Rolling a d_7 in Finite(ish) Time

Kevin Berry

December 1, 2019

Last year, I was hanging out with a couple of friends when they began talking about Dungeons and Dragons. For those unfamiliar with it, Dungeons and Dragons is a tabletop game which requires many different dice for different situations. A typical die set includes a d_4, which is a 4-sided tetrahedral die; a d_6, a standard 6-sided die; a d_8, an octahedrial die; an anomalously non-platonic d_{10}, which is the slightly verbosely named pentagonal trapezohedron; a d_{12}, a dodecahedron; and a d_{20}, which is an icosahedron. Some people also include the degenerate d_2 as part of the die set, even though it is simply a standard 2-sided coin.

Since I actually know very little about the game itself, as my friends were talking I began to wonder if there were any interesting conversations to be had about the dice, with which I am much more familiar. The question that came to mind was essentially this: given only the standard D&D dice, is it possible to "roll" a die with any number of sides? In other words, is it possible to generate a number uniformly at random from 1 to n, using only a finite set of dice, such as a d_2, d_4, d_6, d_8, d_{10}, d_{12}, or d_{20}? I briefly brought up this question, and we discussed it for a moment. Alas, I found it more interesting than they did, so here we are.

The first die missing from a standard set would be the trivial 1-sided die, which always rolls a 1. One can pretend to roll such a die by shouting the number "1", which is satisfactory for our purposes[1].

What about a 3-sided die? This is a much more interesting case to solve. It turns out, there are many solutions to rolling a d_3. One way to do this is to roll a d_6 and divide the rolled value by 2, rounding up. This rule means that a roll of 1 or 2 is counted as a 1, a roll of 3 or 4 is counted as a 2, and a roll of 5 or 6 is counted as a 3. This nicely produces a uniform distribution from 1 to 3, since 1 and 2 are just as common as 3 and 4 or 5 and 6 on a fair d_6. In fact, any function which maps 3 disjoint pairs of faces of a d_6 onto the numbers 1 to 3 will work for simulating a roll of a d_3 from a roll of a d_6.

Similarly, one could roll a d_3 by mapping 3 disjoint sets of 4 faces from a d_{12} onto the numbers 1 to 3. What's more, a d_4 can be rolled by remapping the faces of a d_8, a d_5 can be rolled by remapping the faces of a d_{10}, and a d_{10} could be rolled by remapping the faces of a d_{20}. In general, given a

[1]One could also consider rolling a zero sided die, which one rolls by not considering it.

die d_{f_1} with f_1 faces, it is possibly to roll any other die with a number of faces f_2 such that f_2 evenly divides f_1. One sufficient mapping in this case is $R_{f_1 \to f_2}(r) = \lceil r/(f_1/f_2) \rceil = \lceil (r/f_1) \cdot f_2 \rceil$, which takes a roll r from a group of size f_1 and maps it uniformly to a group of size f_2, producing the desired distribution. For the sake of concreteness, this mapping for rolling a 5 sided die with a 10 sided one is $R_{10 \to 5} = \lceil r/(10/5) \rceil = \lceil r/2 \rceil = \lceil (r/10) \cdot 5 \rceil$, which I've written in 3 ways to hopefully reveal my intuitions for the correctness of the mapping.

While this trick is neat (and will later come in handy), it only allows us to roll two more dice at the moment. This brings us to the next challenge: rolling a die with 7 sides.

I've asked this question to several friends and family members, and the first approach that tends to be given is to do something like rolling 7 six-sided dice and dividing by 6, rounding down when necessary[2]. At first this may seem to work, but fails upon closer inspection. The lowest value it outputs is $\lfloor 7 \cdot 1/6 \rfloor = 1$, and the highest value it outputs is $\lfloor 7 \cdot 6/6 \rfloor = 7$, so the function outputs values in the correct range. However, the distribution of these values is uneven. For instance, there are many ways of rolling a 1, such as by rolling six 1s and one 2 or five 1s and two 2s, but the only way to roll a 7 is by rolling 7 sixes. Clearly, then, this does not generate a random number uniformly from 1 to 7.

What about something like rolling a d_3 (using a d_6, via our earlier method) and a d_4 and adding them together? Well, this nicely has a maximum value of 7, but falls apart when you realize that the minimum value is 2, which is no good. However, rolling these two dice in this way has another, more insightful flaw. When rolling a d_3 and a d_4, there are $3 \cdot 4 = 12$ possible outcomes: (1, 1), (1, 2), (1, 3), (1, 4), (2, 1), (2, 2), (2, 3), (2, 4), (3, 1), (3, 2), (3, 3), (3, 4). However, the distribution we aim to recreate has 7 options which must be equally likely. By the pigeonhole principle, it is clear that an onto mapping from 12 inputs to 7 outputs cannot be done in such a way that all 7 outputs are equally likely; there will be at least two outputs which only have one input mapped to them, and there will be at least one output which has more than one output mapped to it, making that output more likely than at least one other output. This observation is true for any possible way of rolling a d_{f_1} to produce the outputs of a d_{f_2}, where f_2 is not divisible by f_1. Clearly, then, any rolling scheme which we come up with must have a number of outcomes which is divisible by the number of faces on the die that we want to roll, as was the case with our trick for rolling a d_3.

While the d_3, d_4 combination failed to produce a way of rolling a d_7, there is another insight to be gleaned from this attempt. Namely, that we can roll a die whose face count is any product of the face counts of the dice we have. This follows from the fact that we can simply enumerate f outcomes of equal probability in order to roll a die with f faces, and rolling a set of dice yields a number of equally likely outcomes which is the product of their number of faces. With this technique, and the fundamental theorem of algebra, we now

[2]This was also my first approach.

realize that we only need to be concerned with being able to roll dice with a prime number of faces, since we can generate all other numbers from these.

Which brings us back to rolling a 7. Despite the niceness and utility of our previous observations, we will need some new technique to solve this problem. Because of my familiarity with d_6s, I first thought of a solution which used only them. Namely, we can roll two d_6s and treat one as the ones digit of a base 6 number (offset by 1) and the other as the sixes digit of a base six number (also offset by 1), and we can just re-roll if we get a value greater than or equal to 7. This, perhaps, seems a bit obtuse, but is illustrated well by example. Lets say that we rolled a 4 and then a 3. We treat the first number as the number of sixes that we have, plus one, giving us $(4-1)\cdot 6 = 18$ to start. Then, we treat the second number as the number of ones we have, plus one, bringing our total up to $18 + (3-1)\cdot 1 = 20$. If we had rolled two sixes, we would get $(6-1)\cdot 6 + (6-1)\cdot 1 = 35$, and if we had rolled two ones, we would have gotten $(1-1)\cdot 6 + (1-1)\cdot 1 = 0$. Because each roll in this fashion gives us a unique number on this interval, this scheme gives us each number from 0 to 35 exactly once. Thus, each of the numbers from 0 to 6 is equally likely, and we can generate a number from 1 to 7 uniformly at random by remapping the numbers 0 through 6 and ignoring and re-rolling for numbers greater than 6. It may seem like cheating to re-roll, but the reason that it works is that at any time that the dice are rolled, there is always an equal chance of getting the 7 outcomes we care about, and we ignore all other outcomes. It is also easy to see how we can use this technique to roll **any die we want.** For any desired die d_{f_2} and a given die d_{f_1}, we can roll d_{f_1} a total of $\lceil \log_{f_1} f_2 \rceil$ times and treat each roll as a digit of a base f_1 number, re-rolling if we roll a number greater than or equal to f_2.

At this point, an astute reader may notice some peculiarities with this scheme. First, wouldn't it have been simpler to roll a d_7 by rolling a d_8 and re-rolling only when an 8 is rolled? Yes, it would've been, but I didn't think of that until later. Second, can't this rolling scheme go on infinitely if we always roll a value which is too high? Yes! However, thankfully, this almost surely never happens, which is to say that the probability that this happens approaches 0 in the limiting case of the number of rolls approaching infinity. For any scheme where each roll is independent of the previous and we have T possible outcomes and we accept t of them and re-roll on the other $T-t$, the probability of rolling n rejected values in a row is $\left(\frac{T-t}{T}\right)^n$, which goes to 0 as n approaches infinity, since $T-t$ is clearly positive and less than T, making $0 < \frac{T-t}{T} < 1$.

We can also compute the expected number of throws (how many times we have to roll a set of dice) based on the probability of needing to throw again. Using the same t and T as before, we have to consider the number of throws if we succeed on the first throw and the number of throws if we fail on the first throw. In either case, we have to throw at least once. In the case that we succeed, we're done, so our total is one throw. If we fail, we're back at the beginning, so we have one throw, and we expect to throw the same number of times as if we hadn't thrown at all. Mathematically, this leads to the equation

$E(\theta) = \frac{t}{T} \cdot (1 + E(\theta|\text{success})) + \left(1 - \frac{t}{T}\right) \cdot (1 + E(\theta|\text{failure}))$. Since $E(\theta|\text{success}) = 0$ and $E(\theta|\text{failure}) = E(\theta)$, this equation simplifies to be $E(\theta) = \frac{t}{T} + \left(1 - \frac{t}{T}\right) \cdot E(\theta)$. We can actually solve this algebraically to find that $E(\theta) = T/t$. In the case of rolling a 7 sided die with a 6 sided die with 36 outcomes, as described above, the expected number of throws is thus 36/7, which is about 5.14 throws, while rolling a d_7 with a d_8 results in only $8/7 + 1 \approx 1.14$ throws. Thus, rolling an 8 sided die is clearly better in expectation.

This leads us to a scarier question: can we do better? Clearly, if we were to get T closer to t, then the fraction T/t would decrease, and our expected number of throws would decrease accordingly. We can do this by being slightly clever. While a d_8 seems to do better than two d_6s at first, we can reuse the trick we used for rolling a d_3 with a d_6 to make rolling two d_6s the better option. If we take each of our 36 outcomes and map 5 of them to the value 1, 5 of them to the value 2, etc., then we end up with 7 equally likely groups of 5 outcomes each, with only one outcome left over where we need to re-throw. This means that we only need to throw $36/35 \approx 1.03$ times on average! We could even resort to using different pairs of dice, such as a d_6 and a d_{20}. This gives us 120 outcomes, which we can break up into 7 groups of 17 with one left over, giving us an expectation of only $120/119 \approx 1.01$ throws.

The next follow-up question to this is whether or not we can achieve arbitrarily few throws for 7, or any number in general. It's actually fairly easy to show that this is the case. Take a die with any number of faces f_2 that we want to roll, and n dice with f_1 faces each. We can throw all of the n dice, giving us f_1^n outcomes. Assuming f_1^n is greater than f_2, we can then break these outcomes into f_2 groups with k outcomes left over, where we would need to re-throw. This gives us an expected number of throws $E_n[\theta] = \frac{T}{t} = \frac{f_1^n}{f_1^n - k} = \frac{1}{1 - k/f_1^n}$, which approaches 1 as n approaches infinity since k/f_1^n approaches 0 as n approaches infinity.

This answer is not fully satisfying, however. While we can get the number of total throws arbitrarily close to 1, the total number of dice rolled simultaneously approaches infinity, which is, in some respects, impractical. A more satisfying answer would minimize the total number of dice ever rolled, rather than the number of throws[3]. The expected number of dice rolled is simply the expected number of throws, weighted by the number of dice rolled in each throw. For instance, $E_n[r] = nE_n[\theta]$ in the scheme above, since we threw n dice each time. While we will briefly analyze the general case of rolling a set of heterogeneous dice later, it turns out that minimizing the number of identical dice rolled is much more interesting than minimizing the number of throws. If we take the definition of $E_n[\theta]$ from above and substitute k with the actual value of the

[3]Another more satisfying answer would be to try to simultaneously minimize the number of throws and the number of dice rolled, but the optimal answer in this case is possibly subject to a preference of which value is more important to minimize

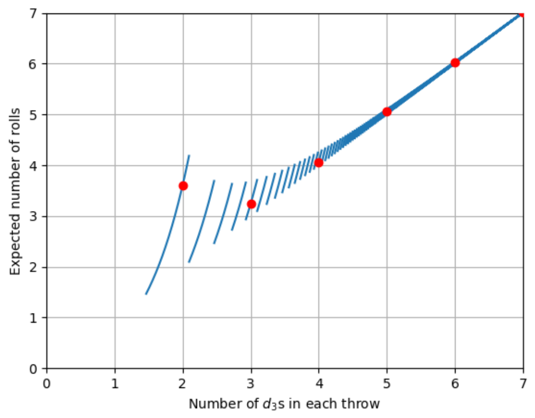

Figure 1: Expected number of dice rolled with $f_1 = 3$ and $f_2 = 5$. Integer values are shown in red.

remainder function, we get the following expression to minimize:

$$E_n[r] = \frac{nf_1^n}{f_1^n - k} = \frac{nf_1^n}{f_1^n - \left(f_1^n - f_2 \left\lfloor \frac{f_1^n}{f_2} \right\rfloor\right)} = \frac{n}{\frac{f_2}{f_1^n} \left\lfloor \frac{f_1^n}{f_2} \right\rfloor}$$

Yuck! That floor function and those exponents don't look very friendly, and it turns out that they complicate things significantly. Some of the problems with this minimization become more apparent when we actually graph this as a function of n for particular values of f_1 and f_2, as shown in figure 2. The most obvious problem with the function is that it is not continuous; the floor function in the denominator makes the function discontinuous for infinitely many values of n. Despite the fact that the graph is piecewise monotone increasing, these discontinuities introduced by the floor function mean that we cannot use information about the slope of the graph in any obvious way to find its minimum values. We also care in particular about integer values of the function, which are evidently not the minima one would obtain from the function when considering non-integer values as well. Quite a predicament!

