# ANALYSIS OF EGG DROPPING PROBLEM

##### **Anvita Gupta**

18070124013

***Abstract*-** This study provides a deep insight into the analysis of a very famous problem ‘Egg dropping Problem’. In this article the reader will be able to learn about the constraints of this problem along with all four different cases available. All the different ways of solving each and every particular type will be studied thoroughly. We will deeply look at all the algorithms and perform their analysis. In the end there will be a comparison based on different methods to solve this problem and their time and space complexities. Lastly, we will have a look at all the applications of this problem in the real world. Results include algorithm validation through experimentation, optimization, and final analysis.

***Index Terms***- Binary Search Method, Dynamic Programming, Egg Dropping Puzzle, Recursive Method.

1. Introduction

Is there any way to determine that at which floor an egg will break by dropping from a particular floor in a multistory building in minimum number of trials? Yes, indeed there is. This puzzle is known as Egg dropping puzzle. It is a really good problem to understand about dynamic programming. It is a really famous problem and asked in interviews of many major companies.

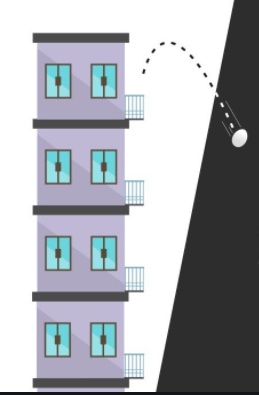
*‘Egg dropping refers to a class of problems in which it is important to find the correct response without exceeding a (low) number of certain failure states.’*

The physical property of an ideal egg is to break from floor n or above and will have no damage if thrown from floor n-1 or below. The problem is to find an optimized strategy such that the eggdropper can perform minimum drops to find the floor from which it breaks.

So summing it all up, the problem is:

**Given a certain amount of eggs and number of floors in a building, what is the least amount of egg drops one has to perform to find out the critical floor?**

**(Critical Floor is the one from which the egg starts breaking and also breaks for all the floors above. If the egg is dropped below the critical floor then it won’t break.It is also known as threshold floor.)**

1. Constraints

The constraints for the given problem are given below:

1. The effect of falling is considered to be same for all eggs.
2. If an egg breaks when dropped from a certain floor, then it would break if it is dropped from a higher floor.
3. An egg that doesn’t break after a fall can be used again.
4. An egg which breaks after falling needs to be thrown away.
5. If an egg doesn’t break from a certain floor then it would not break from floors below.

We will be finding the least amount of tries to find the critical floor and not that floor itself.

Fig 1: Demonstrating Egg Dropping Problem

1. TYPES OF THIS PROBLEM

Egg Dropping Problem has many variations like:

## A.1 Egg and K floors

This is the worst case in which there is only one egg and K number of floors in the building .

## B. 2 Eggs and K floors

In this type there are 2 eggs and K number of floors in the building .

## C. N Eggs and K floors

In this standard type there are n number of eggs and K number of floors in the building .

## D. 2 Eggs and 100 floors

In this standard type there are 2 eggs and 100 floors in the building .

1. SOLVING THE PROBLEM

***A.1 Egg and K floors***

* Firstly we take the worst case i.e. 1 Egg and K floors. Suppose K=6,there is only 1 egg to check and 6 floors to find the threshold floor. One has to try each and every floor until the egg breaks because one can’t take any risks otherwise the egg would break. Starting from first floor like shown in Fig 2 we have to try each floor so as to find the critical floor because if we take like the 5th floor for example and drop the egg from there we might

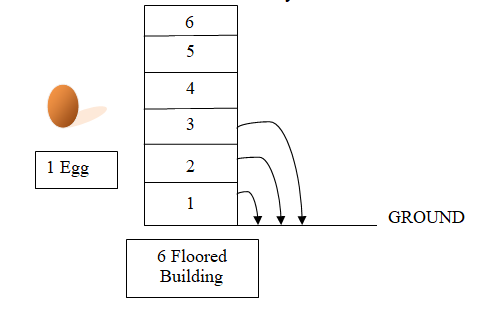


Fig 2:One egg,K floors case

end up breaking it and wasting the only egg we had and thereby we will not be able to find the threshold floor . So there will be 6 trials to find the threshold floor in the worst case. There will be f trials from a f floored building in case of 1 egg.

