Question: https://leetcode.com/problems/super-egg-drop/

It’s similar to Binary Search.

You drop an egg from n/2th floor and if it breaks you check the floors below n/2th floor and if doesn’t break you check the floor above n/2th floor.

For int left = helper(K - 1, i - 1, memo); int right = helper(K, N - i, memo); when i goes up, left goes up and right goes down.  
We can use Binary Search here to get the minimum Math.max(left, right) + 1, when left and right are as close as possible.

Code:  
class Solution {

public int superEggDrop(int K, int N) {

int[][] memo = new int[K + 1][N + 1];

return helper(K, N, memo);

}

private int helper(int K, int N, int[][] memo) {

if (N <= 1) {

return N;

}

if (K == 1) {

return N;

}

if (memo[K][N] > 0) {

return memo[K][N];

}

int low = 1, high = N, result = N;

while (low < high) {

int mid = low + (high - low) / 2;

int left = helper(K - 1, mid - 1, memo);

int right = helper(K, N - mid, memo);

result = Math.min(result, Math.max(left, right) + 1);

if (left == right) {

break;

} else if (left < right) {

low = mid + 1;

} else {

high = mid;

}

}

memo[K][N] = result;

return result;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>