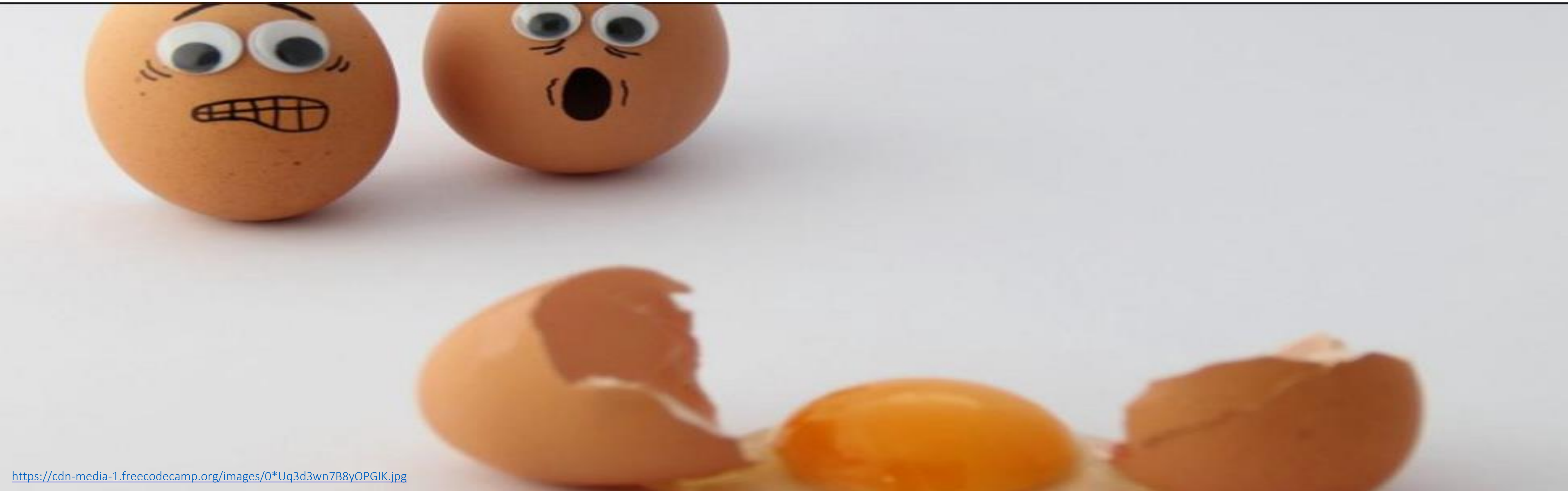
Egg Drop Mathematics:
It IS all its cracked up to be!

Steven J Miller, Williams College (sjm1@williams.edu)
https://web.williams.edu/Mathematics/sjmiller/public_html/



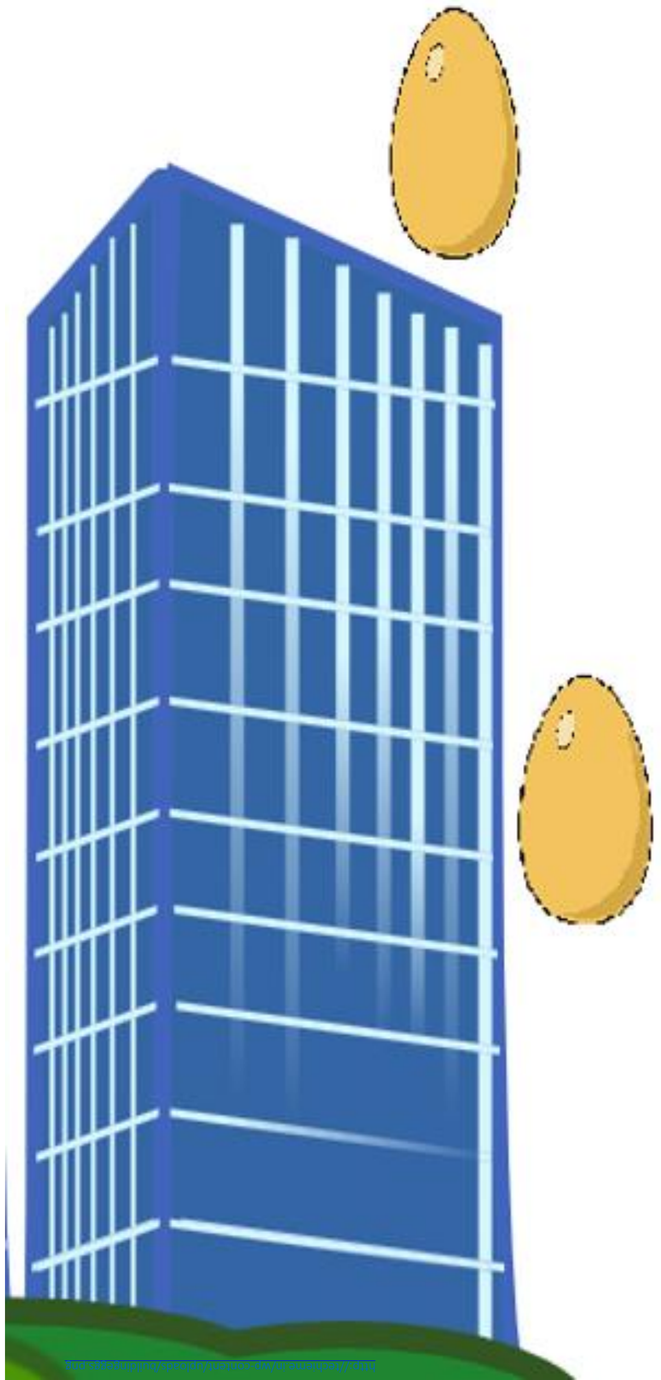
https://cdn-media-1.freecodecamp.org/images/0*Uq3d3wn7B8yOPGIK.jpg

