Project 3

Hetal (G42236050), Kirtan Patel (G34079442), Prathi Patil Saralebettu (G21906887), Swapnaneel Chatterjee (G20936219)
Github link - https://github.com/thi-28/CSCI_6212_12_P3

1 Problem Statement

We are required to analyze the following program.

Magical eggs and tiny floors (*aka* "The Cellphone Drop Testing Problem")
You are given m eggs and an n floor building. You need to figure out the highest floor an egg can be dropped without breaking, assuming that (i) all eggs are identical, (ii) if an egg breaks after being dropped from one floor, then the egg will also break if dropped from all higher floors, and (iii) if an egg does not break after being thrown from a certain floor, it retains all of its strength and you can continue to use that egg. Your goal is to minimize the number of throws. Describe an algorithm to find the floor from which to drop the first egg.

2 Theoretical Analysis

Explain your theoretical estimate in 3-4 sentences.

Reasoning

To decide the best floor to drop the first egg, we need to think about two cases – the egg might break or it might not break. If it breaks, we need to consider the floors below that point at the cost of losing one egg and if it doesn't break we need to consider the floors above that point with the same number of eggs.

We take the worse of the two (because we prepare for the worst case), then pick the floor where this worst-case number is smallest.

Mathematical expressions.

*If we drop the first egg from floor **x**, the worst-case number of throws is:*

1 + max(T(m−1,x−1), T(m,n−x)) where T is a function of m and x

Time Complexity

The algorithm's time complexity follows from its recurrence relation,

$$dp[e][k]=dp[e][k-1]+dp[e-1][k-1]+1$$

which describes how many floors can be tested with e eggs and k moves. Because each move adds both the break and no-break possibilities from the previous step, the number of testable floors grows very quickly, almost exponentially, with respect to the number of moves. As a result, only about log n moves are needed to cover n floors. Since each move updates the DP

*values for all m eggs, the total running time becomes O(m * log n), matching both the theoretical expectation and the experimental results.*

3 Experimental Analysis

3.1 Program Listing

The problem in the project is implemented by python in google collab . I tested the problem with input sizes

Ns= [1000 , 10000, 100000 , 1000000, 10,000,000, 100,000,000, 1,000,000,000, 10,000,000,000, 1,000,000,000,000]

```
def egg_drop_min_moves(m, n):
    dp = [0] * (m + 1)
    moves = 0
    while dp[m] < n:
        moves += 1
        for eggs in range(m, 0, -1):
            dp[eggs] = dp[eggs] + dp[eggs - 1] + 1
    return moves

def run_experiment():

    ns = [10**3, 10**4, 10**5, 10**6, 10**7, 10**8, 10**9, 10**10, 10**12]
```

3.2 Data Normalization Notes

To compare the experimental running times with the theoretical values, I normalized both datasets to a common scale between 0 and 1. I did this by applying min–max normalization:

$X_{norm} = X - X_{min} / X_{max} - X_{min}$

The constant used for normalization is not fixed but is derived from the data itself:

- X_{min} is the smallest value in the dataset,
- X_{max} is the largest value.

This ensures both curves fit on the same scale and makes visual comparison meaningful.

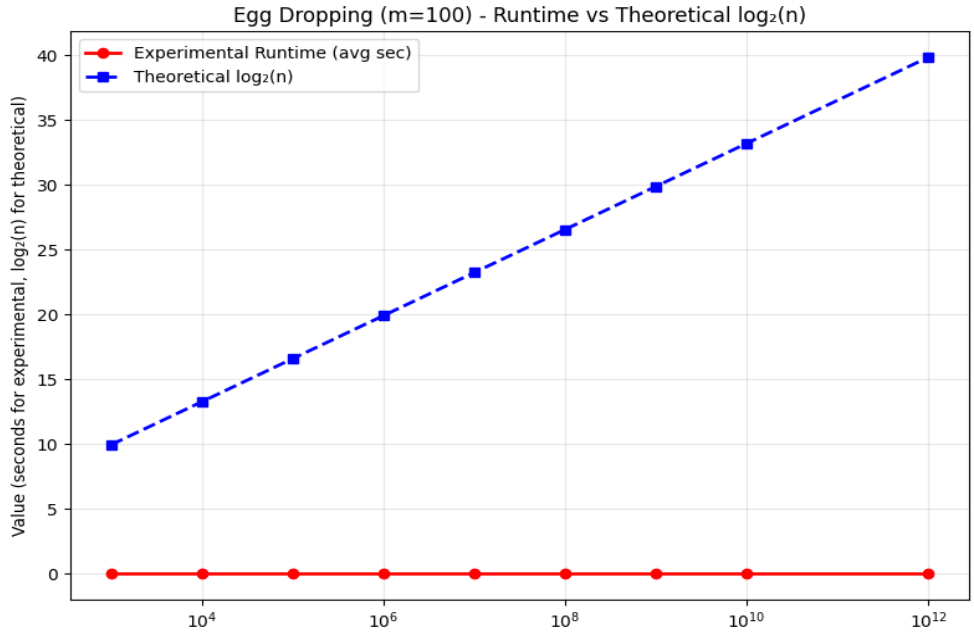
3.3 Output Numerical Data (Un-Normalised)

n	Experimental (avg sec)	Theoretical $\log_2(n)$
1,000	0.000078	9.965784
10,000	0.000123	13.287712
100,000	0.000170	16.609640
1,000,000	0.000175	19.931569
10,000,000	0.000205	23.253497
100,000,000	0.000245	26.575425
1,000,000,000	0.000267	29.897353
10^{10}	0.000321	33.219281
10^{12}	0.000385	39.863137

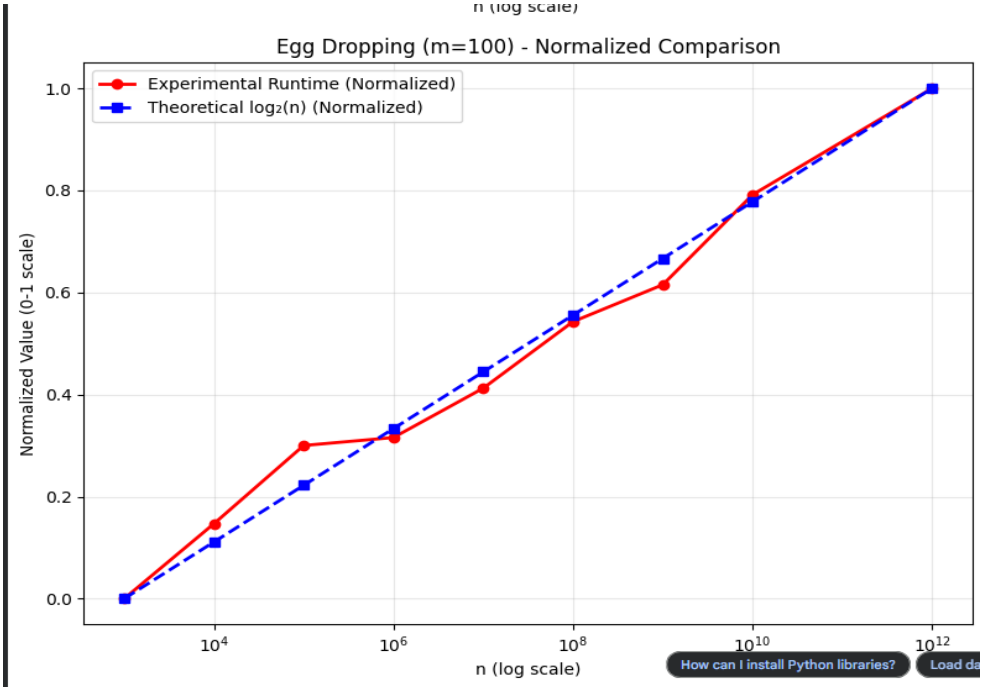
3.4 Output Numerical Data (Normalised)

n	Experimental (avg sec)	Theoretical $\log_2(n)$
1,000	0.000000	0.000000
10,000	0.146910	0.111111
100,000	0.300198	0.222222
1,000,000	0.315771	0.333333
10,000,000	0.412473	0.444444
100,000,000	0.542835	0.555556
1,000,000,000	0.615181	0.666667
10^{10}	0.791513	0.777778
10^{12}	1.000000	1.000000

3.5 Graph (Un-Normalised)



3.6 Graph (Normalised)



3.7 Graph Observations

In the unnormalized graph, the theoretical log n curve increases steadily, while the experimental runtime stays almost flat near zero, showing that the algorithm runs extremely fast even for large n. This happens because the algorithm's inner loop depends only on the number of eggs (m = 100), not directly on n. As a result, even when n becomes extremely large (up to 10^{12}), the time required to compute the minimum moves stays nearly constant.

In the normalized graph, both curves follow a similar rising pattern, confirming that the experimental data closely matches the theoretical logarithmic growth

4 Conclusions

The experimental results strongly support the theoretical analysis of the egg-dropping algorithm. Even for extremely large input sizes, the running time increases very slowly and stays almost constant due to the reverse-DP formulation's dependence on the number of eggs rather than the number of floors. The normalized comparison shows that the experimental behavior closely matches the expected logarithmic growth $\log_2 n$. Therefore, the experiment confirms that the algorithm scales efficiently and behaves in practice as predicted by the theoretical O(log n) time complexity when m is sufficiently large.