In order to make some progress on this, it will help to note the bounds of the function. Since the numerator of $E_n[r]$ is n and the denominator is always between 0 and 1, we find that the function always takes a value which is at least n, i.e., $E_n[r] > n$. We also note that $E_n[r]$ is bounded above by $\frac{nf_1^n}{f_1^n - f_2}$, since the remainder k is at most $f_2 - 1$, and the denominator is minimized when k is maximum[4]. A quick inspection the second derivative of this upper bound with respect to n reveals that this function is concave up in the regions where $E_n[r]$

[4]This upper bound has an asymptote where $f_1^n = f_2$. To the left of this asymptote, values of the function are negative, so we ignore them.

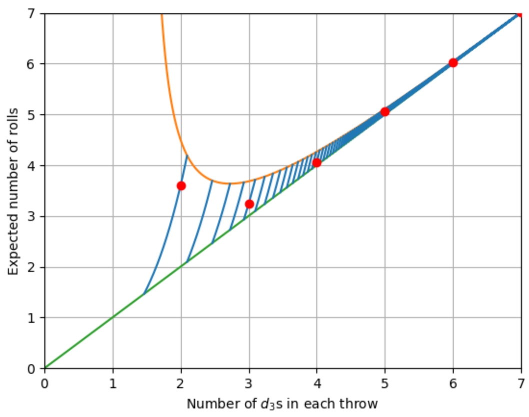

Figure 2: Expected number of dice rolled with $f_1 = 3$ and $f_2 = 5$. Integer values are shown in red. Upper bound is shown in orange, lower bound in green.

is defined. Now, since the function we want to minimize is bounded below by an increasing function and bounded above by a function which is concave up, it follows that there is some minimum value m such that $E_n[r] > E_m[r]$ for all $n > m$. Thus, it seems useful to find when $E_n[r]$ always increases when we add another die, i.e., $E_{n+1}[r] > E_n[r]$ for all positive integers $n > m$. For this, we can subtract the upper bound for $E_n[r]$ from the lower bound for $E_{n+1}[r]$, since this is the smallest the difference between consecutive terms could possibly be. Through some tedious algebra, we find:

$$E_{n+1}[r] - E_n[r] > 0 \tag{1}$$

$$(n+1) - \left(\frac{nf_1^n}{f_1^n - f_2}\right) > 0 \tag{2}$$

$$\frac{nf_1^n + f_1^n - nf_2 - f_2}{f_1^n - f_2} - \frac{nf_1^n}{f_1^n - f_2} > 0 \tag{3}$$

$$\frac{(f_1^n - f_2) - nf_2}{f_1^n - f_2} > 0 \tag{4}$$

$$1 - \frac{nf_2}{f_1^n - f_2} > 0 \tag{5}$$

$$f_1^n > nf_2 + f_2 \tag{6}$$

At this point, it is clear that we cannot easily eliminate the exponential term from this equation. Those familiar with exponential equations may recognize that we will need to resort to using the Lambert W function in order to solve for n. To those unfamiliar with exponential equations, this is the function W such that $W(ve^v) = v$ for any expression v. While the W function cannot be expressed in terms of elementary functions, it is practical to evaluate because its values can be computed easily via Newton's method. It will be useful for our purposes if we can get the exponential portion of our equation into a "canonical" form, i.e., the form $C(n)e^{C(n)}$, where C is some non-exponential function of n, since we can then apply the relation $W(C(n)e^{C(n)}) = C(n)$ to eliminate the exponentials from our equation. This results in the following manipulations, all of which are simply convincing the coefficient of the exponential term to look more and more like the exponent of the exponential term:

$$1 < (nf_2 + f_2)f_1^{-n} \tag{7}$$

$$-1 > (-nf_2 - f_2)f_1^{-n} \tag{8}$$

$$-1 > (-nf_2 - f_2)e^{-n\log f_1} \qquad \text{by } a^x = e^{x\log a} \tag{9}$$

$$-\log f_1 > (-nf_2\log f_1 - f_2\log f_1)e^{-n\log f_1} \tag{10}$$

$$-\frac{\log f_1}{f_2} > (-n\log f_1 - \log f_1)e^{-n\log f_1} \tag{11}$$

$$-\frac{\log f_1}{f_2}e^{-\log f_1} > (-n\log f_1 - \log f_1)e^{-n\log f_1 - \log f_1} \tag{12}$$

$$-\frac{\log f_1}{f_1 f_2} > (-n\log f_1 - \log f_1)e^{-n\log f_1 - \log f_1} \tag{13}$$

