# Algorithm Analysis: Dropping Eggs

In the *two-egg* problem, we have two eggs and a building with $n$ floors; we want to find the building's *drop tolerance*, the highest floor from which we can drop an egg before it breaks, denoted $T$. The only way to know whether an egg will break at some floor is to try dropping it from that floor; if it breaks, of course, we cannot use it again for another test.

One approach is to incrementally drop an egg from increasing floor levels, starting at the first floor, until we drop the egg from some floor $F$ and it breaks; we know that $T$ is given by the last floor from which we dropped the egg without breaking it, i.e. $T = F - 1$. This solution takes $\Theta(T)$ drops, using only one egg.

The question is whether we can solve this problem more efficiently.

## Understanding the constraint

Notice the constraining factor is the fact that we only have two eggs. If we had an unlimited number of eggs, a binary search over the building's floors would solve the problem in $\Theta(\lg T)$ time.

The $O(T)$ solution presented above makes suboptimal use of resources: we are given two eggs, but we only use one to identify $T$. Another approach that makes use of both eggs is as follows:

- Drop the first egg at increasingly higher floors, but instead of climbing one floor at a time, climb $k$ floors on each trial: i.e., drop it from floor $0$, $k$, $2k$, …, until it breaks. We are free to choose the value of $k$, which we refer to as the *drop interval* for the first egg.
- When the first egg breaks on some floor $\alpha k$, we know that $T$ must lie in the range $[(\alpha - 1)k,\ \alpha k)$. So, starting at floor $1 + (\alpha - 1)k$, drop the second egg from incrementally higher floors, one floor at a time, until we identify $T$.

A simple argument shows why we must use a drop interval of 1 for the second egg: say we use some drop interval of $\beta \neq 1$ for the second egg. We drop the second egg from some floor $F$ and it breaks. What can we conclude? The value of $T$ could be anywhere in the range $[F - \beta,\ F)$. This implies that there are $\beta$ possible answers, when we wanted an exact answer—we have not given a sufficiently specific answer to the original question.[1]

We have thus arrived at an algorithm for finding $T$. The question is how to choose the drop interval $k$ so as to minimize the number of drops we need to make.

[1] Note, had we wanted the value of $T$ within a given error tolerance, the value of $\beta$ could indeed be greater than 1.

Finding the optimal drop interval

We will have to do ~$T/k$ drops with the first egg (within a constant factor) and upwards of $k-1$ drops with the second egg to guarantee completion of the algorithm. Asymptotically, the total work done (total number of drops we have to perform) is thus

$$W_{total} = W(\text{first egg}) + W(\text{second egg}) = \Theta\left(\frac{T}{k}\right) + O(k) = O\left(\frac{T}{k} + k\right)$$

Notice the tradeoff in our choice of $k$: if we increase k, $W(\text{first egg})$ is smaller, but $W(\text{second egg})$ is larger. This leads us to the first method for solving this problem: balancing. Given two components of an algorithm that have inverse dependencies on some parameter(s) (i.e., one varies directly with the parameter, while the other varies inversely), the idea of balancing is simply to exchange work between the components such that the work performed by each is equal, so that neither component performs more work than needed. Thus, we want $W(\text{first egg}) = W(\text{second egg})$, or

$$\frac{T}{k} = k \rightarrow T = k^2 \rightarrow k = \sqrt{T}$$

Another approach, the calculus-based perspective, is to minimize the function on the parameter $k$ by finding the zero of its first derivative:

$$\frac{d}{dk}\left[\frac{T}{k} + k\right] = 1 - \frac{T}{k^2} = 0 \rightarrow k = \sqrt{T}$$

We arrive at the same answer via both methods. In either case, choosing $k = \sqrt{T}$ yields a minimum. The work done at this minimum is

$$W = O\left(\frac{T}{k} + k\right) = O\left(\frac{T}{\sqrt{T}} + \sqrt{T}\right) = O\left(2\sqrt{T}\right)$$

If the building has $n$ floors, we know that $T$ is bounded at $O(n)$, so at worst the algorithm will require $O\left(2\sqrt{n}\right)$ steps. The factor of 2 is not omitted for reasons that will be made apparent.

Three eggs

In the three-egg problem, we have an extra egg to work with. Similar to before, the last egg must be dropped one floor at a time, but the first two eggs can be dropped at any intervals $k, \beta$ of our choice. The work done is bounded by

$$W = O\left(\frac{n}{k} + \frac{k}{\beta} + \beta\right)$$

Be careful here: the second term is indeed $k/\beta$, *not* $n/\beta$. The reason is that the series of drops performed with the first egg will have narrowed our search to a range of floors with span $k$; the search with the second egg, having jump interval $\beta$, operates in this narrowed range. Additionally, we use $n/k$ for the first term rather than $T/k$ since $T = O(n)$.

To minimize this, the calculus-based approach is most straightforward: taking $W$ as a function of parameters $\beta, k$, we have

$$\frac{\partial W}{\partial \beta} = -\frac{k}{\beta^2} + 1 = 0 \rightarrow \beta = \sqrt{k}$$
$$\frac{\partial W}{\partial k} = -\frac{n}{k^2} + \frac{1}{\beta} = 0 \rightarrow k = \sqrt{\beta n}$$

From these we have $k = \beta^2 = \sqrt{\beta n}$. Some algebra will show that $\beta = n^{1/3}, k = n^{2/3}$. The total work is then

$$W = \frac{n}{n^{2/3}} + \frac{n^{2/3}}{n^{1/3}} + n^{1/3} = 3n^{1/3}$$

With only one more egg, our solution has taken on a better asymptotic complexity, albeit with a higher coefficient.

Thus for some number of eggs $E$, we have so far found

$$W(E) = O\left(\begin{Bmatrix} 1n^{\frac{1}{1}}, & E = 1 \\ 2n^{\frac{1}{2}}, & E = 2 \\ 3n^{\frac{1}{3}}, & E = 3 \\ & \vdots \end{Bmatrix}\right)$$

Notice a pattern?

<u>Many eggs*</u>
*It may be helpful to have familiarity with mathematical induction for this section.

Say we have $E$ eggs. Letting $N \triangleq E - 1$,[2] then the jump intervals be given by $\vec{\alpha} = (a_1, a_2, \ldots, a_N)$. The worst-case work is

$$W = O\left(\frac{n}{a_1} + \frac{a_1}{a_2} + \frac{a_2}{a_3} + \cdots + \frac{a_{N-1}}{a_N} + a_N\right)$$

We have $W$ as a function of the components of $\vec{\alpha}$ and, as before, pose a minimization constraint based on these components (note, superscripts are exponents, subscripts are indexing labels):

[2] $\triangleq$ may be used to indicate "is defined as". Some authors may also use $\equiv$ to represent this meaning.

$$\frac{dW}{d\vec{\alpha}} = 0 = \begin{bmatrix} \partial W/\partial a_1 \\ \partial W/\partial a_2 \\ \vdots \\ \partial W/\partial a_N \end{bmatrix} = \begin{bmatrix} -\frac{n}{a_1^2} + \frac{1}{a_2} \\ -\frac{a_1}{a_2^2} + \frac{1}{a_3} \\ \vdots \\ -\frac{a_{N-2}}{a_{N-1}^2} + \frac{1}{a_N} \\ -\frac{a_{N-1}}{a_N^2} + 1 \end{bmatrix}$$

