CS4800: Algorithms — S'18 — Jonathan Ullman

Homework 4
Due Friday February 9 at 11:59pm via Gradescope

Name:
Collaborators:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Friday February 9 at 11:59pm via Gradescope. No late assignments will be accepted. Make sure to submit something before the deadline.
- Solutions must be typeset in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly prohibited.

Problem 1. *Armageddon*

The NASA Near Earth Object Program lists potential future Earth impact events that the JPL Sentry System has detected based on currently available observations. Sentry is a highly automated collision monitoring system that continually scans the most current asteroid catalog for possibilities of future impact with Earth over the next 100 years.

This system allows us to predict that i years from now, there will be x_i tons of asteroid material that has near-Earth trajectories. In the mean time, we can build a space laser that can blast asteroids. However, each laser blast will require exajoules of energy, and so there will need to be a recharge period on the order of years between each use of the laser. The longer the recharge period, the stronger the laser blast—after j years of charging, the laser will have enough power to obliterate d_j tons of asteroid material. This problem explores the best way to use such a laser.

The input to the algorithm consists of the vectors $(x_1, \ldots, x_n)$ and $(d_1, \ldots, d_n)$ representing the incoming asteroid material in years 1 to n, and the power of the laser d_i if it charges for i years. The output consists of the optimal schedule for firing the laser to obliterate the most material.

Example: Suppose $(x_1, x_2, x_3, x_4) = (1, 10, 10, 1)$ and $(d_1, d_2, d_3, d_4) = (1, 2, 4, 8)$. The best solution is to fire the laser at times $3, 4$. This solution blasts 5 tons of asteroids, 4 tons at time 3 and 1 ton at time 4.

(a) Construct an input on which the following "greedy" algorithm returns the wrong answer:

```
1: procedure BadLaser($(x_1, \ldots, x_n), (d_1, \ldots, d_n)$)
2:     Compute the smallest $j$ such that $d_j \geq x_n$. Set $j = n$ if no such $j$ exists.
3:     Shoot the laser at time $n$.
4:     if $n > j$ then return BadLaser($(x_1, \ldots, x_{n-j}), (d_1, \ldots, d_{n-j})$)
5:     end if
6: end procedure
```

Intuitively, the algorithm figures out how many years j are needed to blast all the material in the last time slot. It shoots during that last time slot, and then accounts for the j years required to recharge for that last slot, and recursively considers the best solution for the smaller problem of size $n - j$.

Solution:

(b) Let OPT(j) be the maximum amount of asteroid we can blast from year 1 to year j. Give a recurrence to compute OPT(j) from OPT$(j), \ldots,$ OPT$(j-1)$. Justify your recurrence.

Solution:

(c) Describe a dynamic programming algorithm based on your recurrence above, including the base cases and order of evaluation. Analyze the running time of your solution. You may use either a top-down ("memoized") or bottom-up ("iterative") approach.

Solution:

Problem 2. *Box Stacking*

You are given a set of n three-dimensional rectangular boxes, with each box represented by the triple (h_i, w_i, d_i) giving the height, width, and depth of box i. Your goal is to create a stack of boxes that is as high as possible. However, you can only stack box i on top of box j if box j is larger than i in both width and depth—that is, if $w_j \geq w_i$ and $d_j \geq d_i$. Since the boxes have fragile contents, you are not allowed to rotate the boxes. You may assume for simplicity that no two boxes have the same width and depth.

In this problem, you will design dynamic programming algorithm that outputs the height of the tallest stack you can make. The input is a list $b_1, \ldots, b_n$ containing height, width, and depth for the n boxes and the output is an ordered list of the boxes you will stack that maximized the total height of the stack.

Example: Suppose the four boxes are $b_1 = (4, 1, 20), b_2 = (5, 2, 11), b_3 = (2, 10, 3), b_4 = (8, 12, 5)$ then the optimal solution is to stack boxes 3 on top of box 4, achieving a height of 10.

The different parts of the problem correspond to the different steps of designing and specifying a dynamic programming algorithm. The question is split into steps to help you structure your solution, but, like any design process, you may want to go back and forth between these steps when you are formulating your solution.

(a) If you want to sort the input as the first step of your algorithm, specify what you sort by.

Solution:

(b) Define in English, define the set of subproblems your dynamic programming algorithm will consider. That is, describe what you want the function OPT to denote.

Solution:

(c) Give a recursive formula for OPT. Give a justification of your recursive formula in English.

Solution:

(d) Explain in English in which order you intend to fill your dynamic programming table.

Solution:

(e) Give a pseudocode description of your algorithm.

Solution:

(f) State running time of your algorithm, and briefly justify your answer.

Solution: