THE EGG GAME

STUART FLETCHER

ABSTRACT. We present a game and proofs for an optimal solution.

1. The Game

You are presented with a multistory building and some number of SUPERSTRONG eggs. These eggs can withstand drops from some height; you must figure out just what this height is. You are told that there is some floor, f_0, such that the eggs can survive the drop from f_0 but will break when dropped from floor $f_0 + 1$. f_0 may be the top of the building—that is, the eggs may not break at all. (Note that in each game all the eggs you are given have the same SUPERSTRENGTH, i.e., if one breaks when dropped from a certain height, they all will.)

You want to determine the value of f_0 as efficiently as possible, that is, by making the fewest number of droppings. Of course, you *must* guarantee that you find f_0 before you run out of eggs, so any strategy must take this into account.

As an example, take a 1 egg, 100 story game, which we call a (1,100) game. To guarantee finding f_0 you have no choice but to start by dropping the egg from the first floor and dropping from each successive floor in sequence until the egg breaks. This can take as many as 100 drops.

A more interesting example, is the (2,100) game—that is, a 2 egg, 100 story game. Before reading on, try to think of a good strategy to find the break floor. How well can you do?

Many people suggest a strategy seemingly inspired by binary search: do the first drop from the middle floor and cut your search space in half. We are always interested in the worst case performance of a strategy, and in this case, with only two eggs, it is easy to see that at worst the strategy just described will at worst solve the (2,100) game in 50 drops. That is, the first egg breaks when dropped from floor 50, and you make 49 more drops (from floor 1, 2, ..., 49), eventually finding that floor 49 (or confirming that floor 50) is the break floor.

Two eggs is not enough to really execute a binary search. If you have 6 eggs you can do binary search and find the floor in 6 drops. An improvement on this is to start by dropping off of every tenth floor. When the egg breaks you only have nine more drops to make. With this strategy it will take no more than 19 drops. This is an improvement over 50, but it turns out to be suboptimal. The method of dividing the number of stories up into evenly spaced intervals also turns out to be the wrong way to try to think of an optimal solution. We will see later that you can solve the (2,100) game in 14 drops.

2. Preliminaries

We will let $D(e, s)$ be the least number of drops it takes to guarantee finding the floor at which an egg breaks, given e eggs and an s-story building.

Let $FM(e, s)$ be a First Move in an optimal strategy for the $D(e, s)$ game. In general there is more than one good first move for an optimal strategy, so we define $FM(e, s)$ to be the maximum of the set of good first moves.

Let t_i denote the ith triangle number ($t_i = \sum_{k=1}^{i} k$).

Date: May 11, 2004.

We use $\lg s$ to denote the logarithm base 2 of s.

Note that, in general, when $e \geq \lceil \lg s \rceil$ it is straightforward to find $D(e,s)$ in $\lfloor \lg s \rfloor + 1$ steps by (essentially) executing binary search. The number of eggs is not a constraint in this case. So the interesting cases are when $e < \lceil \lg s \rceil$.

Something else to keep in mind is that a $D(e,s)$ game is reduced one of two ways on a drop: the egg either breaks or it does not break. Suppose the move is to drop from floor j. If the egg breaks you are reduced to $D(e-1, j-1)$ since you are left with one less egg (for $e-1$ total eggs) and you only have to check the $j-1$ floors that are lower than j. That is, since the egg broke when dropped from the jth floor, you know it will break for all i, $j < i \leq s$; you no longer care about these floors. If the egg does not break you are reduced to $D(e, s-j)$ since you have lost no eggs and you are now only concerned with floors greater than j, of which there are $s-j$. Note that if the egg breaks (the first case discussed above), the game you are reduced to is independent of s, the number of stories in the original game.

3. The one egg game

Claim 1. *For the one egg game, $D(1,s) = s$, with $FM(1,s) = 1$.*

This is very simple but must be pointed out. If you have only one egg and s stories, you can execute linear search, dropping from floor $1, 2, 3, \ldots$ until the egg breaks or until you reach the sth floor. So s drops are sufficient to guarantee finding the break floor.

To show that s is also a lower bound on the number of drops needed to find the break floor, suppose you make fewer than s drops. Your first move can either be 1 or > 1. If the first move is > 1 it is clear that an adversary could always force a situation where the break floor cannot be determined at all: if the egg breaks on the first drop you cannot say whether the break floor is the floor you dropped from or a lower floor. So $FM(1,s) = 1$ is the more interesting case. In this case, since only $s-1$ drops are made, an adversary can also force a situation where the break floor cannot be determined. As soon as a floor is skipped the adversary forces a break on the next drop. You do not know whether *that* is the break floor or the floor just skipped is the break floor. The situation where no floors are skipped, that is, the $s-1$ drops are made from $1, 2, 3, \ldots, s-1$ is also clear: you do not know whether the egg breaks from the sth floor or if it does not break at all (which is a possibility in how we have defined the game). So s drops is also a lower bound.

4. A naïve approach

Claim 2. $D(e,s) \leq es^{1/e}$. *In other words, for the (e,s) game there is some way to do it in at most $es^{1/e}$ drops.*

Proof by induction on e, the number of eggs. For base cases consider the 1- and 2-egg games. A $(1,s)$ game can be done in s drops. This is less than or equal to $es^{1/e} = s^{1/1} = s$.

For the $(2,s)$ game let $FM(2,s) = \frac{s}{s^{1/2}} = s^{1/2}$. We consider two alternatives, the egg breaking and the egg not breaking. If the egg breaks, we are in a $(1, s^{1/2})$ game which takes a total of $s^{1/2} - 1 + 1 = s^{1/2} \leq 2s^{1/2}$. If the egg does not break we have a $(2, s - s^{1/2})$ game and things get interesting. Instead of calculating a new first move for this game (as we shall see is the normal approach taken in this paper), we will now simply continue to use multiples of the original first move until an egg breaks. So the next drop is from floor $2s^{1/2}$. If this breaks, we have a one egg game and the total number of drops will be $\leq s^{1/2} - 1 + 1 + 1 \leq 2s^{1/2}$. At worst the egg will continue to not break until it reaches the topmost floor that is really a multiple of $s^{1/2}$ (this will not be floor s unless s is a perfect square). In this worst case, we have a maximum length series of no breaks finally followed by a break. This can add up to at most $s^{1/2}$ drops. When the egg breaks we are again in a $(1, s^{1/2} - 1)$ game, so the total droppings is $s^{1/2} - 1 + s^{1/2} = 2s^{1/2} - 1 \leq 2s^{1/2}$.

Assume for $e \geq 1$ that $D(e,s) \leq es^{1/e}$. We will show that $D(e+1,s) \leq (e+1)s^{1/e+1}$. Divide the floors at multiples of $\frac{s}{s^{1/e}}$. We are interested in the worst that can happen. If the egg breaks on the first or second drop, the total number of drops is well under $(e+1)s^{1/e+1}$. At worst there is a maximum length series of no breaks and then a break at the topmost floor that is a multiple of $\frac{s}{s^{1/e}}$ (recall that this is not necessarily the topmost floor). This series of drops has length $s^{1/e+1}$ and we are in an $(e, s^{e/e+1})$ game. By the inductive hypothesis we have that

$$\begin{aligned} totaldrops &\leq s^{1/e+1} + e(s^{e/e+1} - 1)^{1/e} \\ &\leq s^{1/e+1} + e(s^{e/e+1})^{1/e} \\ &= s^{1/e+1} + es^{1/e+1} \\ &= (e+1)s^{1/e+1} \end{aligned}$$

which is what was to be shown. □

5. Bounds on the two egg game

Claim 3. *For the two egg game with $t_{i-1} < s \leq t_i$, $D(2,s) = i$, with $FM(2,s) = i$.*

Proof of the upper bound. We prove the claim by strong induction on s, the number of stories.

The base case is to consider $t_1 = 1 < s \leq t_2 = 3$ (i.e., $i = 2$).

Let $s = 2$. Determine $D(2,2)$. For first drop from floor 2 there are two cases.

i. Egg breaks, reduced to $D(1,1)$ game. $D(1,1) = 1$, resulting in 2 drops total.
ii. Egg does not break, reduced to $D(2,0)$ game, which is solved in 0 drops for a total of 1 drop.

So 2 drops suffices.

Let $s = 3$. Determine $D(2,3)$. Taking $FM(2,3) = 2$ there are again two cases.

i. Egg breaks, reduced to $D(1,1)$ game. $D(1,1) = 1$, so total drops is 2.
ii. Egg does not break, reduced to $D(2,1)$ game. This takes 1 drop, for a total of 2 drops.

So 2 drops suffices here as well.

Now suppose for $s > 3$ (with $t_{i-1} < s \leq t_i$) that $D(2,s) = i$ with $FM(2,s) = i$. We will show that $i+1$ drops suffices for $t_i < s \leq t_{i+1}$ (with $FM(2,s) = i+1$).

Take s so that $t_i < s \leq t_{i+1}$ and use $FM(2,s) = i+1$ as the strategy. There are two cases to consider.

i. If the egg breaks we are reduced to the $D(1,i)$ game, which can be solved in i drops. The total number of drops is $i+1$.
ii. If the egg does not break we are reduced to the $D(2, s-(i+1))$ game. Let $s' = s - i - 1$ so we are in a $D(2,s')$ game, with this bound on s':

$$\begin{aligned} t_i &< s \leq t_{i+1} \\ t_i - (i+1) &< s - (i+1) \leq t_{i+1} - (i+1) \\ t_{i-1} - 1 &< s' \leq t_i \\ t_{i-1} &\leq s' \leq t_i \qquad \textit{Note the change in inequality} \end{aligned}$$

By the inductive hypothesis, $D(2,s')$ is solvable in i drops (or $i-1$ if $s' = t_{i-1}$). So $i+1$ drops suffices.

In either case $e+1$ drops suffices. □

Proof of the lower bound. We will prove that you can do no better than i drops for $t_{i-1} < s \leq t_i$ by supposing you have an algorithm that solves $D(2,s)$ in $i-1$ drops and showing an adversary could force you into a contradiction no matter what first move you make.

There are three cases to consider: let $FM(2,s)$ be i, $< i$, or $> i$.

i. $[FM(2,s) = i]$ If our first move is i and the egg breaks, we are in a $D(1, i-1)$ game, which requires $i-1$ droppings, hence the whole game takes i droppings. $\Rightarrow\Leftarrow$

ii. $[FM(2,s) < i]$ With first move $1 \le j < i$, if the egg does not break, we need to solve $D(2, s-j)$.

$$s - j \le t_i - j \le t_i - 1$$

and

$$s - j > t_{i-1} - j \ge t_{i-1} - (i-1) = t_{i-2}$$

So,

$$t_{i-2} < s - j \le t_i - 1$$

This will take at most $i-1$ droppings to solve, for a total of i drops. $\Rightarrow\Leftarrow$

iii. $[FM(2,s) > i]$ With first move $i < j \le s$, if the egg breaks we must solve $D(1, j-1)$. But

$$i - 1 < j - 1 \Longrightarrow i \le j - 1$$

so it will take at least i drops to determine the proper floor, for a total of $i+1$ droppings. $\Rightarrow\Leftarrow$

So no matter what strategy is adopted there is a pattern of play that makes it impossible for there to be an algorithm that determines $D(2,s), t_{i-1} < s \le t_i$ in fewer than i moves. □

Corollary 1. $D(2,s) = \sqrt{2s} \pm O(1)$

Proof. We know that $D(2,s) = i$. We want to see how i relates to s. Suppose $s = t_i$. Then we have

$$\begin{aligned} s &= t_i \\ &= \sum_{k=1}^{i} k \\ &= \frac{i(i+1)}{2} + O(1) \quad \text{or} \quad O(i) \\ &\approx \frac{i^2}{2} \end{aligned}$$

From this we see that $i \approx \sqrt{2s}$, so $D(2,s) = \sqrt{2s} + O(1)$. □

6. Bounds on the three egg game

Claim 4. *For the three egg game with* $\sum_{k=1}^{3}\binom{i-1}{k} < s \le \sum_{k=1}^{3}\binom{i}{k}$*,* $D(3,s) = i$*, with first move* $FM(3,s) = \sum_{k=0}^{2}\binom{i-1}{k}$*. (Note that this is only* one *good first move; it happens to be the highest good first move.)*

With a little thought it quickly becomes clear that, given this formulation of the solution the problem is to find the i that satisfies the above inequality when you are given s.

Proof of the upper bound. Again we prove the claim by strong induction on s, the number of stories.

The base case is to consider $i = 3$, so we have $3 < s \le 7$. Note that the first move is the same in each case, it is $\sum_{k=0}^{2}\binom{i-1}{k} = \binom{2}{0} + \binom{2}{1} + \binom{2}{2} = 1 + 2 + 1 = 4$. We have four cases, each with two subcases, break and no break. In each of the four cases, though, the "break" case is identical because when an egg breaks we are reduced to a case that is independent of s. The new configuration only depends on the number of eggs and the move that was just made, which is the same over these four base cases. So in each case here, if the egg breaks we are reduced to $D(2,3) = 2$ for a total of three drops made. We will now show the outcome of the "no break" cases.

i. Let $s = 4$. We have $D(3,4)$. If the egg does not break we have $D(3,0) = 0$. Three drops is necessary, so $D(3,4) = 3$.
ii. Let $s = 5$. If the egg does not break we are in $D(3,1) = 1$. Again, accounting for the "break" case, three drops is necessary, so $D(3,5) = 3$.
iii. Let $s = 6$. If the egg does not break we are in $D(3,2) = 2$. One way to think of this is that $D(2,2) = 2$ and the extra egg does not help you. So $D(3,6) = 3$.
iv. Let $s = 7$. If the egg does not break we are in $D(3,3) = 2$. Again, this is evident either from thinking about $D(2,3)$, or just from working it out with pencil and paper. So $D(3,7) = 3$.

Now suppose for $s > 7$ (with $\sum_{k=1}^{3}\binom{i-1}{k} < s \le \sum_{k=1}^{3}\binom{i}{k}$) that $D(3,s) = i$, with first move $FM(3,s) = \sum_{k=0}^{2}\binom{i-1}{k}$. We will show that $i+1$ drops suffices for $\sum_{k=1}^{3}\binom{i}{k} < s \le \sum_{k=1}^{3}\binom{i+1}{k}$ with $FM(3,s) = \sum_{k=0}^{2}\binom{i}{k}$. There are two cases to consider, breaking and not breaking.

i. If the egg breaks we are reduced to

$$\begin{aligned} D\left(2, \sum_{k=0}^{2}\binom{i}{k} - 1\right) &= D\left(2, \sum_{k=1}^{2}\binom{i}{k}\right) \\ &= D\left(2, t_i\right) \quad \textit{Using triangle number notation from 2-egg proof} \\ &= i \end{aligned}$$

So $i+1$ drops suffices.

ii. If the egg does not break we are reduced to $D\left(3, s - \sum_{k=0}^{2}\binom{i}{k}\right)$. Let $s' = s - \sum_{k=0}^{2}\binom{i}{k}$. We see that

$$\sum_{k=1}^{3}\binom{i}{k} - \sum_{k=0}^{2}\binom{i}{k} < s' \le \sum_{k=1}^{3}\binom{i+1}{k} - \sum_{k=0}^{2}\binom{i}{k}$$
$$\sum_{k=1}^{3}\binom{i-1}{k} - 2 < s' \le \sum_{k=1}^{3}\binom{i}{k}$$

By the inductive hypothesis $D(3,s')$ can be solved in i steps, for a total of $i+1$ drops in this case.

So $D(3,s) = i$ (with the previous bounds on s), which is what was to be shown. □

Proof of the lower bound. We will show that you can do no better than i drops for $D(3,s)$ with $\sum_{k=1}^{3}\binom{i-1}{k} < s \le \sum_{k=1}^{3}\binom{i}{k}$ by supposing you have an algorithm that solves $D(3,s)$ in $i-1$ drops and deriving a contradiction based on an adversary argument.

Let j denote our strategy, $FM(3,s)$. There are three cases to consider: $j = \sum_{k=0}^{2}\binom{i-1}{k}$, $j < \sum_{k=0}^{2}\binom{i-1}{k}$, and $j > \sum_{k=0}^{2}\binom{i-1}{k}$.

i. [$j = \sum_{k=0}^{2}\binom{i-1}{k}$] If the egg breaks, we have

$$\begin{aligned} D\left(2, \sum_{k=0}^{2}\binom{i-1}{k} - 1\right) &= D\left(2, \sum_{k=1}^{2}\binom{i-1}{k}\right) \\ &= D\left(2, t_{i-1}\right) \quad \textit{Using notation for triangle numbers} \\ &= i-1 \end{aligned}$$

for a total of i drops. $\Rightarrow\Leftarrow$

ii. $[1 \le j < \sum_{k=0}^{2} \binom{i-1}{k}]$ Here the adversary will force not breaking, as that is the problem case. If the egg does not break, we have a $D(3, s-j)$ game. We see that

$$s - j \le \sum_{k=1}^{3} \binom{i}{k} - j \le \sum_{k=1}^{3} \binom{i}{k} - 1$$

and

$$\begin{aligned} s - j &> \sum_{k=1}^{3} \binom{i-1}{k} - j \\ &\ge \sum_{k=1}^{3} \binom{i-1}{k} - \sum_{k=0}^{2} \binom{i-1}{k} \\ &= \binom{i-1}{3} - 1 \end{aligned}$$

