CS 341, Winter 2022 R. Peng, A. Storjohann

ASSIGNMENT 4

DUE: Wednesday, February 9, 11:59 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read `http://www.student.cs.uwaterloo.ca/~cs341` for general instructions and policies.

Exercises. The following exercises are for practice only. You may ask about them in office hours. Do not hand them in.

1. **Change-Making.** In Lecture 6, we gave an example showing that the greedy change-making algorithm does not always work for every possible coin system. Can you design a dynamic programming algorithm to make changes with the minimum number of coins for all coin systems? More formally, given a non-negative integer $v \in \mathbb{Z}_+$ (the amount that you want to make change) and a coin system $(c_1, c_2, \dots, c_n)$ where c_i is a strictly increasing sequence of positive integers with $c_1 = 1$, design a dynamic programming algorithm to find $x_1, \dots, x_n \in \mathbb{Z}_+$ such that $\sum_{i=1}^n x_i c_i = v$ while minimizing $\sum_{i=1}^n x_i$.

Problems. To be handed in. For the two dynamic programming problems in this assignment, your solution must consist of

- A description of the subproblems you will solve and the order in which you will solve them.
- A formula for solving one subproblem in terms of previously solved subproblems, and a justification of correctness.
- Pseudocode of your algorithm.
- An analysis of run-time using big-O notation.

1. [10 marks] **Garage Sale**

   You are having a garage sale at your house and you are trying to carry items from your attic to the street as efficiently as possible. You are given a suitcase that can not carry more than a total weight W and total volume V and you want to maximize the price of the objects that you can carry with you in a single trip.

   You are given a set of objects: $a_1, a_2, ..., a_n$ of

   - prices p: $p_1, p_2, ..., p_n$
   - weights w: $w_1, w_2, ..., w_n$
   - volumes v: $v_1, v_2, ..., v_n$

   respectively. You can use each object at most once, and want to maximize the sum of prices of objects chosen while making sure the collective weights and volumes of the objects selected do not exceed W and V. All p_i, w_i, v_i, as well as W and V, are non-negative integers.

   Provide an $O(nVW)$ time dynamic programming algorithm to find maximum price of a set of objects whose total weight and volume are within the limits.

2. [10 marks] **Goose Egging**

   You have acquired a stash of k goose eggs, and the MC has n floors. There is some value x between 1 and n such that if a goose egg is dropped from a floor $\geq x$, it breaks. (in other words, it is guaranteed that the goose egg breaks when dropped from floor x or higher.) You want to find this value of x by doing the fewest number of drops.

   The only allowed operation is to drop a goose egg from some floor: if the goose egg breaks when you drop it, you cannot reuse it, otherwise it can be reused.

   Your goal is to devise a dynamic programming algorithm to determine the minimum number of drops needed to determine x in the worst case.

   (a) [1 marks] Consider the $k = 1$ case: you are given 1 goose egg for an n story building. What is the minimum number of drops you have to make from which you are guaranteed to find the lowest floor where the goose egg breaks?

   (b) [1 marks] Suppose you now have 2 goose eggs for a 7 story building, and you just dropped your first goose egg from floor 4. How many more drops are needed in the worst case?

   Note that if the goose egg breaks, floor 4 is still a candidate for x. Furthermore, you need to take worst case over whether this egg broke or not.

   (c) [1 marks] Find, with justification, a way to determine the breaking level of a 7 floor building, when given 2 eggs, using 3 drops.

   (d) [7 marks] Give a dynamic programming solution to this problem that computes the minimum number of drops needed when there are n initial floors and k goose eggs. Your algorithm should run in time $O(n^2k)$ or faster.

Challenge Questions. This is for fun and enrichment only. Do not hand it in.

1. Do Question $2(d)$ in $O(n^{1/2}k)$ time.