Building with N floors, have 2 golden eggs.

Special eggs: some floor n such that if you drop from below n no damage; can drop as many times as wish.

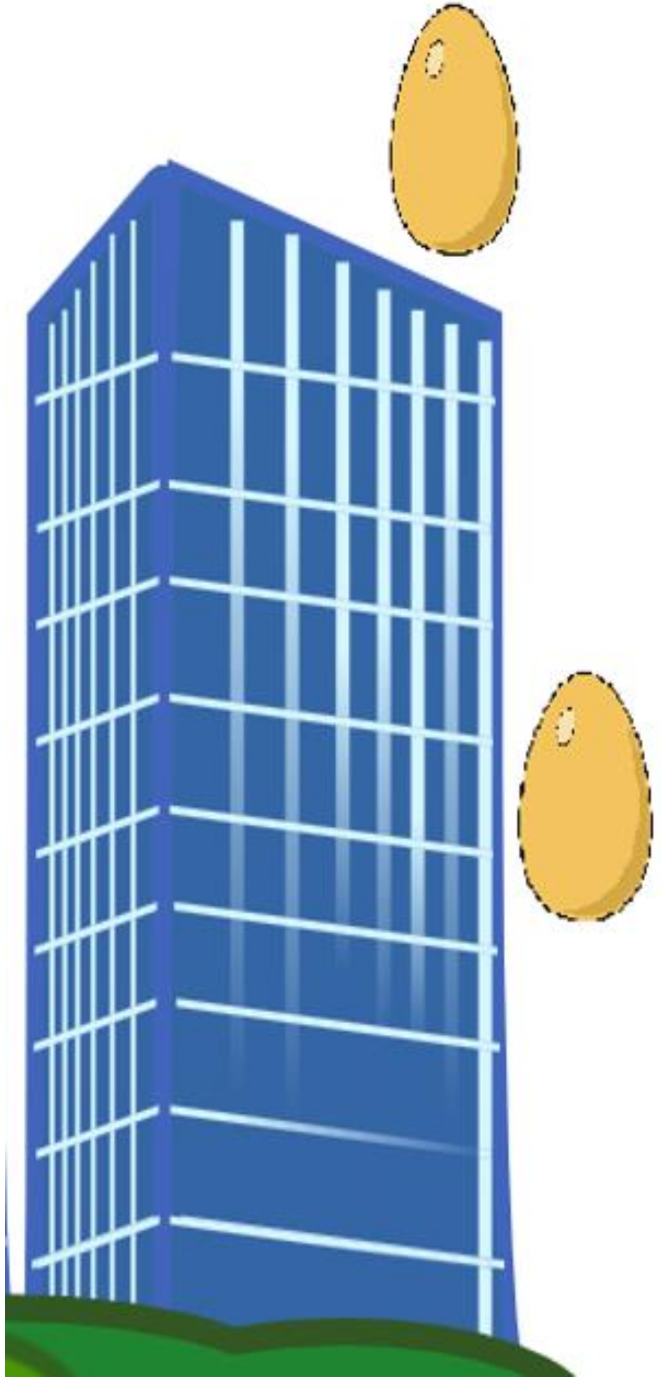
If drop even once from floor n or higher immediately break.

Find in as few drops as you can what n is (the lowest floor where if you drop from there it breaks). Doesn't matter if have any of the golden eggs at the end - just want to know n.



Interpretation:

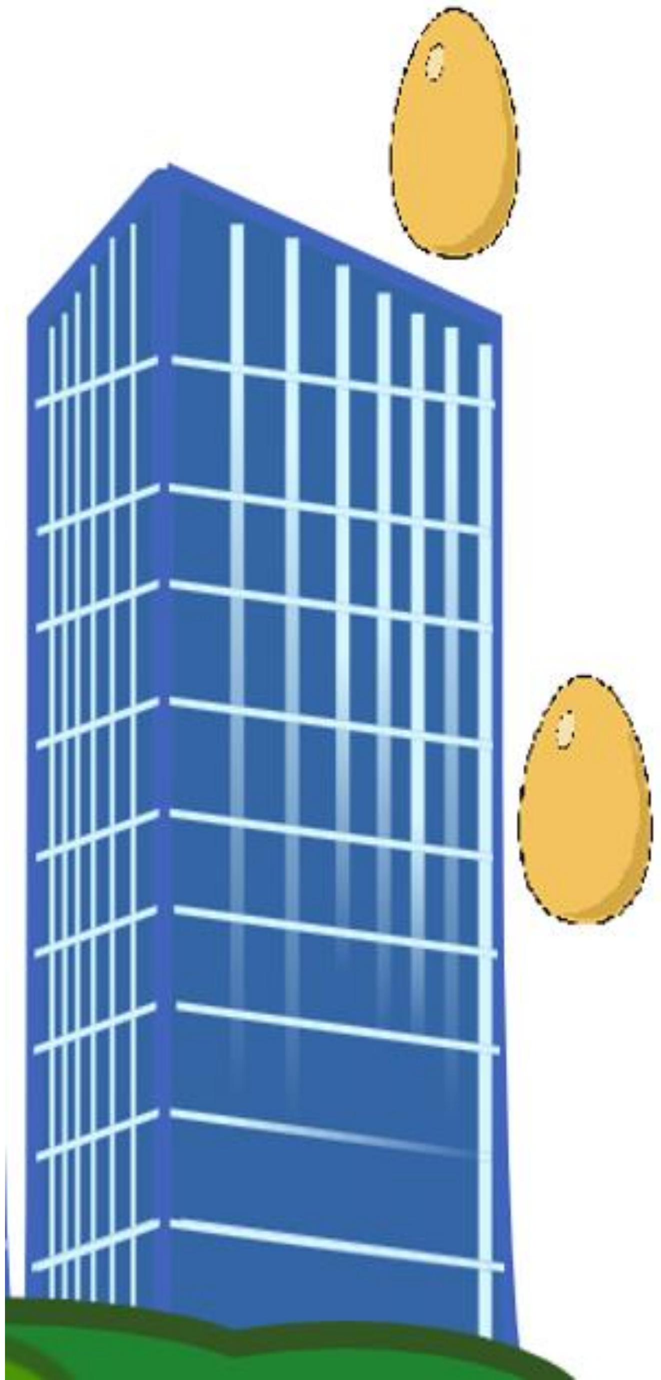
How do you interpret finding n in as few drops as possible?



Interpretation:

How do you interpret finding n in as few drops as possible?

- Minimize worse case.
- Minimize average case.

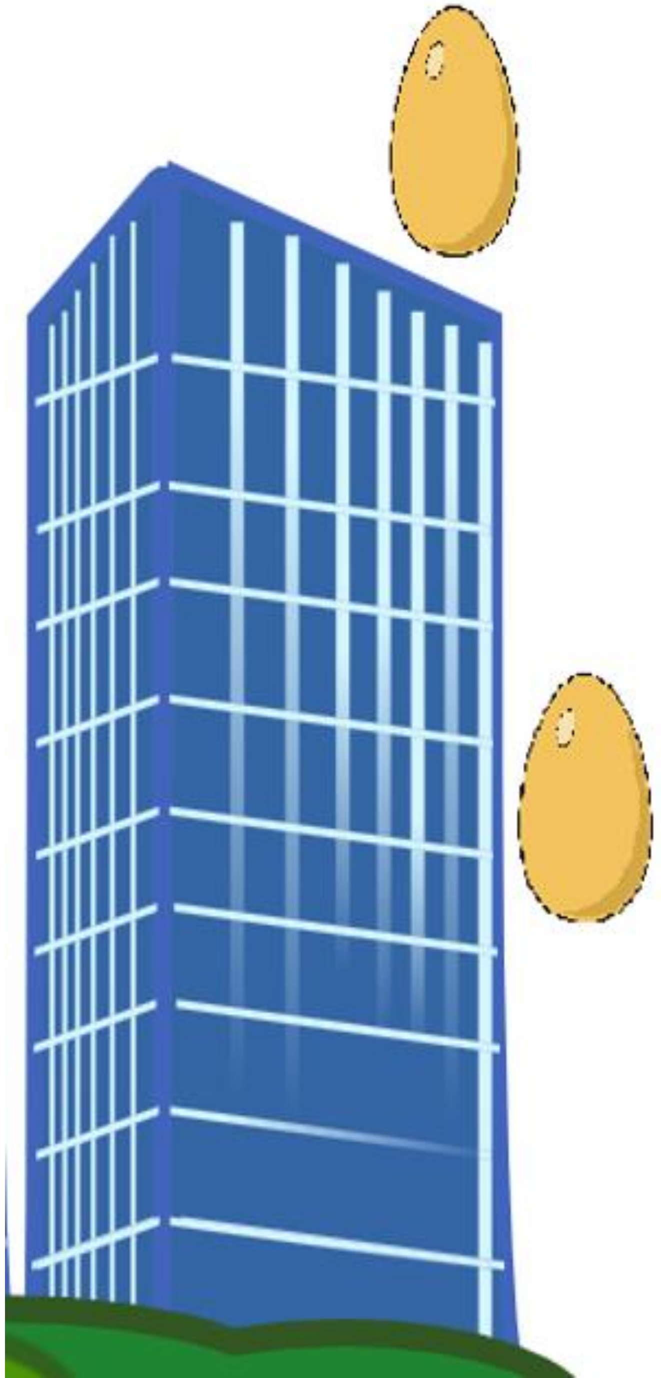


General Advice:

When given a hard problem:

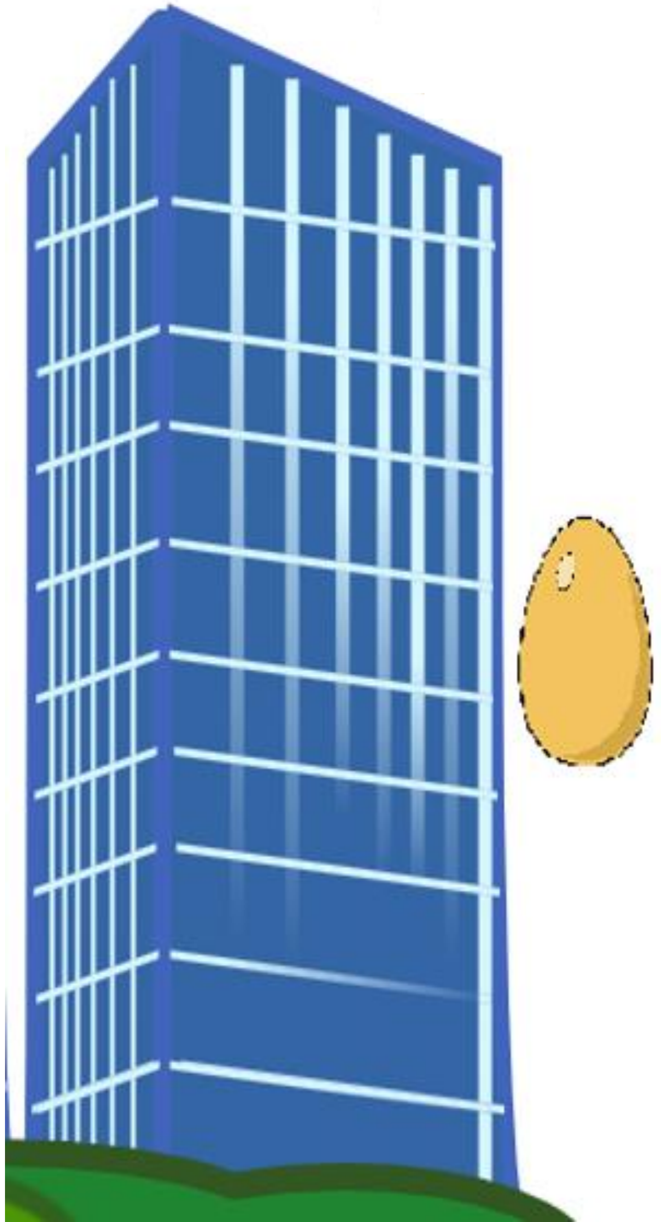
- try to do an easier version first, and
- try to do specific values of parameters.

What is an easier problem?



Simple Case: 1 Egg

What is the solution?

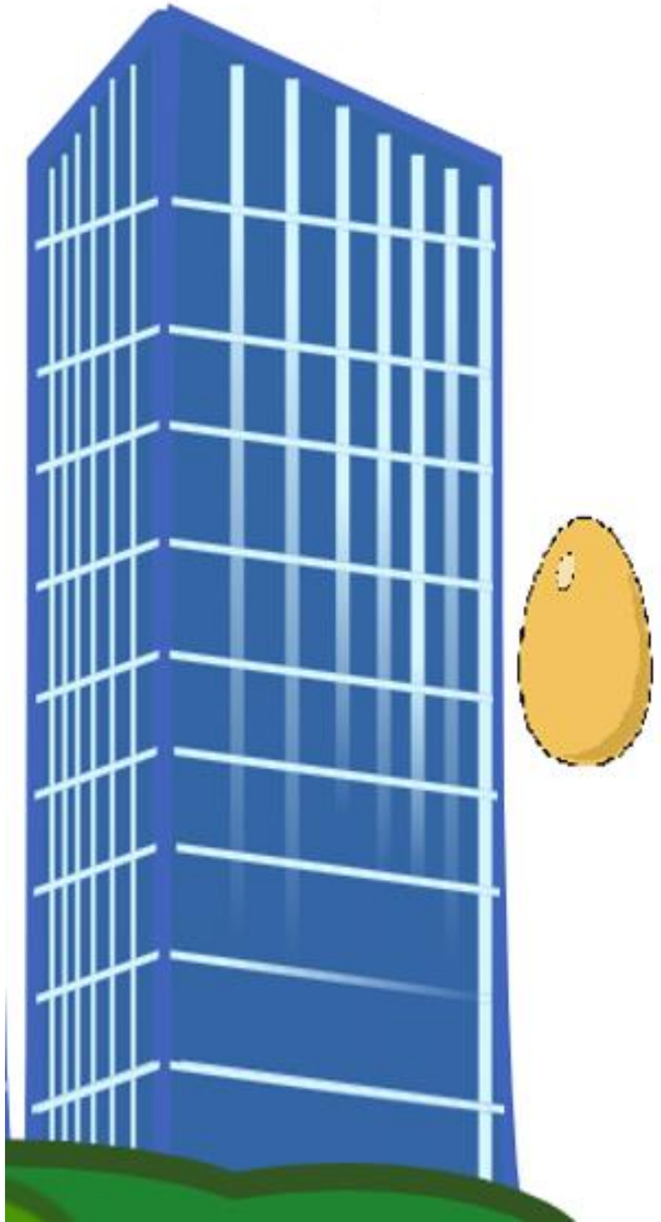


Simple Case: 1 Egg

What is the solution?

Only possibility is go 1, 2, 3, … till break.

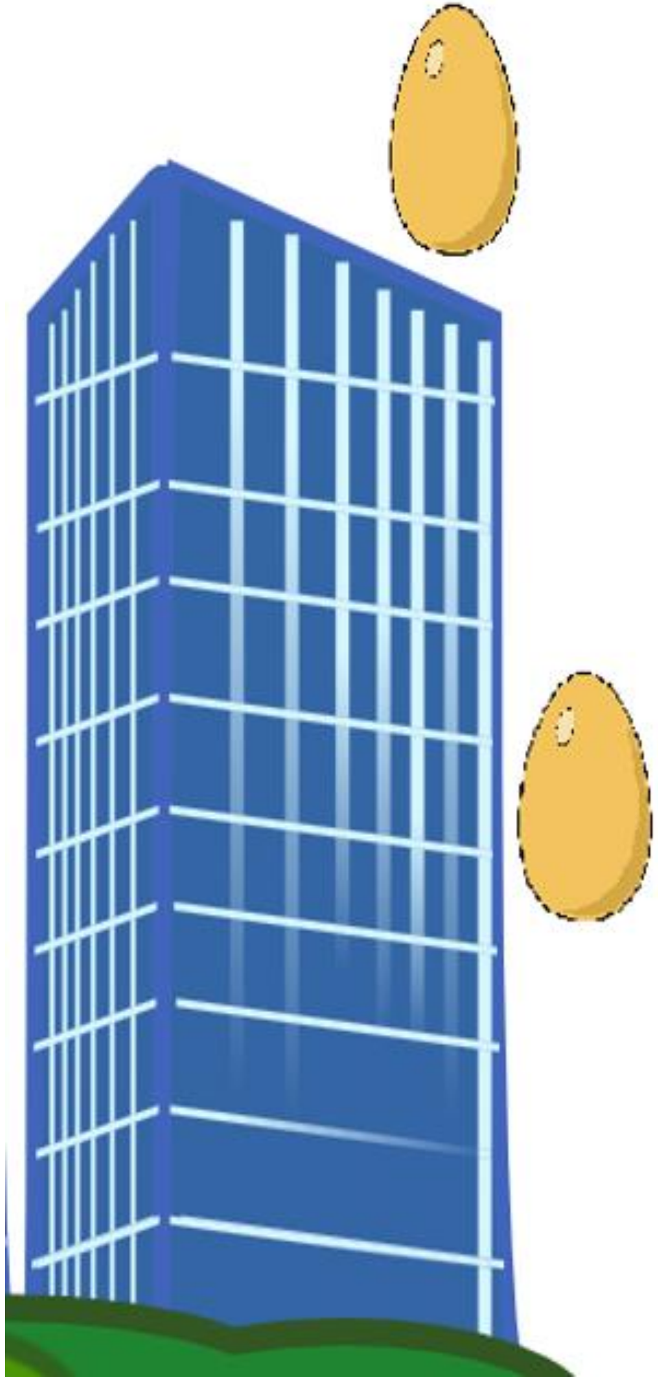
Worse case is order N drops.



Next Case: 2 Eggs

Once one cracks, reduced to 1 egg case.

What are possible strategies?



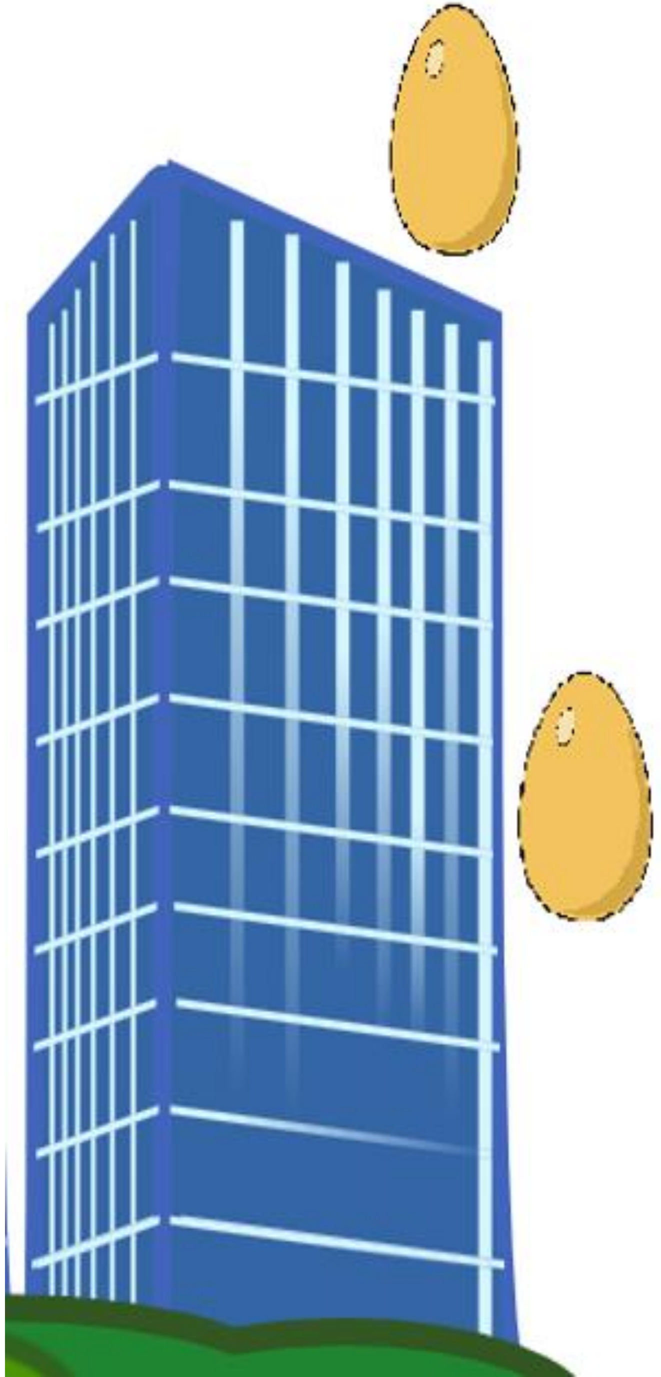
Next Case: 2 Eggs

Once one cracks, reduced to 1 egg case.

What are possible strategies?

Extreme cases:

- Drop every 2nd floor.
- Drop at N/2.
- (more generally drop every x)



Competing Influences

Drop every 2^{nd} floor.

- Once first breaks fast, but could take many drops.
- #Drops = N/2 + 1

Drop at N/2

- If doesn’t crack eliminate a lot, when crack lot to check.
- #Drops = 1 + (N/2 – 1).

Both basically on the order of N/2 drops….

Competing Influences: Balance

Drop every x floors.

- Could take N/x before first breaks, then x-1.
- #Drops = (N/x) + (x-1) = (N/x + x) – 1.

Reduced to choosing x to minimize

$$\frac{N}{x} + x.$$

Thoughts on how to do this?

Competing Influences: Balance

Reduced to choosing x to minimize

$$\frac{N}{x} + x.$$

Set two terms equal to each other to balance:

$$\frac{N}{x} = x \quad so \;\; N = x^2 \quad or \;\; x = N^{1/2}.$$

Competing Influences: Balance

Reduced to choosing x to minimize

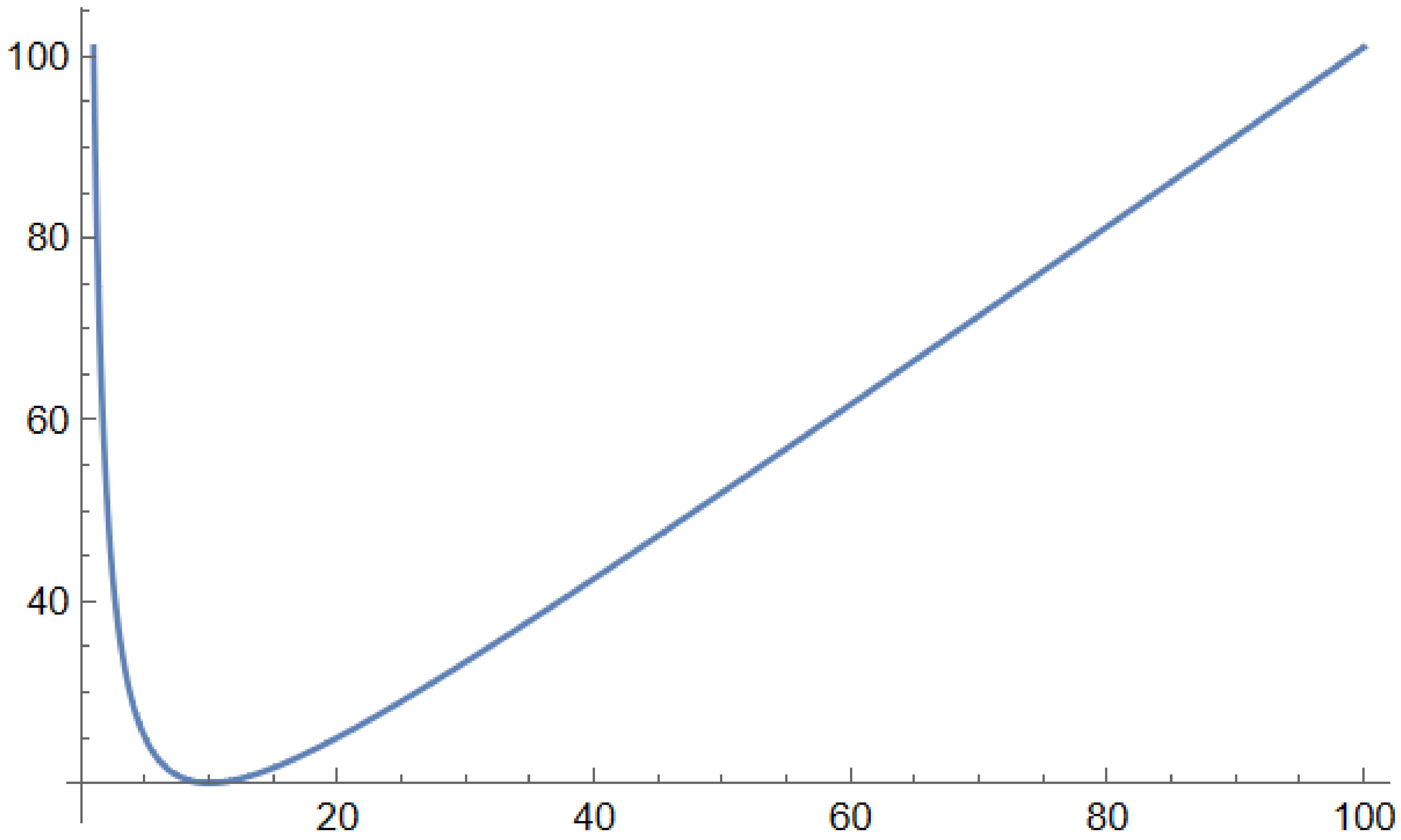
$$\frac{N}{x} + x.$$

Set two terms equal to each other to balance:

$$\frac{N}{x} = x \quad so \;\; N = x^2 \quad or \;\; x = N^{1/2}.$$

Gives #Drops = $\frac{N}{N^{1/2}} + N^{1/2}$ - 1 or about 2 $N^{1/2}$.

```
Plot[100 / x + x, {x, 1, 100}]
```

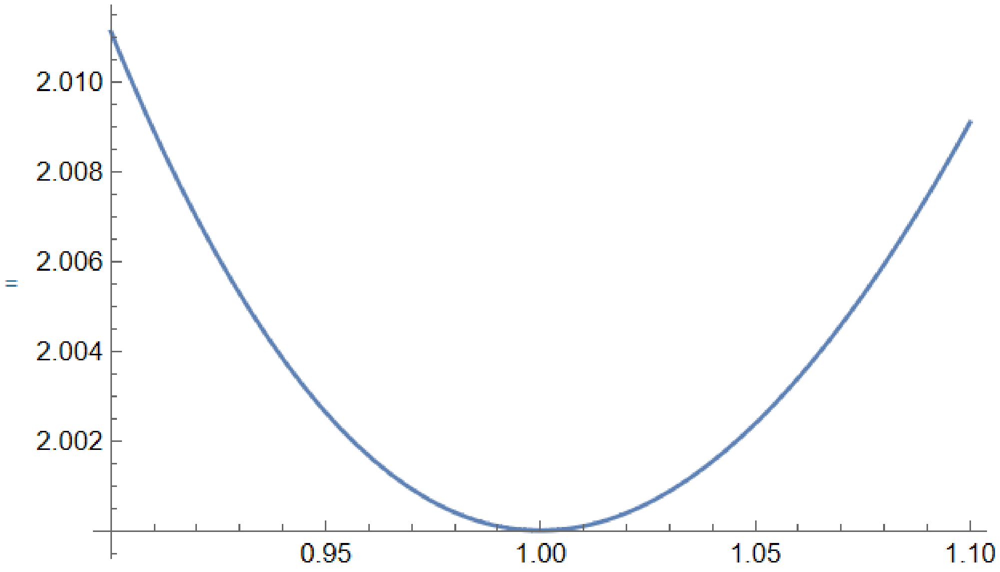


Write $x = t\,N^{1/2}$ in #Drops = $\frac{N}{x} + x - 1$.

