the best way to count

footnotes & transcripts

Lucilla

introduction

[**i.a**] some people don't know Arabic numerals by name:
https://independent.co.uk/news/arabic-numerals-survey-prejudice-bias-survey-research-civic-science-a8918256.html

[**i.b**] Greek decimal mysticism:
https://en.wikipedia.org/wiki/Tetractys

[**i.c**] binary in ancient Egyptian mathematics:
https://en.wikipedia.org/wiki/Ancient_Egyptian_mathematics#Multiplication_and_division

[**i.d**] Leibniz's *Explanation of Binary Arithmetic*:
http://leibniz-translations.com/binary.htm

[**i.e**] Mangarevan binary number system:
https://ncbi.nlm.nih.gov/pmc/articles/PMC3910603/

[**i.f**] binary in the Arecibo message:
https://en.wikipedia.org/wiki/Arecibo_message#Numbers

[**i.g**] BASE OFF:
https://seximal.net/s/BASEOFF.zip

[**i.h**] Artifexian's *Inventing a Number System*:
https://youtu.be/H5EUjnEKzjQ?t=260

chapter zero

[**0.a**] languages and writing systems convey information at about the same rate:
https://pubmed.ncbi.nlm.nih.gov/22486107/
https://scientificamerican.com/article/fast-talkers/
https://content.time.com/time/health/article/0,8599,2091477,00.html
https://persquaremile.com/2011/12/21/which-reads-faster-chinese-or-english/

[**0.b**] because the length of numbers in any base grows logarithmically, and logarithms to any two bases are always proportional to each other, so are the growth rates of numbers in any two bases. so in an ideal situation, if every base could have symbols that are proportionally more lightweight or packed, depending on the size of the base, then their relative lengths would be entirely dependent on their *radix economy*, which is something we'll also get to.

[**0.c**] a decent ASCII approximation of this system is `.` for 0, and `|` for 1. for grouping, you can put spaces to delimit groups, or even use underlining if you're feeling fancy. handwritten forms, if distinct from the "printed" ones at all, could look something like the letters u and ɯ.

[**0.d**] for just one implication, the single most cited paper in psychology (*The Magical Number Seven, Plus Or Minus Two*) states that the optimal number of items to have in short-term memory falls somewhere around seven. many binary numbers will have more digits than that, but just reimagining them as octal indeed seems to make them a lot easier to keep in memory, even though no actual base conversion is required and the information content is still exactly the same.

[**0.e**] in general, whenever you're using digits that cannot be instantly broken down and arbitrarily regrouped, this benefit is lost. this also applies to digits whose design is inspired by the bits, or where the bits are stacked vertically, giving a privilege to one specific grouping scheme over all others, like these:
https://arxiv.org/ftp/arxiv/papers/1707/1707.03751.pdf
https://en.wikipedia.org/wiki/Bibi-binary

[**0.f**] in base b^n, the probability that a k-digit number will have *less* than $n \times k$ digits in base b rapidly converges to $1/b$ as n increases.

[**0.g**] Misali was born in "1JI", three niftimal digits; or "13130", not six, but *five* seximal digits.

[**0.h**] the genetic code represents the roughly twenty amino acids using triplets of nitrogen bases, of which there are four kinds. four cubed is 64, the smallest power of four sufficient to represent all amino acids distinctly. the same number of combinations could be represented using sextuplets if there were only two kinds of bases – but *quintuplets* would already be enough. so barring biological implications such as safety of random mutations, a binary genetic code would be more efficient!

[**0.i**] Ithkuil uses base one hundred:
http://ithkuil.net/newithkuil_13_numbers.htm

chapter one

[**1.a**] to be a bit nitpicky, though, *a better way to count* got the numbers wrong: dozenal out-bases decimal at 10^{13}, not 12^{13}. after all, 12^{13} is when *dozenal* numbers get longer by one digit, so if they've been out-basing decimal by then, they have before as well. similarly, decimal out-bases seximal at 6^4, not 10^4.

[**1.b**] for example, a base-9 digit is worth *two* base-3 digits, not three. accordingly, $\log 9 = 2 \times \log 3$.

[**1.c**] obviously, base e can't possibly be the most efficient. not only does it represent every positive integer greater than 2 as an infinite aperiodic sequence of digits: you can't even have "e distinct digits" in the first place, so base e needs to have 3 digits – which automatically makes it wasteful, since with 3 digits it could be more efficient by having 3 as its base.

[**1.d**] base 3 misconception:
https://en.wikipedia.org/wiki/Ternary_numeral_system#Practical_usage
https://seximal.net/names-of-other-bases

[**1.e**] the number 0 is a bit of a strange case. in some contexts it makes sense that 0 should correspond to the empty string, but that's not actually practically usable. it's an exception, but we can just ignore it.

[**1.f**] logarithm... to what base? we need to make an arbitrary choice here, but it turns out it doesn't actually matter, since in the formula for radix economy we'll divide by such a logarithm again, and this arbitrary choice of base will cancel out.

[**1.g**] as a bonus, if we compare binary and *decimal*, the first range where decimal is more efficient is from 2^{53} to 10^{16}, which has to do with 2^{53} starting with a 9 in decimal. more on that in a bit.

[**1.h**] base 4 only looks more efficient than base 2 when the digits of both are artificially bloated up to the same (decimalish) size: we've already established in chapter zero that this is unfair and wasteful.

[**1.i**] the "$1/\log_{b-1}(n)$" part is equivalent to "$\log_n(b-1)$", which is how it is to be read (and how Desmos evaluates it) for $b = 2$.

[**1.j**] Desmos graphs in this chapter:
https://desmos.com/calculator/piu9zofsdn (out-base)
https://desmos.com/calculator/hqdwv2elke (radix cost)
https://desmos.com/calculator/qbkzr0dywv (radix economy)

[**1.k**] radix economy on Wikipedia:
https://en.wikipedia.org/wiki/Radix_economy

[**1.l**] another way to put this has to do with that 2^{53} thing from earlier. notice how for every base there are small regions where the radix economy function falls below 1? these maximum efficiency regions correspond to numbers where the leading digit has its highest possible value – which it *always* has in binary.

[**1.m**] this fact is actually used in many implementations of floating-point arithmetic.

[**1.n**] to be precise, this is what causes the huge step up in efficiency from base 3 to base 2 – it's not what causes base 2 to be the most efficient in the first place. even without this leading digit thing, binary would still win, it'd just be a lot harder to see.

[**1.o**] by multiplying by b rather than $\log b$, the formula on Wikipedia is actually measuring something entirely different, something like the complexity of a combination lock or tally counter – which, granted, really is minimized in base 3. but we're looking at *information*, and as we've already seen, this corresponds not to the base itself, but its logarithm. this would treat niftimal digits as *six* times as heavy as seximal digits, even though we know a niftimal digit is worth *two* seximal digits.

chapter two

[**2.a**] an excellent introduction to finger binary by 3blue1brown:
https://youtu.be/1SMmc9gQmHQ

[**2.b**] for example, in this base 6 proposal:
http://shacktoms.org/base-six/base-six.htm

[**2.c**] strictly speaking, chisanbop is about pressing fingers down on a table, not extending or retracting them to or from a fist. but it can easily be reimagined this way.

[**2.d**] or, actually, by this classification, the "traditional decimal" counting method is... actually undecimal? it lets you count eleven different numbers, 0 to 10 inclusive, the eleven possible values of an undecimal digit. guess we can be glad that inclusive counting shielded our ancestors from adapting base *eleven*.

[**2.e**] aside from just being a race about who can count highest, these differences have more practical applications. for example, in seximal you can do any one-digit addition with just finger counting. meanwhile, in binary, if you're so inclined, you can do *five-bit* addition by performing bitwise operations on your fingers. it's extremely nerdy and really cool once you get the hang of it.

[**2.f**] the proposal by Shack actually comes up with a seximal workaround: having the thumb touch the remaining four fingers in turn to represent 32, 33, 34, and 35, thus making each hand a niftimal digit. but this destroys the ability to quantize the counting into smaller digits than entire hands (base six *squared*!), which themselves are now a weird amalgamation of binary and some leftovers – and to what end? it only pushes the upper limit by an additional 25%, compared to nearly tripling from seximal to chisanbop and increasing by over ten times from chisanbop to binary.

[**2.g**] moreover, the regularity and repetition of binary counting makes the motions for finger binary extremely easy to pick up on. thanks to this, it's easily possible to count in binary automatically, not thinking about the intermediate steps and instead only interpreting the final result – for instance to count words in a sentence – thus reaping its rewards well before you've familiarized yourself with the ins and outs of binary mental math.

[**2.h**] check out for instance any of these:
https://youtu.be/zELAfmp3fXY (wooden panels)
https://youtu.be/4yBGbozevqs (marble machine)
https://youtu.be/--LolAtwecE (toy train track)
https://youtu.be/pXN1TH0OHT8 (liquids)
https://youtu.be/B3RC8UAd1Mo (Cell Machine)
https://youtu.be/KEhsMZlJFyU (Baba Is You)
https://youtu.be/iIaFcJYtluw (Minecraft; only sticky pistons and observers)

[**2.i**] Nystrom's quote:
https://books.google.com/books?id=aNYGAAAAYAAJ

chapter three

[**3.a**] though even naively, it's a tradeoff between linear complexity of adding each pair of digits and logarithmic complexity of number lengths – so intuitively smaller bases should have an advantage.

[**3.b**] for more details: all you need to do to add two binary digits is to write a 1 if they're different, and carry a 1 if both are 1 (XOR for the answer, AND for the carry). you can also carry all the way across a large streak of 1s, instead of each one individually, which massively speeds things up. (the same thing also happens in decimal, only with nines, and a lot less often.)

[**3.c**] in fact, the expected number of carries per digit is the lowest in binary, at 1/4, while it approaches 1/2 as the base increases.

[**3.d**] Tab's base 6 plea:
https://xanthir.com/b4y30

[**3.e**] as opposed to the values of the carry in addition, which are just... um... 0 and 1...

[**3.f**] natural number multiplication is commutative:
https://math.stackexchange.com/questions/34131/commutativity-of-multiplication-in-mathbbn

[**3.g**] *a better way to count* goes one extra step and emphasizes that only four of seximal's 36 (obfuscated ten) entries "sound" unfamiliar, even though what matters for the algorithm

is their written forms. Shack's proposal even dismisses *every single entry* as "trivial" except for a single "$4 \times 4 = 24$".

[**3.h**] the binary multiplication table is exactly just AND. in fact, the digits 0 and 1 together with the operations XOR as addition and AND as multiplication form the smallest possible finite field.

[**3.i**] this is known variously as "ancient Egyptian" or "Russian" multiplication. the very *existence* of this algorithm implies that instead of multiplying in decimal, it's considered easier to convert to binary, multiply there, and then convert back:
https://en.wikipedia.org/wiki/Ancient_Egyptian_multiplication

[**3.j**] specifically, if a binary number n consists of multiple copies of a smaller pattern x, then it's equal to x times the pattern of starting offsets of x in n. this is similar to how in decimal, you can immediately tell that $777,777 = 7007 \times 111$, but it's a lot simpler and more common with binary since 0 and 1 are the only digits. this is essentially reversing the multiplication algorithm in those cases where the subsequent addition step doesn't involve any carrying – which is often the case.

[**3.k**] since every base makes multiplying by its own powers trivial, and the powers of two are the most dense.

[**3.l**] Misali's stance on long division:
https://seximal.net/math

[**3.m**] as a bonus, since you don't even have to multiply in this algorithm to find the remainders at each step, you can skip the whole part of writing down the answer digit by digit, and just find remainders one after the other to very quickly calculate the modulo operation, sometimes called "the fifth arithmetic operation". of course, dividing and modulo by powers of two is far easier still, requiring you to just chop off a few of the rightmost bits. (you could in theory use this to easily perform a general divisibility test by any arbitrary number, but that's not actually necessary; anyway, there's a *lot* more coming about *that* topic later.)

[**3.n**] if the input to this algorithm isn't a perfect square, the algorithm never terminates, and you can do it for as long as you want to calculate the square root to any arbitrary precision.

[**3.o**] this algorithm is adapted from the Wikipedia article *Methods of computing square roots*. it works, though it has no citations, and is listed as "possibly original research":
https://en.wikipedia.org/wiki/Methods_of_computing_square_roots

chapter four

[**4.a**] by the way, those last two can be done recursively, always reducing the number of possible outcomes all the way down to a fixed set of base cases.

[**4.b**] these work because the powers of ten are all congruent to 1 modulo 9, and they're congruent to the corresponding powers of -1 modulo 11.

[**4.c**] actually, we can know even more: it will always divide $b^n - 1$ for some $n < b$, and if the number for which we're trying to find the divisibility test is prime, then the smallest such n will always be a factor of $b - 1$. and numbers one *more* than powers of the base are relevant here because of the identity $(b + 1) \times (b - 1) = b^2 - 1$, so if the smallest such

n is even, then the number will also divide $b^{n/2}+1$. this is sufficient to give divisibility tests for everything, because any number can be split into the part with prime factors it has in common with the base (which will divide b^n for some n) and the part coprime to the base (which will divide $b^n \pm 1$ for some, possibly different, n). the most "atomic" divisibility tests are for every *prime power*: from them, you can assemble a divisibility test for anything else, but you can't break them down any further.

[**4.d**] there are easier decimal divisibility tests for 7, such as treating the decimal number as if it were a base-3 number (e.g. $231 = 1+(3\times10)+(2\times10^2)$ is divisible by 7 if and only if $1+(3\times3)+(2\times3^2)$ is). also, for every number there are the very popular divisibility tests of the type "multiply the leftmost digit by some constant, then add the remaining digits" – which, reducing the number's size by one digit with each iteration, take roughly as many iterations as the number has digits. but there are none that are as easy as the three types we've looked at, which essentially reduce a number to its *logarithm* with each iteration.

[**4.e**] Misali's tier list argument about factors:
https://seximal.net/factors

[**4.f**] specifically: in dozenal, doing the divisibility test for 5 recursively will eventually reduce to a base case that is still some multiple of 5 up to 144, so you need to know them by heart. in binary, on the other hand, it will recursively reduce to... some multiple of 5 up to 4 – in other words, always to 0. usually, we associate cyclic numbers with numbers adjacent to a power of the base that end up being huge multiples, like how 7 in decimal is first reachable at 1001, which is 143×7 – which makes it so you need to memorize 143 multiples of 7 to use the divisibility test by 1001. meanwhile, binary has the unique situation where, on the one hand, it has plenty of cyclic numbers near the start – 3, 5, and 9 – but on the other hand, all of those are already adjacent to a power of two, so in a way they almost aren't really cyclic in the same way at all.

[**4.g**] take a moment to verify that those are actually just special cases of the divisibility tests succeeding. for example, if a number consists of multiple disjoint copies of the bit pattern "11", then it's divisible by 3; notice how such a number would be guaranteed to make the "alternating sum of single bits" test return 0.

[**4.h**] Leibniz's quote:
http://leibniz-translations.com/prime.htm

[**4.i**] even though seximal might appear superior to all other bases here, there actually isn't any metric with which you can quantify seximal being better than every other base in this regard. primes can only end in 1/3 of all possible digits in seximal, but only 4/15 in base thirty; only two possible digits in seximal, but only one in binary. also, the first number that "looks prime but isn't" would be 11×13 in both binary and seximal if you disallow perfect squares. speaking of detecting perfect squares...

[**4.j**] seximal squares:
https://reddit.com/r/Seximal/comments/top1kk/perfect_squares_only_end_in_13_or_4/

[**4.k**] because in any base b, for any n, n^2 and $(b-n)^2$ are congruent modulo b.

[**4.l**] in other words, every odd perfect square must be congruent to 1 modulo 8 (which, by the way, doesn't work in octal, since the "even number of zeros" in binary can throw it completely off the rigid octal grid). now, every *throdd* perfect square must also be congruent to 1 modulo 3, which overall means that every perfect square coprime to 6 (ending in 1 or 5 in seximal) must be congruent to 1 modulo 24. you *could* consider *this*

as the seximal equivalent of the above, but testing divisibility by 24 requires you to look at the last *three* seximal digits, which can have 9 options (this problem is due to seximal lumping two and three into a single power-of-six component; see more about it below), and this doesn't even exclude perfect squares not coprime to 6, which can't be detected using this method at all.

[**4.m**] seximal has every number divided into a power of six component and a... um... "primes higher than 3, plus at most either a power of two or a power of three, but not both" component... yeah, not that useful.

[**4.n**] for binary, the power of two and odd components can be treated as entirely separate "coordinates" of a vector, which multiply separately: $A = 2^a \times \mathsf{odd}_A$ and $B = 2^b \times \mathsf{odd}_B$ always multiply to $A \times B = 2^{a+b} \times (\mathsf{odd}_A \times \mathsf{odd}_B)$. this is not the case in seximal, where for instance $2 = 6^0 \times 2$ multiplies with $3 = 6^0 \times 3$ not to $6 = 6^0 \times 6$, but to $6 = 6^1 \times 1$. an immediate implication of this "orthogonality" in binary is that, as mentioned, the trailing zeros of a product are equal to the sum of trailing zeros in the multiplicands – which is not true in seximal. in number theory terms, this boils down to the fact that $\mathbb{Z}_2$ is a field, while $\mathbb{Z}_6$ is not.

[**4.o**] you can also alternatively use negative numbers in the magic sequence, for example -1 and -2 instead of 10 and 9 in the magic sequence for 11. this can greatly decrease the sizes of the sums involved, by causing many things to cancel out. (you can actually change any number to any other as long as they're congruent to each other modulo 11. in fact this is how these tests work – by swapping out every power of two for a much smaller number congruent to it modulo 11.)

[**4.p**] even more is true: if the result is congruent to any x modulo n, the original number is also congruent to x modulo n. so in particular, the tests are bijective: they're necessary and sufficient conditions for divisibility.