1egg, f floors f trials

***B.2 Eggs and K floors***

* Next case will be 2 eggs and K floors. So how do we proceed in this particular case? Let’s find out.

The first floor we are trying to say let's say x. Suppose that if that egg breaks, we will still have one egg left and we have to try everywhere from 1 to x-1 as we have seen before. Therefore the total test will become 1 + (x - 1). The next floor we try becomes x + (x - 1) because our answer is x and if the egg breaks from the number x + (x-1) we should try from the number x + 1 to x-2. If the first egg is not broken by now, its test should be from the number x + (x - 1) +… + (x - i - 1). We can see from the above that we can cover x + (x - 1) + (x - 2)…. + 2 + 1 with x trials. The value of this is

**x+(x-1)+(x-2)+(x-3)+……+2+1=**

**)**/ 2

The optimal value of x can be written as,

**x=⌈((-1 + √(1+8k))/2)⌉**

Algorithm:

1.Begin

2.Input number of floors K

3. return (int)Math.ceil((-1.0 +Math.sqrt(1 + 8 \* k)) / 2.0);

4.End

Suppose K=6, there are 2 eggs to check and 6 floors to find out the threshold floor as shown in Fig.3.

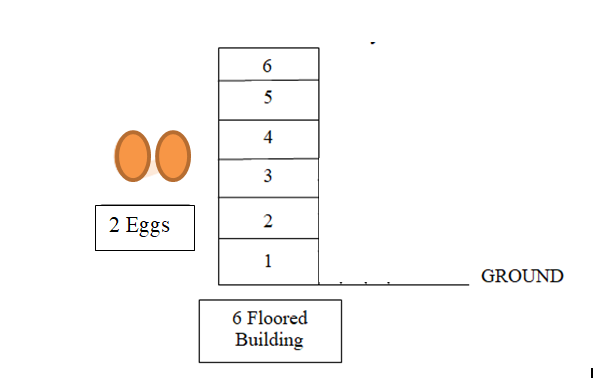


Fig 3: 2 Eggs, K floors Case

Using the given above formula we get:

x=ceil[-1+sqrt(1+48)/2.0)]=ceil[-1+3.5]=ceil[2.5]= 3

So it will take 3 trials to find out the threshold floor using the above algorithm.

**Time Complexity**: There is only one basic operation in this program . There are no loops in this program. Therefore the complexity is O(1).

***C. 2 Eggs and 100 floors***

After knowing about different cases lets come to the one more case i.e. 2 eggs and 100 floors. This is a special case of egg dropping puzzle. Suppose if we had only one egg and 100 floors we would have try each floor from first till the threshold floor. This would be a very tedious process. But in this case we have 2 eggs so let’s find out how we proceed to find the least amount of egg droppings to find the threshold floor. If we use the third method of binary search which we have seen earlier to find the floor then if we start from 50th floor we would end up doing 50 comparisons in the worst case. The worst case happens when the required threshold floor is 49th. The next solution takes a little bit of the linear approach and mixes in a little of the splitting from our binary approach. We can start off by dropping an egg at floor 10, increasing the drop floor by 10 at a time, then going back to drop one floor at a time until we find that n. If our egg breaks at floor 10, we know n is one of the 9 floors below us. Worst case, the egg drops and doesn’t break until floor 100 (10 drops) and we drop the second egg but don’t break it for floors 91–99.

It brings our worst case drop count to 19 drops. Seems relatively straightforward, but we can still improve our number of drops. Therefore, we use an optimized solution here which is as follows:

Let us make our first attempt on x th floor. If it breaks, we try remaining (x-1) floors one by one. So in worst case, we make x trials. If it doesn’t break, we jump (x-1) floors (Because we have already made one attempt and we don't want to go beyond x attempts. Therefore (x-1) attempts are available), Next floor we try is floor x + (x-1)

Similarly, if this drop does not break, next need to jump up to floor x + (x-1) + (x-2), then x + (x-1) + (x-2) + (x-3) and so on. Since the last floor to be tried is 100th floor, sum of series should be 100 for optimal value of x.

