Egg dropping puzzle

The following is a description of the instance of this famous puzzle involving n=2 eggs and a building with H=36 floors:[14]

> Suppose that we wish to know which stories in a 36-story building are safe to drop eggs from, and which will cause the eggs to break on landing. We make a few assumptions:
>
> - An egg that survives a fall can be used again.
> - A broken egg must be discarded.
> - The effect of a fall is the same for all eggs.
> - If an egg breaks when dropped, then it would break if dropped from a higher window.
> - If an egg survives a fall then it would survive a shorter fall.
> - It is not ruled out that the first-floor windows break eggs, nor is it ruled out that the 36th-floor windows do not cause an egg to break.
>
> If only one egg is available and we wish to be sure of obtaining the right result, the experiment can be carried out in only one way. Drop the egg from the first-floor window; if it survives, drop it from the second floor window. Continue upward until it breaks. In the worst case, this method may require 36 droppings. Suppose 2 eggs are available. What is the least number of egg-droppings that is guaranteed to work in all cases?

To derive a dynamic programming functional equation for this puzzle, let the **state** of the dynamic programming model be a pair s = (n,k), where

> n = number of test eggs available, $n = 0, 1, 2, 3, \ldots, N - 1$.
> k = number of (consecutive) floors yet to be tested, $k = 0, 1, 2, \ldots, H - 1$.

For instance, $s = (2,6)$ indicates that two test eggs are available and 6 (consecutive) floors are yet to be tested. The initial state of the process is $s = (N,H)$ where N denotes the number of test eggs available at the commencement of the experiment. The process terminates either when there are no more test eggs ($n = 0$) or when $k = 0$, whichever occurs first. If termination occurs at state $s = (0,k)$ and $k > 0$, then the test failed.

Now, let

> $W(n,k)$:= minimum number of trials required to identify the value of the critical floor under the Worst Case Scenario given that the process is in state $s = (n,k)$.

Then it can be shown that[15]

$$W(n,k) = 1 + \min\{\max(W(n-1, x-1), W(n,k-x)) : x \text{ in } \{1, 2, \ldots, k\}\}, n = 2, \ldots, N;\ k = 2, 3, 4, \ldots, H$$

with $W(n,1) = 1$ for all $n > 0$ and $W(1,k) = k$ for all k. It is easy to solve this equation iteratively by systematically increasing the values of n and k.

An interactive online facility is available for experimentation with this model as well as with other versions of this puzzle (e.g. when the objective is to minimize the **expected value** of the number of trials.[15]

Faster DP solution using a different parametrization

Notice that the above solution takes $O(nk^2)$ time. This can be improved to $O(nk \log k)$ time by binary searching on the optimal x in the above recurrence, since $W(n-1, x-1)$ is increasing in x while $W(n, k-x)$ is decreasing in x, thus a local minimum of $\max(W(n-1, x-1), W(n, k-x))$ is a global minimum. However, there is a much faster solution that involves a different parametrization of the problem:

Let k be the total number of floors such that the eggs break when dropped from the kth floor (The example above is equivalent to taking $k = 37$)

Let m be the minimum floor from which the egg must be dropped to be broken

Let $f(t, n)$ be the maximum number of values of m that are distinguishable using t tries and n eggs

Then $f(t, 0) = f(0, n) = 1$ for all $t, n \geq 0$

Let x be the floor from which the first egg is dropped in the optimal strategy

If the first egg broke, m is from 1 to x and distinguishable using at most $t-1$ tries and $n-1$ eggs

If the first egg did not break, m is from $x+1$ to k and distinguishable using $t-1$ tries and n eggs

Therefore $f(t, n) = f(t-1, n-1) + f(t-1, n)$

Then the problem is equivalent to finding the minimum x such that $f(x, n) \geq k$

To do so we could compute $f(t, i) : 0 \leq i \leq n$ in order of increasing t, which would take $O(nx)$ time

Thus if we separately handle the case of $n = 1$, the algorithm would take $O(n\sqrt{k})$ time

But the recurrence relation can in fact be solved, giving $f(t, n) = \sum_{i=0}^{n} \binom{t}{i}$, which can be computed in $O(n)$ time using the identity $\binom{t}{i+1} = \binom{t}{i} \frac{t-i}{i+1}$ for all $i \geq 0$

Since $f(t, n) \leq f(t+1, n)$ for all $t \geq 0$, we can binary search on t to find x, giving an $O(n \log k)$ algorithm