**CSE421: Introduction to Algorithms** February 10, 2022

# Homework 5

*Yin Tat Lee* Due: Feb 16, 2022 (before class)

Unless otherwise mentioned, you always need to show your algorithm's runtime and prove that it outputs the correct answer. See Homework Guideline on Ed for more details.

1. (10 Marks) Show how to multiply two degree $n$ single-variable polynomials of $x$ in $O(n^\alpha)$ time for some $\alpha < 2$.

2. (10 Marks) You have a pet tortoise, Zippy, that can walk either one step or two steps or three steps at a time before he has to stop for a break. You place him in a corner of your room and keep a slice of cucumber for him to eat $n$ steps away. Write a dynamic programming algorithm to compute the number of ways in which Zippy can walk from his starting point to the cucumber.

   As an example, suppose $n = 4$. Then the number of possible ways is *seven*, as listed below.

   (a) $1 + 1 + 1 + 1$
   (b) $1 + 1 + 2$
   (c) $1 + 3$
   (d) $1 + 2 + 1$
   (e) $2 + 1 + 1$
   (f) $2 + 2$
   (g) $3 + 1$

3. (10 Marks) You are going camping with your friends and need to decide what items to take in your backpack, which has a limit to how much weight it can carry. You may choose from a set of $n$ items, where the $i^{\text{th}}$ item has $a_i$ units of utility A, $b_i$ units of utility B, and $k_i$ units of weight. You want to target certain minimum units of utilities $A$ and $B$, denoted by $\alpha$ and $\beta$, respectively. Assume that all $a_i$, $b_i$ and $\alpha$, $\beta$ are **non-negative integers**.

   Output: A subset $S \subseteq \{1, 2, \cdots, n\}$ of items with *minimum* total weight that fulfills the target requirement of each utility. More formally, your algorithm should return a subset S that *minimizes* $\sum_{i \in S} k_i$ satisfying the constraints $\sum_{i \in S} a_i \geq \alpha$ and $\sum_{i \in S} b_i \geq \beta$. You can assume that such a set always exists. This is an example of a mixed packing covering linear program, a very widely studied problem class.

   Give an algorithm to solve this problem. Note that your algorithm should return an optimal subset, not just the optimal value. The time complexity should depend polynomially on $n, \alpha, \beta$.

4. (**Extra Credit**) Consider again the knapsack problem. There are $n$ items with positive integer weights $w_i$ and positive integer value $v_i$. Our goal is to pick a subset of the items with maximum total values such that the total weight is less than $W$.

- Suppose the optimal solution has total values $V$. Give an $O(nV)$ time algorithm to solve the knapsack problem exactly. (In the class, we gave an $O(nW)$ time algorithm.)
- In practice, usually an approximate optimal solution is good enough. Give an $O(n^2/\epsilon)$ time algorithm for finding $v$ such that $(1-\epsilon)V \leq v \leq (1+\epsilon)V$. (Hints: Wikipedia)