**x + (x-1) + (x-2) + (x-3) + .... + 1 = 100**

**x(x+1)/2 = 100**

**x = 13.651**

Therefore, we start trying from 14th floor. If Egg breaks we one by one try remaining 13 floors. If egg doesn't break we go to 27th floor. If egg breaks on 27th floor, we try floors form 15 to 26. If egg doesn't break on 27th floor, we go to 39th floor and so on...

The optimal number of trials is **14** in worst case. This was really a much better approach to solve this. This is the number of drops required as we move up the floors in the building:

Drop Floor

#1 14

#2 27

#3 39

#4 50

#5 60

#6 69

#7 77

#8 84

#9 90

#10 95

#11 99

#12 100

So what if the first egg breaks at floor 14? Let’s see, if the first egg breaks at the 14th  floor, then we should check the first floor, then the second one, until the 13th floor. Doing this the total number of attempts would be 14.And What if it doesn't break? Then we should check the 27th floor. Why? Because if it breaks, you would have to check all the floors from the 15th until the 26th one (thirteen floors), which keeps the total number of attempts at 14 \big((first attempt at the 14th floor, second at the 27th floor, and the twelve remaining drops from the 15th floor until the 26th floor.

And if it doesn't break, you would have to check the 39th floor; if it breaks you would have to check all the floors from the 28th until the 38th one. Remember, one attempt at the 14th floor, the second attempt at the 27th floor, the third attempt at the 39th floor, and the 11 remaining attempts at the floors {28,29,30,31,32,33,34,35,36,37} and 38,totaling 14 attempts in this case.

Using the same reasoning, you should check the 50th floor, the 60th, the 69th, the 77th, the 84th, the 90th, the 95th, the 99th and finally the 100th one. See? Using this strategy you would cover all the floors and the number of attempts would never be greater than 14, even in the worst cases.

|  |  |  |
| --- | --- | --- |
| 1st egg | If the first egg breaks -> 2nd egg | Drops |
| 14 | 1-2-3-4-5-6-7-8-9-10-11-12-13 | 1+13=14 |
| 27 | 15-16-17-18-19-20-21-22-23-24-25-26 | 2+12=14 |
| 39 | 28-29-30-31-32-33-34-35-36-37-38 | 3+11=14 |
| 50 | 40-41-42-43-44-45-46-47-48-49 | 4+10=14 |
| 60 | 51-52-53-54-55-56-57-58-59 | 5+9=14 |
| 69 | 61-62-63-64-65-66-67-68 | 6+8=14 |
| 77 | 70-71-72-73-74-75-76 | 7+7=14 |
| 84 | 78-79-80-81-82-83 | 8+6=14 |
| 90 | 85-86-87-88-89 | 9+5=14 |
| 95 | 91-92-93-94 | 10+4=14 |
| 99 | 100 | 11+1=12\* |

\*Although the number of drops here is 12 it’s not the worst case

TABLE 1: If the first egg breaks in 2eggs and 100 floors

***D. N Eggs and K floors***

* Let’s come to the main case i.e. N eggs and K floors. This is the standard case. There are different methods to solve this which are as follows:

**1. Recursion:**

This is a general solution to this problem. The solution is to try dropping an egg from every floor (from 1 to k) and then recursively calculate the minimum number of droppings needed in the worst case. The floor which gives the minimum value in the worst case is going to be part of the solution. When we drop an egg from a floor say x, there can be two cases:

(1) **The egg breaks**

--If the egg breaks after dropping from ‘x th’ floor, then we only need to check for floors that are below ‘x’ with the remaining eggs as some floor would exist lower than ‘x’ on which the egg won’t break therefore the problem reduces to (x-1) floors and (n-1) eggs.

(2) **The egg doesn’t break.**

--If the egg doesn’t break after dropping from the ‘xth’ floor, then we only need to check for floors that are above ‘x’; so the problem reduces to ‘k-x’ floors and n eggs.

Since we are minimizing the number of trials in worst case, we will take the maximum of the above two cases for every floor and choose the floor which yields minimum number of trials. Below is the algorithm:

Function eggDrop(eggs, floors)

Input: Number of eggs, maximum floor.

Output: Get a minimum number of trials.

Algorithm:

1.Begin

2. If there are no floors, then no trials needed. OR if there is one floor, one trial needed.

   if (k == 1 || k == 0)  then return k;

3. If one egg id there then there will be k trials for k floors

        if (n == 1)  then return k;

4.Consider all droppings from 1st floor to kth floor and return the minimum of these values plus1.

        for (x = 1; x <= k; x++) do

            res = Math.max(eggDrop(n - 1, x - 1), eggDrop(n, k - x));

            if (res < min)

                min = res;

        done

   return min + 1;

 5.End

This algorithm computes the same problem again and again therefore its recursive. The partial recursion tree is drawn below for 4 floors and 2 eggs:

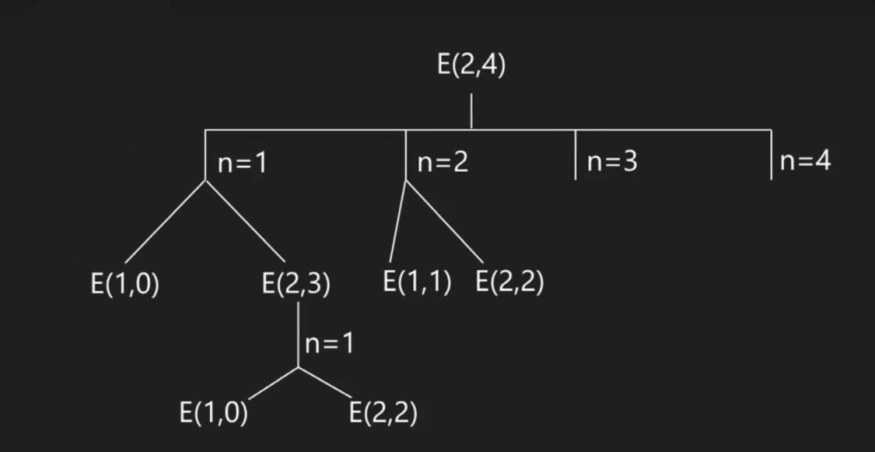


Fig. 4: Partial recursion tree for recursive method

Like shown in Fig.4, E as eggDrop, E(1,0) and E(2,2) are processed two times, this can be solved using dynamic programming, therefore all the results will be stored in an 2-d array of Eggs vs. Floors table. Since same method is called again, this problem has Overlapping Sub problems property. So Egg Dropping Puzzle has both properties of a dynamic programming problem. Like other typical Dynamic Programming(DP) problems, recomputations of same sub problems can be avoided by constructing a temporary array eggFloor[][] in bottom up manner.

**Complexity Analysis:**

Time Complexity: As there is a case of overlapping sub-problems the time complexity is O(nk).

Auxiliary Space: The space complexity is O(1) as there was no use of any data structure for storing values.

**2. DYNAMIC PROGRAMMING:**

The previous recursive solution was very slow, and the same function is called more than once, which is not necessary. We can avoid recalculation of the same sub problems by memoizing the function with a two-dimensional array eggFloor[][]. Then, we just have to fill it up. We will do it by dynamic programming .

*So let’s see first what’s Dynamic Programming?*

Dynamic Programming (DP) is an algorithmic technique for solving an optimization problem by breaking it down into simpler sub problems and utilizing the fact that the optimal solution to the overall problem depends upon the optimal solution to its sub problems. Therefore this approach is really useful in this egg dropping puzzle.

In this approach, we will neglect the cases of calculating the sub problems again. The approach used is to make a table which will store the results of sub-problems so that to solve a sub-problem, it would only require a look-up from the table which will take constant time, which earlier took exponential time. Therefore, it will hugely affect the time taken to solve the problem. Now for filling the array[i][j] state where

i=number of eggs and j = number of floors:

We will have to traverse each floor ‘x’ from ‘1’ to ‘j’ and find minimum of:

**(1 + max( array[i-1][j-1], array[i][j-x] ))**

This will be the basis of solving the problem. Let’s take an example to make things a bit clear:

EXAMPLE:

Let i=number of eggs

j=number of floors

For example:

i=2 eggs

j=4 floors

We firstly put the boundary conditions:

If there is 1 egg and K floors ->Min attempt will be K

If there are n eggs and 1 floor -> Min attempt will be 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eggs /Floors | 1 | 2 | 3 | 4 |
| 1 | (1,1)=1 | (1,2)=2 | (1,3)=3 | (1,4)=4 |
| 2 | (2,1)=1 |  |  |  |

For (2,2):

If egg is dropped from 1st floor ->1+max(0,1)=2

If egg is dropped from 2nd floor ->1+max(1,0)=2

Min(2,2)=2

For(2,3):

If egg is dropped from 1st floor ->1+max(0,2)=3

If egg is dropped from 2nd floor ->1+max(1,1)=2

If egg is dropped from 3rd floor ->1+max(2,0)=3

Min(3,2,3)=2

For(2,4):

If egg is dropped from 1st floor ->1+max(0,2)=3

If egg is dropped from 2nd floor ->1+max(1,2)=3

If egg is dropped from 3rd floor ->1+max(2,1)=3

If egg is dropped from 4th floor ->1+max(3,0)=4

Min(3,3,3,4)=3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eggs /Floors | 1 | 2 | 3 | 4 |
| 1 | (1,1)=1 | (1,2)=2 | (1,3)=3 | (1,4)=4 |
| 2 | (2,1)=1 | (2,2)=2 | (2,3)=2 | (2,4)=3 |

Table 2:Filling up the egg floor table in dynamic programming

Table 3: Completed table of egg floor in dynamic programming

ALGORITHM:

1.BEGIN

 2.Construct a utility function to get maximum of two integers

    static int max(int a, int b)

    {

        return (a > b) ? a : b;

    }

3.Add a Function eggDrop(int n, int k) to get minimum number  of trials needed in worst case with n eggs and k floors .Add 2D table where entry eggFloor[i][j] will represent minimum number of trials  needed for i eggs and j floors.

        int eggFloor[][] = new int[n + 1][k + 1];

        int res;

        int i, j, x;

        // We need one trial for one floor and

        // 0 trials for 0 floors

        for (i = 1; i <= n; i++) {

            eggFloor[i][1] = 1;

            eggFloor[i][0] = 0;

        }

        // We always need j trials for one egg

        // and j floors.

        for (j = 1; j <= k; j++)

            eggFloor[1][j] = j;

        // Fill rest of the entries in table using

        // optimal substructure property

        for (i = 2; i <= n; i++) {

            for (j = 2; j <= k; j++) {

                eggFloor[i][j] = Integer.MAX\_VALUE;

                for (x = 1; x <= j; x++) {

                    res = 1 + max(

                                  eggFloor[i - 1][x - 1],

                                  eggFloor[i][j - x]);

                    if (res < eggFloor[i][j])

                        eggFloor[i][j] = res;

                }

            }

        }

        return eggFloor[n][k];

    }

4.End

Complexity Analysis:

**Time Complexity:** The time complexity of this approach will be based on the two for loops used in this solution. There are two nested for loops ‘K2’ times for each egg.The complexity will be O(n\* K2) where n is the number of eggs and K is the number of floors in the building.

**Space Complexity:** There is one 2-D array of size ‘n\*K’ to store the elements therefore the space complexity will be O(n\*K) where n is the number of eggs and K is the number of floors in the building.

3. Binary Search and Binomial Coefficient Method:

This is another method for n eggs and K floors. It uses binomial coefficient and binary search to solve the egg dropping problem. When we drop an egg, two cases arise.

a. If egg breaks, then we are left with x-1 trials and n-1 eggs.

b. If egg does not break, then we are left with x-1 trials and n eggs

Let maxFloors(x, n) be the maximum number of floors that we can cover with x trials and n eggs. From above two cases, we can write.

maxFloors(x, n) = maxFloors(x-1, n-1) + maxFloors(x-1, n) + 1

For all x >= 1 and n >= 1

Base cases :

We can't cover any floor with 0 trials or 0 eggs

maxFloors(0, n) = 0

maxFloors(x, 0) = 0

Since we need to cover k floors,

maxFloors(x, n) >= k ----------(1)

The above recurrence simplifies to following,

maxFloors(x, n) =∑(xCi)

1 < i < n ----------(2)

Here C represents Binomial Coefficient.From above two equations, we can say.

∑(xCi)>= k

1 < i < n

Basically we need to find minimum value of x that satisfies above inequality. We can find

such x using Binary Search.

Complexity Analysis:

**Time Complexity:** The time complexity of this approach is O(n log K) where n is the number of eggs and K is the number of floors in the building.

1. **COMPARSION OF ALL THE ALGORITHMS:**

Following are the approaches and their respective time complexity to solve this problem:

a. Recursive solution-O(nk)

b. Dynamic Programming- O(n\* K2)

c. Binary Search Method- O(n log K)

Let’s see the values of the time complexities for different amount of eggs and floors:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Values  For n | Values for k | Recursive Solution | Dynamic Programming | Binary Search Method |
| 1 | 5 | 1 | 25 | 0.69897 |
| 2 | 5 | 32 | 50 | 1.39794 |
| 3 | 5 | 243 | 75 | 2.09691 |
| 4 | 5 | 1024 | 100 | 2.79588 |
| 1 | 10 | 1 | 100 | 1 |
| 2 | 10 | 1024 | 200 | 2 |
| 3 | 10 | 59049 | 300 | 3 |
| 4 | 10 | 1048576 | 400 | 4 |
| 2 | 100 | 1.2676506e+30 | 20000 | 4 |

Table. 4 :Comparing Time Complexities of different methods

By looking at the table above we can clearly see that the binary search method is the most efficient in all the cases and it is a lot faster in cases with large values of k. Therefore, the time complexities can be compared like:

**Recursive solution > Dynamic Programming > Binary Search Method**

The best time complexity is of the binary search method.

According to their space complexity the best method would be recursive as its space complexity is O(1) but as it takes time exponentially it makes it worse therefore the binary search method works the best in this case.

1. APPLICATIONS

This problem has many applications in the real world such as:

* It is largely used by car manufacturers in placing airbags in vehicles. They place them in cars to prevent or limit the serious damage that can occur to a person. With the massive force, momentum and pressure a vehicle which has been in massive motion with a spontaneous stop the force increases. Without airbags, people in accidents would hit dashboards, windows, etc. which might be very fatal.
* Solve for the average force an object will experience in different collisions
* Avoiding a call out to the slow HDD.
* Attempting to minimize cache misses.
* Running a large number of expensive queries on a database.

1. CONCLUSION

This article focused on solving all the different cases of the Egg Dropping Puzzle .There are many ways to solve this problem but now you are well equipped with all the necessary information on this topic. We saw recursive method, binary search method and dynamic programming methods. We compared and analyzed their respective space and time complexities and found out that Binary Search method was optimal for n eggs and k floors. We also saw the real world applications of this egg dropping puzzle. So which method of solving intrigued you the most?

References

1. https://en.wikipedia.org/wiki/Dynamic\_programming#Egg\_dropping\_puzzle
2. Moshe Sniedovich, “OR/MS Games: 4. The Joy of Egg-Dropping in Braunschweig and Hong Kong”,Vol 4 No. 1.
3. James Stanton, “Thinking Mathematics”,http://www.jamestanton.com/wp-content/uploads/2019/04/Cool-Math-Essay\_April-2019\_Egg-Dropping.pdf
4. https://sites.google.com/site/eggdropexperiment101/home/egg-case/did-it-break/application-to-the-real-world
5. https://www.geeksforgeeks.org/egg-dropping-puzzle-dp-11/
6. https://brilliant.org/wiki/egg-dropping/#complexity
7. https://www.geeksforgeeks.org/eggs-dropping-puzzle-binomial-coefficient-and-binary-search-solution/?ref=lbp
8. https://www.geeksforgeeks.org/eggs-dropping-puzzle-set-2/?ref=rp
9. https://www.geeksforgeeks.org/puzzle-set-35-2-eggs-and-100-floors/?ref=rp
10. https://hackernoon.com/dissecting-the-famous-egg-dropping-puzzle-eec5064a341b
11. https://blog.hubspot.com/marketing/writing-powerful-conclusions
12. https://www.shoutmeloud.com/improve-article-writing-skills.html