Algorithm Design and Analysis
Assignment 4

1. (15 points) Given two strings $A = a_1a_2a_3a_4...a_n$ and $B = b_1b_2b_3b_4...b_n$, how to find the longest common subsequence between A and B? In particular, we want to determine the largest k, where we can find two list of indices $i_1 < i_2 < i_3 < ... < i_k$ and $j_1 < j_2 < j_3 < ... < j_k$ with $a_{i_1}a_{i_2}...a_{i_k} = b_{j_1}b_{j_2}...b_{j_k}$. Design a DP algorithm for this task.

2. (15 points) Given two teams A and B, who has already played $i+j$ games, where A won i games and B won j games. We suppose that they both have 0.5 independent probability to win the upcoming games. The team that first win $n \geq \max\{i, j\}$ games will be the final winner. Design a DP algorithm to calculate the probability that A will be the winner.

3. (20 points) Formalize the improved DP algorithm for the Longest Increasing Sequence Problem in the lecture, prove its correctness, and analyze its time complexity.

4. (20 points) Optimal Indexing for A Dictionary Consider a dictionary with n different words $a_1, a_2, ..., a_n$ sorted by the alphabetical order. We have already known the number of search times of each word a_i, which is represented by w_i. Suppose that the dictionary stores all words in a binary search tree T, i.e., each node's word is alphabetically larger than the words stored in its left subtree and smaller than the words stored in its right subtree. Then, to look up a word in the dictionary, we have to do $\ell_i(T)$ comparisons on the binary search tree, where $\ell_i(T)$ is exactly the level of the node that stores a_i (root has level 1). We evaluate the search tree by the total number of comparisons for searching the n words, i.e., $\sum_{i=1}^{n} w_i\ell_i(T)$. Design a DP algorithm to find the best binary search tree for the n words to minimize the total number of comparisons.

5. (30 points) Precedence Constrained Unit Size Knapsack Problem Let us recap the classic knapsack problem (unit size): We are given a knapsack of capacity W and n items, each item i has the same size $s_i = 1$ and is associated with its value v_i. The goal is to find a set of items S (each item can be selected at most once) with total size at most W (i.e., $\sum_{i \in S} s_i = |S| \leq W$) to maximize the total value (i.e., $\sum_{i \in S} v_i$). Now we are given some additional precedence constraints among items, for example, we may have that item i can only be selected if we have selected item j. These constraints are presented by a *tree* $G = (V, E)$, which says that a vertex can only be selected if we have selected its parent, (i.e., we need to select all his ancestors). The task is also to maximize the total value we can get, but not violating the capacity constraint and any precedence constraints.

   (a) (20 points) Design a DP algorithm for it in $O(nW^2)$ time.

   (b) (10 points) Can you improve the algorithm and analysis to make it run in $O(n^2)$ time? (When W is sufficient large like $W = \Theta(n)$, it is an improvement.)

   Tips

   - You can write down your best algorithm even if you can not get the required running time.
   - You can improve the running time to $O(nW)$.
   - You can directly finish question (b) (30 points).
   - You may use the following observation: For a subtree T, we can at most select $|T|$ items.

6. How long does it take you to finish the assignment (include thinking and discussing)? Give a score (1,2,3,4,5) to the difficulty. Do you have any collaborators? Write down their names here.