

Proof of Goldbach's Conjecture and Bertrand's Postulate

Using Prime Generator Theory (PGT)

Jabari Zakiya – jzakiya@gmail.com

February 1, 2025

Abstract

Goldbach's Conjecture states every even integer $n > 2$ can be written as the sum of 2 primes, while *Bertrand's Postulate* states for each $n \geq 2$ there is at least one prime p such that $n < p < 2n$. I show both are essentially statements on the primes distribution, and their inherent properties when modeled and understood as the residues of modular groups \mathbb{Z}_n . In addition, a much tighter dynamic bound on p than given by the *BP* will be presented.

Introduction

In 1742, the Prussian mathematician Christian Goldbach wrote a series of letters to the renowned Swiss mathematician Leonhard Euler, stating that every even number $n > 2$ can be written as the sum of prime numbers. In Goldbach's time '1' was considered a prime, so some of his conjectures varied between the sum of 2 or more primes when using it. Its modern statement is: **every even number > 2 can be written as the sum of 2 primes** (not necessarily unique). It's been computer verified to hold for all even n up to 4×10^{18} [1].

In 1845, 23 year old French mathematician Joseph Bertrand stated: **for each $n \geq 2$, there is a prime p such that $n < p < 2n$** . Its 1852 proof made it a fundamental theorem of *Prime Number Theory*.

Prime Generator Theory (PGT) will be the basis of the mathematical framework used to formalize the properties of modular groups \mathbb{Z}_n over the even integers $n > 2$. I provide the minimum necessary axioms and properties here for the purpose of establishing the validity of the *GC*. More extensive expositions of the theory and applications of *PGT* can be found in my papers on the *Twin Prime Conjecture* [2], and my *Twin Primes Segmented Sieve* algorithm and code [3].

I show *Goldbach's Conjecture (GC)* is really a (weak) statement about the inherent properties of primes when modeled and understood as the residues of modular groups \mathbb{Z}_n . I also show *Bertrand's Postulate (BP)* falls directly out of it as a necessary (weaker) condition for the *GC* to be true. You can't have the former without the later. I also present a much tighter, and dynamic, bound on p than given by the *BP*. But both are fundamentally statements on the distribution of the primes, *which do not exist randomly!*

The techniques used here will be "modern", in the sense they don't use or need the classical conceptual and numerical approaches used previously to prove the *BP*, and in the many unsuccessful attempts to prove the *GC*. Instead it will use the conceptual framework, logic, math and properties of the **residues of modular groups** \mathbb{Z}_n over the even integers $n > 2$. In fact, most of what needs to be understood is purely conceptual in nature, requiring very little number crunching. Finally, I present a simple algorithm to identify all the prime pairs for any n , its implementation in software, and empirical data.

Modular Group Residues

Prime Generator Theory is the understanding of prime numbers, their distribution, characteristics, gap structures, and other relationships, thru the properties of modular groups residues. For even values of n , it uses the **residues** of the modular groups \mathbb{Z}_n to construct **Prime Generators**, which efficiently generate every prime that is not a factor of n .

A modular group \mathbb{Z}_n is constructed from the integers $\{0 \dots n-1\}$, with n the group's **modulus**. A **Prime Generator** is constructed from its **residues**, the r_i odd values coprime to n e.g. $\gcd(r_i, n) = 1$. Their r_i values are the: 1) coprime primes $p_j < n$, 2) their powers $p_j^e < n$, or 3) their cross-products $r_k = r_i \cdot r_j < n$. The number of r_i residues in \mathbb{Z}_n is an even value given by $\varphi(n)$, the **Euler Totient Function (ETF)**.

$$\varphi(n) = n \prod_{i=1}^j \frac{(p_i - 1)}{p_i} \quad (1)$$

The p_i are the j unique prime factors of n . Thus for $n = 12$: $\varphi(12) = 12 \frac{(2-1)}{2} \frac{(3-1)}{3} = 4$

We can easily identify the 4 residues with a **greatest common divisor (gcd)** of 1 in \mathbb{Z}_{12} , as: $\{1, 5, 7, 11\}$. Thus the **canonical residues** of \mathbb{Z}_n are '1' and the set of odd integer values $< n$ that are not multiples of its prime factors. Every prime coprime to n exists as a congruent modular value to a residue of \mathbb{Z}_n .

To identify the residues values of n we do not need to know its prime factors. We can simply do a $\gcd(r_i, n) = 1$ test on the odd integers in the half interval $[1, n/2)$ to identify its $\varphi(n)/2$ residue values. These are the **low-half-residues (lhr)** values. The $\varphi(n)/2$ **high-half-residues (hhr)** are then just their **modular complements**.

Modular Complement Properties

A very important property of modular residues is that they exist as **modular complement pairs (mcp)**. For a modulus n , each r_i residue has a modular complement r_i^c , adhering to these 2 modular properties:

$$r_i^c = n - r_i \quad (2)$$

$$r_i + r_i^c = n \quad (3)$$

Conceptually, a modular complement for an r_i is its **modular reflection** into its opposite half residues.

Looking at the residues $\{1, 5, 7, 11\}$ for \mathbb{Z}_{12} , when split into their $lhr||hhr$ values, and starting from the middle 6, we see $\{5, 7\}$ and $\{1, 11\}$ are its 2 mcp , which each sum to 12. **Thus to identify the $\varphi(n)$ residue values of n , we can just find the $\varphi(n)/2$ values $< n/2$, and their $n-r_i$ complements are the rest.**

Also, for every even modulus $n > 2$, the integers pair $\{1, n-1\}$ are automatic mcp residues. For $n > 6$ the closest odd integers less|greater than the midpoint $n/2$ are also mcp residues. For \mathbb{Z}_{12} , $\{1, 11\}$ are mcp of the first type and $\{5, 7\}$ the latter. Thus we automatically know 4 residues, and 2 mcp , for $n > 6$.

Mirror Image Symmetry

An analog clock is the simplest form to show the modular complement properties of residues. For the group size \mathbb{Z}_{12} , we see the parallel horizontal lines connect pairs of hours on the left|right halves of the clock, and each pair sums to the group size 12. This is true for any even group size representation > 2 .

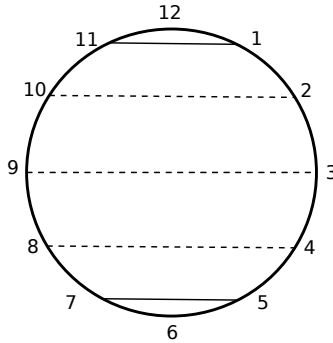


Figure 1.

Here the solid lines connect the two modular complement pair residues $\{1, 11\}$ and $\{5, 7\}$. As the prime factors of 12 are 2 and 3, the dashed lines connect their non-residue composite values, which occur as **modular composite factors pairs**. For $p = 2$, its composite complements must be a multiple of itself to have an even sum, but for $p = 3$ (or any odd prime factor), it must be an odd multiple of itself to have two odd numbers sum to an even n . Thus each r_i must have either an odd prime residue, a prime residue power, or an odd composite product of prime residues, as their modular complements.

For the *lhr* $\{1, 5\}$, $r_i \equiv r_i \pmod n$, but the *hhr* $\{7, 11\}$, can be thought of and used as, $n - r_i \equiv -r_i \pmod n$. We can thus write, use, and conceptually understand the *hhr* as the negative residue values of the *lhr*. Thus conceptually for the *lhr* values $0 < r_i < n/2$, their $-r_i$ equivalents are their **modular reflections** into their *hhr* complements $n/2 < -r_i < n$.

This is a modular arithmetic property analogous to the orientation of angles in trigonometry. Positive angles are defined as counter clockwise (CCW) around the clock, while clockwise (CW) angles are considered as negative. Here the CW residue values are considered positive, and the same units of values CCW around the clock can be referred to as negative values $\pmod n$. Thus if r_i CW units is a right side residue value, then the same units CCW on the left side are residue values too.

This creates a conceptually and arithmetically consistent structure for performing modular math. Thus for \mathbb{Z}_{12} , its *lhr* are $1 \equiv 1 \pmod{12}$ and $5 \equiv 5 \pmod{12}$, while the *hhr* are: $(12 - 1) = 11 \equiv -1 \pmod{12}$ and $(12 - 5) = 7 \equiv -5 \pmod{12}$. And as $n \equiv 0 \pmod n$, we have these conceptually simple relationships:

$$r_i^c = n - r_i \equiv -r_i \pmod n \quad (4)$$

$$r_i + r_i^c = r_i + -r_i = n \equiv 0 \pmod n \quad (5)$$

Figure 1. is a visual metaphor of the symmetry properties of every even \mathbb{Z}_n modular group. Thus we will see *Goldbach's Conjecture* can be restated and proven as the properties of the \mathbb{Z}_n residues and their modular complement pair relationships. Thus from the residue properties of even \mathbb{Z}_n modular groups we will see there is at least one *mcp*, consisting of primes in each half, that sum to every even $n > 6$.

Definitions and Axioms

Here I define the terminology and establish the axiomatic mathematical foundation that will be used.

Definitions

Definition 1: A modular group \mathbb{Z}_n , with modulus n an even integer, is the set of integers $\{0..n-1\}$.

Definition 2: A residue of \mathbb{Z}_n is an integer $r_i < n$ coprime to n , i.e. an integer $r_i < n$ e.g. $\gcd(r_i, n) = 1$.

Definition 3: A modular complement (mc) of an r_i , i.e. r_i^c , is the residue value r_j e.g. $r_j = r_i^c = n - r_i$.

Definition 4: A modular complement pair (mcp) of \mathbb{Z}_n are two residues that sum to n .

Definition 5: A prime complement pair (pcp) of \mathbb{Z}_n is a modular complement pair of two primes.

Definition 6: A trivial prime pair (tpp) is a single prime added to itself, e.g. $n = 2p = p + p$.

Definition 7: The low-half-residues (lhr) are the set of residues in the interval $0 < \{r_{lo}\} < n/2$.

Definition 8: The high-half-residues (hhr) are the set of residues in the interval $n/2 < \{r_{hi}\} < n$.

Axioms

Axiom 1: The number of residues of \mathbb{Z}_n is determined by $\varphi(n)$, the Euler Totient Function (ETF).

Axiom 2: The number of residues is even, as each odd prime factor p_i of n contributes $(p_i - 1)$ to $\varphi(n)$.

Axiom 3: The set of the $\varphi(n)$ residue values $\{r_i\}$ consists of the odd values in $\{1..n-1\}$ coprime to n .

Axiom 4: The set of $\{r_i\}$ residues contain all the coprime primes $< n$.

Axiom 5: The powers of a residue $r_i < n$, i.e. its $r_i^e < n$, are also residues, as they too are coprime to n .

Axiom 6: The products $< n$ of two residues is also a residue, i.e. $r_k = r_i \cdot r_j < n$ is also coprime to n .

Axiom 7: The two residues of an mcp|pcp consist of an $r_i < n/2$ and its complement $r_i^c = n - r_i > n/2$.

Axiom 8: Each residue can only be the modular complement to one residue in its opposite half.

Axiom 9: There are $\varphi(n)/2$ modular complement pairs for each \mathbb{Z}_n modular group.

Goldbach's Conjecture can be restated within the framework of *Prime Generator Theory (PGT)* as:
every even integer $n > 6$ has for its modular group \mathbb{Z}_n at least one prime complement pair (pcp).

Integers 4 and 6 have form $n = 2p$, and thus single *trivial prime pairs*, $4 = 2 + 2$ and $6 = 3 + 3$, to satisfy the general GC for $n > 2$. Every other integer of form $n = 2p$ (of the infinitely many) have a similar *tpp*, in addition to at least one *pcp*. The proof, algorithm, code, and data presented herein ignores them.

PGT Proof of Goldbach's Conjecture

A minimal proof need only show there is at least one *pcp* prime pair that sums to n for all even $n > 6$. This will be established using the residue mirror symmetry|reflection properties of modular groups \mathbb{Z}_n . It will be proved, a simply crafted bounded *hhr* region close to n always contains at least one prime p whose modular complement is some small p_m *lhr* prime, which form a $\{p_m, n - p_m\}$ *pcp* prime pair.

Residue Properties

Lemma 1. For residues $r_i < n/2$, their modular complements $n - r_i$ cannot be multiples of themselves.

Proof: Assume $n - r_i = k \cdot r_i$. Then $k = (n - r_i)/r_i = n/r_i - 1$, requiring n/r_i be an even integer, and r_i a factor of n . But r_i is coprime to n , and thus cannot be a factor of n , thus this is a contradiction.

Therefore the modular complements $n - r_i$ for the residues $r_i < n/2$ cannot be multiples of themselves.

