Examining the Complexity of a Factor-based Numeral System

Mathematics Internal Assessment

by Harrison Davison

14 Pages

1 Inspiration

The modern base-10 numeral system can be traced to India in the 6th or 7th century [1]. It was subsequently introduced to Europeans by Middle Eastern mathematicians, which is why is it known to many as “Arabic numerals” [1]. Base-10 is a positional system, meaning that there is a set of digits that represents different values in different places. Moving to the left, each digit is ten times the previous, so Arabic numerals express all whole numbers as a unique, finite sum of multiples of powers of ten. Binary works similarly, but each digit is two times the previous instead of ten. Thus, binary identifies whole numbers as unique sums of finitely many powers of two.

Binary is a particularly useful numeral system because of its applications in computer science. The most fundamental unit of data is a bit, which is represented by either a one or a zero [2]. As such, one bit of information is one digit of binary, eight bits (a byte) can be represented with eight digits, and so on. It is important to note that while data storage and the size of a number are correlated, they are not the same thing. In binary, writing the number nine uses the same number of digits, and thus the same amount of storage, as the number ten (1001 vs 1010). From this principle, one may get an intuitive notion of the complexity of a number; if it takes more space or storage to record a number, then it is more complex.

While positional digit-based systems like binary and the Arabic numerals are very common, they are certainly not the only way to express numbers. In theory, any method of systematically giving a unique symbol to every natural number could be an alternative. This is how I came up with the idea to develop a numeral system based on prime factors. Every number already had a unique prime factorization, so all that I needed was a method for denoting the prime numbers themselves. After that issue was solved, the system worked, and my attention was naturally brought to the topic of complexity. Hence, my question: “How complex are the the symbolic representations of natural numbers in a factor-based numeral system?”

2 The System

The idea that all prime numbers have a single, unique prime factorization is proven by the fundamental theorem of arithmetic [3]. If there were a set of characters, with one for every prime number, then any number could simply be written as its factors. As an example, let $A = 2$, $B = 3$, $C = 5$, and $D = 7$. The only way to make the number four is to multiply two twos, so $AA = 4$. By the same principle, $AB = 6$, $AD = 14$, and $ABC = 30$. Note that such a system would not be positional, so there is no fundamentally correct order in which to put the characters. For simplicity, the factors are put in ascending order.

Of course, the alphabet can only represent primes up to the 26th prime (which is 101), so it will not work. In fact, no finite system would work. There must instead be a procedural method to draw a unique symbol for every prime number. The following rules will be the solution to create the factor-based numeral system:

- Let the number one be represented by a single vertical mark.
- Let drawing a horizontal mark across the symbol for any number n represent the nth prime.
- Let composite numbers be represented by their prime factors written next to each other in ascending order from left to right.

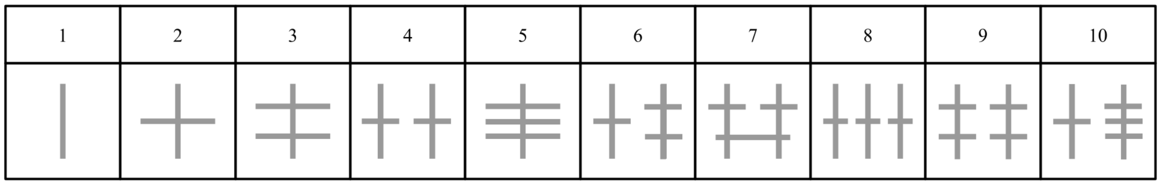


Figure 1: The numbers from one to ten in the factor-based numeral system

As seen in figure 1, composite numbers can be neatly written with their factors, and prime numbers get their own symbols. A possible drawback to this system is that addition appears more difficult. There is little intuitive reason why the symbol for two and the symbol for five add to the symbol for seven if one ignores what the numerals represent. On the other hand, the factor-based numeral system significantly simplifies multiplication. Consider multiplying two and five. Writing the symbols next to each other yields the symbol for ten without needing to do any calculations.

To extend the notion of complexity to this system, a concept analogous to the number of digits used to write a number must be found. The most obvious approach would be to count how many characters are in the factor-based representation of the number. Using this interpretation and referring back to figure 1, the complexity of one would be one, the complexity of six would be two, and the complexity of eight would be three. However, this interpretation conflicts with the intuitive notion of what complexity means; the symbol for seven is obviously more complex than the symbol for one, as it take up more space and would take longer to draw.

A more intuitive notion of complexity comes from considering marks to be the fundamental units of information instead of symbols. There are two types, each with a different purpose: vertical marks and horizontal marks. While vertical and horizontal marks mean different things, so too do the ones and zeros in binary. A definition of complexity follows:

Definition. *The complexity of a number in the factor-based numeral system is equal to the total number of vertical marks and horizontal marks required to write the number.*

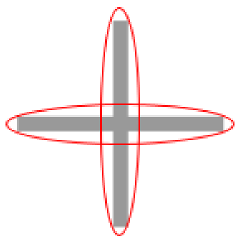
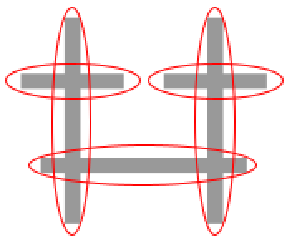
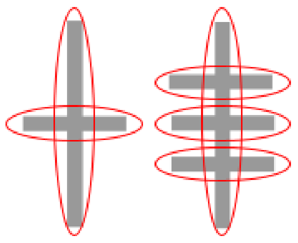
complexity of 2 = 2	complexity of 7 = 5	complexity of 10 = 6
		

Figure 2: Demonstration of the complexity of two, seven, and ten

3 Complexity as a Function

Creating a function that returns the complexity of numbers in the factor-based numeral system will make it easier contemplate the complexity of those numbers without needing to write them down. In order to define such a function, several other functions should be introduced. The prime counting function $\pi(x)$ is defined as the number of prime numbers less than or equal to x [4]. Furthermore, let the function u be defined as $u(x) = p_x$, where p is the sequence of prime numbers $\{2, 3, 5, 7, ...\}$. In other words, $u(x)$ will return the nth prime number for $n = x$.

To see how complexity is related to the function $u(x)$, consider what happens when a horizontal mark is drawn across a number. The nth prime number is written as n with a horizontal mark across it, so drawing a horizontal mark across a number is equivalent to applying the u function to it. Additionally, drawing such a mark increases the complexity of the number by one. Applying the u function to a number therefore increases its complexity.

Theorem 1. *The complexity of $u(x)$ equals one plus the complexity of x.*

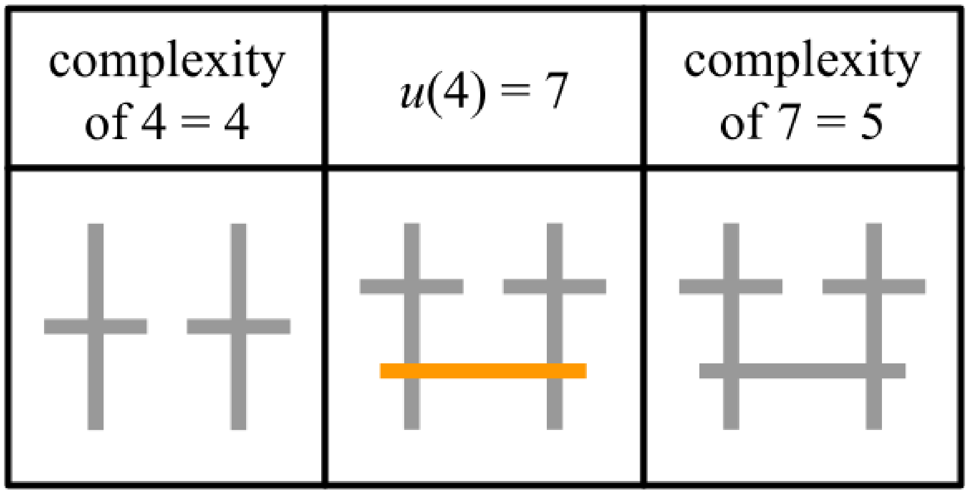


Figure 3: Demonstration of the first theorem

Now consider what happens to numbers in the factor-based numeral system when they are multiplied. As mentioned previously, simply putting two numbers next to each other is sufficient to multiply them, provided that neither equals one and that their symbols are placed in ascending order. Reordering does not change the number of marks involved in drawing a number, so the complexity will be equal to the sum of the original numbers' complexities.

Theorem 2. *The complexity of ab equals the complexity of a plus the complexity of b given $a, b \neq 1$.*

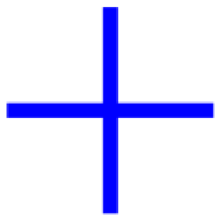
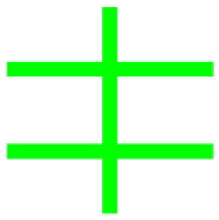
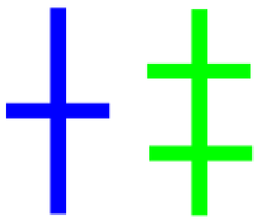
complexity of 2 = 2	complexity of 3 = 3	complexity of 6 = 5
		

Figure 4: Demonstration of the second theorem

A function K will now be created to model the complexity of numbers in the factor-based numeral system by using the above theorems. However, this method will not produce an explicit formula. In the absence of such a formula, the function is best computed in terms of itself, making it recursive [5]. Recursive functions require an initial value to provide a starting point, so $K(2)$ will be defined to match the complexity of two. Taking this all together,

$$K(u(x)) = K(x) + 1$$

$$K(ab) = K(a) + K(b)$$

$$K(2) = 2$$

Two substitutions can make the recursive definition more usable. First, x is replaced with $\pi(p)$, where p is a prime number. The substitution is valid because $\pi(1) = 0$, and $\pi(x)$ increases every time it reaches another prime, so every natural number can be expressed in this manner.

$$K(u(\pi(p))) = K(\pi(p)) + 1$$

To simplify further, $u(\pi(p))$ is reduced using the fact that the $\pi(x)$th prime number is equal to x whenever x is prime [4]. Hence,

$$K(p) = K(\pi(p)) + 1$$

where p is prime. Second, ab is replaced with c. Note that c is divisible by both a and b because they are natural numbers and it is their product. From this substitution, it follows that $b = \frac{c}{a}$. Hence,

$$K(c) = K(a) + K(\frac{c}{a})$$

where a divides c.

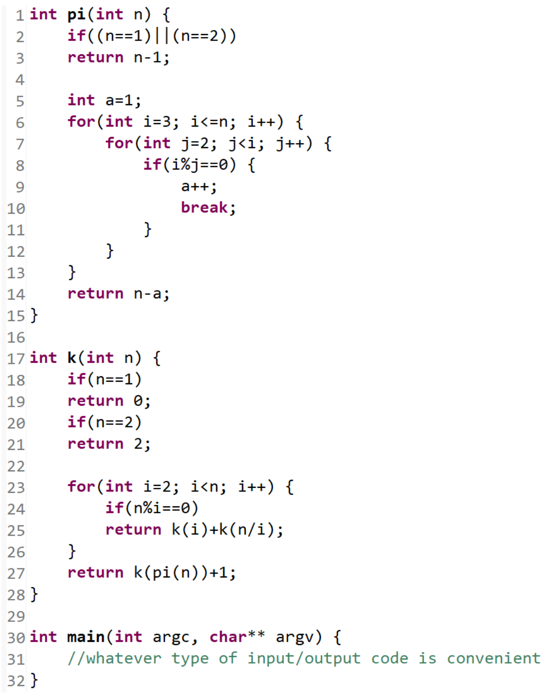
Finally, the value of $K(1)$ will be addressed. The complexity of one is clearly equal to one, but the K function is only defined recursively, not in terms of the original idea of complexity. While $K(x)$ equals the the complexity of x for all $x \geq 2$ because the same rules apply in both cases, the second theorem does not apply to complexity if a or b equals one. No such restriction was put on K, meaning that the equation $K(a \times 1) = K(a) + K(1)$ can be solved for $K(1)$, giving $K(1) = 0$. Therefore, $K(1)$ is a special case in which $K(x)$ is not equivalent to the complexity of x.

4 Computer Program

The simplified recursive rules that define the K function can be summarized in a single piecewise function:

$$K(x) = \begin{cases} 0 & x = 1 \\ 2 & x = 2 \\ K(\pi(x)) + 1 & x \text{ is an odd prime} \\ K(\min\{f \in \mathbb{N} \setminus \{1\} : f \mid x\}) + K(\frac{x}{\min\{f \in \mathbb{N} \setminus \{1\} : f \mid x\}}) & x \text{ is composite} \end{cases}$$

While the last line may look unfamiliar, it is just the rule for composite numbers, with f fixed as the smallest factor of x other than one. The piecewise does not look particularly friendly, but it can be converted into a very effective computer program. Written in C++, the program looks like figure 5.



```cpp
int pi(int n) {
    if((n==1)||(n==2))
    return n-1;

    int a=1;
    for(int i=3; i<=n; i++) {
        for(int j=2; j<i; j++) {
            if(i%j==0) {
                a++;
                break;
            }
        }
    }
    return n-a;
}

int k(int n) {
    if(n==1)
    return 0;
    if(n==2)
    return 2;

    for(int i=2; i<n; i++) {
        if(n%i==0)
        return k(i)+k(n/i);
    }
    return k(pi(n))+1;
}

int main(int argc, char** argv) {
    //whatever type of input/output code is convenient
}
```

Figure 5: A program that nicely implements the above piecewise function

Any modern machine should be able to run the above code, which allows values of the K function to be quickly generated into the thousands. Since $K(x)$ equals the complexity of x for all $x \geq 2$, this also allows the complexities of the factor-based forms of large numbers to be found. It is no longer necessary to write out the factor-based numerals, although it can be helpful.

Using a C++ program like the one above, the first ten thousand values of $K(x)$ were found. Afterwards, they were plotted on a graph to look for any trends.

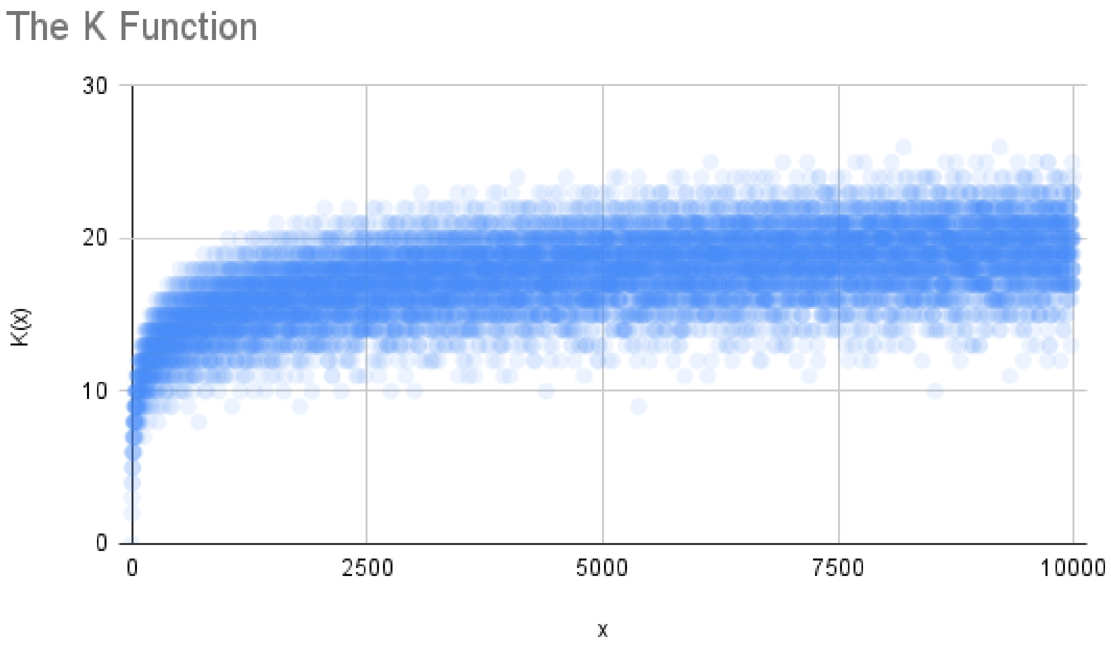


Figure 6: The K function from $x = 1$ to $x = 10000$ in a Google Sheet [6]

The opacity of the points in figure 6 was decreased so that they would not cover each other up as much. As can be seen, the function never rises above thirty. In fact, the maximum value between one and ten thousand is $K(x) = 26$. This result was initially surprising, but it is logical in the context of other numeral systems. The highest amount of digits any binary number between one and ten thousand has is fourteen. Unlike binary, the the complexity of numbers in the factor-based numeral system decreases at some points. For example, the complexity of 5381 is only nine, far less the complexities of any surrounding numbers. K is clearly not a standard function, but its oscillating nature raises the question of what its upper and lower bounds are.

5 Upper Bound

A general answer to the question of how complex numbers are in the factor-based system can be found by determining an upper bound to the K function. Based on figure 6, the upper bound appears to be logarithmic. As such, it is a logical first step to begin examining the relationship between K and exponents. Recall the rule

$$K(ab) = K(a) + K(b)$$

If $K(a^n)$ were evaluated, it would be equal to K of the product of n many a values. Using the above rule, that could be separated into the sum of n many $K(a)$ values. Therefore,

$$K(a^n) = nK(a)$$

Now all even $K(x)$ will be considered. If $K(x)$ is even, then x could potentially be equal to $2^{\frac{K(x)}{2}}$ because $K(2^{\frac{K(x)}{2}}) = \frac{K(x)}{2}K(2) = K(x)$. A number such as x would be written as $\frac{K(x)}{2}$ many symbols for two next to each other in the factor-based numeral system. Now suppose that there were a smaller value of x that gave the same $K(x)$. Its factor-based representation must use the same number of marks as $2^{\frac{K(x)}{2}}$ because its K value is the same. It does not have any extra vertical marks because each vertical mark must have at least one of its own horizontal marks, and there are no spare horizontal marks to go around. Thus, this hypothetical number only has extra horizontal marks. To get more marks to use as horizontals, symbols must be eliminated. Each time one is eliminated, the number is divided by two because all of the symbols are twos. However, each new horizontal that is drawn will multiply the number by some non-constant amount.

Each symbol for two that is eliminated provides two more marks to be used as horizontals. This applies the u function to two parts of the factorization of the number and divides the number by two. The net amount the number is multiplied by after removing a symbol (r) is thus calculated using the equation

$$r = \frac{1}{2}\left(\frac{u(x)}{x}\right)\left(\frac{u(y)}{y}\right)$$

where x represents the smaller part of the factorization that a horizontal is drawn across and y represents the larger part.

No prime number other than two is even, so the rest of the primes are separated by at least two units, so the smallest possible value of $u(x)$ after $u(2)$ would be $2x - 3$. Therefore, $\frac{u(x)}{x}$ is at least $\frac{2x-3}{x}$, and $\frac{u(y)}{y}$ is at least $\frac{2x-3}{x}$. The expression $\frac{2x-3}{x}$ is strictly increasing for all positive x, and $y \geq x$, so $\frac{2y-3}{y}$ is at least $\frac{2x-3}{x}$, so $\frac{u(y)}{y}$ is at least $\frac{2x-3}{x}$. Therefore,

$$r \geq \frac{(\frac{2x-3}{x})^2}{2} = 2 - \frac{6x - 4.5}{x^2}$$

If the subtracted part is less than one, then the overall process will increase the number. The following inequality is solved:

$$\frac{6x - 4.5}{x^2} < 1$$

$$6x - 4.5 < x^2$$

The intersection of $6x - 4.5$ and x^2 is found:

$$x = \frac{6 \pm \sqrt{(-6)^2 - 4(1)(4.5)}}{2(1)} \approx 0.88 \text{ or } 5.12$$

These approximations are sufficient to establish bounds in which r is definitely greater than one. If $x \leq 0$ or $x \geq 6$, r is greater than one. There are five cases in between: $x = 1$, $x = 2$, $x = 3$, $x = 4$, and $x = 5$. The case $x = 1$ can be eliminated because the one symbol would never be present in the factor-based representation of another number.

If the following inequality were true, r would be greater than one:

$$\frac{1}{2}\left(\frac{u(x)}{x}\right)\left(\frac{2y-3}{y}\right) > 1$$

By manipulating the inequality, the minimum condition for this to be true can be found:

$$\frac{2y-3}{y} > \frac{2x}{u(x)}$$

$$2 - \frac{3}{y} > \frac{2x}{u(x)}$$

$$2 - \frac{2x}{u(x)} > \frac{3}{y}$$

$$y > 3(2 - \frac{2x}{u(x)})^{-1}$$

For $x = 2$, the minimum condition is $y > 4.5$. For $x = 3$, the minimum condition is $y > 3.75$. For $x = 4$, the minimum condition is $y > 3.5$. Finally, for $x = 5$, the minimum condition is $y > 2.75$. Now the remaining unproven pairs of x and y can be manually checked. If

$$\frac{1}{2}\left(\frac{u(x)}{x}\right)\left(\frac{u(y)}{y}\right) > 1$$

in each case, then r has been shown to always be greater than one, so redistributing the marks from the symbols into horizontal marks must increase the number, so the hypothetical number does not exist, so the smallest x value corresponding to any even $K(x)$ is a power of two.

Let $R(x,y) = \frac{1}{2}(\frac{u(x)}{x})(\frac{u(y)}{y})$ to decrease the space taken up by the r equation. $R(2,2) = \frac{9}{8} > 1$. $R(2,3) = \frac{5}{4} > 1$. $R(2,4) = \frac{21}{16} > 1$. $R(3,3) = \frac{25}{18} > 1$. All unproven pairs have been shown to be greater than one, so the smallest x value corresponding to any even $K(x)$ is a power of two.

Based on the power rule discovered earlier, it appears that any set of powers (powers of two, powers of three, etc.) must follow a single logarithmic path once put through the K function. Since the following equation can be generally solved for b, the conjecture is true:

$$\log_b(a^n) = K(a^n)$$

$$n\log_b(a) = nK(a)$$

$$a = b^{K(a)}$$

$$b = a^{\frac{1}{K(a)}}$$

Therefore, $\log_{a^{\frac{1}{K(a)}}}(x)$ will intersect $K(x)$ at every coordinate where the x value is a whole power of a. Because of this fact and the fact that $2^{\frac{1}{K(2)}} = \sqrt{2}$, the following statement is proven:

Theorem 3. *All even* $K(x) \leq \log_{\sqrt{2}}(x)$.

It has been shown that eliminating a two from the factor-based representation of a number to create a pair of horizontal marks will always increase the number. If this is the case, then eliminating two horizontal marks to make a two must always decrease the number. No other arrangement of marks could produce a lower value than creating a two because said arrangement could then be made into pairs of twos to decrease it further. This means that the lowest x value for any $K(x)$ can be produced by repeatedly taking away pairs of horizontal marks and constructing twos with them.

If $K(x)$ were odd, there would necessarily be one spare horizontal mark left over after the maximum number of twos were formed. That one horizontal mark would be paired with a vertical mark and another horizontal mark. Together, they would make the symbol for a three. Hence, the lowest possible x value for any odd $K(x)$ is equal to three times a power of two. The expression $3 + \log_{\sqrt{2}}(\frac{x}{3})$ produces three at $x = 3$, and $3 + \log_{\sqrt{2}}(\frac{3\times 2^a}{3}) = 3 + \log_{\sqrt{2}}(2^a) = 3 + 2a$, so the expression produces consecutive odd numbers when consecutive powers of two times three are put into it, starting with $(3, 3)$, which lies on $y = K(x)$. Thus, $3 + \log_{\sqrt{2}}(\frac{3\times 2^a}{3}) = K(3 \times 2^a)$.

Theorem 4. *All odd* $K(x) \leq 3 + \log_{\sqrt{2}}(\frac{x}{3})$.

Now it will be shown that the upper bound of even $K(x)$ is greater than that of odd $K(x)$.

$$3 + \log_{\sqrt{2}}(\frac{x}{3}) = (3 - \log_{\sqrt{2}}(3)) + \log_{\sqrt{2}}(x) < \log_{\sqrt{2}}(x)$$

Theorem 5. *All* $K(x) \leq \log_{\sqrt{2}}(x)$.

An upper bound has been found for $K(x)$, which can also be applied to the complexity of x. The complexity of x does not equal $K(x)$ at one, so the upper bound does not apply there.

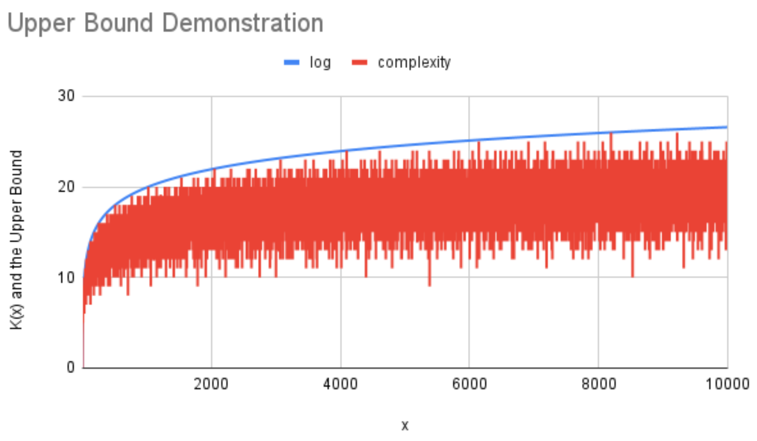


Figure 7: Line graph of $K(x)$ and $\log_{\sqrt{2}}(x)$, with K never exceeding the logarithm [6]

6 Conclusion

The upper bound of the complexity of a number in the factor-based numeral system is equal to the log base square root of two of that number. There are infinitely many numbers with complexities equal to the upper bound, but the density of numbers with lower complexities seems to be much greater. In that regard, exclusively focusing on finding the upper bound is a limitation of the research methodology, as a simple logarithmic function cannot describe the intricacies of the K function. However, the fact that the upper bound function is so simple greatly improves its utility. If one wanted a quick estimate for the K function, one could enter the upper bound equation into a calculator and get back an answer much faster than software could deliver an exact value.

A direction in which the investigation could be continued would be to study the lower bound of complexity. From observations of the graph, it is clear that the bound is not logarithmic, although it looks similar. Finding a function that somehow goes through the middle of the function would also be valuable, as that would be more revealing of the shape of the function. An alternative to these approximation methods would be to find a more efficient algorithms for calculating the K function. If one were found, it would be possible to obtain a much larger data set.

The application of the ideas put forward here could be wide reaching, from studying data compression to the frequency of prime numbers. While some numbers are less complex in the factor-based numeral system than in binary, binary is still generally more efficient. It would be worth studying if there is a system that can consistently express larger numbers in less complicated ways than traditional base systems can. The ability to express numbers using low complexity could have major applications in computer science. For example, such methods could enable the creation of hard drives that are physically small but store large amounts of information. This would connect to my personal interests and hobbies because I develop programs, and the storage capacity of the machine running my software can occasionally impede my ideas.

Another area of study that the information in this paper applies to is the study of recursive functions. Functions such as these frequently behave unexpectedly, as is the case with the K function. It is possible that researching the K function in its own right, apart from number systems, could lead to general insights into this branch of mathematics.

References

[1] The Editors of Encyclopaedia Britannica. Hindu-arabic numerals. https://www.britannica.com/topic/Hindu-Arabic-numerals, Sep 2017.

[2] Ashley Taylor. Bits and bytes. https://web.stanford.edu/class/cs101/bits-bytes.html.

[3] Eric W Weisstein. Fundamental theorem of arithmetic. https://mathworld.wolfram.com/FundamentalTheoremofArithmetic.html.

[4] Eric W Weisstein. Prime counting function. https://mathworld.wolfram.com/PrimeCountingFunction.html.

[5] Walter Dean. Recursive functions. https://plato.stanford.edu/entries/recursive-functions/, Apr 2020.

[6] Google sheets. https://docs.google.com/spreadsheets/.