Tossing a Biased Coin

Michael Mitzenmacher[*]

When we talk about a coin toss, we think of it as unbiased: with probability one-half it comes up heads, and with probability one-half it comes up tails. An ideal unbiased coin might not correctly model a real coin, which could be biased slightly one way or another. After all, real life is rarely fair.

This possibility leads us to an interesting mathematical and computational question. Is there some way we can use a biased coin to efficiently simulate an unbiased coin? Specifically, let us start with the following problem:

Problem 1. Given a biased coin that comes up heads with some probability greater than one-half and less than one, can we use it to simulate an unbiased coin toss?

[*] Digital Equipment Corporation, Systems Research Center, Palo Alto, CA.

A simple solution, attributed to von Neumann, makes use of symmetry. Let us flip the coin twice. If it comes up heads first and tails second, then we call it a 0. If it comes up tails first and heads second, then we call it a 1. If the two flips are the same, we flip twice again, and repeat the process until we have a unbiased toss. If we define a round to be a pair of flips, it is clear that we the probability of generating a 0 or a 1 is the same each round, so we correctly simulate an unbiased coin. For convenience, we will call the 0 or 1 produced by our simulated unbiased coin a *bit*, which is the appropriate term for a computer scientist.

Interestingly enough, this solution works regardless of the probability that the coin lands heads up, even if this probability is unknown! This property seems highly advantageous, as we may not know the bias of a coin ahead of time.

Now that we have a simulation, let us determine how efficient it is.

Problem 2. Let the probability that the coin lands heads up be p and the probability that the coin lands tails up be $q = 1 - p$. On average, how many flips does it take to generate a bit using von Neumann's method?

Let us develop a general formula for this problem. If each round takes exactly f flips, and the probability of generating a bit each round is e, then the expected number of total flips t satisfies a simple equation. If we succeed in the first round, we use exactly f flips. If we do not, then we have flipped the coin f times, and because it is as though we have to start over from the beginning again, *the expected remaining number of flips is still t*. Hence t satisfies

$$t = ef + (1-e)(f+t).$$

or, after simplifying

$$t = f/e.$$

Using von Neumann's strategy, each round requires two flips. Both a 0 and a 1 are each generated with probability pq, so a round successfully generates a bit with probability $2pq$. Hence the average number of flips required to generate a bit is $f/e = 2/2pq = 1/pq$. For example, when $p = 2/3$, we require on average $9/2$ flips.

We now know how efficient von Neumann's initial solution is. But perhaps there are more efficient solutions? First let us consider the problem for a specific probability p.

Problem 3. Suppose we know that we have a biased coin that comes up heads with probability $p = 2/3$. Can we generate a bit more efficiently than by von Neumann's method?

We can do better when $p = 2/3$ by matching up the possible outcomes a bit more carefully. Again, let us flip the coin twice each round, but now we call it a 0 if two heads come up, while we call it a 1 if the tosses come up different. Then we generate a 0 and a 1 each with probability $4/9$ each round, instead of the $2/9$ using von Neumann's method. Plugging into our formula for $t = f/e$, we use $f = 2$ flips per round and the probability e of finishing each round is $8/9$. Hence the average number of coin flips before generating a bit drops to $9/4$.

Of course, we made strong use of the fact that p was 2/3 to obtain this solution. But now that we know that more efficient solutions might be possible, we can look for methods that work for any p. It would be particularly nice to have a solution that, like von Neumann's method, does not require us to know p in advance.

Problem 4. Improve the efficiency of generating a bit by considering the first four biased flips (instead of just the first two).

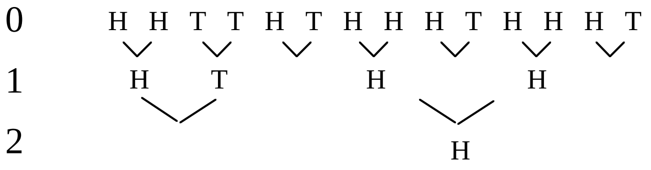

Figure 1: The Multi-Level Strategy.

Consider a sequence of four flips. If the first pair of flips are H T or T H, or the first pair of flips are the same but the second pair are H T or T H, then we use von Neumann's method. We can improve things, however, by pairing up the sequences H H T T and T T H H; if the first sequence appears we call it a 0, and if the second sequence appears we call it a 1. That is, if both pairs of flips are the same, but the pairs are different, then we can again decide using von Neumann's method, *except that we consider the order of the pairs of flips*. (Note that our formula for the average number of flips no longer applies, since we might end in the middle of our round of four flips.)

Once we have this idea, it seems natural to extend it further. A picture here helps– see Figure 1. Let us call each flip of the actual coin a Level 0 flip. If Level 0 flips $2j-1$ and $2j$ are different, then we can use the order (heads-tails or tails-head) to obtain a bit. (This is just von Neumann's method again.) If the two flips are the same, however, then we will think of them as providing us with what we shall call a Level 1 flip. If Level 1 flips $2j-1$ and $2j$ are different, again this gives us a bit. But if not, we can use it to get a Level 2 flip, and so on. We will call this the Multi-Level strategy.

Problem 5a. What is the probability we have not obtained a bit after flipping a biased coin 2^k times using the Multi-Level strategy?

Problem 5b (HARD!). What is the probability we have not obtained a bit after flipping a biased coin ℓ times using the Multi-Level strategy??

Problem 5c (HARDEST!)**.** How many biased flips does one need on average before obtaining a bit using the Multi-Level strategy?

For the first question, note that the only way the Multi-Level strategy will not produce a bit after 2^k tosses is if all the flips have been the same. This happens with probability $p^{2^k} + q^{2^k}$.

Using this, let us now determine the probability the Multi-Level strategy fails to produce a bit in the first ℓ bits, where ℓ is even. (The process never ends on an odd flip!) Suppose that $\ell = 2^{k_1} + 2^{k_2} + \ldots 2^{k_m}$, where $k_1 > k_2 > \ldots > k_m$. First, the Multi-Level strategy must last the first 2^{k_1} flips, and we have already determined the probability that this happens. Next, the process must last the next 2^{k_2} flips. For this to happen, all of the next 2^{k_2} flips have to be the same, *but they do not have to be the same as the first 2^{k_1} flips.* Similarly, each of the next 2^{k_3} flips have to be the same, and so on. Hence the probability of not generating a bit in ℓ flips is

$$\prod_{i=1}^{m} (p^{2^{k_i}} + q^{2^{k_i}}).$$

Given the probability that the Multi-Level strategy requires at least ℓ flips, calculating the average number of flips t_2 before the Multi-Level strategy produces a bit still requires some work. Let $P(\ell)$ be the probability that the Multi-Level strategy takes exactly ℓ flips to produce a bit, and let $Q(\ell)$ be the probability that the Multi-Level strategy takes more than ℓ flips. Of course, $P(\ell) = 0$ unless ℓ is even, since we cannot end with an odd number of flips! Also, for l even it is clear than $P(\ell) = Q(\ell - 2) - Q(\ell)$, since the right hand side is just the probability that the Multi-Level strategy takes ℓ flips. Finally, we previously found that $Q(\ell) = \prod_{i=1}^{m} (p^{2^{k_i}} + q^{2^{k_i}})$ above.

The average number of flips is, by definition,

$$t_2 = \sum_{\ell \geq 2, \ell \text{ even}} P(\ell) \cdot \ell$$

We change this into a formula with the values $Q(\ell)$, since we already know how to calculate them.

$$\begin{aligned} t_2 &= \sum_{\ell \geq 2, \ell \text{ even}} P(\ell) \cdot \ell \\ &= \sum_{\ell \geq 2, \ell \text{ even}} (Q(\ell - 2) - Q(\ell)) \cdot \ell \end{aligned}$$

Now we use a standard "telescoping sum" trick; we re-write the sum by looking at the coefficient of each $Q(\ell)$.

$$\begin{aligned} t_2 &= \sum_{\ell \geq 2, \ell \text{ even}} P(\ell) \cdot \ell \\ &= \sum_{\ell \geq 2, \ell \text{ even}} (Q(\ell - 2) - Q(\ell)) \cdot \ell \\ &= \sum_{\ell \geq 0, \ell \text{ even}} Q(\ell)(\ell + 2 - \ell) \\ &= 2 \sum_{\ell \geq 0, \ell \text{ even}} Q(\ell) \end{aligned}$$

This gives an expression for the average number of biased flips we need to generate a bit. It turns out this sum can be simplified somewhat, as using the expression for $Q(\ell)$ above we have

$$2 \sum_{\ell \geq 0, \ell \text{ even}} Q(\ell) = 2 \prod_{k \geq 1} (1 + p^{2^k} + q^{2^k}).$$

Up to this point, we have tried to obtain just a single bit using our biased coin. Instead, we may want to obtain several bits. For example, a computer scientist might need a collection of bits to apply a randomized algorithm, but the only source of randomness available might be a biased coin. We can obtain a sequence of bits with the Multi-Level strategy in the following way: we flip the biased coin a large number of times. Then we run through each of the levels, producing a bit for each heads-tails or tails-heads pair. This works, but there is still more we can do if we are careful.

Problem 6. Improve upon the Multi-Level strategy for obtaining bits from a string of biased coin flips. Hint: consider recording whether each pair of flips provides a bit via von Neumann's method or not.

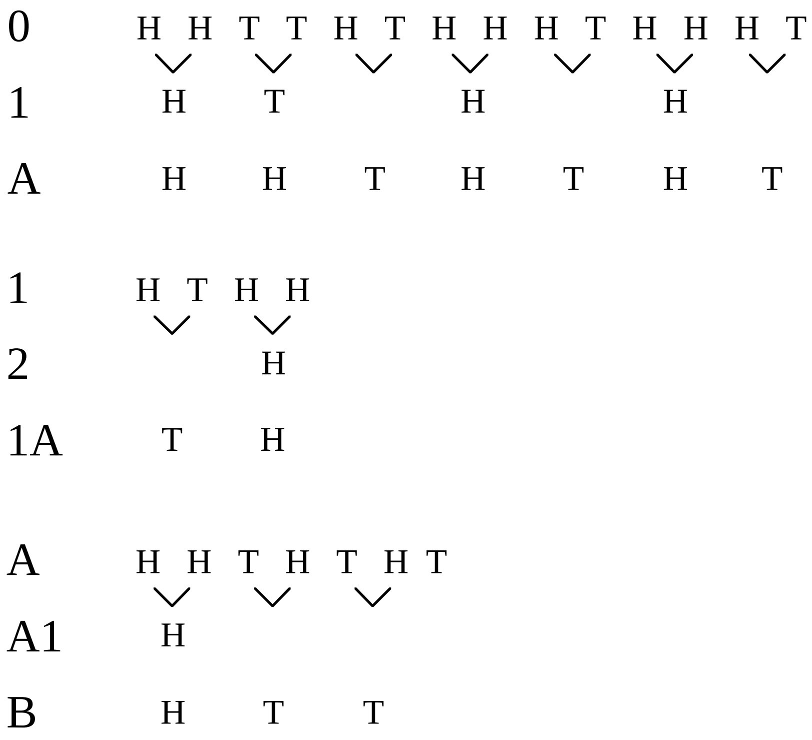

Figure 2: The Advanced Multi-Level Strategy. Each sequence generates two further sequences. Bits are generated by applying von Neumann's rule to the sequences in some fixed order.

The Multi-Level strategy does not take advantage of *when* each level provides us with a bit. For example, in the Multi-Level strategy, the sequences H H H T and H T H H produce the same single bit. However, since these two sequences occur with the same probability, we can pair up these two sequences to provide us with a second bit; if the first sequence comes up, we consider that a 0, and if the second comes up, we can consider it a 1.

To extract this extra randomness, we expand the Multi-Level strategy to the Advanced Multi-Level strategy. Recall that in the Multi-Level strategy, we used Level 0 flips to generate a sequence of Level 1 flips. In the Advanced Multi-Level strategy, we determine two sequences from Level 0. The first sequence we extract will be Level 1 from the Multi-Level Strategy. For the second sequence, which we will call Level A, flip j records whether flips $2j-1$ and $2j$ are the same or different in Level 0. If the flips are different, then the flip in Level A will be tails, and otherwise it will be heads. (See Figure 2.) Of course, we can repeat this process, so from each of both Level 1 and Level A, we can ge two new sequences, and so on. To extract a sequence of bits, we go through all these sequences in a fixed order and use von Neumann's method.

How good is the Advanced Mult-Level Strategy? It turns out that it is essentially as good as you can possibly get. This is somewhat difficult to prove, but we can provide a rough sketch of the argument.

Let $A(p)$ be the average number of bits produced for each biased flip, when the coin comes us heads with probability p. For convenience, we think of ths average over an infinite number of flips, so that we don't have to worry about things like the fact that if we end on an odd flip, it cannot help us. We first determine an equation that describes $A(p)$.

Consider a consecutive pair of flips. First, with probability $2pq$ we get H T or T H, and hence get out one bit. So on average von Neumann's trick alone yields pq bits per biased flip. Second, for every two flips, we always get a single corresponding flip for Level A. Recall that we call a flip on Level A heads if the two flips on Level 0 are the same and tails if the two flips are different. Hence for Level A, a flip is heads with probability p^2+q^2. This means that for every two flips on Level 0, we get one flip on Level A, with a coin that has a different bias– it is heads with probability p^2+q^2. So for every two biased Level 0 flips, we get (on average) $A(p^2+q^2)$ bits from Level A. Finally, we get a flip for Level 1 whenever the two flips are the same. This happens with probability p^2+q^2. In this case, the flip at the next level is heads with probability $p^2/(p^2+q^2)$. So on average each two Level 0 flips yields (p^2+q^2) Level 1 flips, where the Level 1 flips again have a different bias, and thus yield $A(p^2/(p^2+q^2))$ bits on average. Putting this all together yields:

$$A(p) = pq + \frac{1}{2}A(p^2+q^2) + \frac{1}{2}(p^2+q^2)A\left(\frac{p^2}{p^2+q^2}\right).$$

Problem 7. What is $A(1/2)$?

Plugging in yields $A(1/2) = 1/4 + A(1/2)/2 + A(1/2)/4$, and hence $A(1/2) = 1$. Note that $A(1/2)$ is the average number of bits we obtain per flip when we flip a coin that comes up heads with probability $1/2$. This result is somewhat surprising: it says the Advanced Multi-Level strategy extracts (on average, as the number of flips goes to infinity) 1 bit per flip of an unbiased coin, and this is clearly the best possible! This gives some evidence that the Advanced Multi-Level strategy is doing as well as can be done.

You may wish to think about it to convince yourself there is no other randomness lying around that we are not taking advantage of. Proving that the Advanced Multi-Level strategy is optimal is, as we have said, rather difficult. (See the paper "Iterating von Neumann's Procedure" by Yuval Peres, in *The Annals of Statistics*, 1992, pp. 590-597.) It helps to know the average rate that we could ever hope to extract bits using a biased coin that comes up heads with probability p and tails with probability $q = 1 - p$. It turns out the correct answer is given by the entropy function $H(p) = -p\log_2 p - q\log_2 q$. (Note $H(1/2) = 1$; see Figure 3.) We will not even attempt to explain this here; most advanced probability books explain entropy and the entropy function. Given the entropy function, however, we may check that our recurrence for $A(p)$ is satisfied by $A(p) = H(p)$.

Problem 8. Verify that $A(p) = H(p)$ satisfies the recurrence above.

This derivation itself is non-trivial! Let us plug in $A(p) = H(p)$ on the right hand side of the recurrence and simplify. First,

$$\begin{aligned}
\frac{1}{2}H(p^2+q^2) &= -\frac{1}{2}(p^2+q^2)\log_2(p^2+q^2) - \frac{1}{2}(1-p^2-q^2)\log_2(1-p^2-q^2) \\
&= -\frac{1}{2}(p^2+q^2)\log_2(p^2+q^2) - pq\log_2(2pq) \\
&= -\frac{1}{2}(p^2+q^2)\log_2(p^2+q^2) - pq - pq\log_2 p - pq\log_2 q,
\end{aligned}$$

where we have used the fact that $1-p^2-q^2 = 2pq$. Second,

$$\begin{aligned}
\frac{1}{2}(p^2+q^2)H(\frac{p^2}{p^2+q^2}) &= -\frac{1}{2}p^2\log_2\frac{p^2}{p^2+q^2} - \frac{1}{2}q^2\log_2\frac{q^2}{p^2+q^2} \\
&= -\frac{1}{2}p^2\log_2 p^2 + \frac{1}{2}p^2\log_2(p^2+q^2) - \frac{1}{2}q^2\log_2 q^2 + \frac{1}{2}q^2\log_2(p^2+q^2) \\
&= -p^2\log_2 p - q^2\log_2 q + \frac{1}{2}(p^2+q^2)\log_2(p^2+q^2)
\end{aligned}$$

Now the right hand side simplifies dramatically:

$$\begin{aligned}
pq + \frac{1}{2}A(p^2+q^2) + \frac{1}{2}(p^2+q^2)A(\frac{p^2}{p^2+q^2}) &= pq - \frac{1}{2}(p^2+q^2)\log_2(p^2+q^2) - pq - pq\log_2 p - pq\log_2 q \\
&\quad - p^2\log_2 p - q^2\log_2 q + \frac{1}{2}(p^2+q^2)\log_2 p^2 + q^2 \\
&= -pq\log_2 p - pq\log_2 q - p^2\log_2 p - q^2\log_2 q \\
&= -p(p+q)\log_2 p - q(p+q)\log_2 q \\
&= -p\log_2 p - q\log_2 q \\
&= H(p)
\end{aligned}$$

Notice that we used the fact that $p+q=1$ in the third line from the bottom. As the right hand side simplifies to $H(p)$, the function $H(p)$ satisfies the recurrence for $A(p)$.

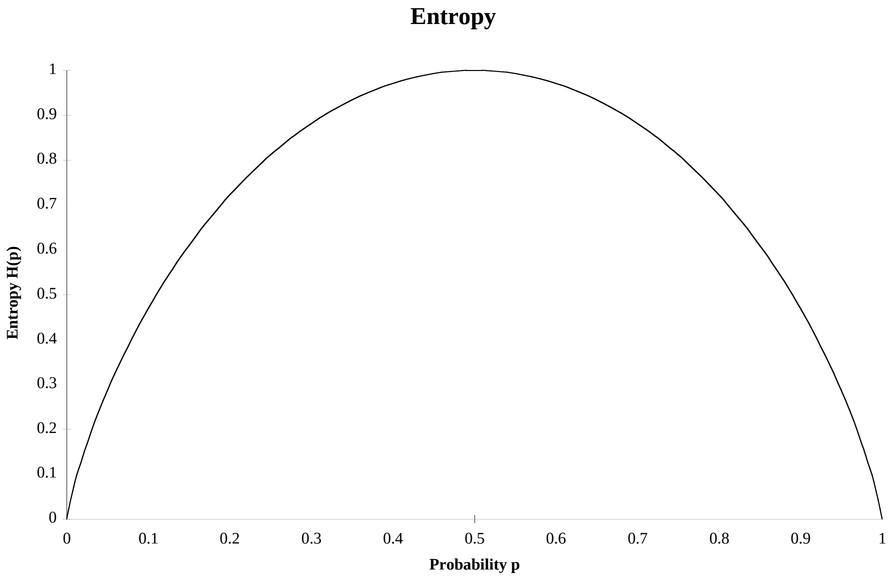

Figure 3: A Graph of the Entropy $H(p)$ vs p.

We hope this introduction to biased coins leads you to more questions about randomness and how to use it. Now, how do you simulate an unbiased die with a biased die...