6. More Boxes

Program Name: More.java **Input File: more.dat**

You've just finished moving into a new apartment and have tons of boxes left over. A friend of yours wants to borrow them and only has access to a compact car to pick them up. So you're trying to pack the boxes into one-another as efficiently as possible.

Given a series of box dimensions, write a program to determine the least possible number of nested boxes you will have to carry to your friend's car. For this problem, a "nested" box is a series of boxes, each nestled inside the next. In the optimal case, you will be able to fit one box inside another until you end up with just 1 nested box. One tactic that is **not** allowed is to put two (or more) boxes "side-by-side" into another. When nesting boxes, you choose exactly one to put inside another, where the dimensions of the outer box are strictly larger than the dimensions of the interior box. Note that it may be necessary to rotate the boxes to make them fit, and you can nest boxes into each other in any order you choose. You are not restricted by the order given in the input.

Input

- The first line will be a single integer `n` that indicates the number of data sets in the input.
- Each data set will consist of
  - A line containing an integer `b` that indicates the number of boxes, 2 <= `b` <= 10.
  - A line containing a space-separated list of `b` box dimensions in the format "`LxWxH`", where `L` represents the length of the box in feet, `W` represents the width of the box in feet, and `H` represents the height of the box in feet, 1 <= `L,W,H` <= 20. Note that although these are the given dimensions, a box can be rotated such that its dimensions change (e.g., a 1x2x3 box can be rotated to be a 1x3x2, 2x1x3, 2x3x1, 3x2x1, or 3x1x2 box).

Output

For each data set in the input, output a single line "`X`", where `X` is the minimum number of nested boxes that can be arrived at given the boxes described.

Example Input File

```
6
2
1x2x3 2x3x4
2
2x3x4 1x2x3
2
2x3x4 2x3x4
2
4x3x2 2x3x4
4
1x2x3 2x3x4 1x2x3 2x3x4
3
2x4x3 9x9x9 1x2x3
```

Example Output to Screen

```
1
1
2
2
2
1
```