CS3000: Algorithms & Data — Fall ’18 — Jonathan Ullman

Homework 3
Due Friday October 5 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 October 5 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 forbidden.

Problem 1. *Pacing Yourself*

You are a freelancer who has to choose which jobs to take on each of n days. On every day $i < n$ you have the option of taking either a one day job, in which case you are paid o_i, or taking a two-day job that pays a larger amount $t_i > o_i$, but will prevent you from taking another job on day $i+1$. On the final day n you only have the option of a one-day job that pays o_n. You have the list of available jobs for the next n days, and need to choose the sequence of one-day and two-day jobs that earns the most total money.

The input to the algorithm is thus a sequence of $2n-1$ positive numbers

$$o_1, t_1, o_2, t_2, \ldots, o_{n-1}, t_{n-1}, o_n.$$

The output should be an optimal sequence of jobs to maximize total earnings, subject to the constraint that if you select a two-day job and earn t_i on day i, then you cannot take any job on day $i+1$.

(a) Let $\mathrm{OPT}(j)$ be the maximum amount of money you can earn for days $1, \ldots, j$. Give a recurrence to compute $\mathrm{OPT}(j)$ from $\mathrm{OPT}(1), \ldots, \mathrm{OPT}(j-1)$. Give a few sentences justifying why your recurrence is correct.

Solution:

(b) Using your recurrence, design a dynamic programming algorithm to output the optimal set of jobs to take. You may use either a top-down or bottom-up approach. Remember that your algorithm needs to output the optimal set of jobs to take.

Solution:

(c) Analyze the running time and space usage of your algorithm.

Solution:

Problem 2. *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 blast—after j years of charging, the laser will have enough power to obliterate d_j tons of asteroid material. You must find the best way to use the 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 a total of 5 tons of asteroids.

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

Algorithm 1: The BADLASER Algorithm

```
Function BADLASER($(x_1, \ldots, x_n), (d_1, \ldots, d_n)$):
    Compute the smallest $j$ such that $d_j \geq x_n$, or set $j = n$ if no such $j$ exists
    Shoot the laser at time $n$
    If $n > j$ :
        Return BADLASER($(x_1, \ldots, x_{n-j}), (d_1, \ldots, d_{n-j})$)
```

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(1), ..., OPT($j-1$). Give a few sentences justifying why your recurrence is correct.

Solution:

(c) Using your recurrence, design a dynamic programming algorithm to output the optimal set of times to fire the laser. You may use either a top-down or bottom-up approach. Remember that your algorithm needs to output the optimal set of times to fire the laser.

Solution:

(d) Analyze the running time and space usage of your algorithm.

Solution: