CS4800 - ALGORITHMS - S'17 — NGUYỄN
DUE FRI FEB 10, 2017 AT 11.59A VIA GRADESCOPE

H4.

- The assignment is due at Gradescope on Friday, February 10 at 11:59am. Late assignments will not be accepted. Submit early and often.
- You are permitted to study with friends and discuss the problems; however, *you must write up you own solutions, in your own words*. Do not submit anything you cannot explain. If you do collaborate with any of the other students on any problem, please do list all your collaborators in your submission for each problem.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly prohibited.
- We require that all homework submissions are prepared in Latex. If you need to draw any diagrams, however, you may draw them with your hand. Please use *a new page to begin each answer.*

PROBLEM 1 *Planet Laser*

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; e.g. 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 which obliterates 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 $3, 4$ in order to blast 5 tons of asteroids (4 tons at time 3 and 1 ton at time 4).

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

procedure BADLASER($(x_1, \ldots, x_n), (d_1, \ldots, d_n)$)
  Compute the smallest j such that $d_j \geq x_n$. Set $j = n$ if no such j exists.
  Shoot the laser at time n.
  if $n > j$ **then return** BADLASER($(x_1, \ldots, x_{n-j}), (d_1, \ldots, d_{n-j})$)
  end if
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 $\textsc{best}_j$ be the maximum amount of asteroid we can blast from year 1 to year j. Give a recurrence to compute $\textsc{best}_j$ from $\textsc{best}_1, \ldots, \textsc{best}_{j-1}$. 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.

Solution:

PROBLEM 2 *Chompo bar*

You are given an $n \times m$ chompo bar. Your goal is to devise an algorithm A that takes as input (n, m) and returns the minimal number of cuts needed to divide the bar into perfect squares of either $1 \times 1, 2 \times 2, 3 \times 3, \ldots, j \times j$. With each cut, you can split the bar either horizontally or vertically. For example, $A(2, 3) = 2$ because $2 \times 3 \to (2 \times 2, 2 \times 1) \to (2 \times 2, 1 \times 1, 1 \times 1)$.

(a) Notice that no matter the rectangle, it is always possible to make a perfect square in the first cut. Show that this greedy strategy fails. Namely, show an input size for which the strategy of picking the cut which creates the largest box leads to extra cuts in total.

Solution:

(b) Let $C(j, k)$ be the minimum number of cuts needed to divide a bar of size $j \times k$ into perfect squares. Give a recurrence to compute $C(j, k)$. Justify your recurrence. Hint: what are the possibilities for the first cut?

Solution:

(c) Give a dynamic programming algorithm to compute the minimum number of cuts using your recurrence. Analyze the running time of your algorithm.

Solution: