Proof of the Hardy-Littlewood K-tuple Conjecture in the Distribution of Numbers Coprime with the Primorial

Tim Samshuijzen
TimSamshuijzen@gmail.com
Wageningen, Netherlands

January 2025

Abstract

In the symmetries in the numbers that are coprime with the primorial we find proof of the Hardy-Littlewood K-tuple Conjecture and, consequently, the Twin Prime Conjecture. Using a primorial-based sieve, called the *bitstring sieve*, we prove that the number of prime k-tuples of size k between p_n^2 and p_{n+1}^2, where p_n is the n-th prime, increases on average with increasing n. Hardy and Littlewood's statistical predictions concerning prime k-tuples and twin primes are correct.

1 Introduction

The Hardy-Littlewood K-tuple Conjecture, or the First Hardy–Littlewood Conjecture, is a long-standing problem in analytic number theory. Proposed by G. H. Hardy and J. E. Littlewood in their 1923 paper *Some problems of 'Partitio numerorum'; III: On the expression of a number as a sum of primes* [1], as part of their investigations into the distribution of primes, using their circle method approach, the conjecture predicts the asymptotic frequency with which prime k-tuples of size k appear.

In this paper, we present a proof of the Hardy-Littlewood K-tuple Conjecture. The starting point of the proof is a description of the *bitstring sieve*. The bitstring sieve is the smallest program of binary digit instructions that generates the primes, running on a digital computing machine with unlimited memory. The bitstring sieve is similar to the Sieve of Pritchard, or wheel sieve, and identical to the sieve conceived by Pete Quinn [2], posted on primegrid.com in 2011.

The bitstrings generated by the bitstring sieve are a recording of the creation and elimination of *candidate primes*. Candidate primes are the numbers coprime with the primorial, represented as bit value 1 in the bitstring. The visual pattern of 1s in the bitstrings clearly show the periodicities and symmetries in the numbers coprime with the primorial. These patterns, often called *primorial patterns*, are well known. Some examples of work on primorial patterns are Dennis R. Martin's *Proofs Regarding Primorial Patterns* 2006 [3] and Fred B. Holt's *Patterns among the Primes* 2022 [4].

2 The Bitstring Sieve

The bitstring sieve is the smallest program of binary digit instructions that generates the primes, running on a computer with unlimited memory. The bitstring sieve is defined as a recurrence relation, such that its output is used as input for the next iteration. Per iteration, the first step is to obtain the next prime from the input, and the second step is to calculate the output, which then serves as input for the next iteration, and so forth.

Let S be the set of all outputs generated by the bitstring sieve. For each p_n, where p_n is the n-th prime, there is a sequence $S_{p_n} \in S$:

$$S = \{S_2, S_3, S_5, S_7, S_{11}, \dots\}$$

Each sequence S_{p_n} is a *bitstring*, a finite-length sequence of binary digits $S_{p_n} \in \{0,1\}^*$.

Let a bitstring be noted as $(b_1, b_2, \cdots, b_n)$, where $b_i \in \{0,1\}$ and n is the length of the bitstring. For example, bitstring $(1,0,0,0,1,0)$ has length 6.

Notation:

- $(b_1, b_2, \cdots, b_n)$: Represents individual bits in the bitstring.
- $|s|$: The length of the bitstring s.
- $s[i]$: The i-th bit in the bitstring s, where indexing starts from 1.

The bitstrings of S are generated by the following recurrence relation.

Initial condition:
Let S_1 be the bitstring (1). S_1 is not a member of S, but it serves to get the recurrence relation started. We regard 1 as the zeroth prime p_0.

$$S_1 = (1)$$

Recurrence relation:
Given bitstring S_{p_n}, where p_n is the n-th prime, the next bitstring $S_{p_{n+1}}$ is obtained by:

$$\begin{aligned} p_{n+1} &= NEXT1(S_{p_n}) \\ S_{p_{n+1}} &= AND(CONCAT(S_{p_n}, p_{n+1}), NOT(STRETCH(S_{p_n}, p_{n+1}))) \end{aligned} \tag{1}$$

The set of functions $NEXT1$, AND, $CONCAT$, NOT and $STRETCH$ is the instruction set of the bitstring sieve. These functions are defined below.

NEXT1
Let $NEXT1(s) : \{0,1\}^* \rightarrow \mathbb{Z}^+$ be a function that takes as input bitstring s, and returns the (1-based) index of the second occurrence of 1 in s (skipping the first 1 at index 1), or the length of the bitstring s plus 1 if such an occurrence does not exist, as defined in:

$$NEXT1(s) = \begin{cases} \text{index of second 1 in } s, & \text{if such 1 exists} \\ |s| + 1, & \text{otherwise} \end{cases}$$

AND
Let $AND(s1, s2) : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be a function that takes as input bitstrings $s1$ and $s2$, where $|s1| = |s2|$, and returns a new bitstring with the same length, where each bit is the result of the logical AND operator applied to the corresponding bits in $s1$ and $s2$, as defined in:

$$OR(s1, s2) = (s1[1] \wedge s2[1], s1[2] \wedge s2[2], \cdots, s1[|s1|] \wedge s2[|s2|])$$

Where $\wedge$ represents the logical AND operator.

CONCAT
Let $CONCAT(s, n) : \{0,1\}^* \times \mathbb{Z}^+ \rightarrow \{0,1\}^*$ be a function that takes as input bitstring s and positive integer $n > 0$, and returns a new bitstring with length $|s| \times n$, filled with n concatenated copies of s, as defined in:

$$CONCAT(s, n) = s \circ s \circ \cdots \circ s \quad \text{(n times)}$$

Where $\circ$ denotes concatenation.

NOT
Let $NOT(s) : \{0,1\}^* \rightarrow \{0,1\}^*$ be a function that takes as input bitstring s, and returns a new bitstring with the same length, where each bit is the logical inverse of corresponding bit in s, as defined in:

$$NOT(s) = (\neg s[1], \neg s[2], \cdots, \neg s[|s|])$$

Where $\neg$ represents the logical NOT operator.

STRETCH
Let $STRETCH(s, n) : \{0,1\}^* \times \mathbb{Z}^+ \rightarrow \{0,1\}^*$ be a function that takes as input bitstring s and positive integer $n > 0$, and returns a new bitstring with length $|s| \times n$, where the bits from s are mapped to a position n times farther than their original position, and the positions in between are padded with 0s, as defined in:

$$STRETCH(s,n) = (b_1, b_2, b_3, \cdots, b_{|s| \times n})$$

Where:

$$b_i = \begin{cases} s[\frac{i}{n}], & \text{if } i \text{ is a multiple of } n \\ 0, & \text{otherwise} \end{cases}$$

In words, the recurrence relation works as follows. Given bitstring S_{p_n} as input, the output bitstring $S_{p_{n+1}}$ is obtained in two steps. The first step is to determine the next prime p_{n+1}, that is, to find the index of the second occurrence of 1 in S_{p_n}, that is, to skip the first 1 at index 1. If such a 1 is not found, which only happens when iterating from S_1 to S_2, or from S_2 to S_3, then continue searching and counting back from the start of the bitstring, where the first bit is always 1. The second and last step is to create the output bitstring $S_{p_{n+1}}$, by concatenating p_{n+1} copies of S_{p_n}, and then for each 1 in the original S_{p_n}, say at index i, invert the 1 in $S_{p_{n+1}}$ that is at index $p_{n+1} \cdot i$. This resulting $S_{p_{n+1}}$ then serves as input for the next iteration to calculate $S_{p_{n+2}}$, and so on.