Solving equations in this vector from the bottom up:

$$a_{N-1} = a_N^2$$

$$a_{N-2} = \frac{a_{N-1}^2}{a_N} = \frac{a_N^4}{a_N} = a_N^3$$

$$a_{N-3} = \frac{a_{N-2}^2}{a_{N-1}} = \frac{a_N^6}{a_N^2} = a_N^4$$

We begin to see a pattern that suggests the following recursive hypothesis: [3]

$$a_{N-k} = a_N^{k+1} \; \forall \, k \in [0, N-1]$$

We can prove this using induction. On the interval $k \in [1, N-1]$ we have

$$a_{N-k} = \frac{a_{N-(k-1)}^2}{a_{N-(k-2)}}$$

Our first base case is verified trivially by $a_N = a_{N-0} = a_N^{0+1} = a_N^1 = a_N$. We also have verified the second base case above, $a_{N-1} = a_N^{1+1} = a_N^2$. Now assume the hypothesis is true for some $k-2, k-1$; we want to show it true for $k$:

$$a_{N-k} = \frac{a_{N-(k-1)}^2}{a_{N-(k-2)}}$$

$$a_N^{k+1} \overset{?}{=} \frac{a_N^{2(k-1+1)}}{a_N^{k-2+1}}$$

$$a_N^{k+1} \overset{?}{=} \frac{a_N^{2k}}{a_N^{k-1}} = a_N^{2k-(k-1)}$$

$$a_N^{k+1} \overset{\checkmark}{=} a_N^{k+1}$$

We find the value of $a_N$ in two steps. First, use the first equation in the gradient vector:

[3] ∀ means "for all"

$$-\frac{n}{a_1^2}+\frac{1}{a_2}=0 \to a_1=\sqrt{a_2 n}$$

Second, use the results of the induction:

$$a_1 = a_{N-(N-1)} = a_N^{(N-1)+1} = a_N^N$$
$$a_2 = a_{N-(N-2)} = a_N^{(N-2)+1} = a_N^{N-1}$$

Combining these results gives

$$a_N^N = \sqrt{a_N^{N-1} n} \to a_N = n^{\frac{1}{N+1}} = n^{1/E}$$

From this we deduce the following:

$$a_i = a_{N-(N-i)} = a_N^{N-i+1} = \left(n^{\frac{1}{E}}\right)^{E-i} = n^{1-\frac{i}{E}}$$

$$\frac{a_i}{a_{i+1}} = \frac{a_N^{N-i+1}}{a_N^{N-(i+1)+1}} = a_N = n^{1/E}$$

$$a_1 = a_N^N = n^{\frac{N}{N+1}} \to \frac{n}{a_1} = n^{\frac{1}{N+1}} = n^{1/E}$$

Recall what we were trying to accomplish: we wanted to minimize the total work $W$ as a function of the components of $\vec{\alpha}$, which gave the jump intervals $(a_1, a_2, \dots, a_N)$. From the above results, we have all the information we need. Returning to the original worst-case expression for $W$:

$$W = \frac{n}{a_1} + \frac{a_1}{a_2} + \frac{a_2}{a_3} + \cdots + \frac{a_{N-1}}{a_N} + a_N$$

Every term in this sum equates to $n^{1/E}$, and there are $E$ terms, so we have $W = En^{1/E}$. Due to the requirement that each $a_i$ be an integer, we might have some $a_i, a_{i+1}$ that are rounded to the same values, which is redundant; the algorithm would have to be redesigned to handle this situation. One option is to calculate $a_i = \text{round}(n^{1-i/E})$, and then prune duplicate values; the number of operations would then be given by the following:

$$W^* = \left\lfloor\frac{n}{a_1}\right\rfloor + \left\lfloor\frac{a_1}{a_2}\right\rfloor + \left\lfloor\frac{a_2}{a_3}\right\rfloor + \cdots + \left\lfloor\frac{a_{N-1}}{a_N}\right\rfloor + a_N$$

The floor operator $\lfloor \quad \rfloor$ ensures an integral result. The following example shows why the ceiling operator would not work: say at some point in the algorithm we were using jump intervals of 7 and then 4; if an egg breaks at some floor $f + 7$, we would only need to check $f + 4$ with the interval of 4, because we know the egg would break on floor $f + 8$. The only other modification that would be needed would be if some $a_i, a_i + 1$ were multiples of one another, in which case that pair of jump intervals would only contribute $\frac{a_i}{a_{i+1}} - 1$; however, this would not happen very often, so its effect is negligible.

<u>What-sized carton?</u>
Given some number of eggs $E$, we have found the optimal choices of jump intervals to minimize the overall work done for that $E$. But what is the optimal value of $E$?

We want to minimize $W = En^{1/E}$ over $E$:

$$W' = \frac{dW}{dE} = n^{1/E} + E\frac{d\left[n^{1/E}\right]}{dE} = 0$$

We seek the derivative of $f(E) = n^{1/E}$. We use the logarithm of this function to find its derivative:

$$\frac{d}{dE}\ln f = \frac{f'}{f} \rightarrow f' = f\frac{d[\ln f]}{dE} = n^{1/E}\frac{d\left[\ln n^{1/E}\right]}{dE} = n^{1/E}\frac{d}{dE}\left[\frac{\ln n}{E}\right] = -\ln(n)\,n^{1/E}E^{-2}$$

So that $W' = n^{1/E} - \ln(n)\,n^{1/E}E^{-1} = 0$. This requires $1 - \ln(n)\,E^{-1} = 0$, hence $E = \ln(n)$. The asymptotic worst-case work is then

$$W = \ln(n)\,n^{1/\ln n}$$

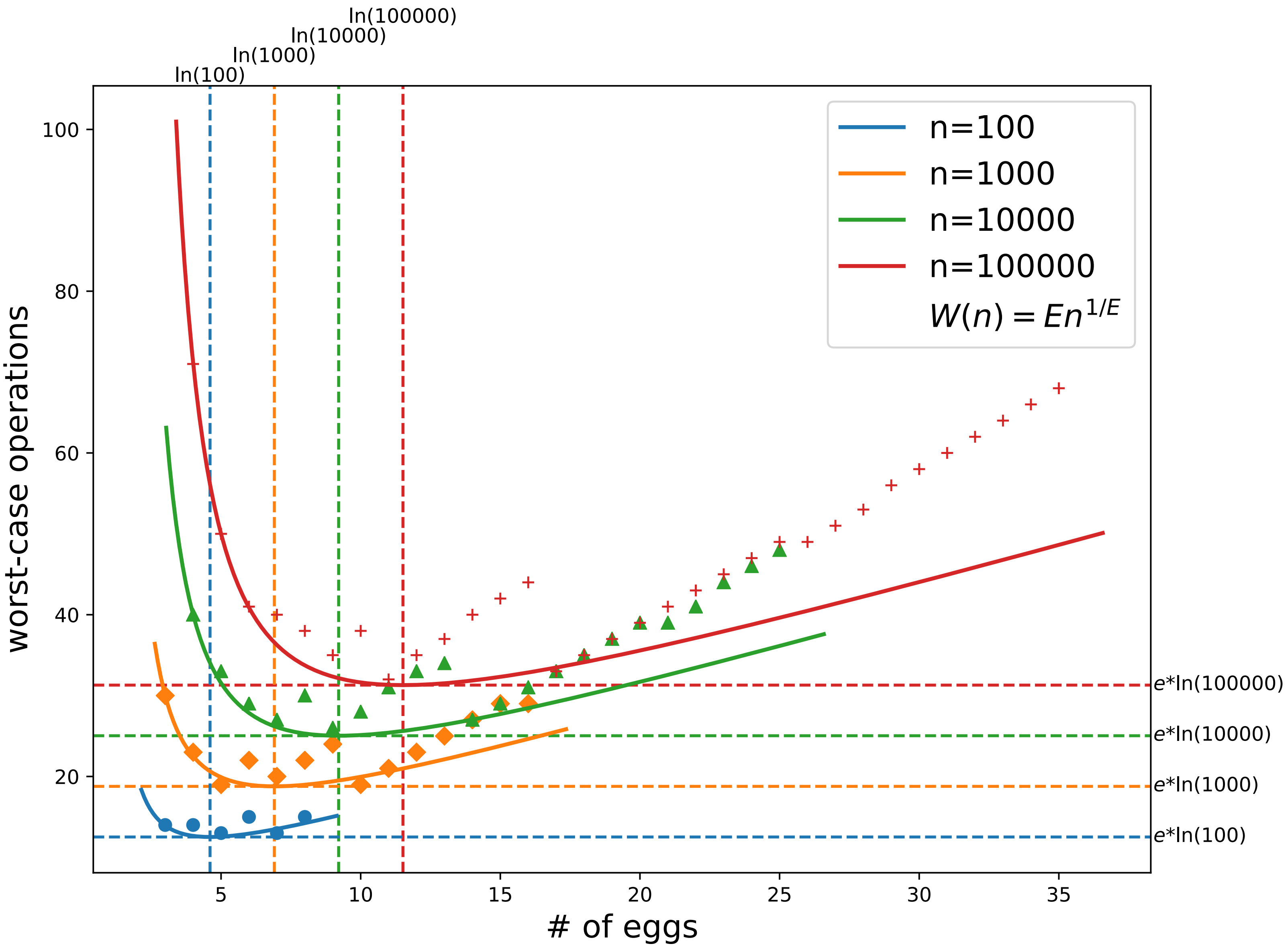
To resolve this expression, take the natural log of both sides:

