Algorithm Design and Analysis
Assignment 4
Deadline: May 16, 2023

1. (25 points) A *palindrome* is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes are all strings of length 1, `civic`, `racecar`, and `aibohphobia` (fear of palindromes).

   Give an efficient algorithm to find the longest palindrome that is a subsequence of a given input string. For example, given the input `character`, your algorithm should return `carac`. What is the running time of your algorithm?

2. (25 points) Consider the following 3-PARTITION problem. Given positive integers $a_1, ..., a_n$, we want to determine whether it is possible to partition of $\{1, ..., n\}$ into three disjoint subsets I, J, K such that

$$\sum_{i \in I} a_i = \sum_{j \in J} a_j = \sum_{k \in K} a_k = \frac{1}{3} \sum_{i=1}^{n} a_i.$$

   For example, for input $(1, 2, 3, 4, 4, 5, 8)$ the answer is yes, because there is the partition $(1, 8)$, $(4, 5)$, $(2, 3, 4)$. On the other hand, for input $(2, 2, 3, 5)$, the answer is no. Devise and analyze a dynamic programming algorithm for 3-PARTITION that runs in time polynomial in n and in $\sum_i a_i$.

3. (25 points) Recall the TSP problem we studied in the lecture. Now we consider one of its variants. Given a non-negative weighted graph $G = (V, E, c)$, a start point $s \in V$, and a profit function $p(u)$ for each $u \in V$ (also non-negative). Unlike TSP, we are not required to go through all vertices exactly. We only need to select a subset of vertices U (other than v), design a walk from s, go through every vertex in U, and return to s. Also, different from TSP, we can go through a vertex or an edge multiple times. Your total profit is $\sum_{v \in U} p(u)$, and your total cost is the sum of $c(e)$ on every edge e you go through. The objective is to maximize your total profit minus the total cost. Design a DP algorithm for it. (Your running time can be exponential.)

4. (25 points) We are given a sequence of integers $a_1, a_2, ..., a_n$, a lower bound and an upper bound $1 \leq L \leq R \leq n$. An (L, R)-step subsequence is a subsequence $a_{i_1}, a_{i_2}, ..., a_{i_\ell}$, such that $\forall 1 \leq j \leq \ell - 1$, $L \leq i_{j+1} - i_j \leq R$. 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. (Tips: Refer to the "Priority Queue" technique of the k-largest number problem in the lecture.)

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.