Playfulness and Mental Methods for Divisibility by Small Numbers

by John M. Boyer, PhD, MSPE

Introduction

Play. We all do it. It's fun. Quite a while ago, I came to the conclusion that playfulness is an essential survival skill. Currently, one can easily find supporting academic literature[1] and general-interest literature[2] using web search queries along the lines of "playfulness as a survival skill." Play helps us to sharpen skills and widen the number and variety of scenarios in which we can properly apply those sharpened skills. Playfulness in thinking, then, can result in sharper and more fluid thought processes in real-life scenarios. And there's a name I use for the reward component of the operant conditioning mechanism that biologically reinforces playfulness: It's *fun*.

Throughout everyday life, I like to fill in the thinking blanks by having fun with words and numbers. Although many opportunities for elaborate mental play arise, in this paper I am referring to mental games that are relatively simple and quickly executed. For example, quick word plays are often puns, like thinking *I wonder what's "happy Ning" today?* while "opeNing" the ISPE social media platform. Puns are formed by making small changes at symbolic levels, such as lexical, aural, or grammatical changes, that produce larger effects at the semantic level. They seem to me to be miniature practice runs of the creativity it takes to think laterally of a tactical action that helps achieve an objective one deems important. For a pun, the objective is to get you to laugh or, at least, to enjoy the groan while you're groaning.

My number plays are analogous to puns, except that the small tactical actions typically involve variations of arithmetical operations. I start by picking a number from my surroundings, such as a house address, a phone number, or the concatenation of the digits of distance traversed and calories burned during exercise. Next, I'll pick a single-digit number as an objective, such as a favorite number from childhood. Then, I'll make new numbers from parts of the start number by inserting arithmetical operations, such as adding and multiplying, and by using basic functions, like averaging. I iterate this process and, at each step, choose operations and insertion points that appear to help generate numbers that get closer to, and finally arrive at, the single-digit objective.

Recently, I ran across an unexpected fact that caused me to add divisibility to my repertoire of toys for arriving at the single-digit objective. Let me tell you about it. Why? Because if you're still reading this, then you might find that… it's *fun*.

Simple Divisibility Methods

To set a baseline for computational complexity, we'll start with 1. Is a given number divisible by 1? The answer is, of course, yes. No matter the number, this just mentally returns true. That's OK, though, because I never pick 1 as the objective, anyway.

Slightly more complex is divisibility by 2 and 5, which involves simple conditional logic to select the rightmost digit of the base-10 expression of the number and to perform a set membership test on {0, 2, 4, 6, 8} for divisibility by 2 and on {0, 5} for divisibility by 5. We typically learn this so early in life that it just comes naturally; but the reason it's true is that all natural-number powers of 10 are divisible by 2 and 5, so we can focus on the ones place of a number because we're just getting a whole number of additional twos or fives from higher place-value digits. For example, each hundred is just 20 more fives or 50 more twos.

The case for 4 is only a little more complex. If a number's rightmost two digits form a number that is even when halved, then the number is divisible by 4. The reason this works is an extension of the reasoning that worked for divisibility by 2 and 5. All digits of a number in the hundreds place or higher express values that yield a whole number of additional fours, so we can focus on the value formed by just the tens and ones places of a number.

The last simple single-digit case is 8. Every incremental value of 200 expresses exactly 25 more eights, so we can ignore all digits in the thousands place and higher, and we can form a value with the tens and ones places, plus 100 if the original number's hundreds place contains an odd value. For example, 2968 is divisible by 8 because 168 is 21 more eights than the 350 in 2800.

I've referred to these methods as simple for two reasons. First, they require only a constant amount of mental effort no matter the size of the number being considered. Second, one is convinced of their correctness by an informal, plainspoken explanation. These two reasons do not characterize the remaining single-digit divisibility methods.

Divisibility by 9 and Proof by Mathematical Induction

Many years ago, I ran across the following recreational mathematics practice problem: use mathematical induction to prove that a number is divisible by 9 if, and only if, the sum of its digits is also divisible by 9. For example, the number 3843 is a multiple of 9 because 3+8+4+3=18, and 18 is a multiple of 9. To me, this seemed like an isolated, but still interesting, piece of information worth doing the exercise to prove it to be true by the method of mathematical induction.

In non-mathematical circles, inductive reasoning has a bit of a bad reputation because humans use available data to infer generalizations that may or may not, in fact, be true.[3] A faulty generalization occurs when one jumps to a conclusion that fits available data but may not fit all data about a phenomenon. In the world of artificial intelligence, this same kind of inductive reasoning is performed by methods of machine learning that seek to learn coefficients or weight values that cause the machine-learned model to most accurately fit the available training data. Like a human's faulty generalization, a machine-learned model may very accurately map to available data but then perform poorly when faced with new data about the same phenomenon.[4]

Mathematical induction, on the other hand, takes the extra step of rigor needed to prove that a generalization from initial data is appropriate. Mathematical induction is performed on phenomena that are parameterized by the natural numbers by showing two things:

(1) that the phenomenon is true for some base case, such as when n=1, and

(2) that the phenomenon generalizes to the case n+1 if we assume it is true for case n.

The analogy often used is that of a domino effect. If the phenomenon is true for n=1, and it's true for the successor of n (i.e., for n+1), then it is true for case n=2. Then, if the phenomenon is true for n=2 and for n+1, then it is true for n=3. This logic applies repeatedly without limit, like a sequence of dominoes falling in succession, to inexorably establish the truth of the phenomenon for each higher natural-number value.[5]

Consider this phenomenon: For any positive multiple of 9, the sum of its digits is also a positive multiple of 9. Let's start with the first multiple of 9 (n=1). With no doubt, the sum of its digits is a multiple of 9. Now, with the second multiple of 9 (n=2), we see that 1+8 is also 9; and for n=3, we see that the sum of digits of the third multiple of 9, 2+7, again equals 9. It's tempting to check the next few values of n

and then jump to the conclusion that the phenomenon is true in general. But mathematical induction is more rigorous. We must prove that the phenomenon is true for each and every next value of *n*.

The key to creating a proof by mathematical induction is to *use* the *assumed fact* that the phenomenon is true for a case *n*, without picking a specific value for *n*, to help prove that the phenomenon is true for case *n*+1. So, without picking a specific value for *n*, we can assume that the n^{th} multiple of 9 has digits that sum to a multiple of 9. Now, imagine how the n^{th} multiple of 9 is written. It's a base-10 number that has a ones place, a tens place, a hundreds place, and so on, for as many places as are needed to represent it. Because we didn't pick a specific *n*, we don't know the specific digit values in the n^{th} multiple of 9, but we do know how addition works on base-10 numbers. This enables us to figure out how the sum of the digits will change when we add 9 and do any necessary carry operations to complete the addition.

Let X_n denote the n^{th} multiple of 9. This means $X_n = 9 \times n$ (e.g., X_{427}=3843). Let S_n denote the sum of the digits of X_n (for example, S_{427} = 3+8+4+3=18). Using this notation, X_{n+1} is the $(n+1)^{th}$ multiple of 9, and so it equals X_n+9. Similarly, S_{n+1} denotes the sum of the digits of X_{n+1}, and the question is whether we can use the assumed fact that S_n is a multiple of 9 to prove that S_{n+1} is also a multiple of 9. Enter the rules of addition.

Consider the ones place digit of X_n. It's either a 0, or it is not a 0. If it is a 0, then adding 9 to X_n has the effect of changing the ones place from 0 to 9, which adds 9 to the sum of the digits. Therefore, S_{n+1} is 9 greater than S_n, and so it is a multiple of 9. If, on the other hand, the ones place digit of X_n is not 0, then adding 9 to X_n has a wraparound effect on the ones place that decrements the digit and produces a carry of one to the higher-placed digits of X_n. In the earlier example of 3843, if we add 9, then we get 3852. The ones place was reduced by 1, and we carried one to the tens place, which incremented it. Note that a carry operation may carry the one across several digits if they were originally nines. Those digits change from nines to zeros, so the sum of digits will still be a lesser multiple of 9. For example, with X_{433}=3897, S_{433}=27, X_{434}=3906, and S_{434}=18, the tens place changed from 9 to 0, which reduced the sum of digits by 9. So, according to the rules of addition, when the ones place of X_n is not 0, adding 9 to X_n has the following effects:

1) The ones place of X_n is decremented by 1,

2) A higher digit of X_n is incremented by the carried one, and

3) Any digits between where the carrying starts and ends are reduced from 9 to 0.