$$\ln W = \ln\left[\ln(n)\,n^{1/\ln n}\right] = \ln\ln n + \ln n^{1/\ln n} = \ln\ln n + \frac{\ln n}{\ln n} = \ln\ln n + 1$$

Now re-exponentiate:

$$W = e^{\ln W} = e^{\ln\ln n+1} = e\cdot e^{\ln\ln n} = e\ln n = \Theta(\ln n)$$

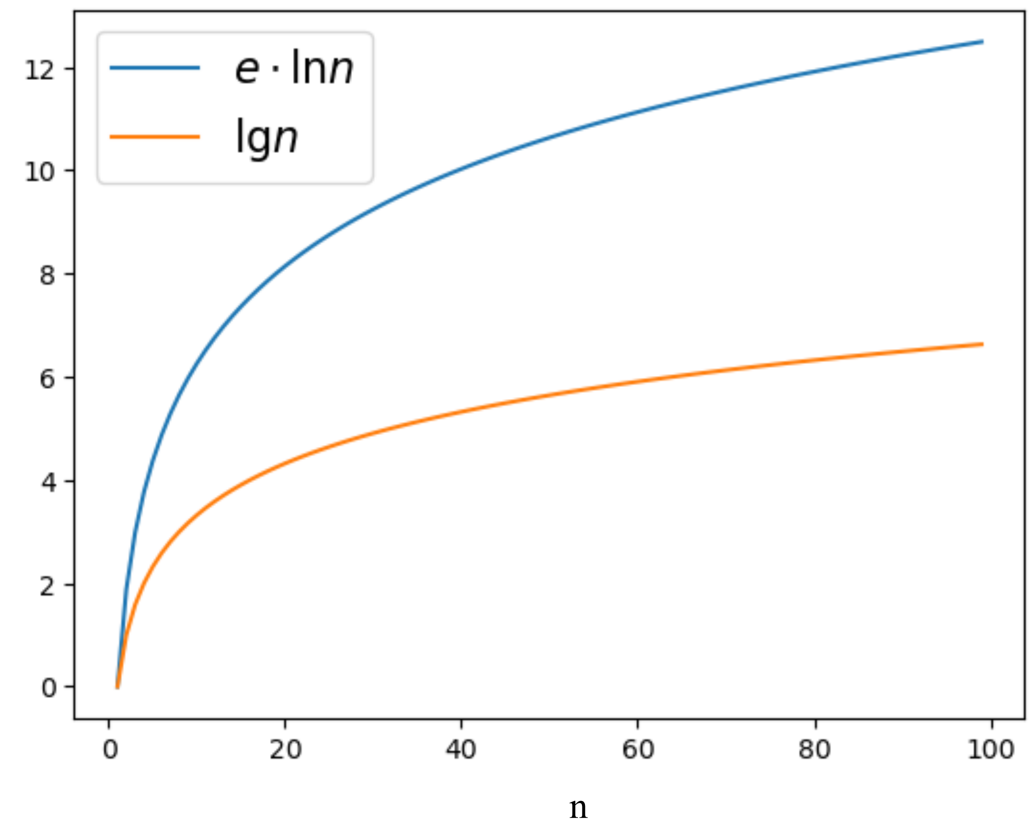
To visualize the results, a plot follows of some theoretical results for several $n$, each assigned to a differently colored line, with vertical/horizontal bars denoting the optimal values of $E$ and $W$. The smooth lines show the hypothetical results without taking integer logic into account; the plotted markers show values of $W^*$ as defined earlier:



Comment: if you have a building with $10^i$ floors for any integer $i > 2$, that is news in itself.

The theoretical predictions tend to underestimate the actual operation counts, increasing with increasing $E$, but the theoretical optimum ($E = \ln n \to W = e \ln n$) gives a good approximation of the empirical optimum. Note, for the tallest buildings currently in existence (upwards of ~160 floors), roughly a handful of eggs gives the best results; any more is unnecessary.

Thus, for the incremental approach defined earlier—dropping each egg at a consistent drop interval until it breaks, and iteratively decreasing the drop interval until we narrow down the exact drop tolerance with the last egg—we have found that $W = \Omega(e \ln n)$ for a building of height $n$. Notice that this is the same asymptotic class that would have arisen had we used binary search, which requires $\lg n$ operations. However, if we had $\lg n$ eggs in the first place, then binary search would still have been the way to go: for any $n > 1$, $\lg n < e \ln n$:



The takeaways

A few summary points to take from this exercise:

*Use more resources only if they improve the target complexity*. We saw from the start that the overall asymptotic time complexity was much better when we used two eggs instead of using only one, and it improves with more eggs available until we have $E = \ln n$. However, we saw that if we had $\lg n$ eggs, binary search would be an even better approach.

*Knowing how to optimally utilize resources depends on the algorithm*. The asymptotic complexity involved with having $E$ eggs at our disposal arises from the choice of the jump intervals for each egg dropped. Working out the algorithm first allowed us to derive the optimal parameters for it.

*Parameters may be optimized at several levels*. Notice, even though we found that for $E$ eggs, the optimal work performed is asymptotically $En^{1/E}$, this did not tell us what the optimal number of eggs was for the jump-interval approach. This was a separate optimization problem entirely, though it required an analytical form for $E$ to be solved.

*Consider practical constraints*. This has two realizations in this problem.
**First**, the theoretical prediction of $W = En^{1/E}$ seems to systematically underestimate the actual work required for given values of $E$ that deviate far from the theoretical minimum of $E = e \ln n$; this is because in theory, $E$ was treated as a continuous variable for the sake of optimization, but in practice, $E$ is a discrete variable. Note, the predictions for $W$ were quite good near the theoretically minimizing $E$.
**Second**, although in theory buildings could be arbitrarily tall, in practice buildings only will be only so tall on the Earth; this implies that there is some rough upper bound on the resources we would need to solve this problem in any realistic situation.

*      *      *

In short: resolve a problem to all its constraints, and test the theoretical solutions against the practical problems to which they will be applied. Only when we have built as complete a picture of possible of how our resource usage depends on our inputs can we justify the resources that we use.