Equivalently, more compactly, a bitstring S_{p_n} can be defined as follows:

$$S_{p_n} = (b_1, b_2, b_3, \cdots, b_{p_n\#})$$

Where p_n is the n-th prime, $p_n\#$ is the product of all the primes up to and including the n-th prime, and:

$$b_i = \begin{cases} 1, & \text{if } i = 1 \\ 1, & \text{if } i \text{ is coprime with } p_n\# \\ 0, & \text{otherwise} \end{cases}$$

The first bitstrings are:

$$\begin{aligned}
&S_2 = (1,0) \\
&S_3 = (1,0,0,0,1,0) \\
&S_5 = (1,0,0,0,0,0,1,0,0,0,1,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,1,0) \\
&S_7 = (1,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,1,0,\dots)
\end{aligned}$$

The indices of the 1s in bitstring S_{p_n} are the numbers that are coprime with $p_n\#$.

Figure 1 shows the operations performed by the bitstring sieve when advancing from bitstring S_3 to bitstring S_5. The 0s are represented as white squares with black text, and the 1s are represented as black squares with white text (a convention used throughout this paper). The numbers in the squares indicate the 1-based index of the bit in the bitstring. A, B and C are intermediate registers to show what happens at each step. Note that intermediate step B marks the periodic pattern of integers which have p_n as its least prime factor. The periodic pattern of prime p_n in the sequence of least prime factors (OEIS A020639) is exactly bitstring $S_{p_{n-1}}$ stretched by a factor of p_n.

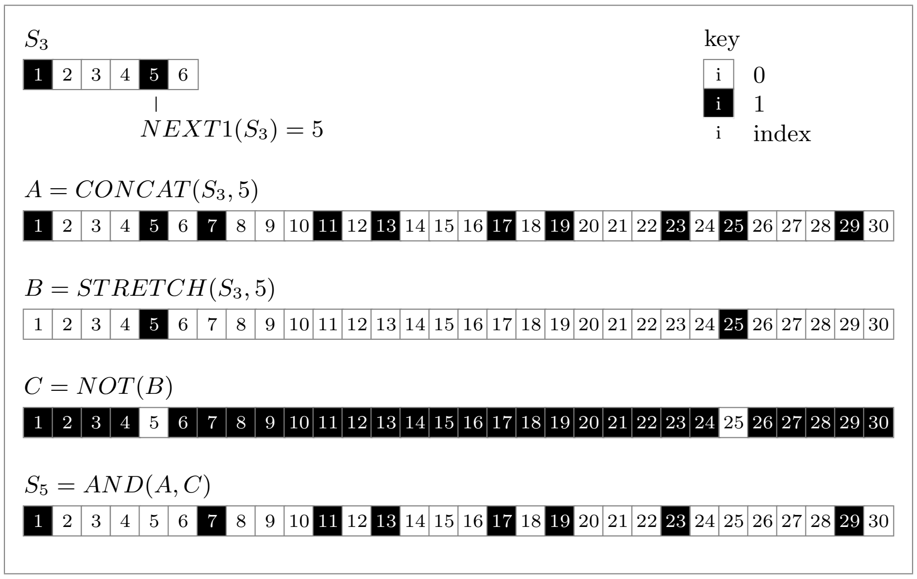


Figure 1: Recurrence relation applied to S_3 to obtain S_5

The 0s (the white squares) in a bitstring S_{p_n} have indices that are not coprime with $p_n\#$, which are either the numbers between 1 and p_n, or the numbers we call *definite composites*. The indices of the 1s (the black squares) in a bitstring S_{p_n} are the numbers that are coprime with $p_n\#$. The 1s after index 1 represent the *candidate primes*. A candidate prime is either a composite, in which case it will at some iteration be marked as a definite composite, or it is a prime, in which case it will survive all iterations until it is found by the $NEXT1$ operation, and eliminated in the next bitstring.

3 Symmetry in the bitstrings

The length of bitstring S_{p_n}, where p_n is the n-th prime, is equal to the primorial function $p_n\#$, the product of all primes up to and including the n-th prime.

$$|S_{p_n}| = p_n\# = \prod_{i=1}^{n} p_i \tag{2}$$

The sequence of primorial numbers is listed in OEIS A002110.

The index of the bit halfway a bitstring at $\frac{|S_{p_n}|}{2}$ is its *index of symmetry.* The pattern of 1s and 0s are mirror-symmetric on either side of this index. In other words, each bitstring S_{p_n} is palindromic. This is because each function ($CONCAT$, NOT, $STRETCH$ and AND) in the recurrence relation (1) conserves symmetry given symmetric input.

A method of visualizing the overall structure and symmetry of the bitstrings in S is to render the bitstrings as rows of black and white squares, where each bitstring is scaled to a common width, and drawn beneath each other. The symmetry in this fractal-like structure becomes apparent when aligning the indices of symmetry in each bitstring, by shifting each bitstring by half a square width to the right, in modular fashion (as if the structure is cylindrical). The result is shown in Figure 2. Each horizontal row corresponds to a bitstring in S. The first row is S_2, the next row is S_3, the next row is S_5, and so forth.

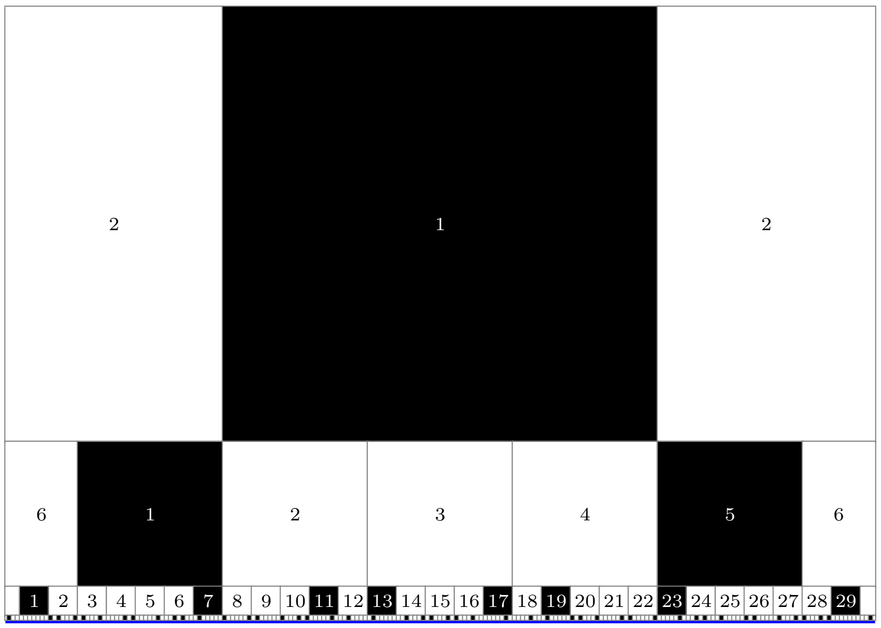


Figure 2: Fractal-like structure of the bitstrings in S

Suppose the width of this fractal-like structure is 1, then its height is the sum of the reciprocals of the primorials, which converges to $0.70523\cdots$.

$$\sum_{n=1}^{\infty} \frac{1}{p_n\#} \approx 0.70523\cdots$$

The Engel expansion of this value is the sequence of prime numbers. See its decimal expansion in OEIS A064648. The heights of the bitstrings after S_7 are too small for print, so in the above figure we represent this convergent area at the bottom as a blue horizontal line. That blue line, slightly thickened to make it visible, contains all the bitsrings from S_{11} to S_{p_∞}. The surface of the bottom of this structure is undefined, as there is no such thing as the largest prime.

4 Candidate prime k-tuples

Let us investigate the recurrence relation (1) and derive formulations for the distribution of 1s in the bitstrings of S.

When iterating from S_{p_n} to $S_{p_{n+1}}$, the *CONCAT* function outputs p_{n+1} times as many 1s as there are in S_{p_n}, and the *NOT-STRETCH* operation eliminates as many 1s as there are in S_{p_n}. Therefore, the number of 1s in bitstring S_{p_n}, denoted as $p_n\#_1$, is as follows:

$$p_n\#_1 = \prod_{i=1}^{n}(p_i - 1) \tag{3}$$

The sequence of $p_n\#_1$ per n is listed in OEIS A005867.

$p_n\#_1$ relates to Euler's totient function ϕ as follows:

$$\begin{aligned} p_n\#_1 &= \prod_{i=1}^{n}(p_i - 1) \\ &= \prod_{i=1}^{n} p_i \prod_{i=1}^{n}\left(1 - \frac{1}{p_i}\right) \\ &= p_n\# \prod_{i=1}^{n}\left(1 - \frac{1}{p_i}\right) \\ &= \phi(p_n\#) \end{aligned}$$

Let a *candidate twin prime* be a sequence in a bitstring that matches $(1, 0, 1)$. When iterating from S_{p_n} to $S_{p_{n+1}}$, the *CONCAT* function creates p_{n+1} copies of the candidate twin primes in S_{p_n}, and the *NOT-STRETCH* operation eliminates 2 candidate twin primes for each candidate twin prime in S_{p_n}. Therefore, the number of candidate twin primes in S_{p_n}, denoted as $p_n\#_2$, where $p_n > 2$, is as follows:

$$\begin{aligned}
p_n\#_2 &= \prod_{i=2}^{n}(p_i - 2) \\
&= \prod_{i=2}^{n}(p_i - \frac{2 \cdot p_i}{p_i}) \\
&= \prod_{i=2}^{n} p_i \prod_{i=2}^{n}(1 - \frac{2}{p_i}) \\
&= \frac{p_n\#}{2} \prod_{i=2}^{n}(1 - \frac{2}{p_i})
\end{aligned}$$

A bitstring S_{p_n} is periodic over the entire natural number line, beyond the length of the bitstring, so we include in our count the candidate twin prime that would be formed at index 1 and index $(p_n\# - 1)$. Note that $p_1\#_2 = 1$, because in S_2 we encounter $(3, 5)$ when wrapping around in modular fashion. The sequence of $p_n\#_2$ per n is listed in OEIS A059861. In addition to counting the number of candidate twin primes, $p_n\#_2$ also counts the number of *candidate cousin primes*, that is, occurrences of bit pattern $(1, 0, 0, 0, 1)$, and also the number of *candidate sexy primes*, that is, occurrences of bit pattern $(1, 0, 0, 0, 0, 0, 1)$.

The bit sequence (1,0,1,0,0,0,1,0,1) is a *candidate prime quadruplet*. For example, this sequence can be found in S_5 at index 11, corresponding with prime quadruplet $(11, 13, 17, 19)$, a constellation of the form $(p, p + 2, p + 6, p + 8)$. This candidate prime quadruplet is copied 7 times into S_7, of which $(7 - (4 \cdot 1)) = 3$ survive, at indices $11, 101, 191$. These 3 candidate prime quadruplets are copied 11 times into S_{11}, of which $(33 - (4 \cdot 3)) = 21$ survive. These 21 candidate prime sextuplets are copied 13 times into S_{13}, of which 189 survive. Therefore, the number of candidate prime quadruplets in S_{p_n}, where $p_n > 4$, denoted as $p_n\#_4$, is as follows:

$$p_n\#_4 = \prod_{i=3}^{n}(p_i - 4)$$

We define $2\#_3 = 1$ and $3\#_3 = 1$, because in S_2 we encounter $(3, 5, 7, 9, 11)$, and in S_3 we encounter $(5, 7, 11, 13)$. The sequence of $p_n\#_4$ per n is listed in OEIS A059863.

The bit sequence (1,0,0,0,1,0,1,0,0,0,1,0,1,0,0,0,1) is a *candidate prime sextuplet*. For example, this sequence can be found in S_5 at index 7, corresponding with the prime sextuplet $(7, 11, 13, 17, 19, 23)$, a constellation of the form $(p, p + 4, p + 6, p + 10, p + 12, p + 16)$. This candidate prime sextuplet is copied 7 times into S_7, of which only 1 survives, at index 97. This candidate prime sextuplet is copied 11 times into S_{11}, of which 5 survive. These 5 candidate prime sextuplets are copied 13 times into S_{13}, of which 35 survive. Therefore, the number of candidate prime sextuplets in S_{p_n}, denoted as $p_n\#_6$, where $p_n > 6$, is as follows:

$$p_n\#_6 = \prod_{i=4}^{n}(p_i - 6)$$

The sequence of $p_n\#_6$ per n is listed in OEIS A059865.

From here we generalize to *candidate prime k-tuples.* In the bitstrings we observe the following:

> *Whatever pattern of* 1*s and* 0*s can be found in bitstring* S_{p_n}*, all occurrences of this pattern is copied* p_{n+1} *times into* $S_{p_{n+1}}$*, and eliminated as many times as the number of occurrences of this pattern in the original* S_{p_n} *multiplied by the number of* 1*s in the pattern.*

The number of candidate prime k-tuples in bitstring S_{p_n}, denoted as $p_n\#_k$, where $k > 0$ is the number of 1s in the common pattern, is as follows:

$$p_n\#_k = \prod_{i=\pi(k+1)}^{n}(p_i - k) \tag{4}$$

A candidate prime k-tuple is a pattern in a bitstring S_{p_n} that is counted by the function $p_n\#_k$. For each $k > 0$ there may be more than one pattern that is counted, such as when $k = 2$ it counts candidate twin primes, candidate cousin primes, candidate sexy primes, and so on.

5 Density of candidate prime k-tuples and the Twin Prime Constant

The average distance between the centers of two nearest candidate prime k-tuples of size $k > 0$ in bitstring S_{p_n}, denoted as $G_{p_n,k}$, is simply the length of the bitstring divided by the number of occurrences of candidate prime k-tuples of size k.

$$\begin{aligned} G_{p_n,k} &= \frac{p_n\#}{p_n\#_k} \\ &= \frac{\prod_{i=1}^{n} p_i}{\prod_{i=\pi(k+1)}^{n}(p_i - k)} \\ &= \frac{p_{(\pi(k+1)-1)}\# \cdot \prod_{i=\pi(k+1)}^{n} p_i}{\prod_{i=\pi(k+1)}^{n}(p_i - k)} \\ &= p_{(\pi(k+1)-1)}\# \cdot \prod_{i=\pi(k+1)}^{n} \frac{p_i}{p_i - k} \\ &= p_{(\pi(k+1)-1)}\# \cdot \prod_{i=\pi(k+1)}^{n} \frac{1}{1 - \frac{k}{p_i}} \end{aligned} \tag{5}$$

Where π is the prime counting function.

When $k = 1$, i.e. 1-tuples, then $G_{p_n,1}$ equals the average distance between candidate primes in bitstring S_{p_n}.

$$\begin{aligned} G_{p_n,1} &= p_{(\pi(2)-1)}\# \cdot \prod_{i=\pi(2)}^{n} \frac{1}{1 - \frac{1}{p_i}} \\ &= p_0\# \cdot \prod_{i=1}^{n} \frac{1}{1 - \frac{1}{p_i}} \\ &= \prod_{i=1}^{n} \frac{1}{1 - \frac{1}{p_i}} \end{aligned}$$

Note that $G_{p_\infty,1}$ is equivalent to $\zeta(1)$, the pole of the Riemann zeta function ζ, or the harmonic series.

Let $\rho_{p_n,k}$ be the reciprocal of $G_{p_n,k}$, such that $\rho_{p_n,k}$ is a measure for the average density of candidate prime k-tuples in bitstring S_{p_n}.

$$\begin{aligned} \rho_{p_n,k} &= \frac{1}{G_{p_n,k}} \\ &= \frac{p_n\#_k}{p_n\#} \\ &= \frac{1}{p_{(\pi(k+1)-1)}\#} \cdot \prod_{i=\pi(k+1)}^{n} (1 - \frac{k}{p_i}) \end{aligned} \tag{6}$$

The density of candidate prime k-tuples tends to zero as n goes to infinity. For any $k > 0$:

$$\lim_{n\to\infty} \rho_{p_n,k} = \lim_{n\to\infty} \frac{1}{p_{(\pi(k+1)-1)}\#} \cdot \prod_{i=\pi(k+1)}^{n} (1 - \frac{k}{p_i}) = 0$$

$\rho_{p_n,k}$ is never zero because for any value of n there is a slice of candidate primes that is being eliminated from the number line. Consider the distribution of the least prime factors on the natural number line, as in OEIS A020639. Let ρ_{lpf,p_n} be the general density of positive integers that have p_n as its least prime factor. We define ρ_{lpf,p_n} as follows:

$$\begin{aligned} \rho_{lpf,p_n} &= \frac{p_{n-1}\#_1}{p_n\#} = \frac{\phi(p_{n-1}\#)}{p_n\#} \\ &= \prod_{i=1}^{n} \frac{1}{p_i} \cdot \prod_{i=1}^{n-1} (p_i - 1) \\ &= \frac{1}{p_n - 1} \cdot \prod_{i=1}^{n} \frac{p_i - 1}{p_i} \\ &= \frac{1}{p_n - 1} \cdot \prod_{i=1}^{n} (1 - \frac{1}{p_i}) \end{aligned} \tag{7}$$

Every positive integer has exactly one least prime factor; therefore, the sum of densities ρ_{lpf,p_n} over all $n > 0$ converges to 1.

$$\sum_{n=1}^{\infty} \frac{p_{n-1}\#_1}{p_n\#} = \sum_{n=1}^{\infty} \left(\frac{1}{p_n - 1} \cdot \prod_{i=1}^{n} (1 - \frac{1}{p_i}) \right) = 1 \tag{8}$$

Where $p_0\#_1 = 1$.

The expression above shows that, for each iteration of the recurrence relation, a thin slice of the candidate primes is eliminated from the concatenated bitstring. For any iteration, the candidate primes eliminated are the composites with the new prime as its least prime factor. The pattern of eliminations is just a scaled-up version of the pattern of candidate primes in the previous bitstring, and just as symmetric and uniform. We therefore observe the following pattern.

> *During the sieving process, as the candidate primes are gradually and being thinned out, the average distance between nearest candidate prime k-tuples gradually increases, resulting in ever sparser clusters of intact candidate prime k-tuples, of which a deterministic number survive in each iteration.*

The rate of change in average distance between neighboring candidate prime k-tuples depends on k because a candidate prime k-tuple has k opportunities of getting eliminated per iteration. A candidate single prime has one opportunity of getting eliminated per iteration, and a candidate twin prime has two opportunities of getting eliminated per iteration (and never a double elimination in a single iteration). This implies that, per iteration, the rate of change in distance between candidate twin primes is proportional to the rate of change in distance between single candidate primes squared. We can express this as follows.

$$G_{p_n,2} \approx \frac{G^2_{p_n,1}}{2 \cdot C_2}$$

Here C_2 is the Hardy-Littlewood twin prime constant, and the factor of 2 in the denominator is to align with their formulation. Solving for C_2:

$$\begin{aligned}
C_2 &= \lim_{n\to\infty} \frac{1}{2} \cdot G^2_{p_n,1} \cdot \frac{1}{G_{p_n,2}} \\
&= \lim_{n\to\infty} \frac{1}{2} \cdot \left(\frac{p_n\#}{p_n\#_1}\right)^2 \cdot \frac{p_n\#_2}{p_n\#} \\
&= \lim_{n\to\infty} \frac{1}{2} \cdot \left(\prod_{i=1}^{n} \frac{p_i}{p_i - 1}\right)^2 \cdot 2 \cdot \prod_{i=2}^{n} \frac{p_i - 2}{p_i} \\
&\equiv \prod_{i=2}^{\infty} \frac{p_i}{p_i - 1} \cdot \prod_{i=2}^{\infty} \frac{p_i}{p_i - 1} \cdot \prod_{i=2}^{\infty} \frac{p_i - 2}{p_i} \\
&\equiv \prod_{i=2}^{\infty} \frac{p_i}{p_i - 1} \cdot \prod_{i=2}^{\infty} \frac{p_i - 2}{p_i - 1} \\
&\equiv \prod_{i=2}^{\infty} \frac{p_i \cdot (p_i - 2)}{(p_i - 1)^2} \\
&\equiv \prod_{i=2}^{\infty} \left(1 - \frac{1}{(p_i - 1)^2}\right) \\
&\approx 0.6601618\cdots
\end{aligned} \tag{9}$$

6 At the border of candidate prime elimination

The candidate primes (the 1s) in bitstring S_{p_n} after index p_n and before p_n^2, are all prime numbers. These candidate primes are prime because in S_{p_n} the the first 1 that is composite is at index p_{n+1}^2. The next composite after p_{n+1}^2 is at index $(p_{n+1} \cdot p_{n+2})$, followed by either p_{n+2}^2 or $(p_{n+1} \cdot p_{n+3})$. Figure 3 shows the general anatomy of a bitstring.

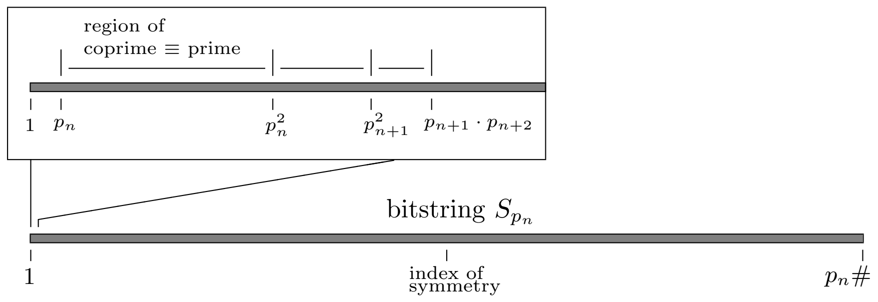


Figure 3: Generic structure of a bitstring S_{p_n}

When the bitstring sieve completes the creation of bitstring S_{p_n}, then from the sieve's perspective, not yet knowing p_{n+1}, the candidate primes between p_n and p_n^2 are *definite primes*, and for any candidate prime beyond p_n^2 it is not yet known whether it is a definite prime or a definite composite. p_n^2 is at the border between the candidate primes and the definite primes, which we call the *border of candidate prime elimination*. When observing the progression of bitstring S_{p_n} as n increases, the border of candidate prime elimination travels with a velocity of p_n^2 along the number line, into a growing structure that is gradually being thinned out throughout. At each iteration, the border of candidate prime elimination passes over a non-empty set of 1s, which in itself proves there are infinitely many primes. In this way, the proof of the infinitude of primes can be written as follows:

(Yet another) proof of the infinitude of primes
The distance between p_n^2 and p_{n+1}^2, or the sequence OEIS A069482, is relatively smallest when p_n and p_{n+1} are twin primes, at which point the distance between p_n^2 and p_{n+1}^2 is $4p_n + 4$, or $4p_{n+1} - 4$. A jump from p_n^2 to p_{n+1}^2 therefore always covers a distance of at least $4p_n + 4$. The largest distance between candidate primes in S_{p_n} is $2p_{n-1}$. Therefore, the number of candidate primes between p_n^2 and p_{n+1}^2 is at least $\frac{4p_n+4}{2p_{n-1}}$, which is at least 2, which is more than 1. $\square$

The largest distance between candidate twin primes is not always smaller than $4p + 4$, so there is no equivalent and easy proof for the twin primes. For example, in S_{17} the largest distance between candidate twin primes (from center to center) is 108, which is greater than $19^2 - 17^2 = 72$. The largest distance between candidate twin primes (from center to center) in S_{p_n} per n is listed in OEIS A144311 (plus 1).

Statistically, however, the average number of candidate twin primes per randomly selected region of size $(p_{n+1}^2 - p_n^2)$ increases with increasing n. In other words, with ever more iterations of the bitstring sieve, we expect ever more candidate twin primes to survive the border of candidate elimination and reach p_n^2. Assuming a uniform distribution of candidate twin primes throughout a bitstring, then an estimate for how many candidate twin primes exist on average in the region between p_n^2 and p_{n+1}^2 is as follows:

$$\pi_2(p_{n+1}^2) - \pi_2(p_n^2) \approx \frac{(p_{n+1}^2 - p_n^2)}{2} \cdot \prod_{i=2}^{n} \frac{p_i - 2}{p_i} \tag{10}$$

Where $\pi_2(x)$ is the actual number of twin primes less than x.
If the distribution of candidate twin primes in the bitstrings is uniform on average, then we expect:

$$\lim_{n\to\infty} \frac{\pi_2(p_{n+1}^2) - \pi_2(p_n^2)}{\frac{(p_{n+1}^2 - p_n^2)}{2} \cdot \prod_{i=2}^{n} \frac{p_i-2}{p_i}} = 1 \tag{11}$$

Let us extend this approach to candidate k-tuples of any size $k > 0$.
If the candidate prime k-tuples of size k are, on average, uniformly distributed throughout bitstring S_{p_n}, then A_{k,p_n}, the average number of candidate prime k-tuples between p_n^2 and p_{n+1}^2, is as follows:

$$\begin{aligned}
\pi_k(p_{n+1}^2) - \pi_k(p_n^2) &\approx A_{k,p_n} \\
A_{k,p_n} &= \frac{p_{n+1}^2 - p_n^2}{G_{p_n,k}} \\
&= \frac{p_{n+1}^2 - p_n^2}{\frac{p_n\#}{p_n\#_k}} \\
&= (p_{n+1}^2 - p_n^2) \cdot \frac{p_n\#_k}{p_n\#} \\
&= \frac{p_{n+1}^2 - p_n^2}{p_{(\pi(k+1)-1)}\#} \cdot \prod_{i=\pi(k+1)}^{n} \frac{p_i - k}{p_i}
\end{aligned} \tag{12}$$

Where $\pi_k(x)$ is the number of prime k-tuples of size k with a center index less than x.

A_{k,p_n} increases with increasing n, albeit slowly for large k. On average, assuming a uniform distribution of candidate prime k-tuples throughout S_{p_n}, the region swept by p_n^2 at each iteration captures ever more candidate prime k-tuples with increasing n. For any prime k-tuple of size $k > 0$:

$$\lim_{n\to\infty} A_{k,p_n} = \lim_{n\to\infty} \left(\frac{p_{n+1}^2 - p_n^2}{p_{(\pi(k+1)-1)}\#} \cdot \prod_{i=\pi(k+1)}^{n} \frac{p_i - k}{p_i} \right) = \infty \tag{13}$$

This result is already very close to a proof of the K-tuple Conjecture. The above result states that if the density of candidate prime k-tuples of size k in the region between p_n^2 and p_{n+1}^2 in bitstring S_{p_n} is, on average and in the long run, representative of its density in the whole bitstring, then the K-tuple Conjecture is true. The only way for the K-tuple Conjecture to be false is if after some large prime there somehow emerges an everlasting wave of bias toward eliminating all candidate prime k-tuples of particular size k just ahead of p_n^2, thus preventing any of them from becoming prime. In order to study whether such phenomena are even possible, let us delve deeper into the elimination process, by rearranging the bitstrings into *bitmatrices*.

