Practicum Assignment 1, A&D, 2023-2024

The problem

One of the Computing Science students is moving to a smaller student house and has called the Affordable&Decent (A&D) Storage Company to temporarily store some of their stuff. A&D Storage Company said they currently cannot store any more boxes because their storage is full of empty boxes. However, A&D Storage Company provides the student with a unique business deal: If they can figure out a way to clean up as much storage as possible, they can store their boxes for free! Clearly, this is a great deal and the student happily accepts.

A&D Storage Company provides the following additional information about the boxes in their storage. All boxes are rectangular and can be expressed by the side lengths (b_1, b_2, b_3) for box b. Moreover, all side lengths are longer than half a meter and shorter than one meter.

The student has already figured out that *storing* boxes in other boxes is a good way to save space. One box can be stored inside another box if it can be rotated such that the larger box is larger in each dimension. Formally, we say that box b with dimensions (b_1, b_2, b_3) can be stored in box c with dimensions (c_1, c_2, c_3) if there exists a permutation x, y, z of dimensions $\{1, 2, 3\}$ such that $b_x < c_1$, $b_y < c_2$ and $b_z < c_3$. The storing of boxes is recursive, if b can be stored in c and c can be stored in d, then storing b in c in d clears up the most space. Due to the minimum and maximum side lengths, it is not possible to store two boxes next to each other in one box. If box b is inside c, then nothing else can be stored in c if it cannot be stored in b.

We say that a *storing strategy* for a set of n boxes is a sequence of actions where box b is put inside another box c. If other boxes are stored in b, then they also end up in box c. A box d is *final* if it is never put into another box in the storing strategy.

The student wants to determine a storing strategy that minimizes the number of final boxes. Can you help them solve this problem?

Example. Consider three boxes with dimensions $(.7, .6, .6)$, $(.85, .6, .8)$, and $(.9, .9, .9)$. Then either the first or second box fits into the third box; but the first two boxes do not fit into one another. So the minimum possible number of final boxes, for any storing strategy, is two. This can be achieved by storing either the first or second box inside the third.

Instructions

You may work in groups of two. For this assignment, you have to hand in two things:

- Source code. We will run the code for grading.
- A report. In the report, you have to explain the algorithm and analyse it.

Source code You are allowed to submit a solution in either C, C++, Java or Python 3. If you prefer another language, please contact teaching assistant Sybrand Aarnoutse asap via Discord or `sybrand.aarnoutse@ru.nl` (no guarantees!) You must write all code yourself, copying code from fellow students or the internet is considered plagiarism. You can only use libraries that come with a minimal installation of the language. We will soon set up a test website on which you can upload and test your code.

Report Besides handing in code, we would like to receive a report in which you explain your algorithm and analyse its correctness and runtime complexity. The report is important and determines 40% of the final grade! We expect a clearly written high-level explanation of how your algorithm works, and a convincing analysis of correctness and asymptotic complexity.

Submission The deadline for sending in your solution is on November 17. You must submit your solutions via Brightspace as a group assignment for the group "Practical 1" (for which all team members need to enroll, even if you submit alone!). Only one team member has to submit a solution; the names and student numbers of both team members must be mentioned in the report. Please make sure your source code compiles and runs on domjudge, is directly accessible (i.e. not in a zip file) and contains only a single file (i.e. not a complete Java project). We will run the code for grading.

Grading Grades will be determined as follows. You may earn up to 100 points for your solution:

- 20 points for the explanation of your algorithm.
- 10 points for the correctness analysis.
- 10 points for the complexity analysis.
- 50 points for the test results. We will be running several tests and you will get points for every correct answer within the time limit. (The exact time limit will be determined after the submission deadline, but it will be in the order of 1sec.) If your code does not compile or does not read and

write via `stdin` and `stdout`, you will get zero points on the test cases. So please test your code well!
- 10 points for the quality of the code.

If you have any questions, do not hesitate to contact teaching assistants Sybrand Aarnoutse and Cas Visser via Discord (there is a special channel practical-assignments) or via email `sybrand.aarnoutse@ru.nl`. You may also contact lecturer Frits Vaandrager via `F.Vaandrager@cs.ru.nl`.

Input

- The first line has a number $1 \leq n \leq 5000$, the number of boxes.
- Then follow n lines with three floating point numbers each; the i^{th} line contains i_1, i_2 and i_3, the dimensions of box i.

Output Output the minimum number of final boxes as an integer.

Sample inputs and outputs Here are some examples:

Sample 1	
Input	Output
4 0.9 0.9 0.9 0.8 0.8 0.8 0.7 0.7 0.7 0.6 0.6 0.6	1

Sample 2	
Input	Output
3 0.6 0.6 0.6 0.75 0.75 0.75 0.9 0.7 0.7	2

Sample 3	
Input	Output
7 0.75 0.75 0.75 0.80 0.70 0.74 0.73 0.65 0.85 0.60 0.90 0.72 0.95 0.55 0.71 0.52 0.98 0.70 0.70 0.75 0.69	6