CCU Undergraduate Algorithm 2024
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*?

2. (20%) What is the Big O complexity of following recurrence? Justify your answer.

(a) (10%) $T(n) = 2T\left(\frac{n}{4}\right) + T\left(\frac{2}{3}n\right) + cn$, where *c* is a constant.

(b) (10%) $T(n) = 2T\left(\sqrt{n}\right) + \lg n$ (hint: you may replace *n* with other form)

3. (10%) Prove that an *n*-element heap has at most $\lceil n/2^{h+1} \rceil$ nodes at height *h*.

4. (5%) Mr. Stupid claims we can sort *n* real numbers in *linear* time by multiplying a large integer to each real number such that all of them become integers. Then, the counting sort can be used to sort these integers in linear time. What is the problem of this method?

5. (35%) Longest common subsequence.

(a) (10%) Write the optimal substructure (recurrence) of computing LCS of *k* sequences, where $k = 3$ (5%). What is the time complexity of computing LCS for *k* sequences of length *n* (5%).

(b) (5%) Mr. Smart claimed that the LCS of three sequences can be obtained by first computing LCSs of any two sequences (say LCS_{12}), and then compute LCS of LCS_{12} and the remaining sequence. What's the bug of this method? Give an example.

(c) (10%) Given a string, find the longest subsequence occurring at least twice in the string, requiring their indices must not overlap. e.g., Given ATACTCGAG, the answer is 4 since ATCG occurs twice and their indices (i.e., (1,2,4,7) and (3,5,6,9)) do not overlap. Describe a dynamic programming (recurrence) for this problem. Illustrate a bottom-up DP using the string ATACTCGAG.

(d) (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 computation.

6. (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 for solving this problem. Illustrate your algorithm using the following example (W=500).

Item	Weight	Value
1	100	1
2	200	2
3	250	4
4	300	5

7. (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.