Algorithm Design and Analysis
Assignment 3
Deadline: Dec 11, 2022

1. (20 points) Usually, we want to improve the time complexity. But now, let us talk about space complexity.

   (a) (10 points) Recall the knapsack DP algorithm in the lecture, which has nW subproblems totally. We need to maintain a subproblem table with size $n \times W$. Can we use only $O(W)$ space (i.e., maintain only a $1 \times W$ size subproblem table) to implement the DP algorithm (still runs in $O(nW)$)?

   (b) (10 points) Recall the Edit Distance DP algorithm in the lecture, which has nm subproblems totally. Can we use only $O(\min\{n, m\})$ space to implement the DP algorithm (still runs in $O(nm)$)?

2. (25 points) Given a sequence of integers $a_1, a_2, ..., a_n$, a lower bound and an upper bound $1 \le L \le R \le n$. An (L, R)-step subsequence is a subsequence $a_{i_1}, a_{i_2}, ..., a_{i_\ell}$, such that $\forall 1 \le j \le \ell - 1$, $L \le i_{j+1} - i_j \le R$. The revenue of the subsequence is $\sum_{j=1}^{\ell} a_{i_j}$. Design a DP algorithm to output the maximum revenue we can get from a (L, R)-step subsequence.

   (a) (5 points) Suppose $L = R = 1$. Design a DP algorithm in $O(n)$ to find the maximum $(1, 1)$-step subsequence.

   (b) (10 points) Design a DP algorithm in $O(n^2)$ to find the maximum (L, R)-step subsequence for any L and R.

   (c) (10 points) Design a DP algorithm in $O(n)$ to find the maximum (L, R)-step subsequence for any L and R.

3. (25 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.

4. (30 points) Collecting Gift On a Grid Given n gifts located on a $(m \times m)$ grid. The i-th gift is located at some point (x_i, y_i) (integers chosen in $1 \cdots m$) on the grid. A player at $(1, 1)$ is going to collect gifts by several *Upper-Right Move*. In particular, assuming the player is currently located at (x, y), he can make one *Upper-Right Move* to another point (x', y') where $x' \geq x$ and $y' \geq y$. The cost of this movement is $(x' - x)^2 + (y' - y)^2$. The player will collect the i-th gift when he is at point (x_i, y_i). There is no restriction for the number of *Upper-Right Move* and the final location of the player.

   (a) (15 points) Design an $O(m^2)$ algorithm to maximize the player's profit, i.e., the sum of value he collects minus the sum of cost he pays for his *Upper-Right Move*.

   (b) (15 points) Sometimes, n can be much smaller than m. Can you design another algorithm that runs in $O(n^2)$ for this situation?

   (c) (0 Points. It is for fun, you can discuss your idea with me.) Is there any difference if the player can only make *Upper-Right Move* among gifts? Can we still design efficient algorithm runs in $O(n^2)$, $O(nm)$, and $O(m^2)$?

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