Dynamic Programming Homework

In this assignment you will use the provided code template to implement solutions to each given problem. The solutions you submit must follow the guidelines provided in this document as well as any given in code.

Your solutions must implement non-recursive, bottom-up dynamic programming approaches to each given problem.

Restrictions

- Template code is provided and must be used.
    - ***Note: The templates are refreshed each semester. Be sure to use the current version.***
- Your code must be compatible with **Python 3.10**.
- Do not add imports for any libraries beyond the Python standard library.
- Do not modify the function name or definition.

Testing

For each problem, one or more base test cases are provided. A properly implemented solution will pass any provided base test cases, but their trivial nature is not intended to ensure the overall correctness of your algorithm.

These are the same test cases that will be used to provide assurance that your submitted solutions run using the autograder. We do not release any other test cases.

In problem instances where more than one solution is optimal, *any* optimal solution will be considered correct, assuming that is the solution your algorithm produces.

The test cases can - and should! - be expanded while developing your solutions.

You can run your test cases by issuing the `python3 -m unittest` command from the directory containing your solution files.

Submission

Submit only - and all of - your solution files (i.e., **cs6515_[problem_name].py**) to the Gradescope assignment on or before the posted due date. Do not submit a zip file or any other files except those matching the naming convention defined.

You may submit your solutions as many times as desired and the most recent submission is what will be graded. Execution of your code may take up to 60 minutes. If not completed before the due date, your previous submission will be used.

Faster and correct solutions are worth more credit.

After your submission completes execution, you will see a score of `- / 20` listed until grading is completed.

Longest Increasing Subsequence

Dr. Vigoda has created lectures to help students learn about dynamic programming.

In the lectures, he discusses a classical dynamic programming concept - the Longest Increasing Subsequence.

In the Longest Increasing Subsequence problem, we want to find a subsequence, where the numbers in a subsequence increase, and for which the length is the longest possible such subsequence, in the given sequence.

Develop an efficient dynamic programming algorithm to find the length of the longest increasing subsequence as well as list of indices that make up the subsequence.

Notes:

- The input is given in an immutable format.
- The output list must be in ascending order.

Implement your algorithm for solving this problem in **cs6515_hw0.py**.