Now, the right hand side is in the canonical form with $C(n) = -n\log f_1 - \log f_1$, so we can apply the W function to both sides and simplify:

$$W\left(-\frac{\log f_1}{f_1 f_2}\right) > -n\log f_1 - \log f_1 \tag{14}$$

$$\frac{W\left(-\frac{\log f_1}{f_1 f_2}\right) + \log f_1}{\log f_1} > -n \tag{15}$$

$$\frac{W\left(-\frac{\log f_1}{f_1 f_2}\right)}{\log f_1} + 1 > -n \tag{16}$$

$$-\frac{W\left(-\frac{\log f_1}{f_1 f_2}\right)}{\log f_1} - 1 < n \tag{17}$$

meaning that for all n greater than the value of the left hand expression, $E_{n+1}[r] > E_n[r]$. With this (fairly complicated) expression[5], we now have a limited set of candidates to check to find the integer number of dice which globally minimizes the expected number of dice rolled. We only need to check the few values between $\lceil \log_{f_1} f_2 \rceil$ and this value[6], since all possible larger numbers will require more rolls in expectation.

[5] This expression is even more complicated than I'm letting on. Since the W function is multi-valued, we need to be careful to take the value which makes the number of expected rolls positive.

[6] Because xe^x grows faster than e^x, the W function grows slower than the logarithm function, meaning that we only have to consider $O(\log f_1 + \log f_2)$ possible minimum values.

In general, for a given set of heterogenous dice D from which we must choose a subset P to roll, the optimal subset is algorithmically simple to define, but computationally inefficient to construct. We can simply enumerate all $2^{|D|}$ subsets $P \subseteq D$ of the dice and find the subset which minimizes the value:

$$E_{P\subseteq D}[r] = \frac{|P|}{\frac{f_2}{\prod_{d_{f_1}\in P} f_1}\left\lfloor \frac{\prod_{d_{f_1}\in P} f_1}{f_2} \right\rfloor}.$$

However, this obviously takes exponential time and becomes intractible for even small sets of dice, e.g., 20 dice. A slightly less naïve algorithm for computing the optimal subset would only consider the subsets whose product is at least f_2, though this could still comprise every or nearly every subset of D, such as in the event that each element of D has a face value of at least f_2. Because of the floor function and the cyclic nature of the resulting function expectation function, there is no optimal substructure to the problem which would allow us to eliminate groups of candidate subsets at a time, as is normally done by dynamic programming. Thus, it seems that there is no easy way to compute the optimal number of rolls for an arbitrary subset of given dice efficiently.

There is one last trick[7] that we can use to reduce the expected number of rolls in general[8]. When we have a set of dice D such that $p = \prod_{d_f\in D} d_f = f_2 + k$, we can "reuse" rejected rolls. Specifically, there will be k possible rolls which are rejected, which we can enumerate. Since each of these rolls is equally likely, we can treat the enumerated value of the rejected roll as an extra roll of a k sided die on our next roll. For instance, if we were rolling a d_{13} with a d_{24} (rolled as a d_6 and a d_4) and we rolled a 16, instead of throwing out the value of 16, we can treat it as if we rolled a 3 on an 11 sided die. This means we will have to modify our next roll, e.g., by only rolling the d_4, giving us effectively a d_{44} when combined with the roll of the d_{11}. This complicates the analysis significantly, but essentially gives free rolls, thus reducing the total number of rolls in some cases. It is worth noting that because the "free die" you receive will never divide f_2, none of the modified rolls will ever divide f_2, and, as such, this scheme can still technically take infinitely many rolls to produce a valid value.

Well, that's all I have on this topic so far. I'm not sure what this could possibly be used for. It's technically a form of rejection sampling for some very constrained discrete distributions, but in any sort of computational application it's faster to just generate a continuous random value from 0 to 1 and scale it to the size that you need. I would like to find an algorithm which will guarantee a valid value in finitely many rolls, but I'm not sure whether or not an algorithm exists (I'm inclined to think it doesn't). Either way, I had fun thinking about this problem, and I hope you enjoyed reading about it!

[7] As far as I know.

[8] Credit for this idea goes to Kenneth Adams.