Problem Set 7
data structures and algorithms 2 - spring 2024

mst3k
due march 27, 2024 at 11:59p

Collaboration Policy: You are encouraged to collaborate with up to 4 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: list your collaborators

Sources: list your sources

problem 1 *Dynamic Programming*

1. If a problem can be defined recursively but its subproblems do not overlap and are not repeated, then is dynamic programming a good design strategy for this problem? If not, is there another design strategy that might be better?

   Solution:

2. As part of our process for creating a dynamic programming solution, we searched for a good order for solving the subproblems. Briefly (and intuitively) describe the difference between a top-down and bottom-up approach.

   Solution:

problem 2 *Birthday Prank*

Prof Hott's brother-in-law loves pranks, and in the past he's played the nested-present-boxes prank. I want to repeat this prank on his birthday this year by putting his tiny gift in a bunch of progressively larger boxes, so that when he opens the large box there's a smaller box inside, which contains a smaller box, etc., until he's finally gotten to the tiny gift inside. The problem is that I have a set of n boxes after our recent move and I need to find the best way to nest them inside of each other. Write a **dynamic programming** algorithm which, given a $fits(b_i, b_j)$ function that determines if box b_i fits inside box b_j, returns the maximum number of boxes I can nest (i.e. gives the count of the maximum number of boxes my brother-in-law must open).

Solution:

problem 3 *Stranger Things*

The town of Hawkins, Indiana is being overrun by interdimensional beings called Demogorgons. The Hawkins lab has developed a Demogorgon Defense Device (DDD) to help protect the town. The DDD continuously monitors the inter-dimensional ether to perfectly predict all future Demogorgon invasions.

The DDD allows Hawkins to predict that i days from now a_i Demogorgons will attack. The DDD has a laser gun that is able to eliminate Demogorgons, but the device takes a lot of time to charge. In general, charging the laser for j days will allow it to eliminate d_j Demogorgons.

Example: Suppose $(a_1, a_2, a_3, a_4) = (1, 10, 10, 1)$ 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 eliminate 5 Demogorgons.

1. Construct an instance of the problem on which the following "greedy" algorithm returns the wrong answer:

   BADLASER$((a_1, a_2, a_3, \ldots, a_n), (d_1, d_2, d_3, , \ldots, d_n))$:

   Compute the smallest j such that $d_j \geq a_n$, Set $j = n$ if no such j exists

   Shoot the laser at time n

   if $n > j$ then BADLASER$((a_1, \ldots, a_{n-j}), (d1, \ldots, d_{n-j}))$

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

   Solution:

2. Given an array holding a_i and d_j, devise a dynamic programming algorithm that eliminates the maximum number of Demogorgons. Analyze the running time of your solution. *Hint: it is always optimal to fire during the last time slot.*

   Solution: