. (25%) Longest common subsequence.

   (a) (5%) Write the optimal substructure (recurrence) of computing LCS of k sequences, where $k = 3$

   (b) (10%) Given a string, find the longest subsequence occurring at least twice in the string, requiring their indices must not overlap. e.g., Given ATATAGAGGC, the answer is 4 since ATAG occurs twice and their indices (i.e., (1,2,5,6) and (3,4,7,8)) do not overlap. Describe a dynamic programming (recurrence) for the string ATTAATAT. You should show the bottom-up tabular computation.

   (c) (10%) Compute the Longest Palindrome Subsequence (LPS) in any sequence using dynamic programming. Given a string "character," the LPS is "carac." You should write down the recurrence and bottom-up tabular computation.
7. (10%) Consider the knapsack problem of n items and W pack size. Suppose the pack/item sizes are very large and the item values are very small. Give a dynamic programming (recurrence) for solving this problem. Illustrate your tabular computation using the following example (W=350).

Item	Weight	Value
1	100	1
2	150	2
3	200	4
4	300	5

8. (10%) Consider the following six activities with (start time, finish time, and value): (2, 4, 3), (5, 5, 5), (3, 4, 2), (1, 4, 3), (1, 3, 1), (3, 5, 4). Illustrate a dynamic programming algorithm for computing the mutually-exclusive subset of activities of maximum total values using the above example.