Egg Dropper

Advanced - java

Imagine that you have 2 eggs and are standing at the bottom of a very large skyscraper. Let's say that skyscraper has `n` floors.

There is a certain floor, `f`, where if you drop an egg from that height (or a height above that floor), the egg will break. If you drop an egg below floor `f`, the egg won't break.

To find this floor `f`, you decide to start dropping your eggs from different floors. If you drop an egg from a floor, and that egg breaks, you can no longer use that egg.

To find floor `f`, one strategy you could use would be to start dropping your egg from the bottom of the skyscraper, and keep going up a floor one at a time until the egg breaks. But that sounds like a lot of work dropping all of those eggs! You want to know what the minimum number of drops it would take to find floor `f`.

Complete the method `eggDrop(int n)` that returns the minimum number of egg drops it would take to discover what floor of your tower is floor `f`, given that the tower has `n` total floors. Remember, don't return floor `f` — return the number of egg drops it would take to find floor `f`.

For example `eggDrop(2)` should return `2`. If the tower has `2` stories, then we need to commit to dropping an egg twice. We could drop an egg from the first floor. If the egg breaks, we got lucky, and now know that `f = 1`. But if the egg doesn't break, then we need to drop an egg from floor `2` to see if that egg breaks. If it does, then we know `f = 2`. If it doesn't break, then we know `f` is greater than the height of the tower.

Things get trickier as `n` increases. For example, if `n` is `100`, then the minimum number of drops it will take to find `f` is `14`.

This challenge was reported to have been asked at interviews with Microsoft. If you've covered the material in [Pass the Technical Interview with Java](#) or an equivalent, you should be able to solve this challenge. If you have trouble, try thinking about this problem recursively. Our [lesson on recursion might help!](#)