Egg Dropping Puzzle | DP-11

The following is a description of the instance of this famous puzzle involving n=2 eggs and a building with k=36 floors.

Suppose that we wish to know which stories in a 36-story building are safe to drop eggs from, and which will cause the eggs to break on landing. We make a few assumptions:

…..An egg that survives a fall can be used again.
…..A broken egg must be discarded.
…..The effect of a fall is the same for all eggs.
…..If an egg breaks when dropped, then it would break if dropped from a higher floor.
…..If an egg survives a fall then it would survive a shorter fall.
…..It is not ruled out that the first-floor windows break eggs, nor is it ruled out that the 36th-floor do not cause an egg to break.

If only one egg is available and we wish to be sure of obtaining the right result, the experiment can be carried out in only one way. Drop the egg from the first-floor window; if it survives, drop it from the second floor window. Continue upward until it breaks. In the worst case, this method may require 36 droppings. Suppose 2 eggs are available. What is the least number of egg-droppings that is guaranteed to work in all cases?
The problem is not actually to find the critical floor, but merely to decide floors from which eggs should be dropped so that total number of trials are minimized.

Source: Wiki for Dynamic Programming

Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

In this post, we will discuss solution to a general problem with n eggs and k floors. The solution is to try dropping an egg from every floor (from 1 to k) and recursively calculate the minimum number of droppings needed in worst case. The floor which gives the minimum value in worst case is going

can be easily modified to print floor numbers of every trials also.

1) Optimal Substructure:
When we drop an egg from a floor x, there can be two cases (1) The egg breaks (2) The egg doesn't break.

1) If the egg breaks after dropping from xth floor, then we only need to check for floors lower than x with remaining eggs; so the problem reduces to x-1 floors and n-1 eggs
2) If the egg doesn't break after dropping from the xth floor, then we only need to check for floors higher than x; so the problem reduces to k-x floors and n eggs.

Since we need to minimize the number of trials in *worst* case, we take the maximum of two cases. We consider the max of above two cases for every floor and choose the floor which yields minimum number of trials.

k ==> Number of floors
n ==> Number of Eggs
eggDrop(n, k) ==> Minimum number of trials needed to find the critical
floor in worst case.
eggDrop(n, k) = 1 + min{max(eggDrop(n - 1, x - 1), eggDrop(n, k - x)):
x in {1, 2, ..., k}}

2) Overlapping Subproblems
Following is recursive implementation that simply follows the recursive structure mentioned above.

C

```
# include <stdio.h>
# include <limits.h>

// A utility function to get maximum of two integers
int max(int a, int b) { return (a > b)? a: b; }

/* Function to get minimum number of trials needed in worst
  case with n eggs and k floors */
int eggDrop(int n, int k)
{
    // If there are no floors, then no trials needed. OR if there is
    // one floor, one trial needed.
    if (k == 1 || k == 0)
        return k;

    // We need k trials for one egg and k floors
    if (n == 1)
```

```
                _

    // Consider all droppings from 1st floor to kth floor and
    // return the minimum of these values plus 1.
    for (x = 1; x <= k; x++)
    {
        res = max(eggDrop(n-1, x-1), eggDrop(n, k-x));
        if (res < min)
            min = res;
    }

    return min + 1;
}

/* Driver program to test to pront printDups*/
int main()
{
    int n = 2, k = 10;
    printf ("nMinimum number of trials in worst case with %d eggs and "
             "%d floors is %d \n", n, k, eggDrop(n, k));
    return 0;
}
```

C#

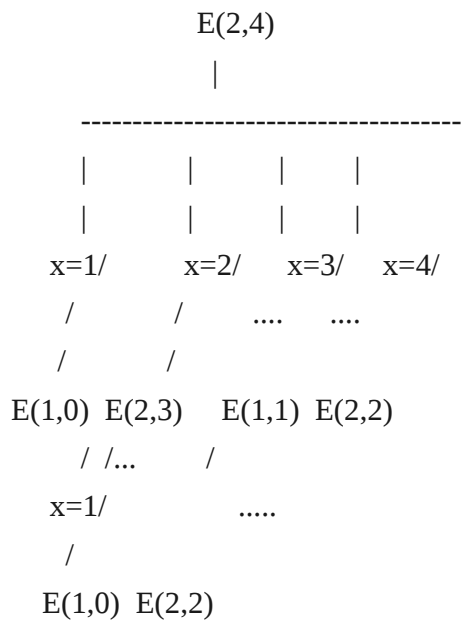
```
using System;

class GFG {

    /* Function to get minimum number of
    trials needed in worst case with n
    eggs and k floors */
    static int eggDrop(int n, int k)
    {
        // If there are no floors, then
        // no trials needed. OR if there
        // is one floor, one trial needed.
        if (k == 1 || k == 0)
            return k;

        // We need k trials for one egg
        // and k floors
        if (n == 1)
            return k;

        int min = int.MaxValue;
        int x, res;

        // Consider all droppings from
        //1st floor to kth floor and
        // return the minimum of these
```

```
            res = Math.Max(eggDrop(n-1, x-1),
                           eggDrop(n, k-x));
            if (res < min)
                min = res;
        }

        return min + 1;
    }

    // Driver code
    static void Main()
    {
        int n = 2, k = 10;
        Console.Write("Minimum number of "
            + "trials in worst case with "
                   + n + " eggs and " + k
          + " floors is " + eggDrop(n, k));
    }
}

// This code is contributed by Sam007.
```

Output:

Minimum number of trials in worst case with 2 eggs and 10 floors is 4

It should be noted that the above function computes the same subproblems again and again. See the following partial recursion tree, E(2, 2) is being evaluated twice. There will many repeated subproblems when you draw the complete recursion tree even for small values of n and k.



Partial recursion tree for 2 eggs and 4 floors.

Since same suproblems are called again, this problem has Overlapping Subprolems property. So Egg Dropping Puzzle has both properties (see this and this) of a dynamic programming problem. Like other typical Dynamic Programming(DP) problems, recomputations of same subproblems can be avoided by constructing a temporary array eggFloor[][] in bottom up manner.

Dynamic Programming Solution

Following are the implementations for Egg Dropping problem using Dynamic Programming.

C++

```
// A Dynamic Programming based C++ Program for the Egg Dropping Puzzle
# include <stdio.h>
# include <limits.h>

// A utility function to get maximum of two integers
int max(int a, int b) { return (a > b)? a: b; }

/* Function to get minimum number of trials needed in worst
  case with n eggs and k floors */
int eggDrop(int n, int k)
{
    /* A 2D table where entery eggFloor[i][j] will represent minimum
       number of trials needed for i eggs and j floors. */
    int eggFloor[n+1][k+1];
    int res;
    int i, j, x;

    // We need one trial for one floor and0 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] = INT_MAX;
```

```
                if (res < eggFloor[i][j])
                    eggFloor[i][j] = res;
            }
        }
    }

    // eggFloor[n][k] holds the result
    return eggFloor[n][k];
}

/* Driver program to test to pront printDups*/
int main()
{
    int n = 2, k = 36;
    printf ("nMinimum number of trials in worst case with %d eggs and "
             "%d floors is %d \n", n, k, eggDrop(n, k));
    return 0;
}
```

Java

```
//A Dynamic Programming based Java Program for the Egg Dropping Puzzle
class EggDrop
{
    // A utility function to get maximum of two integers
    static int max(int a, int b) { return (a > b)? a: b; }

    /* Function to get minimum number of trials needed in worst
    case with n eggs and k floors */
    static int eggDrop(int n, int k)
    {
       /* A 2D table where entery 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 and0 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
```

```
            {
                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;
                }
            }
        }

        // eggFloor[n][k] holds the result
        return eggFloor[n][k];

    }

    /* Driver program to test to pront printDups*/
    public static void  main(String args[] )
    {
        int n = 2, k = 10;
        System.out.println("Minimum number of trials in worst case with "+n+"
                 " floors is "+eggDrop(n, k));
    }
}
/*This code is contributed by Rajat Mishra*/
```

Python

```
# A Dynamic Programming based Python Program for the Egg Dropping Puzzle
INT_MAX = 32767

# Function to get minimum number of trials needed in worst
# case with n eggs and k floors
def eggDrop(n, k):
    # A 2D table where entery eggFloor[i][j] will represent minimum
    # number of trials needed for i eggs and j floors.
    eggFloor = [[0 for x in range(k+1)] for x in range(n+1)]

    # We need one trial for one floor and0 trials for 0 floors
    for i in range(1, n+1):
        eggFloor[i][1] = 1
        eggFloor[i][0] = 0

    # We always need j trials for one egg and j floors.
    for j in range(1, k+1):
        eggFloor[1][j] = j

    # Fill rest of the entries in table using optimal substructure
    # property
    for i in range(2, n+1):
```

```
                res = 1 + max(eggFloor[i-1][x-1], eggFloor[i][j-x])
                if res < eggFloor[i][j]:
                    eggFloor[i][j] = res

    # eggFloor[n][k] holds the result
    return eggFloor[n][k]

# Driver program to test to pront printDups
n = 2
k = 36
print("Minimum number of trials in worst case with" + str(n) + "eggs and "
      + str(k) + " floors is " + str(eggDrop(n, k)))

# This code is contributed by Bhavya Jain
```

C#

```
// A Dynamic Programming based C# Program
// for the Egg Dropping Puzzle
using System;

class GFG {

    // A utility function to get maximum of
    // two integers
    static int max(int a, int b)
    {
        return (a > b) ? a: b;
    }

    /* Function to get minimum number of
    trials needed in worst case with n
    eggs and k floors */
    static int eggDrop(int n, int k)
    {

        /* A 2D table where entery 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 and0
        // trials for 0 floors
        for (i = 1; i <= n; i++)
        {
            eggFloor[i,1] = 1;
            eggFloor[i,0] = 0;
        }
```

```
        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] = int.MaxValue;
                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;
                }
            }
        }

        // eggFloor[n][k] holds the result
        return eggFloor[n,k];

    }

    // Driver function
    public static void Main()
    {
        int n = 2, k = 36;
        Console.WriteLine("Minimum number of trials "
           + "in worst case with " + n + " eggs and "
                 + k + "floors is " + eggDrop(n, k));
    }
}

// This code is contributed by Sam007.
```

PHP

```
<?php
// A Dynamic Programming based PHP
// Program for the Egg Dropping Puzzle

/* Function to get minimum number
   of trials needed in worst
   case with n eggs and k floors */
function eggDrop($n, $k)
{

    /* A 2D table where entery eggFloor[i][j]
```

```
    // We need one trial for one
    // floor and0 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][$j] = 999999;
            for ($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;
            }
        }
    }

    // eggFloor[n][k] holds the result
    return $eggFloor[$n][$k];
}

    // Driver Code
    $n = 2;
    $k = 36;
    echo "Minimum number of trials in worst case with " .$n. " eggs and "
                                    .$k. " floors is " .eggDrop($n, $k) ;

// This code is contributed by Sam007
?>
```

Output :

Minimum number of trials in worst case with 2 eggs and 36 floors is 8

As an exercise, you may try modifying the above DP solution to print all intermediate floors (The floors used for minimum trial solution).

More Efficient Solution : Eggs dropping puzzle (Binomial Coefficient and Binary Search Solution)

Egg Dropping Puzzle with 2 Eggs and K Floors
2 Eggs and 100 Floor Puzzle

Egg Dropping Problem - Approach to write the code (Dynamic Program...

References:
http://archive.ite.journal.informs.org/Vol4No1/Sniedovich/index.php

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

Egg Dropping Puzzle with 2 Eggs and K Floors
Eggs dropping puzzle (Binomial Coefficient and Binary Search Solution)
The Two Water Jug Puzzle
Puzzle | Set 35 (2 Eggs and 100 Floors)
Paths with maximum number of 'a' from (1, 1) to (X, Y) vertically or horizontally
Ways to place K bishops on an N×N chessboard so that no two attack

Minimum cost to reach a point N from 0 with two different operations allowed

Treasure and Cities

Memoization using decorators in Python

Count ways to reach a score using 1 and 2 with no consecutive 2s

Sum of Fibonacci numbers at even indexes upto N terms

K- Fibonacci series

Improved By : Sam007, sachinhaldavanekar

Article Tags : Dynamic Programming Egg-Dropping

Practice Tags : Dynamic Programming

4

4.2

To-do Done

Based on **343** vote(s)

Feedback Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org