Therefore, S_{n+1} is also a multiple of 9 when the ones place digit of X_n is not 0 because it is either equal to S_n or to S_n minus a 9 for each digit that was reduced from 9 to 0.

The same ideas can be applied to prove by induction that any natural number that is not divisible by 9 has a sum of digits that is not divisible by 9. Specifically, the starting case would be one of the digits 1 to 8, none of which sum to 9. Then, if you successively add nines to any of those digits, the resulting numbers continue to have digit sums that are not multiples of 9.

The proof above demonstrates that, qualitatively, more formality is needed to convince someone that the summing-digits method for testing divisibility by 9 is correct. The technique is also more complex because, although the addition is simple for each digit, one must mentally process all the digits of a number to get an answer.

Divisibility by 3: A Game Changer

Because the number base, 10, figured prominently in the proof of correctness for the

method of divisibility by 9, I found it surprising recently when a friend commented that the digit-summing method also works for divisibility by 3. Once you know this, proving it involves an easy adjustment to the proof for divisibility by 9, so I will leave it as an exercise for the reader.

Learning that summing digits could be used to test divisibility by 3 was, however, a game changer, literally. Beforehand, the divisibility-by-9 test had seemed isolated. Afterward, a new goal emerged: add single-digit divisibility testing to my number-play repertoire!

Divisibility by 6 and Proof by Contradiction

Given the divisibility-by-3 method, divisibility by 6 is now easy to solve. A number divisible by 6 must be divisible by both 2 and 3. Hence, one simply determines that the number is even and that its digits sum to a multiple of 3. To exemplify another proof technique, we'll prove this by contradiction. In a proof by contradiction, one initially assumes the negation of a desired statement and then takes logical steps from that assumption until an obvious absurdity results, like the truth and fallacy of the same statement. Working the logic backwards from the absurdity establishes the absurdity of the initially assumed negation and, hence, the truth of the desired statement.[6]

Initially assume that a number, X, exists which is a multiple of 6 but is *not* divisible by both 2 and 3. Since X is a multiple of 6, it can be written as $X = 6K = 2\times3\times K$ for some whole number, K. Since K is a whole number, so are $2K$ and $3K$. Since X divided by 2 gives $3K$ with no remainder, and X divided by 3 gives $2K$ with no remainder, we reach a contradiction of the initial assumption because X cannot be both divisible and not divisible by 2 and 3.

Divisibility by 7 and Creativity by Composition

Upon learning of the easy method for testing divisibility by 3, the question of how to test for divisibility by 7 hit like the smack of a glove on the cheek. A challenge. The only digit left. But this time, the question wasn't how to prove correct a given method for testing divisibility. The question was how to create a provably correct method. Granted, one could just do a web search to look for one, but there's a reason we climb mountains even though others have already climbed them, or play games that others have already played. We do it because it's *fun*.

My first thought about creating a method was consistent with the creative mode of a software architect: I thought about building the new thing out of existing things without changing the internal operations of the existing things. To me, it was clear that the summing-digits method for divisibility by 9 worked because numbers are expressed in base 10. So my first thought was that the same method would work for divisibility by 7 if only the input number were changed to be expressed in base 8, which is called the octal number system.[7] While that may sound a bit far-fetched, it's not as preposterous as a first reaction by those with a computer engineering background who have training in number bases that are powers of two. For example, the numeric part of the "Inchworm" song[8] inflicted on—ahem—*sung* to my daughter in her childhood, typically ended not with "*sixteen* and *sixteen* are *thirty-two*" but with "sixteen million, seven hundred seventy-seven thousand, two hundred *sixteen* and sixteen million, seven hundred seventy-seven thousand, two hundred *sixteen* are thirty three million, five hundred fifty-four thousand, four hundred *thirty-two*."

The typical way that a computer engineer converts a number to octal is to first convert it to binary digits called bits, and then convert groups of 3 bits into octal digits. The decimal number system that we normally use has the ones, tens,

hundreds, and thousands places, corresponding to powers of 10. By comparison, binary has the ones (2^0), twos (2^1), fours (2^2), and eights (2^3) places; and octal has the ones (8^0), eights (8^1), sixty-fours (8^2), and five hundred twelves (8^3) places. In the decimal number system, the digit values are 0 to 9. In binary, the valid bit values are only 0 and 1, and octal digits range from 0 to 7. For an example, the base-10 number 3899 in the binary representation is as follows:

2048 + 1024 + 512 + 256 + 0×128 + 0×64 + 32 + 16 + 8 + 0×4 + 2 + 1 = 111 100 111 011

Each three binary bits are capable of representing a number from 0 to 7, which maps exactly to the range of an octal digit because $8 = 2^3$. For this reason, the bits above are arranged in groups of three so that it is easier to see that they map to the octal digits 7 4 7 3. The sum of these four octal digits is 21 in decimal, which is 25 in octal ($2\times8^1 + 5\times8^0$). The sum of the digits in the octal number 25 is 7, and, sure enough, it is the third multiple of 7. So the original number, 3899, in decimal is also divisible by 7. As you can see, by converting the decimal number first to binary and then to octal, we were able to use digit summing to test for divisibility by 7.

Although this method works, the problem is that it is more complex than just doing division. As an example, let's compare the above calculations to the operations one performs when mentally testing 3899 for divisibility by 7. One can easily test this number as follows:

1) 3500 is five hundred sevens, so look at what's left: 399 (3899 – 3500);

2) 350 is fifty sevens, so look at what's left: 49 (399 – 350); and

3) 49 is the square of seven, so 3899 is divisible by 7.

More generally, to test divisibility by 7 using division, we find the closest multiple of the divisor that doesn't exceed the value of the first one or two digits of the dividend, subtract the multiple from the value to get the remainder, append the next digit of the dividend, and repeat these steps until the digits of the dividend are processed. In essence, we perform about three simple operations per step, and the number of steps is approximately equal to the number of digits in the dividend.

By comparison, testing divisibility by 7 by converting to binary, then octal, then summing octal digits takes many more operations. In the analysis of the approximate number of operations, we will rely on two facts:

Fact 1. The base *b* logarithm of a number *X*, denoted $log_b(X)$, approximates how many digits are in the base *b* representation of *X*. This is true because a logarithm indicates the power to which the base must be raised in order to get a number, and the place values of the number represent successive powers of the number base.

Fact 2. A logarithm can be converted from base *b* to base *a* using $log_b(X)=log_a(X) / log_a(b)$.[9]

Converting a number, *X*, from decimal to binary involves working with descending powers of 2 starting with the one that doesn't exceed *X*. For each power of 2, a 0 bit results if *X* is less than the power of 2. Otherwise, a 1 bit is the output, and the power of 2 is subtracted from *X* before proceeding to the next-lower power of 2. For each binary bit, we perform a comparison and also a subtraction (when generating a 1). Since there is a 50% probability that a randomly selected bit in a randomly selected number will be a 1, the average number of operations to convert to binary is 1.5 times the number of bits. By Fact 1, the number of bits is approximately $log_2(X)$, and the number of decimal digits is approximately $log_{10}(X)$. By Fact 2, $log_{10}(X)=log_2(X) / log_2(10)$. Solving algebraically for the ratio $log_2(X) / log_{10}(X)$ gives $log_2(10)$, which is about 3.32 and, based on Fact 1, is the approximate ratio of bits to decimal digits. Despite being a slight

overestimation, it suffices for estimating that conversion of a number X to binary requires about 5 (≈3.32×1.5) operations per decimal digit of X. Using similar calculations, there are approximately 1.1 octal digits per decimal digit in a number, so converting to octal requires about 1.1 operations per decimal digit, and summing the octal digits requires about 1.1 more operations per decimal digit. In the final tally, this new method requires about 7 mental operations per decimal digit compared with only 3 operations per decimal digit for using division to test divisibility by 7.

Divisibility by 7 and Creativity by Adaptation

In search of a simpler approach, I switched to a creative mode consistent with being a software algorithm developer. In this mode, one begins to "dream up" or imagine or create a new algorithm by first trying to adapt a known algorithm, i.e., by "playing with" or adjusting its internal operations to make it applicable to the problem at hand. Instead of just thinking about known algorithms from the outside, based on *what* they do, we switch to thinking about known algorithms from the inside, based on *how* and *why* they work. For testing divisibility by 9, a digit-summing method worked, so let's see how that might be adjusted to test for divisibility by 7.

For divisibility by 9, the analysis in the proof by induction revealed two cases: adding 9 to a number, X_n, either didn't cause a carry or did cause a carry. When a carry occurred, there was a balance: the carry added 1, but the ones place decreased by 1, and any other digit changes were from 9 to 0. This balance meant that there was no change between X_n and X_n+9 in whether the sum of digits was a multiple of 9.

For divisibility by 7, the same two cases emerge. If adding 7 to a number, X_n, doesn't cause a carry, then the sum of digits of X_{n+1} (i.e., of X_n+7) is increased by 7. But when the ones place of X_n is in the range 3 to 9, then two problems occur. First, any extra digits changed by a carry operation are changed by 9, which doesn't help divisibility by 7 quite as it did divisibility by 9. Second, there is an imbalance, because a carry of 1 to the higher digits is mismatched with a decrease of 3 in the ones place. For an example of both problems, 3899 + 7 = 3906.

To solve the first problem, let's amend the algorithm. *What if*, instead of summing the digits above the ones place, we just take a sum that involves a single number formed by all digits of X_n except the ones place? For example, let the revised summing algorithm operate on 3899 by taking 389 plus something involving the 9 in the ones place, and let it operate on 3906 by taking 390 plus something involving the 6 in the ones place.

We could refer to the higher digits of X_n using the notation H_n and the ones place using O_n. Then, the revised summing algorithm is simply to take $H_n+f(O_n)$ for some function f. This solves the first problem in which the value of the higher digits of X_{n+1}, denoted H_{n+1}, increments by only 1 due to a carry no matter how the digits within H_{n+1} change relative to H_n. The notation also allows us to characterize the second problem as an imbalance in the changes that occur to H_n and O_n when adding 7 to X_n. Specifically, although $H_{n+1} = H_n+1$, the ones place digit of X_{n+1}, denoted O_{n+1}, is 3 less than O_n.

Rather than decreasing O_n by 3, we need a decrease of 1 more than some multiple of 7. That decrease, when combined with increasing H_n by 1, would mean that the sum would decrease by an exact multiple of 7. *What if* we use H_n+5O_n as the revised summing algorithm? This way, a decrease of 3 in O_n corresponds to a decrease of 15 in $5O_n$, where 15 is one more than a multiple of 7. Put another way, $H_{n+1}+5O_{n+1}=H_n+5O_n-14$. And, when adding 7 to X_n produces no carry in the ones place, then $H_{n+1}+5O_{n+1}=H_n+5O_n+35$. In both cases, the sum's divisibility or non-divisibility by 7 is preserved between X_n and $X_{n+1} = X_n+7$.

To test an example, 3899 would be processed as 389+45=434. If it is not visually apparent that the result is divisible by 7, one can iterate the processing: 434 would be processed as 43+20=63, and we know from the normal 12×12 multiplication table that 63 is divisible by 7.

Divisibility by 7: There is No Spoon

The summing method we have just created for testing divisibility by 7 is easier than the earlier method that required binary and octal conversions, but how easy is it? Let's do a little more algorithm analysis. For the divisibility-by-9 test, the sum of the digits of X won't exceed 9×floor(1+$log_{10}(X)$), where "floor" means truncate any decimal part. This is true because base-10 digits cannot exceed 9 and because floor(1+$log_{10}(X)$) more accurately characterizes the base-10 digit count of X than did Fact 1 above. Thus, the sum of digits of any number under a trillion will still be in the range of the products appearing in the normal 12×12 multiplication table. For example, to do mental calculations on phone numbers, you'll only have to sum the digits once to get an answer. By comparison, the summing method above for testing if X is divisible by 7 produces an answer that is never smaller than one digit less than X. This means that you must mentally perform the summing method at least once for each digit of X more than two digits, in order to reduce the answer down to something from the normal 12×12 multiplication table.

On the surface, it seems that the complexity difference is that of performing only one summing operation (for divisibility by 9) versus performing a summing operation for each digit (for divisibility by 7). However, for divisibility by 9, the complexity is hidden inside the method. One must still perform an operation with each digit, namely, to add it to an accumulated total. The reason that the divisibility-by-7 summing method is more complex is because it also requires more mental work for each digit. One must multiply O_n by 5 before adding it to anything, and then the addition of $5O_n$ to H_n is harder because the numbers are larger than just adding a digit value to an accumulated sum of digits.

Clearly, the summing method for testing divisibility by 7 is harder than just summing digits, but is it at least easier than using division? Well, as shown above, testing divisibility by 7 with division requires a number of steps commensurate with the number of digits and about 3 simple operations per step. With the new summing method, there are two steps to compute $5O_n$ and H_n, and a third to add them. So, it's close. Adaptation yielded a better creative result than composition. However, division is simpler because the work is more compartmentalized and performed on smaller numbers. With the new summing method, the numbers are larger, so adding $5O_n$ most often involves two single-digit additions, and carry operations may extend through the length of the sum.

This begs the question: is there a divisibility-by-7 testing method that is simpler than division? It's hard to prove the non-existence of a method, but let's check if there are any giants upon whose shoulders we can stand. By extending to the web our search for a simpler method, we discover in the public domain not only one but a list of eight "shorthand" methods for testing divisibility by 7, including the method we created.[10] Upon close examination, none of these methods that have been popularized over the last 50 years lives up to the descriptor of being a "shorthand" for testing divisibility by 7, which provides strong evidence supporting the assertion that there is no shorthand method. We are left with the surprising result that division is the best mental method for testing divisibility by 7.

Conclusion

In this paper, I began by asserting that playfulness is essential as it helps us sharpen and widen the critical skills we need to achieve our objectives and goals in life. While searching

for a new way to play with numbers, we took quite a tour of topics, including mathematical proof techniques, artificial intelligence, number representation schemes, algorithm analysis, and creative processes. A key observation is that the imaginative "what if" processing used to create a new algorithm by adaptation is strikingly similar to the processing involved in performing word plays and number plays. Often, playing with available concepts by making small tactical changes can, indeed, result in achieving an important objective. And regardless of whether achieving that objective supports the intended strategic goal or produces a surprising alternative result, it is worth the effort because… it's *fun*.

NOTES

1. Patrick Bateson, "Playfulness and Creativity," *ScienceDirect* 25, no. 1 (Jan 2015): R12-R16, https://www.sciencedirect.com/science/article/pii/S0960982214011245.

2. Bernard L. De Koven, "The Underrated Importance of Being Playful," *Psychology Today*, August 6, 2015, https://www.psychologytoday.com/us/blog/having-fun/201508/the-underrated-importance-being-playful; Bernard L. De Koven, "Survival of the Playfullest," *A Playful Path*, October 10, 2014, https://www.aplayfulpath.com/playfulness-is-survival-skill/.

3. Wikipedia contributors, "Faulty generalization," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Faulty_generalization&oldid=841195109.

4. Wikipedia contributors, "Overfitting," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Overfitting&oldid=864108319.

5. Wikipedia contributors, "Mathematical induction," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Mathematical_induction&oldid=866872907 (accessed November 7, 2018). This website gives more information about mathematical induction, including the analogy with falling dominoes.

6. Wikipedia contributors, "Proof by contradiction," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Proof_by_contradiction&oldid=864365322.

7. Wikipedia contributors, "Octal," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Octal&oldid=862649931. This website gives more information about octal (base 8) numbers as well as hexadecimal (base 16) and binary (base 2) numbers, including numeric conversions between them.

8. Danny Kaye, "Inchworm Lyrics," LyricWiki, http://lyrics.wikia.com/wiki/Danny_Kaye:Inchworm.

9. Wikipedia contributors, "Change of Base Logarithm," Wikipedia, The Free Encyclopedia, https://proofwiki.org/wiki/Change_of_Base_of_Logarithm. This website provides a simple proof of the logarithm change-of-base formula.

10. Wikipedia contributors, "Divisibility Rule," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Divisibility_rule&oldid=865687326.