7 The Bitmatrix Sieve

A bitstring S_{p_n} of length $p_n\#$ can be shaped into a $p_n \times p_{n-1}\#$ matrix. Such a matrix we call a *bitmatrix*.

The *bitmatrix sieve* is defined as follows. Let M be the set of outputs generated by the bitmatrix sieve. For each prime p_n there is a matrix M_{p_n} in M:

$$M = \{M_2, M_3, M_5, M_7, M_{11}, \dots\}$$

Each matrix M_{p_n} is a *bitmatrix*, a matrix of binary digits. The referencing of entries in the bitmatrix is by a single 1-based index, where $index = column + ((row - 1) \times width)$. For example, in bitmatrix M_3 with 6 columns, bit b_7 at $M[7]$ refers to the first bit (column 1) in the second row. The format of a bitmatrix is:

$$\begin{bmatrix} b_1 & \cdots & b_{columns} \\ \cdots & \cdots & \cdots \\ b_{((rows-1)\times columns)+1} & \cdots & b_{rows\times columns} \end{bmatrix}$$

Where $b_i \in \{0, 1\}$.

The recurrence relation that generates the set M is as follows.

Initial condition:
Let M_2 be a 2×1 bitmatrix.

$$\mathbf{M_2} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Recurrence relation:
Given bitmatrix M_{p_n}, where p_n is the n-th prime, the next bitmatrix $M_{p_{n+1}}$ is obtained by:

- In bitmatrix M_{p_n}, starting after index 1, locate the index of the next occurrence of bit value 1. Let p_{n+1} be this index.
- Let $M_{p_{n+1}}$ be a $p_{n+1} \times p_n\#$ bitmatrix. The contents of $M_{p_{n+1}}$ is filled as follows.
  - Fill each row in $M_{p_{n+1}}$ with a flattened copy of M_{p_n}. To flatten is to reshape the matrix such that all rows are concatenated to form a single row (effectively forming bitstring S_{p_n}).
  - For each column in $M_{p_{n+1}}$ that is filled with 1s, zero the entry that has an index that is divisible by p_{n+1}.

Equivalently, bitmatrix M_{p_n} can be defined as:

$$\mathbf{M_{p_n}} = \begin{bmatrix} b_1 & ... & b_{p_{n-1}\#} \\ ... & ... & ... \\ b_{((p_n-1)\times p_{n-1}\#)+1} & ... & b_{p_n\#} \end{bmatrix}$$

Where:

$$b_i = \begin{cases} 1, & \text{if } i \text{ is coprime with } p_n\# \\ 0, & \text{otherwise} \end{cases}$$

The bitmatrices M_5 and M_7 are shown in Figure 4.

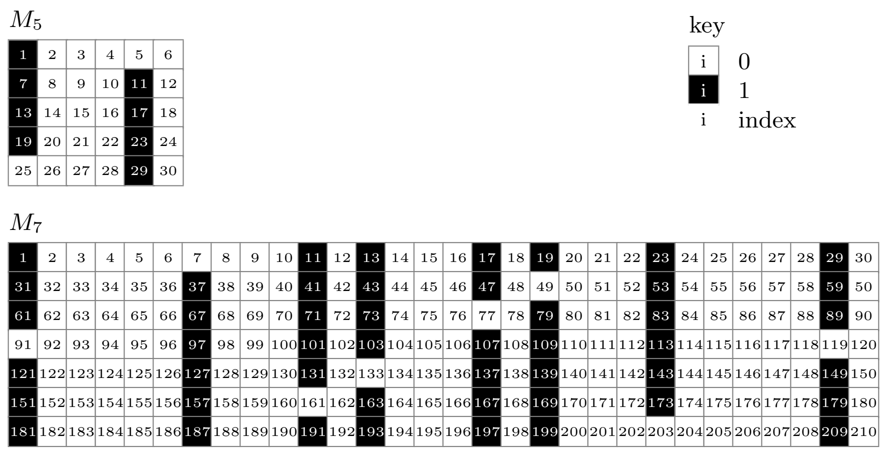


Figure 4: Bitmatrices M_5 and M_7

Zooming out, we can just about get bitmatrix M_{11} in full view, as shown in Figure 5.

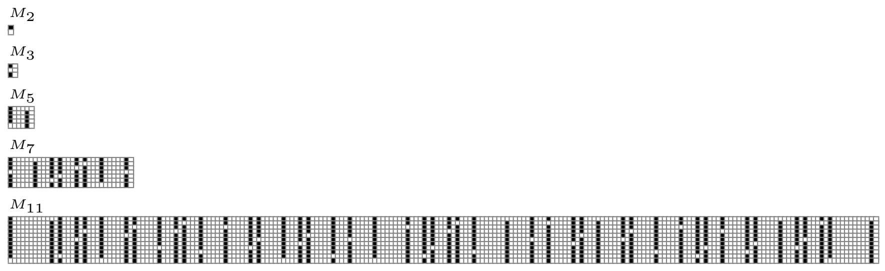


Figure 5: Bitmatrices M_2 - M_{11}

A bitmatrix's index of symmetry is halfway its bitstring length. For example, the index of symmetry of M_7 is at index $\frac{7\#}{2} = 105$. A bitmatrix is centrosymmetric, meaning that pairs of entries that are on opposite sides of the index of symmetry, i.e. having indices that add up to $p\#$, always have the same bit value. A bitmatrix has an even number of columns, and its index of symmetry lies half a column width to the left of its geometric center.

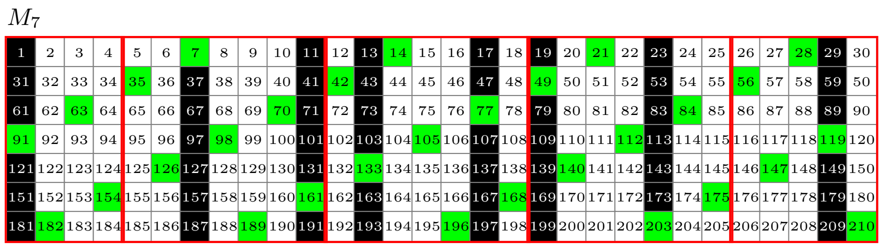
In this matrix format, the candidate primes (black squares) are arranged in columns. Each black column has exactly one white square, because exactly one of these indices will be divisible by prime p_n. These single white squares per black column is the process of eliminating candidate primes in action. The multiples of p_n are distributed as diagonal lines across the table because the width of the table is not divisible by its height.

8 Residue Systems and Elimination Masks

As is visible in Figure 4 and Figure 5, the candidate primes (black squares) are grouped in *black columns.* In bitmatrix M_{p_n} there are $p_{n-1}\#_1$ such black columns. Each black column has exactly one white square because the indices in each column of M_{p_n} form a complete residue system $\pmod{p_n}$, such that each column has exactly one entry that is divisible by p_n. Furthermore, any horizontal sequence of p_n entries also form a complete residue system $\pmod{p_n}$. Therefore, any $p_n \times p_n$ section of M_{p_n} contains a set of all rotations of the complete residue system $\pmod{p_n}$. We can therefore interpret the elimination process as a $p_n \times p_n$ *elimination mask* being applied sequentially along the matrix.

In bitmatrix M_{p_n}, the elimination mask is applied $\frac{p_{n-1}\#}{p_n}$ many times, which is never a whole number, leaving a relatively small fractional part of $\frac{p_n \bmod p_{n-1}\#}{p_n}$. The elimination masks are center-aligned around the index of symmetry. The eliminations are therefore centrosymmetrically placed around the bitmatrix's index of symmetry. Note that the elimination mask is similar to the *NOT-STRETCH* operation in the bitstring sieve, except that it will eliminate numbers that were already marked as composite, whereas the *NOT-STRETCH* operation does not.

Figure 6 shows the E_7 elimination masks highlighted in green in bitmatrix M_7. Red borders are drawn around each elimination mask.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	50
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210

Figure 6: M_7 with E_7 elimination masks (i.e. multiples of 7) highlighted in green

The elimination mask E_{p_n} for M_{p_n} is a $p_n \times p_n$ bitmatrix, defined as:

$$\mathbf{E_{p_n}} = \begin{bmatrix} b_1 & ... & b_{p_n} \\ ... & ... & ... \\ b_{((p_n-1)\times p_n)+1} & ... & b_{p_n \times p_n} \end{bmatrix}$$

Where:

$$b_i = \begin{cases} 0, & \text{if } T_{p_n}(i) \text{ is divisible by } p_n \\ 1, & \text{otherwise} \end{cases}$$

Where:

$$T_{p_n}(i) = \frac{p_{n-1}\#}{2} - \frac{p_n - 1}{2} + ((i-1) \bmod p_n) + \left\lfloor \frac{i-1}{p_n} \right\rfloor \cdot p_{n-1}\#$$

9 The candidate prime eliminator

The candidate prime eliminator (the green squares in Figure 6) can be described as a diagonal line wrapping as a coil around a cylinder. A bitmatrix is modular in horizontal and vertical directions, such that its ends can be joined together in two ways to form a cylinder, either by joining the horizontal ends, or by joining the vertical ends. In this way, a bitmatrix is a torus. There are two ways of wrapping the coil of elimination around the bitmatrix and end up with the same result, either by $\pmod{p_{n-1}\#}$ or by $\pmod{p_n}$. We can either wrap around the bitmatrix in horizontal direction while stepping down, or wrap around vertically while stepping right. The first option rotates around the cylinder with a period of $\pmod{p_{n-1}\#}$ steps, while the second option rotates around the cylinder with a period of $\pmod{p_n}$ steps. We shall focus on the second option, because we are interested in an expression for determining which row in a given black column is eliminated in the next iteration.

The right side of Figure 7 shows an illustration of the candidate prime eliminator of M_7. The other side of the cylinder (within the same torus), shown on the left, hosts the candidate primes (before the elimination step). The coordinate systems of the two cylinders are inverses of each other, and transforming one into the other is akin to turning a punctured torus inside out, whilst ensuring the symmetries are maintained.

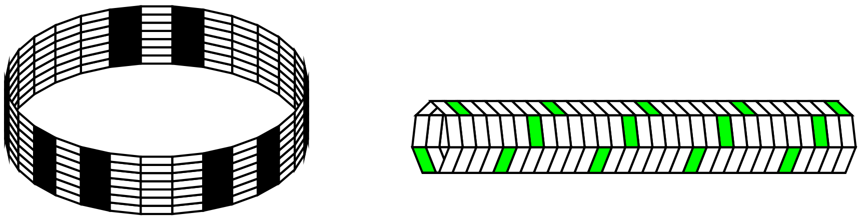


Figure 7: The two cylinders in the torus of M_7. The left cylinder contains the candidate primes (before the elimination step), and the right cylinder contains the candidate prime eliminator.

Let J_{p_n} be a function that returns the increment in row index modulo p_n per increment in column index of bitmatrix M_{p_n}. We know that J_{p_n} is an integer greater than 0 and less than p_n. The congruence relations are as follows.

$$T_{p_n}(\frac{p_n^2}{2}+1) \equiv 0 \pmod{p_n}$$
$$T_{p_n}(\frac{p_n^2}{2}+1)+1+J_{p_n}\cdot p_{n-1}\# \equiv 0 \pmod{p_n}$$

Therefore:

$$1+J_{p_n}\cdot p_{n-1}\# \equiv 0 \pmod{p_n} \tag{14}$$

The candidate prime eliminator in bitmatrix M_{p_n} can be modeled as a rotating object that passes from left to right over the bitmatrix, rotating with a frequency of $\frac{2\cdot\pi\cdot J_{p_n}}{p_n}$ radians per column shift. J_{p_n} can be any integer value greater than 0 and less than p_n, any choice will ensure a periodic visit to each row per shift in p_n columns, but J_{p_n} is the only integer greater than 0 and less than p_n such that the eliminator passes through both p_n and the index of symmetry.

J_{p_n} answers the question: how many times to add $p_{n-1}\#$ to p_n for it to be divisible by p_n. To calculate J_{p_n} requires finding the modular inverse of $p_{n-1}\#$ modulo p_n. Finding the modular inverse requires knowing the specific numbers of the congruence relations, implying there is no simple or direct formula for calculating J_{p_n}. It is a puzzle that you can only start to attempt solving after first knowing $p_{n-1}\#$. When $p_{n-1}\#$ and p_n are both known, J_{p_n} can be calculated by an algorithm, such as by sieving, or by exhaustively searching by adding and checking divisibility, or by the Extended Euclidean Algorithm.

When J_{p_n} is known, we have a formula for $R_{p_n}(c)$, the "row index of elimination" in M_{p_n}, for any given column index $c \geq 1, c \leq p_{n-1}\#$.

$$R_{p_n}(c) = 1 + ((J_{p_n} \cdot c) \bmod p_n)$$

Values of J_p for the first 6 primes are:

$$J_2 = 1$$
$$J_3 = 1$$
$$J_5 = 4$$
$$J_7 = 3$$
$$J_{11} = 10$$
$$J_{13} = 10$$

The sequence of J_{p_n} per n is listed in A081617.

In the long run, with ever increasing n, the distribution of J_{p_n} over n is statistically the same as the distribution of throwing $(p_n - 1)$-sided dice, because, by definition, primes do not divide each other. The distribution of J_{p_n} is stochastic, such that:

$$\lim_{n\to\infty} \frac{\sum_{q=1}^{n} \frac{J_{p_q}}{p_q}}{n} = \frac{1}{2} \tag{15}$$

On average, especially so when p_n is large, each row in a bitmatrix will have near-equal numbers of eliminations, implying that there cannot emerge a sustained bias for near-future eliminations over far-future eliminations, and therefore it is impossible for some everlasting rogue wave of bias to appear after some large prime, that somehow purposely eliminates all candidate prime k-tuples of chosen size $k > 0$ just ahead of p_n^2. Furthermore, even if the values of J_{p_n} were not stochastic, its impact is not enough to create any significant bias. The value of J_{p_n}, particularly when n is large, has very little impact on the distribution of eliminations per row in the bitmatrix. With increasing n, whatever the behavior is of J_{p_n}, the number of eliminations in each row approaches the average value of $\frac{p_{n-1}\#_1}{p_n}$.

To illustrate this, imagine there is a demon in the sieve that manipulates the values of J_{p_n}, in an attempt to eliminate the candidate prime k-tuples of size k that lie ahead of p_n^2, by targeting more prime k-tuples of size k in the first row than in the other rows. The demon will find that, particularly when n is large, the number of eliminations per row remains nearly the same for each setting of J. Table 1 shows the number of candidate primes eliminated per row in bitmatrix M_{11}, for all possible manipulations of J. The column $J_{11} = 10$ represents the actual value for J_{11}. Notice that, in this model, manipulating J has no impact on the number of eliminations in the first row, only from the second row onward.

		manipulated J									J_{11}
		1	2	3	4	5	6	7	8	9	10
row	1	4	4	4	4	4	4	4	4	4	4
	2	4	5	6	3	5	5	4	4	3	4
	3	5	4	4	5	4	6	3	3	4	5
	4	3	4	4	5	3	5	5	4	6	4
	5	6	5	3	4	3	4	4	5	5	4
	6	3	4	5	6	4	4	4	5	3	5
	7	5	3	5	4	4	4	6	5	4	3
	8	4	5	5	4	4	3	4	3	5	6
	9	4	6	4	5	5	3	5	4	4	3
	10	5	4	3	3	6	4	5	4	4	5
	11	4	3	4	4	5	5	3	6	5	4

Table 1: Number of candidate prime eliminations per row in M_{11} per manipulation of J. Actual value of $J_{11} = 10$.

In the above table for bitmatrix M_{11}, with 11 being a low prime, the differences in the number of eliminations per row are still relatively large compared to the average value of $4\frac{4}{11}$, but in the tables for larger primes the differences in the number of eliminations per row get relatively smaller, with all rows gradually approaching the average value of $\frac{p_{n-1}\#_1}{p_n}$ as n increases. For example, in M_{19} the average number of eliminations per row is $4850\frac{10}{19}$, and the actual values range from 4846 to 4854.

We summarize this as follows. Let $E_{p_n,row(r)}$ be the number of eliminations in row index $r \in \mathbb{N}; r \geq 1, r \leq p_n$ in bitmatrix M_{p_n}. Then, as n increases, the number of eliminations in each row converges to the same value of:

$$\begin{aligned} \lim_{n\to\infty} E_{p_n,row(x)} &= \frac{p_{n-1}\#_1}{p_n} \\ &= \frac{1}{p_n}\prod_{i=1}^{n-1}(p_i - 1) \end{aligned} \tag{16}$$

The numbers which have p_n as its least prime factor are uniformly distributed at the primorial scale, as is evident in the sequence of least prime factors. This rules out any possibility for a sustained local phenomenon to appear after some large prime. This means that the density of a pattern of 1s in bitstring S_{p_n} just ahead of p_n^2 is statistically representative of the density of the same pattern throughout the entire bitstring. Therefore, we can say for sure that the number of prime k-tuples of size k between p_n^2 and p_{n+1}^2 increases on average with increasing n. There are infinitely many prime k-tuples, which are distributed as statistically predicted by Hardy and Littlewood.

10 Proof of the Hardy-Littlewood K-tuple Conjecture

In the recurrence relation that defines the primes, such as in the bitstring sieve or bitmatrix sieve, symmetry and modularity of the candidate primes (coprimes with the primorial) is conserved at the primorial scale. During the process of sieving, when the candidate primes are gradually and uniformly (uniform at the primorial scale) being thinned out, the average distance between nearest candidate prime k-tuples gradually increases, resulting clusters of intact candidate prime k-tuples, of which a deterministic number survive into the next iteration. The "border of candidate prime elimination" passes into this structure with a "speed" of p_n^2 (relative to the natural number line), passing over gradually increasing numbers of candidate prime k-tuples.

A_{k,p_n}, the average number of candidate prime k-tuples of size k in S_{p_n} between p_n^2 and p_{n+1}^2, increases with increasing n.

$$\lim_{n\to\infty} A_{k,p_n} = \lim_{n\to\infty}\left(\frac{p_{n+1}^2 - p_n^2}{p_{(\pi(k+1)-1)}\#}\cdot\prod_{i=\pi(k+1)}^{n}\frac{p_i - k}{p_i}\right) = \infty$$

As n increases, the area between p_n^2 and p_{n+1}^2 sweeps up ever more expected candidate k-tuples. The distribution of primes in the sequence of least prime factors, which account for the eliminations, are also uniform at the primorial scale. The creation process and elimination process are both symmetric and uniform at the primorial scale, and the creation process is always one step ahead of the elimination process.

The symmetries maintained by the recurrence relation that defines the primes prohibit any possibility of a sustained local phenomenon, such as a rogue "bow wave" of elimination, to somehow emerge ahead of p_n^2 after some large n. Such sustained local phenomena are guaranteed not to happen, guaranteed by the fact that primes do not divide each other. In the recurrence relation that defines the primes there is no mechanism to target and stop all candidate k-tuples of size k from becoming prime at p_n^2.

11 Conclusion

In the recurrence relation that defines the primes there is no mechanism to target and stop all candidate k-tuples of size k from becoming definite primes at p_n^2. There exist infinitely many twin primes and prime k-tuples, occurring at asymptotic frequency, as predicted by Hardy and Littlewood.

References

[1] G. H. Hardy and J. E. Littlewood. “Some problems of ‘Partitio numerorum’; III: On the expression of a number as a sum of primes”. In: *Acta Mathematica* 44 (1923), pp. 1–70. DOI: 10.1007/BF02403921. URL: https://doi.org/10.1007/BF02403921.

[2] Pete Quinn. *Newbie Question - Sieving based on primorial patterns and symmetry*. 2011. URL: https://www.primegrid.com/forum_thread.php?id=2991.

[3] Dennis R. Martin. *Proofs Regarding Primorial Patterns*. 2006. URL: https://oeis.org/A005867/a005867.pdf.

[4] Fred B. Holt. *Patterns among the Primes: A study of Eratosthenes sieve*. Kindle Direct Publishing, 2022. ISBN: 9798831607314. URL: https://www.amazon.com/dp/B0B2TY72XJ/.