
Collaboration Policy: You are encouraged to collaborate with up to 3 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 by naming the book etc. or listing a website's URL. 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 collaborators's computing IDs

Sources: Cormen, et al, Introduction to Algorithms. (*add others here*)

PROBLEM 1 *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 list of dimensions (length, width, and height) of the n boxes, 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: [Your answer here!](#)

PROBLEM 2 *Arithmetic Optimization*

You are given an arithmetic expression containing n integers and the only operations are additions (+) and subtractions (-). There are no parenthesis in the expression. For example, the expression might be: $1 + 2 - 3 - 4 - 5 + 6$.

You can change the value of the expression by choosing the best order of operations:

$$((((1 + 2) - 3) - 4) - 5) + 6 = -3$$

$$(((1 + 2) - 3) - 4) - (5 + 6) = -15$$

$$((1 + 2) - ((3 - 4) - 5)) + 6 = 15$$

Give a **dynamic programming** algorithm that computes the maximum possible value of the expression. You may assume that the input consists of two arrays: `nums` which is the list of n integers and `ops` which is the list of operations (each entry in `ops` is either '+' or '-'), where `ops[0]` is the operation between `nums[0]` and `nums[1]`. *Hint: consider a similar strategy to our algorithm for matrix chaining.*

Solution: [Your answer here!](#)

PROBLEM 3 *Optimal Substructure*

Please answer the following questions related to *Optimal Substructure*.

1. Briefly describe how you used *optimal substructure* for the Seam Carving algorithm.

Solution: [Your answer here!](#)

2. Do we need optimal substructure for Divide and Conquer solutions? Why or why not?

Solution: Your answer here!

PROBLEM 4 *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: Your answer here!

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. Do both approaches to the same problem produce the same runtime?

Solution: Your answer here!

PROBLEM 5 *Gradescope Submission*

Submit a version of this `.tex` file to Gradescope with your solutions added, along with the compiled PDF. You should only submit your `.pdf` and `.tex` files.