PA7: Birthday Prank

This programming assignment is to be completed in either Java or Python (your choice) and submitted on Gradescope.

Collaboration is restricted to just 2 people (other than yourself), and it must be "Whiteboard Only". In summary, this means that you may discuss the assignment with 2 others, but you cannot preserve any artifacts (digital or physical) from these discussions. In particular, this means you may not take and keep photos, notes, video, audio recordings, whiteboard contents, etc. from any discussions.

All collaborators and other resources consulted should be listed as a comment at the top of your code submission. Course staff and official course materials do not need to be listed.

Motivation

My brother's birthday is coming up. As a prank, I've decided to put his gift inside a box, put that box inside another box, that inside yet another, etc. nesting as many boxes as I can. My gift is small enough to fit in any of my boxes. For this assignment you will write a dynamic programming algorithm which determines the maximum nesting depth of my boxes.

Problem Description

The input to our algorithm is going to be n total boxes. Each box has a length, width, and height. We will always call the largest dimension the box's length and the smallest dimension the box's height (i.e. for each box, length $\geq$ width $\geq$ height).

For two boxes, A and B, box A fits inside of box B provided A's length is strictly less than B's length, A's width is strictly less than B's width, and A's height is strictly less than B's height. For example, a box with

dimensions $(20, 19, 18)$ fits inside a box with dimensions $(20.1, 19.5, 19)$, but it does not fit into a box with dimensions $(20, 20, 20)$.

Given a list of n boxes, each with a length, width, and height, you are to write a dynamic programming algorithm which returns the maximum nesting depth for that collection of boxes. In other words, your algorithm should return just a single integer representing the the largest possible number of boxes that can be nested.

Input: A list of n box objects, each box object has a length, width, and heigh attribute.

Output: Your algorithm should return just the integer representing the maximum nesting depth. You algorithm should have no print statements.

Input Format

Input will be given in the console (same as with the previous PAs). To provide an input with n items, you will have $n + 1$ lines in your input as follows:

– The first line contains the value of n, i.e. the number of boxes in the list.

– The remaining n lines contain a space-separated list of 3 floats/doubles, each box's dimensions. The first number (which is the largest of the three) represents the length, the second represents the width, the third (which is the smallest) represents the height.

Test Cases Summary

There are 15 test cases provided to you. These represent all of the test cases that the autograder will use to check your code. There are no hidden tests. Three of the tests are designed to help you develop an intuition about this problem by illustrating counterexamples for three reasonable guesses at a greedy choice function. In each case, the greedy choice would give an answer of 1, whereas the correct answer is 4.

– The test `largestheight_counterexample.txt` demonstrates that the greedy choice of selecting the box with the largest height first does not produce the correct answer.

– The test `largestlength_counterexample.txt` demonstrates that the greedy choice of selecting the box with the largest length first does not produce the correct answer.

– The test `largestvolume_counterexample.txt` demonstrates that the greedy choice of selecting the box with the largest volume (i.e. length * width * height) first does not produce the correct answer.

The answers for the 12 other tests are listed below:

– `test1.txt` has answer 5

– `test2.txt` has answer 9

– `test3.txt` has answer 8

– `test4.txt` has answer 15

– `test5.txt` has answer 16

– `test6.txt` has answer 25

– `test7.txt` has answer 24

– `test8.txt` has answer 2

– `test9.txt` has answer 1

– `test10.txt` has answer 13

– `test11.txt` has answer 41

– `test12.txt` has answer 1000

Starter Code Summary

The starter code provided consists of two files:

– `Box.java/box.py`: This is the program you will use to run your code. You will not submit this program, instead the autograder will be using its own version to interact with your code. This program implements the box class and reads the input file and then invokes the function you will implement.

– `NestBoxes.java/nest_boxes.py`: This is the program that you will edit and submit. There is just one function in the program. This function, called `max_depth` takes in the list of boxes. It then returns the maximum nesting depth. As always, you're welcome to add any "helper functions" as you see fit.