Recall the introductory lecture where I discussed the game 20 questions and what it had to do with Data Compression. One example I gave is that person A thinks of a specific very large number. Even if person B knows that A is thinking of a number, it is clear that B has no strategy that guarentees it can get the number by asking just 20 questions.

In the last lecture, we discussed possible ways of coding the set of positive integers. Let's think of this as a game of "20 questions" where person A chooses a positive integer i and B is allowed as many questions as he wants. The sequence of answers is $C(i)$. We would like a strategy for asking questions so that the answer uses as few bits a possible on average, i.e. the average codeword length is as small as possible.

A unary code for i may be interpreted as B asking on the i^{th} question whether the number is i. A Golomb code may be interpreted as as asking on the i^{th} question whether $i \in \{b(i-1)+1, bi\}$ (and if the answer is yes, then $\log b$ questions are used to find i. (Recall the unary code is a specific case of the Golomb code, where $b = 1$.)

Golomb codes can be quite large for large numbers, since λ_i grows linearly with i. Today we will consider alternative codes for the set of positive numbers, which grow sub-linearly with i.

In particular, Golomb codes work well when the probabilities of integers i falls off exponentially fast with i. However, there are many other fall off rates for the probabilities on the integers. For example, what if $p(i)$ is roughly ai^{-2} for some constant a? This is possible, since $\sum_{i=1}^{\infty} i^{-2}$ is finite.[1] Or what about some other falloff rate for $p(i)$ that is slower than exponential?

Elias code 1

Because we are restricting ourselves to prefix codes, we are limited in how slowly we can grow the codeword lengths. The slowest possible growth rate for λ_i, *if we want to ensure each i has a different codeword* would be to let $C(i)$ be the binary representation of i. The binary representation of an integer $i > 0$ has $\lfloor \log i \rfloor + 1$ bits. So this is clearly a lower bound on λ_i. However, the binary representation of i does not give us a prefix code $C(i)$. So we cannot achieve this lower bound.

We can this idea to define prefix codes which come closer to this lower bound than the Golomb code does. I will present two such codes, which were introduced by Peter Elias in the 1970s. Here is the first code.

Assume the leftmost bit of the binary representation of i is 1. The *Elias1 code* encodes i in two parts: the first part is $\lfloor \log i \rfloor$ 0's; the second part is the binary representation of i.

The codeword lengths are

$$\lambda_i = 2\lfloor \log i \rfloor + 1$$

which is obviously sub-linear. i.e. the codeword lengths grow slower than the Golomb codewords.

For what probabilities $p(i)$ does the Elias code produce an average code length that is equal to the entropy? In order for $\lambda(i) \approx \log \frac{1}{p(i)}$, we want $p(i) \approx 2^{-\lambda_i}$. Plugging in λ_i from above, we get

$$p(i) \approx \frac{1}{2\ i^2}.$$

which clearly falls off much less slowly than exponential.

The first several codewords (and lengths) are:

[1] Euler proved in 1735 that $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$. For a very nice description of the proof and other fun stuff, see the book "Euler: the master of us all".

i	$C(i)$	λ_i
1	1	1
2	01 0	3
3	01 1	3
4	001 00	5
5	001 01	5
6	001 10	5
7	001 11	5
8	0001 000	7
$\vdots$		

A second way to interpret this code is to form groups of integers as in a Golomb code, but to let the group size increase as 2^{g-1} where g is the group number. That is, let the group sizes be $1, 2, 4, 8, 16, 32, \ldots, 2^{g-1}$.

$$\{1\},\ \{2,3\},\ \{4,5,6,7\},\ \{8,9,10,11,12,13,14,15\},\quad \text{etc.}$$

The group number g for integer i is $\lfloor \log i \rfloor + 1$.

We can now define the Elias code in two parts: first, a unary code for the group number that i belongs to; second, a fixed length binary code that specifies the element within the group. To encode the group number that i belongs to requires $\lfloor \log i \rfloor + 1$ bits. To encode the position of i within the group requires $\lfloor \log i \rfloor$ bits.

Elias code 2

The above Elias code works great if the falloff rate of probabilities $p(i)$ is about i^{-2}. But what if the falloff rate is slower than that. The falloff rate cannot be as slow as i^{-1} since $\sum_{i=1}^{\infty} i^{-1}$ is infinite, and we do want a code that works for all i. [2] But there are falloff rates between i^{-2} and i^{-1} where the sum does converge.

For this, Elias extended a second code which I'll call Elias2. The idea of the Elias2 code is to replace the unary code prefix of the Elias1 code with the Elias1 codeword of the length of that prefix. The second part of the Elias2 code which specifies the element within the group is unchanged.

The unary prefix for the Elias1 code was $\lfloor \log i \rfloor + 1$ bits, and so the Elias1 codeword that replaces this prefix is $2\lfloor \log(\lfloor \log i \rfloor + 1) \rfloor + 1$ bits. The second part of the Elias2 code is the same as the second part of the Elias1 code and hence uses $\lfloor \log i \rfloor$ bits. Thus,

$$\lambda(i) = 2\lfloor \log(\lfloor \log i \rfloor + 1) \rfloor + 1 + \lfloor \log i \rfloor$$

For what probability $p(i)$ would this new Elias code result in an average code length that is near the entropy? Consider $p(i) \approx 2^{-\lambda(i)}$. Ignoring the floors in the above equation, we would get

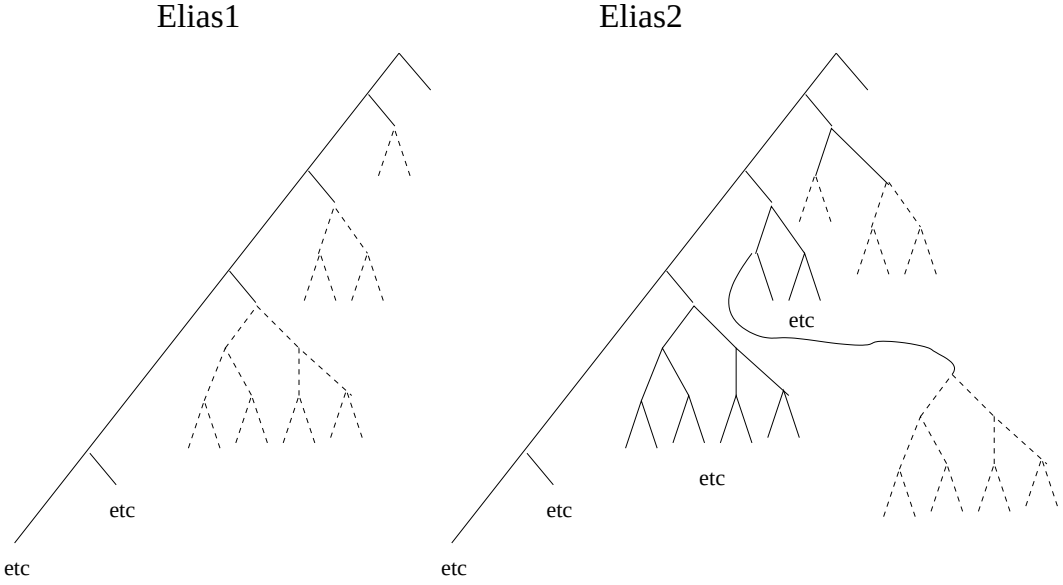
$$p(i) \approx 2^{-(2\log(\log i + 1) + 1 + \log i)} \approx \frac{1}{2i(\log i + 1)^2}$$

Notice that these probabilities fall off slower that the probabilities for the Elias1 code. The slower falloff rate in probabilities is a direct result of the slower growth rate of $\lambda(i)$, which is due to the slower growth of the prefix.

[2] If we restrict i to be less than some finite N, then we can indeed consider a falloff rate of $p(i) = i^{-1}$.

i	$C(i)$	λ_i
1	1	1
2	010 0	4
3	010 1	4
4	011 00	5
5	011 01	5
6	011 10	5
7	011 11	5
8	00100 000	8
$\vdots$		

The figure below illustrates the two Elias codes. The solid lines represent the prefix part of the code, which codes the group number g. The dashed lines represent the suffix part of the code, specifying one element within group g. Notice that the Elias1 code is used within the Elias2 code, namely it encodes the group number g. As is evident from the figure, the Elias2 code grows more slowly than the Elias1 code (though at first the Elias2 code ground more quickly e.g. compare the codewords for $i = 2, 3$).



Is it possible to grow slower than this? Yes. You could apply the same idea again, replacing the unary code within the Elias2 code by some other code whose codeword length grows more slowly than the unary code.

Notice that no matter how clever or complicated the scheme gets, we cannot have a code such that λ_i grows at the rate $\log i$. As we mentioned earlier, the binary representation of i has $\lfloor \log i \rfloor + 1$ bits. But the binary representation of i is not a prefix code for i. Far from it! What we are trying to do is come up with a prefix code for the integers i which grows only slightly faster than $\log i$.