CSCI_6212_11 - Project 3 - Magical Eggs and Tiny Floors (*aka* The Cell Phone Drop Testing Problem)

Team 10 - Namana Y Tarikere – G21372717, Siddhesh Kocharekar – G41505497, Parithosh Dharmapalan – G27479699, Pooja Srinivasan – G33105018

1. Problem Statement: We are required to analyse the below problem statement:
Input: 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.
Objective: The goal is to minimize the number of throws and to describe an algorithm to find the floor from which to drop the first egg.

2. Problem Analysis: The above problem statement can be solved using binary search with a dynamic programming approach. In this implementation, we use n to represent the number of eggs and k denotes the number of floors. The function minimumAttempts(n, k) is responsible for determining the minimum number of attempts required to find the floor from which an egg can be dropped without it breaking. As for the efficiency of this algorithm, it operates with a time complexity of O(nk). The pseudocode of this algorithm is given below:

```
def minimumAttempts(n, k):
    # Initialize 2D array of size (k+1) * (n+1).
    dp = [[0 for x in range(n + 1)] for y in range(k + 1)]
    m = 0 # Number of moves
    while dp[m][n] < k:
        m += 1
        for x in range(1, n + 1):
            dp[m][x] = 1 + dp[m - 1][x - 1] + dp[m - 1][x]
    return m
```

3. Time Complexity:
In the above algorithm, the outer while loop will run k times and the inner while loop will run n times. Hence, the overall time complexity of the algorithm will be **O(n * k)**.

Example Test Case: With 3 eggs and 5 floors, we follow the provided pseudocode. If the first egg breaks on the 3rd floor, we check lower floors, potentially using all 3 eggs (minimum 3 throws). If the first egg doesn't break on the 3rd floor, we still have 3 eggs, and even if the worst-case happens (egg breaks on the 5th floor), we still need a minimum of 3 throws.

4. Numerical Results

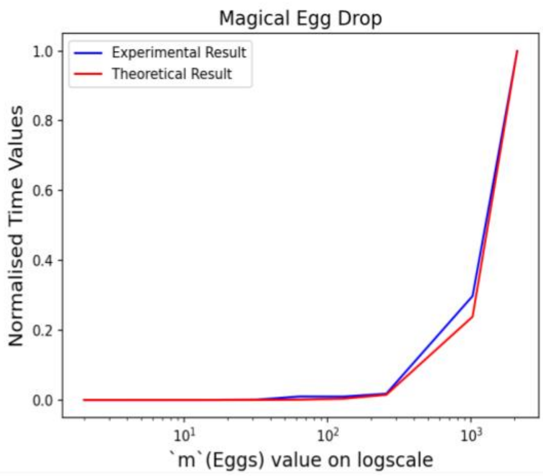
Average of Experimental results is: **26654976.15**
Average of Theoretical result is: **552917.2**
Scaling constant = Average of Experimental result / Average of Theoretical result
Therefore, Scaling Constant for normalizing the theoretical results = **48.20789831**

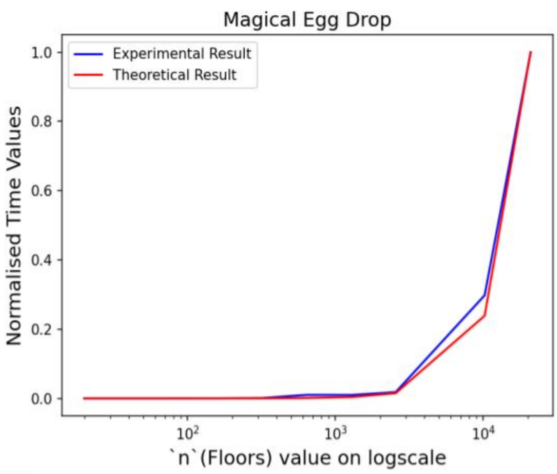
4.2 Output Numerical Data

Eggs (n)	Floors (k)	Experimental Result in ns	Theoretical Result (n * k)	Scaling Constant	Scaled Theoretical Result
2	20	15631.8	40		1928.315932
4	40	16159.2	160		7713.26373
8	80	15721.3	640		30853.05492

16	160	78131.6	2560		123412.2197
32	320	299124	10240		493648.8787
64	640	1046687.3	40960		1974595.515
128	1280	8627083.6	163840		7898382.059
256	2560	34754853.5	655360		31593528.24
1024	10240	670861589	10485760		505496451.8
2096	20960	1949782634	43932160		2117877102
	Average	**26654976.15**	**552917.2**	**48.20789831**	

5. Graph and Observations





The plot of the experimental results **increases exponentially** along with the increase in theoretical result at all the points on the graph for respective increase in values of eggs and floors.

6. Conclusions

The line plots of the experimental and theoretical analysis are intersecting and seem to be convergent. Hence the Big O notation for the above problem statement using dynamic programming is **O(n * k)** where n is the number of eggs, and k is the number of floors.

7. Github Link: https://github.com/namanayt/CSCI_6212_11_Algorithms_Project_3_Team_10