So,

$$\binom{i-1}{3} - 1 < s - j \le \sum_{k=1}^{3} \binom{i}{k} - 1$$

This can take at most $i-1$ drops (with the "super" algorithm) for a total of i drops. $\Rightarrow\Leftarrow$

iii. $[\sum_{k=0}^{2} \binom{i-1}{k} < j \le s]$ The adversary forces a break here, so we have $D(2, j-1)$. We have

$$\sum_{k=0}^{2} \binom{i-1}{k} - 1 < j - 1$$

$$\sum_{k=1}^{2} \binom{i-1}{k} < j - 1$$

This is a 2-egg game taking $i-1$ moves for a total of i drops. $\Rightarrow\Leftarrow$

So there is no way for there to be an algorithm that solves $D(3, s)$ (with our bounds on s) in $i-1$ drops. Hence i drops is sufficient and necessary for the game. □

Corollary 2. $D(3, s) = \sqrt[3]{6s} + O(1)$

Proof. We know $D(3, s) = i$ where $\sum_{k=1}^{3} \binom{i-1}{k} < s \le \sum_{k=1}^{3} \binom{i}{k}$. Suppose $s = \sum_{k=1}^{3} \binom{i}{k}$. So

$$\begin{aligned} s &= \sum_{k=1}^{3} \binom{i}{k} \\ &= \binom{i}{1} + \binom{i}{2} + \binom{i}{3} \\ &= \frac{i^3 + 5i}{6} \\ &\approx \frac{i^3}{6} \end{aligned}$$

This implies that $i \approx \sqrt[3]{6s}$, so that $D(3, s) = \sqrt[3]{6s} + O(1)$. □

7. Bounds on the e egg game

Claim 5. *For the e-egg game with s bounded by* $\sum_{k=1}^{e}\binom{i-1}{k} < s \leq \sum_{k=1}^{e}\binom{i}{k}$*,* $D(e,s)=i$*, with first move* $FM(e,s)=\sum_{k=0}^{e-1}\binom{i-1}{k}$*.*

Proof of the upper bound. We will prove this by induction on the well-ordered set

$$(1,1) < (1,2) < \cdots < (2,1) < (2,2) < \cdots < (3,1) < (3,2) < \cdots$$

The base case is essentially that $D(1,s)=s$, and we have also proven the results for $D(2,s)$ and $D(3,s)$. So we will show that $D(e,s)=i$ with the above bounds on s. To do this we may assume we have the result for all $e-1$-egg games and all e-egg games with fewer than s stories. There are two cases to consider, breaking and not breaking. The first move is always $\sum_{k=0}^{e-1}\binom{i-1}{k}$.

i. If the egg breaks, we have $D\left(e-1,\sum_{k=0}^{e-1}\binom{i-1}{k}-1\right)$. This is equivalent to $D\left(e-1,\sum_{k=1}^{e-1}\binom{i-1}{k}\right)$, which we can do in $i-1$ drops by our hypothesis, so i drops suffices.

ii. If the egg does not break, we have $D\left(e,s-\sum_{k=0}^{e-1}\binom{i-1}{k}\right)$. Call this number of stories s'. We have that

$$s' \leq \sum_{k=1}^{e}\binom{i}{k} - \sum_{k=0}^{e-1}\binom{i-1}{k}$$

which gives us

$$s' \leq \sum_{k=1}^{e}\binom{i-1}{k}$$

By the inductive hyposthesis, this new game, $D(e,s')$, can be done in $i-1$ drops for a total of i drops.

In either case, i drops suffices. ☐

Proof of the lower bound. As in the 2- and 3-egg games, we will prove you can do no better than i drops for $D(e,s)$ where $\sum_{k=1}^{e}\binom{i-1}{k} < s \leq \sum_{k=1}^{e}\binom{i}{k}$ by supposing you have an algorithm that solves $D(e,s)$ in $i-1$ drops and deriving a contradiction based on an adversary argument.

Again, let j denote the strategy for the first move (i.e., $FM(e,s)$). There are three cases to consider. They are $j=$, $j<$, and $j>\sum_{k=0}^{e-1}\binom{i-1}{k}$.

i. $[j=\sum_{k=0}^{e-1}\binom{i-1}{k}]$ If the egg breaks in this case, we are in $D\left(e-1,\sum_{k=1}^{e-1}\binom{i-1}{k}\right)$. This new configuration itself takes $i-1$ drops, for a total of i drops. $\Rightarrow\Leftarrow$

ii. $[1 \leq j < \sum_{k=0}^{e-1}\binom{i-1}{k}]$ Here the adversary forces not breaking. We then have $D(e,s-j)$.

$$s-j \leq \sum_{k=1}^{e}\binom{i}{k} - j \leq \sum_{k=1}^{e}\binom{i}{k} - 1$$

and

$$
\begin{aligned}
s-j &> \sum_{k=1}^{e}\binom{i-1}{k} - j \\
&\geq \sum_{k=1}^{e}\binom{i-1}{k} - \sum_{k=0}^{e-1}\binom{i-1}{k} \\
&= \binom{i-1}{e} - 1
\end{aligned}
$$

So,

$$\binom{i-1}{3} - 1 < s-j \leq \sum_{k=1}^{e}\binom{i}{k} - 1$$

This $D(e, s-j)$ game can take $i-1$ drops for a total of i drops. $\Rightarrow\Leftarrow$

iii. $[\sum_{k=0}^{e-1}\binom{i-1}{k} < j \leq s]$ Here the adversary forces a break, so we have $D(e-1, j-1)$.

$$
\begin{aligned}
j-1 &> \sum_{k=0}^{e-1}\binom{i-1}{k} - 1 \\
&= \sum_{k=1}^{e-1}\binom{i-1}{k}
\end{aligned}
$$

This gives us an $(e-1)$-egg game that, by our hypothesis, will take $i-1$ drops, for a total of i drops. $\Rightarrow\Leftarrow$

So i drops is a lower bound on the game. $\square$

Corollary 3. *$D(e,s) = \sqrt[e]{e!s}$ + something or $D(e,s) = e\sqrt[e]{s}$ + something or $D(e,s) \leq e\sqrt[e]{s}$ + something.*

We would like to say that the corollaries for the two and three egg cases suggest a pattern that we can apply to the e egg game. At this point it is not clear to me that the same pattern applies, however.

Proof. n.p. $\square$

8. Discussion

These results were obtained with the help of some experimentation, that is, writing programs. All these binomial coefficients didn't just come out of thin air! We started with the recurrence for $D(e,s)$:

$$D(e,s) = \begin{cases} 1 + \min_{1\leq f\leq s}\left(\max\left(D(e-1, f-1), D(e, s-f)\right)\right) & : \quad \text{for } e < \lceil \lg s \rceil \\ \lfloor \lg s \rfloor + 1 & : \quad \text{for } e \geq \lceil \lg s \rceil \end{cases}$$

where $D(1,s) = s$ and $D(e,1) = 1$. We also defined $D(e,0) = 0$ and $D(0,s) = \infty$.

Using the recurrence directly to compute $D(e,s)$ is not very efficient, primarily because finding $D(e,s)$ requires finding $D(e', s')$ for all $1 \leq e' \leq e$ and $1 \leq s' \leq s$. A naïve implementation runs in $O(es^2)$, which looks even worse when we consider that, in general, $s \gg e$.

It is more efficient to find $D(e,s)$ using binomial coefficients. A method that works well for finding $D(e,s)$ for multiple (e,s) pairs is to construct Pascal's Triangle and search for the first row r, such that $\sum_{k=1}^{e} \binom{r}{k} > s$. Then $r = D(e,s)$ is the answer and the first move can be read directly off the $(r-1)$st row of the triangle.

The time for this method is the time it takes to construct the triangle plus the time to find the desired row in the triangle. To construct the triangle requires an estimate on the number of rows needed. This we get from the naïve approach outlined in section 3. For the (e,s) game we use the upper bound $D(e,s) \leq es^{1/e}$. The running time for this algorithm is then $O(e^2 s^{1+1/e})$, a significant improvement over the above bound of $O(es^2)$. Note that this running time is for an algorithm that computes all $D(e',s')$ for $1 \leq e' \leq e$ and $1 \leq s' \leq s$. In other words, it builds up an $e \times s$ table with all the $D(e',s')$ information in it.

If only one or a few values of $D(e,s)$ are needed there is a method that improves on the above in terms of space (and the time of calculating unused rows of the triangle). The idea is to store only two rows of Pascal's Triangle at any time, and to store only the first $e+1$ entries of each of these rows. This reduces the running time to $O(e^2 s^{1/e})$. This timing, however, is for a version that *only* computes a single $D(e,s)$ value, unlike the previous two algorithms. To do the work of the previous two algorithms takes this space-efficient version $O(e^3 s^{1+1/e})$.