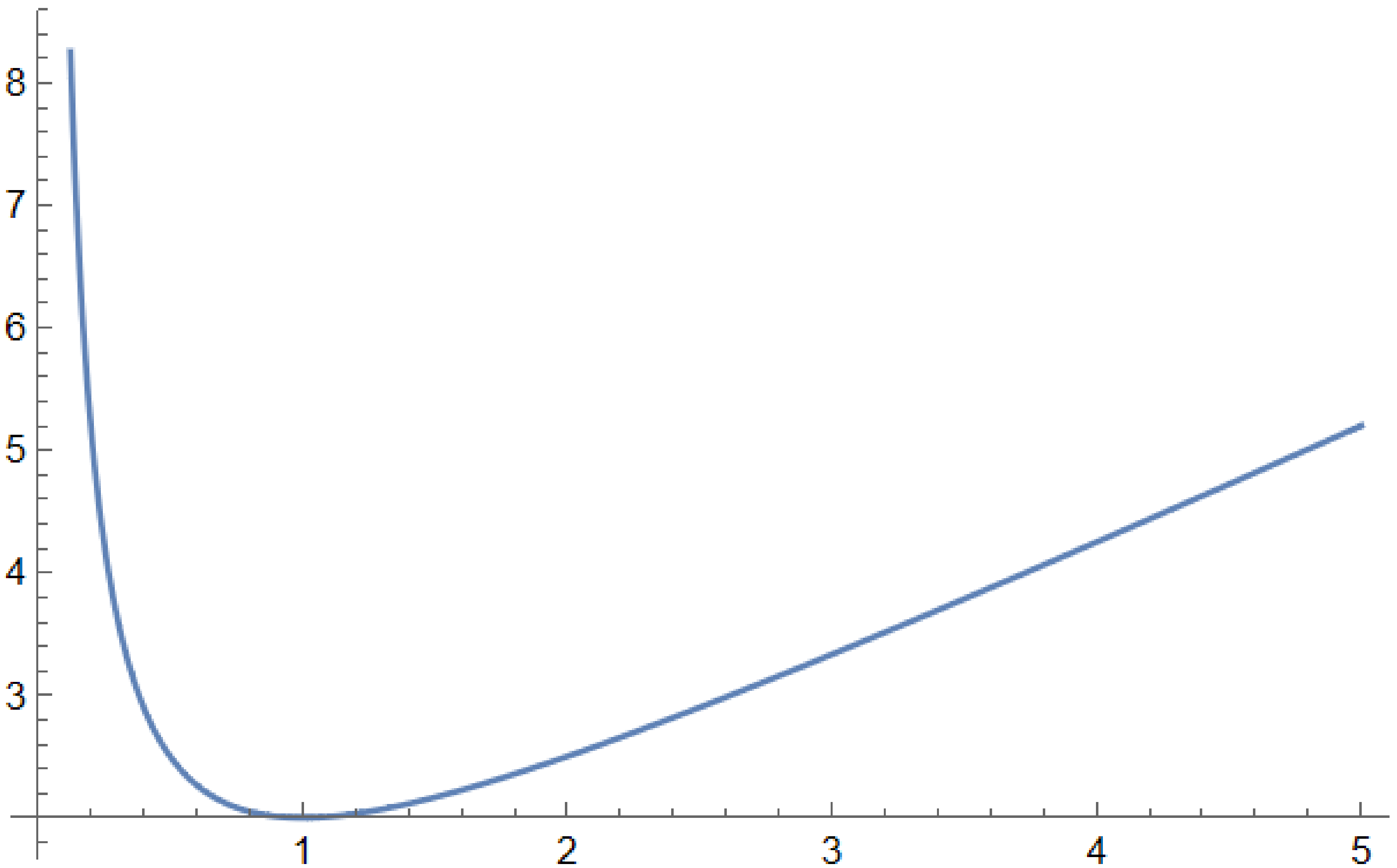
Gives #Drops = $\frac{N}{t\,N^{1/2}} + t\,N^{1/2}$ - 1.

This is just $N^{1/2}\,(\frac{1}{t} + t)$,

so on the order of $N^{1/2}$!




```
Plot[1 / t + t, {t, 0, 5}]
```



If know calculus: want to minimize $f(x) = N/x + x$:

- Endpoints: $f(1)$ and $f(N)$ are of order N.
- $f'(x) = -N/x^2 + 1$, critical point $f'(x) = 0$ or $x = N^{1/2}$.
- Easily see minimum, or note $f''(x) = 2N/x^3 > 0$.

Balancing Application

Imagine have two algorithms:

- One always takes 1000 seconds.
- One takes 1 second except one in a million inputs take 1,000,000,000 seconds.

Both take on average approximately 1000 seconds....

Balancing Application

Imagine have two algorithms:

- One always takes 1000 seconds.
- One takes 1 second except one in a million inputs take 1,000,000,000 seconds.

Both take on average approximately 1000 seconds....
...but what if run algorithm 1 and if takes more than 2 seconds on an input switch to first? Average of about 1 second!

Improving Strategy with 2 Eggs

Consider triangular numbers and dynamic rescaline.

- Do not move in constant steps of x floors.
- Do x, then x-1 if doesn't crack, then x-2….
  - Advantage is always same number of drops!
  - Basically if doesn't crack doing 2 egg problem but now with N-x floors (after first drop).

Improving Strategy with 2 Eggs
Consider triangular numbers and dynamic rescaline.

- Do not move in constant steps of x floors.
- Do x, then x-1 if doesn't crack, then x-2....
  - Advantage is always same number of drops!
  - Basically if doesn't crack doing 2 egg problem but now with N-x floors (after first drop).

Example: N = 105 = 14 + 13 + 12 + … + 1:

(1 + 13) or (2 + 12) or (3 + 11) ….

All are 14 drops, better than 2 $105^{1/2}$ (about 20).

What if we have 3 Eggs? Or k eggs?

What if we have 3 Eggs? Or k eggs?

For 3 eggs: once one cracks, 2 egg problem.

If do every x it would be, worse case: