Dynamic Programming | Set 11 (Egg Dropping Puzzle)

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

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

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

case is going to be part of the solution.
In the following solutions, we return the minimum number of trails in worst case; these solutions 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 trails 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.

```
# 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 trails 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.
```

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding "extern" keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
    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_MAX, x, res;

    // 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;
}
```

```
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.

```
                      E(2,4)
                        |
         --------------------------------
         |           |          |        |
         |           |          |        |
```

```
      x=1/\          x=2/\      x=3/ \    x=4/ \
        /  \           /  \       ....      ....
       /    \         /    \
 E(1,0)  E(2,3)     E(1,1)  E(2,2)
          /\  /\...         /  \
      x=1/  \               .....
        /    \
     E(1,0)  E(2,2)
            /   \
            ......

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 is C/C++ implementation for Egg Dropping problem using Dynamic Programming.

```
# 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 trails 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;
```

705 Subscribe

Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 38 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 42 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...
Find subarray with given sum · 1 hour ago

AdChoices

► Math Puzzles
► Egg Container
► Break the Egg

```
        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;
            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*/
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;
}
```

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

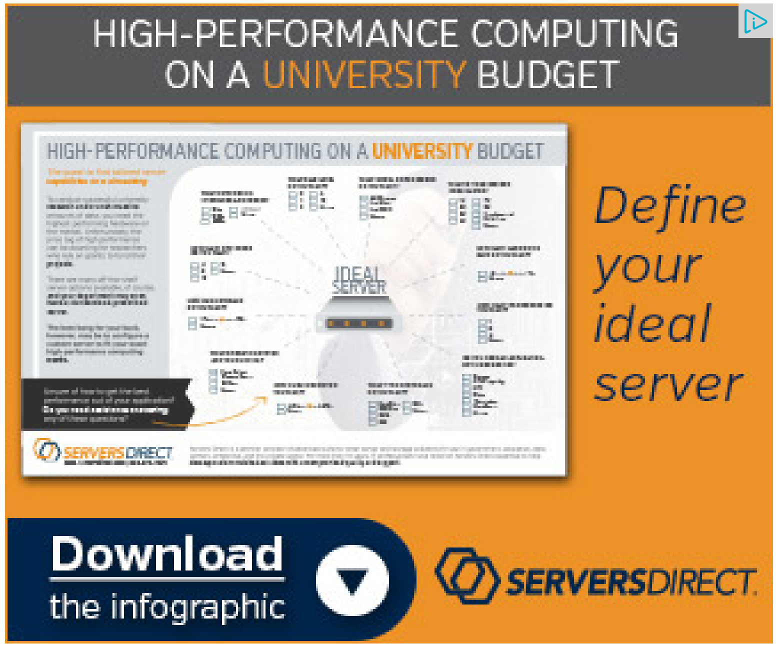
Time Complexity: O(nk^2)
Auxiliary Space: O(nk)

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

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.



Related Tpoics:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation

13 Tweet 1 1

Writing code in comment? Please use **ideone.com** and share the link here.

32 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Akash Agrawal · 19 days ago
Here is a generalized solution for any number of eggs:

http://tech-queries.blogspot.i...

Venu Gopal Akash Agrawal · 19 days ago
TCS ke bahut maje le rahe ho :P

Akash Agrawal Venu Gopal · 19 days ago
???

Venu Gopal Akash Agrawal · 18 days ago
Test ka result aaya ki nahi abhi tak

ameya · 8 months ago
This problem can be solved in O(1) using just a simple formula:
seal(sqrt(2*n)-0.5) // n = number of floors in building

```
#include "stdio.h"
#include "math.h"
```

```
{
int n,ans;
float x;

Printf("Enter Number of floors in building: ");
scanf("%d",&n);
x=sqrt(2*n)-0.5;
ans=x;
if(x>ans)
++ans;
printf("%d\n",ans);

return 0;
}
```

1

Code_Addict ameya • 5 months ago

Thanks for sharing above formula for cross-checking the answer, but c
that minimum number of trials in worst case is independent of number
of eggs is 1).

ameya Code_Addict • 5 months ago

http://www.datagenetics.com/bl...

Code_Addict ameya • 4 months ago

Thanks for link!

Code_Addict Code_Addict • 5 months ago

Formula above gives wrong result for case:

no. of eggs=3 and no. of floors = 92.
By using DP: answer is 8
By using formula: answer is 14

Mihir ameya • 6 months ago
What you call a 'Simple Formula' has a derivation behind it. If this is an
have to mathematically prove this formula first or do the dynamic progr

anju Mihir • 5 months ago
r u trying to say something similar to this post::

http://bit.ly/1biTdXo

prakash • 10 months ago
plz clarify my doubt DP version of this problem solution,
why we need 3 for loops?(.ie,x=1;x<=j;x++)

since this is overlapping sub problem,
res = 1 + max(eggFloor[i-1][x-1], eggFloor[i][j-x]);
itself will give min &#039res&#039 for eggfloor[i][j]
why are we doing it in for loop?

AMIT • 11 months ago
also wikipedia gives various other methods to modify time complexity upto nlo

http://en.wikipedia.org/wiki/D...

rajx • 11 months ago

```
int rj_eggDrop(int eggs, int floors)
{
if(eggs==1)return floors;
int i, step=2;
int arr1[eggs], arr2[eggs];
for(i=0;i<eggs;i++)arr1[i]=1;
int *v1=arr1, *v2=arr2, *tmp;
while(1)
{
v2[0] = step;
for(i=1;i= floors)
return step;
}
step++;
tmp = v1;v1=v2;v2=tmp;
}
}
```

Time Complexity: O(floors)
Auxiliary Space: O(eggs)

1

kavita • 11 months ago

@venki how u arrived at this solution:General solution - n * k^(1\n), where n is

I am not getting results. please explain.

ronny • 11 months ago

Can anyone explain this problem for k eggs.

Niks · a year ago

What is the time complexity of recursive solution??

```
/* Paste your code here (You may delete these lines if not writing c
```

AG · a year ago

x is out of scope in the line :
int min = INT_MAX, x, res;
of the recursive solution.

leet · 2 years ago

How to do the exercise that is to print the floors? Please give some guidance.

```
/* Paste your code here (You may delete these lines if not writing c
```

parashu · 2 years ago

plz,anybody can paste the code using greedy method.

Cameron91 ➦ parashu · 2 years ago

@Parashu,

The first approach is:
The Egg-Drop Numbers
Author(s): Michael BoardmanSource: Mathematics Magazine, Vol. 77,
372Published by: Mathematical Association of AmericaStable URL: htt

The second approach is:

Suppose we have k = 3 eggs.
We drop the first egg at Floor FL/2 and it breaks. Then we have 2 eggs
at which an egg will break is floor FL/2

Now to solve the remaining 2 egg drop problem . we solve the equatior
sqrt(FL) - sqrt(sqrt(FL)).

So the total eggs dropped is i + 1.

e. g. k = 3 eggs FL = 92 Floor

see more

parashu · 2 years ago
nice solution

GeeksforGeeks · 2 years ago
@Hari: Thanks for pointing out the typo. We have corrected it. Keep it up!

Hari · 2 years ago
At the start of dynamic programming solution there is a comment saying "func
anagram of each other". It needs to be deleted.

Venki · 2 years ago
General solution - n * k^(1\n), where n is # eggs, k is # of floors.

AMIT ➦ Venki • 11 months ago

I verified by putting n=2,k=36 in arun's equation..no other source

AMIT ➦ Venki • 11 months ago

it should be (1+n*k)^(1/n),i think...please check

kartik • 2 years ago

@Arun: Thanks for sharing this solution for 2 Egg problem. Can we generalize

Arun • 2 years ago

```
 /*
 #include<iostream>
using namespace std;

int main()
{
cout<<"Enter the No of floors: ";
int fl;
cin>>fl;

if(fl < 0)
{
        cout<<"Not a valid no\n";
        return 0;
}

int ans;
ans=(int) ((-1+sqrt(1+8*fl))/2 + 0.5);
return 0;
}
 */
```

Arun • 2 years ago

[sourcecode language="C++"]
/* #include<iostream>
using namespace std;

int main()
{
cout<<"Enter the No of floors: ";
int fl;
cin>>fl;

if(fl < 0)
{
cout<<"Not a valid no\n";
return 0;
}

int ans;
ans=(int) ((-1+sqrt(1+8*fl))/2 + 0.5);
cout<<"Minimum number of trials in worst case : "<<ans<<endl;

see more

cp • 2 years ago

good to see this problem coded nicely.

the 2 egg problem can be solved using greedy.
http://classic-puzzles.blogspot.in/2006/12/google-interview-puzzle-2-egg-prob

Guest ➜ cp • 8 months ago

This problem can be solved in O(1) -constant time using just a simple seal(sqrt(2*n)-0.5); // n = #of floors in building

// program is as follows

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

int main()
{
int n,ans;
float x;

Printf("Enter Number of floors in building: ");
scanf("%d",&n);
x=sqrt(2*n)-0.5;
ans=x;
if(x>ans)
++ans;
printf("%d\n",ans);
return 0;
}
```