

# Problem 887: Super Egg Drop

## Problem Information

**Difficulty:** Hard

**Acceptance Rate:** 29.54%

**Paid Only:** No

**Tags:** Math, Binary Search, Dynamic Programming

## Problem Description

You are given `k` identical eggs and you have access to a building with `n` floors labeled from `1` to `n`.

You know that there exists a floor `f` where `0 <= f <= n` such that any egg dropped at a floor \*\*higher\*\* than `f` will \*\*break\*\* , and any egg dropped \*\*at or below\*\* floor `f` will \*\*not break\*\*.

Each move, you may take an unbroken egg and drop it from any floor `x` (where `1 <= x <= n`). If the egg breaks, you can no longer use it. However, if the egg does not break, you may \*\*reuse\*\* it in future moves.

Return \_the\*\*minimum number of moves\*\* that you need to determine \*\*with certainty\*\* what the value of \_`f` is.

**Example 1:**

**Input:** k = 1, n = 2 **Output:** 2 **Explanation:** Drop the egg from floor 1. If it breaks, we know that f = 0. Otherwise, drop the egg from floor 2. If it breaks, we know that f = 1. If it does not break, then we know f = 2. Hence, we need at minimum 2 moves to determine with certainty what the value of f is.

**Example 2:**

**Input:** k = 2, n = 6 **Output:** 3

**Example 3:**

**\*\*Input:\*\*** k = 3, n = 14 **\*\*Output:\*\*** 4

**\*\*Constraints:\*\***

\* `1 <= k <= 100` \* `1 <= n <= 104`

## Code Snippets

### C++:

```
class Solution {  
public:  
    int superEggDrop(int k, int n) {  
  
    }  
};
```

### Java:

```
class Solution {  
public int superEggDrop(int k, int n) {  
  
}  
}
```

### Python3:

```
class Solution:  
    def superEggDrop(self, k: int, n: int) -> int:
```