Egg Drop Problem

Swapnamoy Samadder

January 2024

1 Introduction

The ball-drop problem is a classic problem in algorithm design and optimization. The objective is to determine the highest floor of a building from which a ball can be dropped without breaking, using the fewest number of drops in the worst-case scenario. This problem offers valuable insight into decision-making strategies under constraints and has applications in fields such as quality testing, risk assessment, and resource allocation.

In the standard formulation, one has a fixed number of balls and a building with a finite number of floors. The goal is to devise an optimal strategy that minimizes the maximum number of drops required to identify the critical floor, the lowest floor from which a dropped ball breaks.

$$c\left(A, \mathbf{f}\right) := \text{ Number of drops required to determine } \mathbf{f}$$

$$\arg\min_{A \in \mathcal{A}_k} \{ \max_{1 \leq \mathbf{f} \leq n} c(A, \mathbf{f}) \}$$

$$f_k\left(n\right) := \min_{A \in \mathcal{A}_k} \{ \max_{1 \leq \mathbf{f} \leq n} c(A, \mathbf{f}) \}$$

Definition 1. The function $f_k\left(n\right)$ represents the minimum number of drops required to determine the critical floor ($\mathbf{f}$) in the worst-case scenario, using k balls in a tower with n floors.

1.1 The Main Results

Theorem 1. *Given n floors and k balls,*

$$f_k\left(n\right) \sim \left(k!n\right)^{1/k}$$

Theorem 2. *The optimal algorithm to determine $\mathbf{f}$ for an unknown tower height requires at most*

$$2(k!\ \mathbf{f})^{1/k}(1 + o(1))$$

drops in the worst case, where $k \geq 2$.

2 Preliminaries

Lemma 2.1. *Let $a_1, a_2, \ldots, a_k \in \mathbb{N}$ such that $a_1 + a_2 + \cdots + a_k = n$, where n is a fixed integer. Then:*

i) $\max(a_1, a_2, \ldots, a_k) \geq \frac{n}{k}$.

ii) $\max(a_1, a_2, \ldots, a_k) = \frac{n}{k}$ *if and only if* $a_1 = a_2 = \cdots = a_k = \frac{n}{k}$.

Proof. *i)* **Proof by contradiction:** Suppose $\max(a_1, a_2, \ldots, a_k) < \frac{n}{k}$. Then $a_i < \frac{n}{k}$ for all $1 \leq i \leq k$. Summing over all i, we have:

$$a_1 + a_2 + \cdots + a_k < k \cdot \frac{n}{k} = n,$$

which contradicts the given condition $a_1 + a_2 + \cdots + a_k = n$. Hence, $\max(a_1, a_2, \ldots, a_k) \geq \frac{n}{k}$.

ii) **Proof of equivalence:**

- ($\Leftarrow$) If $a_1 = a_2 = \cdots = a_k = \frac{n}{k}$, then clearly $\max(a_1, a_2, \ldots, a_k) = \frac{n}{k}$.

- ($\Rightarrow$) Suppose $\max(a_1, a_2, \ldots, a_k) = \frac{n}{k}$ but not all a_i are equal to $\frac{n}{k}$. Then at least one $a_i < \frac{n}{k}$, which implies:
$$a_1 + a_2 + \cdots + a_k < n,$$
contradicting the condition $a_1 + a_2 + \cdots + a_k = n$. Thus, all a_i must be equal to $\frac{n}{k}$.

□

Lemma 2.2. *Let $T(a, b)$ be defined by the following recurrence relation:*

$$T(a, b) = T(a - 1, b - 1) + T(a - 1, b),$$

with base cases $T(0, b) = 1$, $T(a, 0) = 1$, and $T(0, 0) = 1$ for all $a, b \in \mathbb{N}$. Then:

$$T(a, b) = \sum_{i=0}^{b} \binom{a}{i}.$$

Proof. Define the generating function:

$$G(x, y) := \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} T(i, j) x^i y^j, \quad \forall x \in [0, 1), \forall y \in [0, 1).$$

We expand $G(x, y)$ using the recurrence relation and base cases:

$$\begin{aligned} G(x, y) &= \sum_{i=0}^{\infty} T(i, 0) x^i + \sum_{j=0}^{\infty} T(0, j) y^j + \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} T(i, j) x^i y^j - T(0, 0) \\ &= \frac{x}{1 - x} + \frac{1}{1 - y} + \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \left(T(i - 1, j - 1) + T(i - 1, j)\right) x^i y^j \end{aligned}$$

Substituting the recurrence relation:

$$G(x, y) = \frac{x}{1 - x} + \frac{1}{1 - y} + xyG(x, y) + x \left(G(x, y) - \frac{1}{1 - x} \right)$$

Rearranging and solving:

$$G(x, y) = \frac{1}{(1 - y)\left(1 - x\left(1 + y\right)\right)}$$

Expanding the solution:

$$G(x, y) = \left(\sum_{i=0}^{\infty} y^i \right) \left(\sum_{j=0}^{\infty} x^j \left(1 + y\right)^j \right)$$

The coefficient of $x^a y^b$ in this expansion is:

$$T(a, b) = \sum_{i=0}^{b} \binom{a}{i}.$$

□

3 Proof of Theorem 1

Proof. Let $F(d, k)$ denote the maximum number of floors in a building such that the $\mathbf{f}$ can be determined using k balls and d ball-drops. The function satisfies the following recurrence:

$$\begin{aligned} &F(d, k) = F(d - 1, k - 1) + F(d - 1, k) + 1, \\ &F(0, k) = 0, \\ &F(d, 0) = 0. \end{aligned} \tag{3.1}$$

Define $T(d,k) := F(d,k)+1$. Substituting this into (3.1), we have:

$$\begin{aligned} T(d,k) &= T(d-1,k-1)+T(d-1,k),\\ T(0,k) &= 1,\\ T(d,0) &= 1. \end{aligned}$$

By **Lemma 2.2**, this recurrence implies:

$$T(d,k) = \sum_{i=0}^{k} \binom{d}{i}.$$

Thus:

$$F(d,k) = \sum_{i=1}^{k} \binom{d}{i}.$$

To solve the $f_k(n)$ problem, we seek the smallest d such that:

$$\sum_{i=1}^{k} \binom{d-1}{i} < n \leq \sum_{i=1}^{k} \binom{d}{i}. \tag{3.2}$$

Equivalently, this can be expressed as:

$$\sum_{i=1}^{k} \binom{f_k(n)-1}{i} < n \leq \sum_{i=1}^{k} \binom{f_k(n)}{i}. \tag{3.3}$$

To complete the proof, we need the following two lemmas:

Lemma 3.1. *For any $k \in \mathbb{N}$,*

$$\lim_{n\to+\infty} f_k(n) \to +\infty.$$

Proof of Lemma 3.1. Suppose the contrary. Then, for some fixed $M \in \mathbb{N}$, there are infinitely many $n \in \mathbb{N}$ such that $f_k(n) \leq M$. This implies:

$$n \leq F(M,k)$$

for infinitely many $n \in \mathbb{N}$. However, $F(M,k)$ is a constant, contradicting the fact that natural numbers are unbounded (Archimedean property of $\mathbb{R}$). □

Lemma 3.2. *Let $p : \mathbb{R} \to \mathbb{R}$ be a polynomial of degree r with leading coefficient l. Then, for any fixed $c \in \mathbb{R}$,*

$$\lim_{x\to+\infty} \frac{p(x-c)}{x^r} = l.$$

Using **Lemma 3.1** and **Lemma 3.2**, and observing that $F(d,k)$ is a polynomial of degree k with leading coefficient $\frac{1}{k!}$, we conclude:

$$\lim_{n\to+\infty} \frac{F(f_k(n)-1,k)}{f_k(n)^k} = \frac{1}{k!}, \quad \lim_{n\to+\infty} \frac{F(f_k(n),k)}{f_k(n)^k} = \frac{1}{k!}.$$

Using Equation (3.3) and the Sandwich Theorem, we find:

$$\lim_{n\to+\infty} \frac{n}{f_k(n)^k} = \frac{1}{k!}.$$

Therefore:

$$f_k(n) \sim (k!n)^{1/k}. \tag{3.4}$$

□

Corollary 3.1. *For any k, $f_k(n)$ satisfies:*

$$f_k(n) \geq \log(n+1)$$

Proof. By definition, $f_k(n)$is the smallest number of drops required to determine the critical floor in a tower of height n using k balls. Hence:

$$n \leq \sum_{i=1}^{k} \binom{f_k(n)}{i}.$$

Since:

$$2^{f_k(n)} - 1 \geq n,$$

it follows that:

$$f_k(n) \geq \log(n+1).$$

This bound is independent of k, meaning the worst-case performance of any algorithm cannot be improved beyond $\log(n+1)$. □

4 Algorithm

We present our algorithm in Algorithm 1

Algorithm 1: Optimal Algorithm

```
Data: n [floors] ≥ 0, k [balls] ≥ 1
Result: f, d [number of drops required]
f ← NULL;
d ← −1;
L ← 1;
R ← n;
while d < 0 do
    t ← ⌊(L + R)/2⌋;
    if F(t − 1, k) ≥ n then
        R ← t;
    else if F(t, k) < n then
        L ← t;
    else
        d ← t;
    end
end
b ← 1;
m ← 0;
u_l ← n;
l_l ← 0;
while f = NULL do
    m += F(d − 1, k − 1) + 1;
    drop b−th ball on m−th floor;
    d −= 1;
    if ball doesn't break then
        l_l = m;
    else
        b += 1;
        u_l = m;
        m −= F(d, k − 1) + 1;
        k −= 1;
    end
    if u_l = l_l + 1 then
        f = u_l;
    end
end
```

5 Unknown Tower Height and Optimal Drop Strategy

The previous discussion assumes that the height of the tower is known in advance. However, this raises a natural question: *What happens if the height of the tower is not specified beforehand?* In such a scenario, determining the critical floor ($\mathbf{f}$) becomes more challenging, as the problem now involves finding $\mathbf{f}$ for an unknown tower height. This section addresses this variant of the problem and provides an optimal solution.

Theorem 3. *The optimal algorithm to determine* $\mathbf{f}$ *for an unknown tower height requires at most*

$$2(k!\,\mathbf{f})^{1/k}(1+o(1))$$

drops in the worst case, where $k \geq 2$.

Proof. Let $\{a_m\}_{m\geq 1}$ be a sequence of natural numbers. We drop the first ball on floors

$$a_1,\ a_1+a_2,\ \cdots,\ \sum_{i=1}^{t} a_i,$$

until the ball shatters on floor $\sum_{i=1}^{t} a_i$. This ensures:

$$\sum_{i=1}^{t-1} a_i < \mathbf{f} \leq \sum_{i=1}^{t} a_i$$

Next, we apply the best algorithm (Algorithm 1) to the floors between $\left(\sum_{i=1}^{t-1} a_i\right)+1$ and $\sum_{i=1}^{t} a_i$ to precisely determine $\mathbf{f}$. This step requires:

$$((k-1)!\,a_t)^{1/(k\,-\,1)}$$

drops.

Hence, the total number of drops is:

$$t + ((k-1)!\,a_t)^{1/(k\,-\,1)}$$

To optimize, set:

$$t = ((k-1)!\,a_t)^{1/(k\,-\,1)}$$

which implies

$$a_t = \frac{t^{(k-1)}}{(k-1)!}$$

Define the cumulative sum:

$$s_n = \sum_{i=1}^{n} a_i$$

By bounding summation by integration we get

$$\frac{(t+1)^k}{k!} > s_t > \frac{t^k}{k!}$$

Thus:

$$\frac{(t-1)^k}{k!} < \mathbf{f} < \frac{(t+1)^k}{k!}$$

which implies:

$$t \approx (k!\,\mathbf{f})^{1/k}$$

Therefore, the total number of drops becomes:

$$t + ((k-1)!\,a_t)^{1/(k\,-\,1)} = 2t = 2(k!\,\mathbf{f})^{1/k}$$

Thus, the upper bound for the optimal algorithm is $2(k!\,\mathbf{f})^{1/k}(1+o(1))$ □

Definition 2. Let $\mathcal{A}_K$ represent the set of all deterministic algorithms that use at most k balls to identify $\mathbf{f}$.

Theorem 4. *For all $K \in \mathbb{N} \setminus \{1\}$ and for all $A \in \mathcal{A}_K$, there exists a subsequence of natural numbers $\{t_n\}_{n\in\mathbb{N}}$ such that for large n we need at least*

$$\left[\frac{(K-1)!}{2}t_n\right]^{1/K}$$

drops to determine the $\mathbf{f}$ if $\mathbf{f}$ is at t_n.

Proof. Fix $K \in \mathbb{N} \setminus \{1\}$. Let us , for the sake of contradiction, assume that the statement is false for this K, i.e. $\exists$ an $A \in \mathcal{A}_K$ such that for large enough values of $\mathbf{f}$ (say $\mathbf{f} \geq$ M) this algorithm takes at most

$$\left[\frac{(K-1)!}{2}\mathbf{f}\right]^{1/K}$$

many drops to determine $\mathbf{f}$.

Define the cumulative sum,

$$S_n = \sum_{i=1}^{n} a_i \; \forall \; n \in \mathbb{N}, \; S_0 = 0$$

Then, we must have

$$\limsup_{n\to\infty} a_n = \infty$$

as, otherwise, if a_n is bounded by some real number R we have

$$D_t > N_t \geq \frac{t}{R}$$

where, D_t denotes number of drops to determine $\mathbf{f}$ if $\mathbf{f}$ is at t using A and $N_t = \arg\min_{n\in\mathbb{N}}\{S_n \geq t\}$. Then, our assumption for the sake of contradiction false.

Thus,

$$\lim_{n\to\infty} a_{m_n} = \infty; \quad m_n = \arg\max_{1\leq u\leq n}\{a_u\}$$

Define,

$$r_n = \arg\max_{S_{n-1}<t\leq S_n} \{D_t\}$$

Then, for large enough n, $r_n \geq M$. So,

$$D_{r_n} < \left[\frac{(K-1)!}{2}r_n\right]^{1/K}.$$

As, $D_t > N_t \; \forall \; t$ and $N_{r_n} = n$ we have

$$n < \left[\frac{(K-1)!}{2}r_n\right]^{1/K}$$

Since, $S_n \geq r_n$ and $S_n \leq n a_{m_n}$, we have

$$a_{m_n} \geq \frac{S_n}{n} > \frac{r_n^{1-1/K}}{\left[\frac{(K-1)!}{2}\right]^{1/K}}$$

Hence, by Theorem 1, $\mathbf{f} = r_{m_n}$,

$$\begin{aligned} D_{r_{m_n}} - m_n &\geq [(K-1)!a_{m_n}]^{\frac{1}{K-1}}(1+o(1)) \\ &> 2^{\frac{1}{K-1}}\left[\frac{(K-1)!}{2}r_n\right]^{\frac{1}{K}}(1+o(1)) \\ &\geq 2^{\frac{1}{K-1}}\left[\frac{(K-1)!}{2}r_{m_n}\right]^{\frac{1}{K}}(1+o(1)) \end{aligned}$$

Setting $t_n = r_{m_n}$, results in contradiction. □

Algorithm 2: Deterministic Algorithm

Data: k [balls] ≥ 2
Result: $\mathbf{f}$, d [number of drops required]
$\mathbf{f} \leftarrow$ **NULL**;
$t \leftarrow 0$;
$\mathbf{U} \leftarrow -1$;
$\mathbf{L} \leftarrow 1$;
while $\mathbf{U} < 0$ **do**
  $t \mathrel{+}= 1$;
  drop 1-st ball on s_t th floor;
  if *ball doesn't break* **then**
    $\mathbf{L} = s_t$;
  else
    $\mathbf{U} = s_t$;
  end
end
Apply Algorithm 1 on the range $[\mathbf{L}+1, \mathbf{U}]$ to get $\mathbf{f}$ and add t to d to get the total number of drops required.

6 Randomized Algorithms for the Egg-Drop Problem

In this section, we explore the randomized variant of the classical egg-drop problem. Our goal is to identify the optimal randomized algorithm that minimizes the expected number of drops required to determine $\mathbf{f}$.

Definition 3. Let $\mathcal{A}_K^*$ represent the set of all randomized algorithms that use at most k balls to identify $\mathbf{f}$.

Lemma 6.1. *Consider a tower of height n and k balls. Then for any algorithm $A \in \mathcal{A}_k$, there exists $N \in \mathbb{N}$, such that for all $n > N$,*

$$S_n^A \geq 0.0875 \cdot k!^{1/k} n^{1+1/k}$$

where S_n^A represents the total number of drops required by algorithm A over all possible values of $\mathbf{f}$ in a tower of height n.

Proof. Let $1 \leq d' < d < f_k(n)$. Then for all $A \in \mathcal{A}_k$ at least $F(d,k) - F(d',k)$ values of $\mathbf{f}$ require at least $d' - d$ drops to precisely determine $\mathbf{f}$. Hence,

$$S_n^A \geq (d - d')\left(F(d,k) - F(d',k)\right)$$

Taking $d' = \dfrac{d}{2}$, we get

$$\begin{aligned} S_n^A &\geq \left(d - \frac{d}{2}\right)\left(F(d,k) - F\left(\frac{d}{2},k\right)\right) \\ &= \frac{d}{2}\left(F(d,k) - F\left(\frac{d}{2},k\right)\right) \end{aligned}$$

By Lemma 3.2,

$$\lim_{d\to\infty} \frac{F(d,k)}{\dfrac{d^k}{k!}} = 1$$

Hence, there exists $D \in \mathbb{N}$ such that for all $d > D$,

$$0.9\frac{d^k}{k!} \leq F(d,k) \leq 1.1\frac{d^k}{k!}$$

Thus for all $d > D$,

$$\begin{aligned} S_n^A &\geq \frac{d}{2}\left(0.9\frac{d^k}{k!} - \frac{1.1}{2^k}\frac{d^k}{k!}\right) \\ &= \frac{d^{k+1}}{k!}\frac{1}{2}\left(0.9 - \frac{1.1}{2^k}\right) \geq 0.175 \cdot \frac{d^{k+1}}{k!} \end{aligned} \tag{6.1}$$

By (3.4), we get

$$\lim_{n\to\infty} \frac{f_k(n) - 1}{(k!n)^{1/k}} = 1$$

which implies

$$\lim_{n\to\infty} \frac{(f_k(n) - 1)^{(k+1)}}{(k!n)^{(k+1)/k}} = 1$$

Thus, there exists $N' \in \mathbb{N}$, such that for all $n > N'$

$$(f_k(n) - 1)^{(k+1)} \geq \frac{1}{2}(k!n)^{1+1/k}$$

Setting $d = f_k(n) - 1$ in (6.1), yields that there exists $N \in \mathbb{N}$, such that for all $n > N$

$$S_n^A \geq 0.0875 \cdot \frac{(k!n)^{1+1/k}}{k!} = 0.0875 \cdot k!^{1/k} n^{1+1/k}$$

□

Theorem 5. *For any randomized algorithm $A^* \in \mathcal{A}_K^*$, the expected number of drops required in a tower of height n is at least*

$$\Omega\left((K!n)^{1/K}\right)$$

Proof. Let $A \in \mathcal{A}_\mathcal{K}$. So by lemma 6.1

$$S_n^A \geq 0.0875 \cdot K!^{1/K} n^{1+1/K}$$

Assume the inputs are uniformly distributed. Then

$$\mathbb{E}_{x\sim U}(c(A, x)) = \frac{S_n^A}{n} \geq 0.0875(K!n)^{1/K}$$

for any such algorithm. Then,

$$\min_{A\in\mathcal{A}_\mathcal{K}} \mathbb{E}_{x\sim U}\left[c(A, x)\right] \geq 0.0875(K!n)^{1/K}$$

So using Yao's principal we get our desired result. □

Theorem 6. *Now consider the scenario in which the tower height is unknown. Then for any randomized algorithm to determine* $\mathbf{f}$ $\exists\{t_n\}_{n>0}$*(where,* $\lim_{n\to\infty} t_n = \infty$*) and* $c > 0$ *such that if* $\mathbf{f}$=t_n*, then the algorithm requires at least*

$$c(K!t_n)^{\frac{1}{K}}$$

many drops to determine $\mathbf{f}$*.*

Proof. Let $K^{n-1} + 1 <$**f**$\leq K^n$.If $A \in \mathcal{A}_\mathcal{K}^*$ is an algorithm to determine **f**. Then without loss of generality we may assume A drops the balls between the given range.Now using previous theorem the worst case expected number of drops to determine **f** using A is lower bounded by

$$0.0875(K!K^{n-1}(K-1))^{\frac{1}{K}}$$

i.e.

$$0.0875(1 - \frac{1}{K})^{\frac{1}{K}}(K!K^n)^{\frac{1}{K}}$$

now **f**$\leq K^n$ So $\exists u \in \{K^{n-1} + 1, \cdots, K^n\}$ for which we need atleast $0.0875(1 - \frac{1}{K})^{\frac{1}{K}}(K!u)^{\frac{1}{K}}$ expected number of drops to determine **f**.
So our claim has been proved. □