[**4.q**] for example, the magic sequence for 7 is $[1, 2, 4, 1 \ldots]$, which obviously corresponds to the sum of triplets of bits. and one way of writing the magic sequence for 5 is $[1, 2, -1, -2, 1 \ldots]$, which corresponds to the alternating sum of pairs of bits. and the magic sequence for 2 is $[1, 0, 0 \ldots]$, showing that divisibility by two depends only on the least significant bit and is unaffected by all the others. binary is simple enough where these simple divisibility tests we considered at the beginning can be extended to their most general form and applied to any number whatsoever.

[**4.r**] this website actually talks about a *decimal* magic sequence to test divisibility for 7: https://math.hmc.edu/funfacts/divisibility-by-seven/

chapter five

[**5.a**] the converse is true as well: the rational numbers are precisely those numbers which are terminating or recurring in some (integer) base, or equivalently, all bases.

[**5.b**] the value of this maximum is given by the Carmichael function. this function, at x, is equal to the smallest n for which, for any b coprime to x, $b^n - 1$ is guaranteed to be divisible by x. so "x is cyclic in base b" means that this maximum n is actually attained at b. for a concrete example, the Carmichael function of 11 is 10; so 11 is cyclic in seximal, because 11 divides $6^{10} - 1$ and does not divide any $6^n - 1$ with $n < 10$; but 11 is not cyclic in decimal, because although 11 does divide $10^{10} - 1$, it also divides $10^2 - 1$. the Carmichael function of any prime p is equal to $p - 1$, hence 4 digits for 1/5 in binary and dozenal, or 6 digits for 1/7 in decimal and dozenal.

[**5.c**] the values of the Carmichael function for 3, 5, 7, 9, and 11 are 2, 4, 6, 6, and 10 respectively. the values for 3, 5, 9, and 11 match the period lengths in binary.

[**5.d**] this is because the Carmichael function always takes on even values for all inputs greater than 2.

[**5.e**] cyclic numbers in various bases (note that Wikipedia calls them "full reptends", and "cyclic numbers" their multiplicative complements to $b^n - 1$):
https://en.wikipedia.org/wiki/Full_reptend_prime

[**5.f**] this deceptive position of square bases when viewed through the narrow fraction lens of base comparison possibly accounts for more errors in *seximal responses* than any other detail. nonary is said to be the "most usable odd base", because it only has one red ratio – namely 1/2, because its only cyclic number is 2, because it's a square. and quaternary and hex, but not octal, are said to be "great at compressing binary" because they have no red ratios at all, because they're squares – as opposed to octal, which does have red ratios, because it's not a square. (you know what base is *even better* at compressing binary? that's right, *binary*!)

[**5.g**] again, logarithm to what base? like in chapter one, we need to make an arbitrary choice, and this time, it won't cancel out nicely in the end. it still doesn't matter, though, because we'll only be comparing if one score is bigger than another, and if so, then how many times – both of those remain constant regardless of which base greater than 1 we use for the logarithm. when you see "log" in this chapter, it refers to this kind of base-agnostic logarithm – and "exp" to its inverse function.

[**5.h**] beyond 1/9, the next number that seximal does better by this metric is 1/18, and then 1/25. and in general, by looking at numbers up to one thousand, it seems neither base eventually wins out.

[**5.i**] if a higher base b attains for some n the maximum period length m, then its score will be $m \log b$. but then the number of digits in *any* base to write n will be at most m, so binary's score will be at most $m \log 2$, which is guaranteed to be less. so since every non-square base will have infinitely many cyclic numbers, for every such base greater than 2, there will be infinitely many cases where binary is more efficient.

[**5.j**] the proportion of denominators that result in terminating expansions converges to 0, and the proportion of those that result in recurring expansions converges to 1.

[**5.k**] the digits after the recurring point always come from the odd part, and the power of two part gives just a few leading zeros – in bases with multiple distinct prime factors, like seximal, this is far more complicated.

[**5.l**] the balanced dozenal website ("Janus numbers"):
https://musa.bet/reverse.htm

[**5.m**] remember how base 4 made things worse for binary fractions with odd periods? well, it causes problems here too, once again. while in binary, it's clear that 1/11 consists of two halves which are bitwise negations of each other, this insight goes missing in base 4, where 1/11 is r01131. so in this case, the 5-digit representation in base 4 isn't even *equally* good, but actually *worse* than the 10-digit representation in binary.

[**5.n**] Midy's theorem:
https://en.wikipedia.org/wiki/Midy%27s_theorem

[**5.o**] in fact, we could do what that balanced dozenal notation did, and introduce a special shorthand for a period that repeats, but negating its bits every time. if we ignore the question of whether it's just as easy to read in seximal – where it stands for taking the fives' complement – then updating our original metric with this shorthand makes binary perform *even better* than before compared to seximal.

[**5.p**] as with divisibility tests, even though this is most useful for cyclic numbers, it works with any number. decreasing steps correspond to ones, and increasing or constant steps correspond to zeros. this also means the maximum possible period length of a number's magic sequence (in any base, not just binary) is given by that number's Carmichael function – and that, just like fractional expansions, magic sequences for cyclic numbers have complementary halves.

[**5.q**] you can even start the magic sequence at any number s instead of 1, and the same procedure will give you the binary expansion of s/n.

[**5.r**] hang on, isn't that seximal magic sequence for 11 just the reverse of the binary one? yes, but this is true only for 11 and no other number; it's because coincidentally 2 and 6 happen to be multiplicative inverses of each other in the field $\mathbb{Z}_{11}$... but 2×6 does not equal 1 in any other field.

chapter six

[**6.a**] indeed, the "digit-power" approach can be seen as a lossless compression algorithm, relative to the "digit-by-digit" approach; the pigeonhole principle implies that any such algorithms must make some inputs longer than their uncompressed variants. their, um, "neverthelessy" usefulness comes from the fact that the more common inputs – such as numbers with many zeros – are made shorter. "0" digits are dropped entirely, while "1" digits, except for the least significant one, omit the word "one" and just state the power. binary, of course, has only those two options.

[**6.b**] 10^4 is technically called "myriad", but that word's original meaning has essentially disappeared entirely (though the SI actually used to have "myria-" and "myrio-" prefixes). meanwhile, in some regions of India, 10^5 is called "lakh".

[**6.c**] Chinese long scale:
https://en.wikipedia.org/wiki/Chinese_numerals#Large_numbers

[**6.d**] the -yllion system:
https://en.wikipedia.org/wiki/-yllion

[**6.e**] if we name every power of each base, the advantage of the higher base at n simplifies as:

$$\log_{b_y}(n) - \log_{b_x}(n) = \frac{\log n}{\log b_y} - \frac{\log n}{\log b_x} = \left(\frac{1}{\log b_y} - \frac{1}{\log b_x}\right) \times \log n$$

which is a constant times $\log n$, meaning it'll grow unbounded as n increases, and will always outweigh the constant disadvantage of more digit names for a large enough value of n. with only power-of-two powers of each base, however, now the advantage of the higher base at n simplifies as follows:

$$\log_2 \log_{b_y}(n) - \log_2 \log_{b_x}(n) = \log_2 \left(\frac{\log_{b_y}(n)}{\log_{b_x}(n)} \right) = \log_2 \left(\frac{\log b_x}{\log b_y} \right) = \log_2 \log b_x - \log_2 \log b_y$$

an expression independent of n; a constant advantage, which is much smaller than the constant disadvantage for the digit names.

[**6.f**] originally, the proposed name for 2^4 was "nybble", continuing the convention of naming double powers of two after units of information (or C-like integer data types) which can have that number of possible states. it was changed to "hex", from the name of the base, to increase recognizability, with an added benefit of shortening it to one syllable.

[**6.g**] using the number names up to and including *byteplex* for 2^{256}, the highest number that can be represented is $2^{512} - 1$, which is a number with 155 decimal digits. meanwhile, the number of particles in the observable universe is estimated to be around 10^{80} – that's roughly two byte byteplex.

[**6.h**] recursion: see footnote 6.h.

[**6.i**] previously, when we looked at digit strings, using base 2^n over base 2 always entailed at least two problems: rounding the number of digits up to a multiple of n, and requiring a grouping by n-tuples when it isn't always the most useful. in this case, neither problem arises. the substitution between *two one* and *three* is a simple, mechanical, reversible find-and-replace that doesn't break any of the system's neatness, and only makes many number names shorter, at the cost of introducing only one new word; apart from simplicity, there's little reason not to use it. (if anything, this case shows how lethal it can be to get trapped in one mode of thought when arguing for or against different bases, or anything for that matter.) one could go further and extend it to hex – octal won't work, because 8 isn't a *double* power of two – but that requires *eleven* new words, including names for over-decimal digits, which is far more awkward, and probably not worth it.

[**6.j**] do note, however, that this substitution isn't as compatible with the system as using *three* for *two one*. in practice, irregular power of two names like this should probably only be used in isolation, kind of like *dozen* is an "irregular" name for twelve, or *forty two* is an "irregular" name for *the answer to the ultimate question of life, the universe, and everything* (lojban moment).

[**6.k**] jokes aside, *perstack* works well as the equivalent of percentages in binary, because many common simple fractions can be expanded to a denominator of 63, 64, or 65. this works so well because none of them are prime, which – ironically – is because 64 is the *sixth* power of two, and six is the smallest number which is neither prime (thus preventing a Mersenne prime) nor a power of two (preventing a Fermat prime).

chapter seven

[**7.a**] tally marks are based on the principle that you count only by drawing additional strokes. this doesn't work with normal binary counting, which requires you to erase strokes. instead, binary tally marks work using a zigzag-like fractal pattern. imagine you have a vertical binary number, with the least significant bit on the bottom. to count 1, draw a tiny line segment that ends on that imaginary bottom bit. to count from n to $n+1$, nudge your imaginary binary number to the right, then draw a line from where you currently are to the most significant bit that has flipped. here are some examples, where a digit represents the height of a local peak: 0, 01, 010, 0102, 01020, 010201, 0102010, 01020103

(https://oeis.org/A007814). thus every next line will alternate between ascending and descending, and counting up to higher powers of two makes local peaks that are higher and higher. every odd number ends up on the bottom, while every even number ends up on an unresolved peak. the pattern for counting to one power of two looks like a scaled-up version of the pattern for the previous power of two, making this pattern a fractal. all lines are connected, making this system of tally marks maximally efficient, as you never have to lift your pen. to read a binary number from this system, first find the largest power of two and write a 1 for it, then for every smaller power of two, if its rightmost instance is to the right of the previous power of two's rightmost instance, its bit is a 1; if to the left, it's a 0.

[**7.b**] balanced ternary, with digits whose absolute value doesn't exceed 1, works in many ways like binary, and shares many advantages, such as simple arithmetic, good radix economy, and possibly even better finger counting if you're that dexterous. sadly not *all* the advantages carry over, such as ultra-dense powers of the base letting you work at lots of different scales or have many factors be adjacent to a power of the base. and honestly, it being an odd base is kind of a death sentence, as it makes 2 need a digit sum test. still, it's probably the second best base after binary – and *definitely* the most usable odd base.

[**7.c**] two's complement notation is a great way to write negative numbers with lots of really neat properties. it arises naturally from the "$1 + 2 + 4 + 8 + \ldots = -1$" pseudo-equality (let $1 + 2 + 4 + 8 + \ldots = S$, then S has the property that $2 \times S = S - 1$, so $S = -1$), and extending it to the negative powers of two gives $\ldots 111.111 \ldots = 0$, thus $\sim x = -x$ (the formula "$\sim x = -x - 1$" in computing comes from ignoring the fractional part .000... turning into .111...). two's complement preserves modular arithmetic: -7 is ...111001, indicating that it's congruent to 1 modulo 8 (last three bits are 001). technically n's complement works in all bases, but the leftmost recurring digit is always either ...000 or ...ZZZ, so binary is the only base that uses it "to its full potential".

[**7.d**] we can compare a hypothetical binary system of money with a decimal and seximal one by looking at minimal sets of nominals that can be used to represent any possible value. the decimal one is inefficient because the nominals 1, 2, 5, and 10 are insufficient to represent all values from 1 to 20; you either need to double the 2 or double the 1. the seximal one is also inefficient for the opposite reason: to represent all possible values from 1 to 12, you can use the values 1, 2, 3, and 6, but then the 3 is redundant, already being the sum of 1 and 2. the results of these inefficiencies are also transparently visible in the value of the largest nominal in a "complete" set of eight: for binary, it's 128; for decimal, 50; for seximal, 72. (in addition, there's a mental math advantage; every nominal is just a power of two, so to make a transaction, you just spell out a number's bits. in contrast, something like $5 + 2 + 1 = 8$ is in no way made easier by using decimal.)

[**7.e**] dividing the day into groups of 16 four times gives a binary second only slightly longer than the traditional sexagesimal second (1/65536 instead of 1/86400 of a day). the in-between time units are also surprisingly useful: the "binary hour" and half of it are typical durations of lectures and lessons respectively; the "binary maxime", around 5 traditional minutes, is the "square root of a day" (there are as many maximes in a day as seconds in a maxime, namely 256) and is a typical small amount of time to ask someone to wait; the "binary minute", or 16 binary seconds, is the typical amount of time you can expect someone's attention span to last.

[**7.f**] in music, a notehead can be followed by a dot to increase the note's duration by half, in other words, multiply it by 3/2. that can be followed by another dot to increase it by another quarter, so overall multiplying by 7/4, and another dot to increase it by another eighth, multiplying by 15/8, and so on. these dots can be interpreted as spelling

out the binary fractions 1.1, 1.11, and 1.111, for the numbers 3/2, 7/4, and 15/8. by introducing a hollow dot for 0, you can create all sorts of binary fractions, like 5/4, 13/8, and 9/8, which can otherwise only be represented by adding note values using tie bars – and without making it any harder to sight-read, either, as you can just imagine the hollow dots as being placeholders for filled dots that have been omitted.

[**7.g**] you can also prove that the exponents of Mersenne primes must themselves be prime more directly with number theory, without using binary representations: the differences of two powers $(a^n - b^n)$ can always be factorized as $(a - b)$ times something else; if the exponent is composite, say, $2^6 - 1$, we can represent it as $(2^3)^2 - 1^2$ and thus extract a factor of $2^3 - 1$. the only way to prevent such a factorization is if the exponent is prime – and the base is two.

[**7.h**] this too can be proved directly with a number-theoretic method. the *sums* of two powers $(a^n + b^n)$, unlike differences, can only be factorized as $(a + b)$ times something else if the power is *odd* (this has to do with how -1 is equal to all of its own odd powers, but not its even powers). so if the exponent has a nontrivial odd part, such as $2^6 + 1$, then it's possible to represent the number as a sum of two odd powers and factorize it: $(2^2)^3 + 1^3$ has a factor of $2^2 + 1$. this is impossible only if the exponent has no odd part – in other words, it must be a power of two.

[**7.i**] for just a few examples: they're the "additive primes"; they're the sums of all entries in the rows of Pascal's triangle; they're precisely the numbers that cannot be written as sums of consecutive positive integers ("polite numbers"); they're the lexicographically earliest sequence of which no subset forms an arithmetic progression; they're a sequence whose first differences are the sequence itself; they're the number of elements in power sets, and are so crucial to the concept of power sets that one notation for the power set of S is literally "2^S".

[**7.j**] by the way, if you're an enthusiast of memorizing the circle constant's digits, then in binary you don't have to pick a side on the pi vs tau debate, since it amounts to just a bit shift (https://tauday.com/tau-manifesto). in fact, Bob Palais wrote in *Pi Is Wrong* (http://math.utah.edu/%7Epalais/pi.pdf):

> *What really worries me is that the first thing we broadcast to the cosmos to demonstrate our "intelligence," is* 3.14... *I am a bit concerned about what the lifeforms who receive it will do after they stop laughing at creatures who must rarely question orthodoxy. Since it is transmitted in binary, we can hope that they overlook what becomes merely a bit shift!*

[**7.k**] for those interested, Mathologer's video on "Newton's calculus" is excellent:
https://youtu.be/4AuV93LOPcE

[**7.l**] this game is, of course, none other than "2048".

[**7.m**] seximal powers of two (of course, you could *instead* switch to a base where you *don't* need to memorize the most important integer sequence...)
https://seximal.net/math

[**7.n**] the dual number – no, wait, *this* one:
https://en.wikipedia.org/wiki/Dual_number
https://en.wikipedia.org/wiki/Dual_(grammatical_number)

[**7.o**] "but wait, aren't there twelve notes?" glad you asked! yes, Western music does indeed use twelve notes per octave, but this has nothing to do with 12 being a highly composite

number. rather, if we must reduce it to one reason, it's because $\log_2(3)$ is much better approximated by a fraction with denominator 12 (namely 19/12) than any other small denominator; in number theory lingo, 12 is the denominator of a *convergent* to $\log_2(3)$. if anything, it's highly *unusual* for a highly composite number to show up here – most other numbers of notes per octave famous for "sounding good" are primes or semiprimes (since the numerator and denominator of these convergents must be coprime). if you're curious, *please* do read up more on this question because this answer doesn't even begin to do it justice. (in contrast to the prevalence of twelve notes in Western music specifically and few other cultures, binary octave equivalence seems to be far more widespread across musical traditions – maybe because the ranges of people and instruments fail to overlap, so there's the need for some interval of equivalence; and $2:1$ is the most natural choice, being the simplest integer ratio after $1:1$.)

[**7.p**] in decimal – or seximal, for that matter – there's a very clear dividing line between math where the powers of the base appear, which is always "recreational", and "serious" math, which always lacks them. we recognize that our base is really nothing more than an arbitrary convention, and "objective" math is just entirely devoid of these conventions. in contrast, powers of two permeate math in all its aspects, and let us reconcile what math itself finds important with what we find important. so if anything else other than numerological mysticism surrounding the number two, *this* is what it means for base two to be natural.

transcripts

[**2:46**] Misali and their fanbase are jokingly represented here as a seximal organization. its logo is depicted.
the logo's shape is a bright yellow circle. in the center is a yellow hexagon with the corners numbered 0, 1, 2, 3, 4, 5. inside the hexagon is a small hexagonal grid of 7 cells resembling a honeycomb, and overlaid on top of that grid is a simplistic geometric representation of a bee. around the circumference of the circle, there is black text in Comic Sans in all caps: on top, it says `SEXIMAL.` and on the bottom, `012345 MAL BEST MAL.`
this logo is a joke on many levels:

- its visual layout is a reference to the logo of Volapük, which is also a circle with text on the top and bottom, saying `VOLAPÜK.` and `MENEFE BAL PÜKI BAL.` respectively. the Seximal logo has the same number of letters.

- the enumeration of seximal digits is a reference to the thumbnail of *a better way to count*, which enumerates the twelve dozenal digits in a diagram listing the twelve chromatic notes.

- the "mal" comes from the ending of "seximal", but it also refers to the prefix *mal-* in Misali's base-naming system, which uses it to mean "times 17". this, in turn, comes from the prefix *mal-* in Esperanto, which confusingly means "opposite" despite looking like it means "bad", and the fact that 17 is a bad choice of base.

[**3:43**] the binary representation of the number 37 is interpreted as a decomposition of 37 into "additive primes". the set of "additive primes" under 37 is 1, 2, 4, 8, 16, and 32; and the "prime summandization" of 37 is $32 + 4 + 1$. 37, then, can be represented by starting from the largest additive prime not greater than it, and descending all the way to 1, marking the presence and absence of each additive prime in the target number. in this way, 37 is represented as "32, no 16, no 8, 4, no 2, 1", which becomes "yes, no, no, yes, no, yes", or 100101.

[**3:50**] this same binary representation of 37 can also be interpreted another way: as a game of *twenty questions* attempting to guess its position on the number line. since zero questions are sufficient to guess a number from a range of one, and each question has the maximum potential to double the guessing space, we can surpass 37 with a minimum of six questions, which puts us in a range of $[0, 64)$. then each question asks if the target number is within the higher half of the current range:

- $37 \geq 32$: range is now $[32, 64)$;

- $37 < 48$: range is now $[32, 48)$;

- $37 < 40$: range is now $[32, 40)$;

- $37 \geq 36$: range is now $[36, 40)$;

- $37 < 38$: range is now $[36, 38)$;

- $37 \geq 37$: we guessed 37.

as before, the answers to the questions if 37 is greater than or equal to whatever number spell out "yes, no, no, yes, no, yes", which is 100101: the same representation as above.

[**8:43**] description of proposed binary digits: both are vertical bars. their bottom ends are aligned with each other, and 0 is a lot shorter than 1. groups of bars can be joined by connecting their bottom ends with a horizontal line.

[**10:35**] consider the number one hundred. in binary, it's written 1100100. grouping by threes, it's 1,100,100. in base 8, it's 144, and translating those digits into triplets of binary digits, it's 001,100,100. this is the same as the binary from earlier, except it has two redundant zeros at the beginning.

[**18:08**] description of binary finger counting: each finger of one hand represents a bit. the bit's value is 1 when the finger is extended, and 0 if it's curled. the least significant bit is the thumb, representing 1, all the way to the pinky, representing sixteen. so a clenched fist is 0, then a fist with the thumb extended is 1, then curling the thumb back in and extending the index finger gives 2, and so on, up to 31 with all five fingers extended. you can use your other hand for five more bits to count up to 2^{10}.

[**20:32**] if someone shows you a seximal number with their hands that consists of two fingers in one hand and three in the other, you can't tell if they aligned the digits left to right from your perspective or theirs – if the number is seximal 23 or seximal 32. even if you establish a convention about which of the two perspectives it should be, it's very easy to make a mistake.

[**25:09**] some binary sums of points from a Scrabble game are shown, annotated to explain how they were calculated. in most cases, the numbers can be mentally split into segments with no carrying, where the addition is just bitwise OR; segments where the bits of both numbers are identical, so the sum just shifts them all to the left by one; and segments where a single 1 is added to a series of 1s, equivalent to adding a 1 to a series of nines in decimal. sometimes, the common "idiom" $5 + 3 = 8$ appears, and occasionally, the easiest strategy really is to add in a power-of-two base – for instance, when one of the summands is one less than a power of two.

[**27:49**] the binary number 11011 is depicted. the pattern 11, which represents 3, occurs twice. the first time, it starts at position 0 from the right; the second time, at position 3 from the right. a binary number that contains "1" bits at exactly those positions is 1001, which means 9. so the original number, 11011, must be equal to 3 times 9, which is 27.

[**30:12**] the example for how to take a binary square root is the binary number 10101001 (split into pairs: 10,10,10,01). we start by writing 1 above the first pair, 10, and subtracting 1 from it to get $10 - 1 = 1$. joining on the next pair gives 110, our current working value. the answer we know so far is 1, and joining on the digits 01 gives 101. we test if the working value, 110, is greater or equal – which it is. so the next pair will get 1 as its digit, and we'll subtract the 101 from our working value, getting $110 - 101 = 1$. joining on the next pair gives 110. the answer we know so far is 11; joining on 01 gives 1101. this time the working value is less, so the next digit is 0 and we don't subtract anything from the working value. finally, we join on the final pair to get 11001, and the answer we know so far is 110; joining on 01 gives 11001. if we test whether the working value is greater or equal, we find that it is; in fact it's exactly equal, so the final digit of the answer is 1, and subtracting gives a remainder of zero, so the algorithm stops. thus our answer for the square root of 10101001 is 1101, which checks out: in decimal, the square root of 169 is 13.

[**34:27**] the background turns purple and the following title appears on the screen: "What's the Most Commonly Used Prime Which Is Incompatible with This Particular Base?™". this is a joke that refers to Misali's *Conlang Critic* series. in videos about international

auxiliary languages, there is a recurring gameshow-like segment with a similar ridiculously long name, where the language's consonant inventory is tested for compatibility with the most commonly spoken languages.

[**40:45**] this demonstration involves the binary number 1000010, which is 66. writing the magic sequence underneath the bits, from right to left, puts the number 2 underneath the first 1, and the number 9 underneath the second 1. adding them up, $2 + 9 = 11$, which verifies that the original number is divisible by 11.

[**43:40**] description of binary fraction symbols: the radix point is a vertical bar of about the same length as the digit 0, but extending below the baseline instead of above. the recurring point is like the radix point, but with an additional horizontal line attached to the bottom end and extending towards the right. the length of this line doesn't matter; in particular, it doesn't necessarily extend over the entire period of the fraction.

[**48:07**] a screenshot of seximal.net appears, showing the page that contains Misali's brief summaries of other bases. there are also comments in magenta added into the screenshot. the bases featured are all the powers of two on the site:

- **binary**: bases don't get much smaller than this, so it's really bad when it comes to compactness. – `:(`
- **quaternary**: four is a highly composite number, and it's right between two primes, so it's really almost as good as seximal, just a bit smaller. – `c:`
- **octal**: sometimes used for binary compression, but it isn't really all that good at that. like, quaternary is better at compressing binary even though it's a smaller base. – `who woulda guessed`
- **hex**: everyone's favorite way to compress binary, and for good reason! – `;)`
- **tetroctal**: another power of two, and this one is the WORST ONE!! – `>:(`
- **octoctal**: yet *another* power of two base. this one is used for youtube URLs and stuff. it's not super practical as a numbering system, but as a way of representing binary numbers in a way that's human readable while not taking up too much space it's pretty good. – `youtube transfers audio data as base64`

[**48:10**] a screenshot of BASE OFF is shown, where a user has eliminated every base in the range of two to ten except for two and four. now, no matter how the yes/no questions are answered, the only possible outcomes are to end with base four or be stuck forever.

[**48:15**] the image from the Artifexian video is a table of fractions from 1/2 to 1/20 and their representations in various power of two bases, from 2 up to 32. the original gives a base one point for a given fraction if no other considered base's representation has fewer digits; so for example all bases get a point for 1/2, since each represents it with one digit, but only base 32 gets a point for 1/11, writing it with two recurring digits, while all the others need either ten or five. base 16 wins with 13 points, and base 2 is the worst with only 1 point. the "corrected" version instead takes into account the relative sizes of the digits; so now only base 2 gets a point for 1/2, and bases 2, 4, and 32 all get a point for 1/11. binary is the new winner, trivially getting a point for every base, and the bigger the base, the fewer points it scores, down to just 1 point for base 32.

[**51:37**] among the first ten positive integers, seven are compatible denominators for seximal. however, the figure is only 20 out of the first hundred; starting at 40, each decade of numbers only contains at most one compatible denominator.

[**52:07**] the comment says the following:

> *Balanced dozenal is like two back-to-back seximal systems, with the best of both worlds: the size and factors of dozenal, but the products and fractions of seximal. Fifths are two digits; sevenths are three. Check out*
> `http://www.musa.bet/reverse.htm`

following the website, it turns out one seventh in balanced dozenal is written as a recurring "$2, -3, -5, -2, 3, 5$", with six recurring digits, of which the second half are the negatives of the first half. the website counts this as three recurring digits – which, if you do that, there's no reason not to count the two complementary halves in any base as half the length.

[**55:56**] most languages don't pronounce 42 as their equivalent of *four two* (represented as "insert words"), but as the equivalent of *four ten two* (represented as "insert more words"). this is a joke about a hypothetical language where those could be literal glosses if "insert" was the word for four, "words" for two, and "more" for ten.

[**1:00:17**] some examples of spoken binary are listed, both before and after abbreviating *two one* to *three*:

number	**name**	
1	one	one
2	two	two
3	two one	three
4	four	four
5	four one	four one
6	four two	four two
7	four two one	four three
8	two four	two four
9	two four one	two four one
10	two four two	two four two
11	two four two one	two four three
12	two one four	three four
13	two one four one	three four one
14	two one four two	three four two
15	two one four two one	three four three

[**1:00:53**] a thumbnail of a Minecraft gameplay video is depicted. in the thumbnail, there is a player wearing full diamond armor in a dungeon and a representation of 64 diamonds in item form. the title of the video is "I've got a stack of diamonds to spend!", indicating that the proposal is to give the number 64 the irregular name "stack".

[**1:07:15**] this is a joke about how looking up "dual number" on Wikipedia brings up an article about mathematics instead of linguistics. the correct article's title is "Dual (grammatical number)".

[**1:07:19**] Toki Pona only has three number words: *wan* meaning one, *tu* meaning two, and *mute* meaning any number greater than two, which is functionally identical to having a singular-dual-plural number distinction. this is contrasted with a proposal of a seximal number system for Toki Pona described on seximal.net, where the word *luka*, which literally means "hand", is confusingly used for 6.

[**1:08:00**] a quote from John Nystrom's book about a binary system is shown on screen, the same one as from chapter two, talking about introducing a decimal system into music:

> *In music, the tonal system is in full operation; the notes are divided, as regards time, into halves, quarters, eighths, etc. A bar of music is generally expressed by quarters or eighths, and a burden has generally 8 or 16 bars.*

[**1:11:48**] right at the very end, right before the "thank you for watching" screen, an image is displayed that humorously summarizes *a better way to count*. ("KönKlwJun" is how the constructed language Vötgil, repeatedly made fun of in Misali's *Conlang Critic* series, would spell "conclusion".)

```
a beder wey 2 count lol xd
- haha look at me i make fun of dozenal
- haha look i give my new numbers funny names
part 1:  six is smol
- excuses why seximal numbers take up more space.
oh and if you don't like that you can use this ebcdic clone i made
part 2:  most people have 6 fingers
- no they don't but who cares?  seximal is better full stop
part 3:  seximal fractions
- haha boppy tune
- we don't talk about elevenths
because both decimal and dozenal do them well
part 4:  KönKlwJun
- seximal is the bestimal because i threw some smart words at you.
now go preach the truth about haha funny sex word base
```