Lemma 2. For residues $r_i < n/2$, their modular complements $n - r_i$ cannot be powers of themselves.

Proof: Assume $n - r_i = r_i^e$. Thus $n = r_i^e + r_i = r_i(r_i^{e-1} + 1)$, giving $n/r_i = (r_i^{e-1} + 1)$ for $e \geq 2$. Then n/r_i must be an integer, requiring r_i be a factor of n . But r_i is coprime to n , thus this is a contradiction.

Therefore the modular complements $n - r_i$ for the residues $r_i < n/2$ cannot be powers of themselves.

Corollary 1.

For the $\varphi(n)/2$ *lhr* $r_i < n/2$, their *hhr* modular complements $n - r_i$ can only be either:

1) a prime p_r , 2) a prime power p_r^e , or 3) the cross-product $r_j \cdot r_k$ of residues not including themselves.

Define $r_0 = 1$ as the first canonical residue of n .

Define $r_1 = p_i$ as the first|smallest prime not a factor of n .

Lemma 3. The set of residues $\{r_i\} < r_i^2$, i.e. in $r_1 \leq \{r_i\} < r_1^2$, are the coprime primes $< r_1^2$.

Proof: As r_1 is the first prime residue, r_1^2 is the first multiple of any residue in \mathbb{Z}_n .

Thus, every residue in $1 < r_k < r_1^2$ cannot be a multiple of r_1 , or any other residue, thus is a prime $< r_1^2$.

If $r_1 = 3$, then $\{5, 7\} < 9$ are the next ordered residues if not factors of n . If $\{5, 7\}$ are factors, the next residue is $r_2 = r_1^2 = 9$ if $< n$, because 11 is the next larger prime. In this case, if $r_3 = 11$, then all the coprime primes residues $\{r_i\} < (r_3^2 = 121)$ are also primes $< n$. And we continue in this manner.

We can extend this for all $r_m < n/2$, to establish all the coprime prime $p_r \leq \{p_m\} < p_r^2$ are residues to n . The residues also contain all the powers of the prime residues p_r e.g. $\{p_r, p_r^2, \dots\} < n$, in addition to the residue cross-products $r_j \cdot r_k < n$. However, once for some r_n (*nth* residue), $r_n^2 > n$, no larger *lhr* r_{n+1} prime power, or cross-product, can be a residue $< n$. Then for r_{n-1} (previous residue), r_{n-1}^2 is the largest square that can be a *hhr* $< n$. Let's designate this residue as: $r_{max} = r_{n-1}$ for some *lhr* $r_i < n/2$.

Thus the *hhr* in $r_{max}^2 < \{p_r || r_{cp}\} < n$ can only be p_r primes or cross-products of the type $r_{cp} = r_{max-} \cdot r_{max+}$, with $r_{max-} \leq r_{max}$ and $r_{max+} > r_{max}$. Thus no larger *lhr* power can be a residue e.g. $r_{max}^2 < r_i^e < n$, for $e \geq 2$.

When n is small, only primes exist in the region. As n increases, cross-products, and more primes, will populate it, as a function of the prime factorization of n . When the cross-products and prime powers are filtered out, only the primes in both halves are left, and form the *pcp* prime pairs for even $n > 6$.

Optimum Bounded Regions

The optimum bounded *hhr* region that contains prime residues that always form at least one *pcp* is:

$$r_{max}^2 < \{p_r\} < n \quad (6)$$

It **dynamically adjusts** to create the smallest bounded *hhr* region of *pcp* primes, and has generic form:

$$\text{LowBound} < \{p_i\} < \text{HighBound} \quad (7)$$

High Bound Limit

Normally the *hhr* high bound is just n . But we want the smallest region a prime *hhr* can exist within.

Because $n - 1$ is the modular complement for $r_0 = 1$, whether its prime or not, it cannot be part of a *pcp*. We thus set the *hhr* high bound to $n - 2$, the next smaller even value from n . Thus any *hhr* $r_i < n - 2$ can possibly be a prime *hhr* that can form a *pcp* with a small *lhr* prime.

Low Bound Limit

From the low bound of (6), the largest *lhr* value whose square satisfies $r_{max}^2 < n - 2$, is given by:

$$r_{max} = \text{isqrt}(n - 2) \quad (8)$$

where $\text{isqrt}(x)$ is the integer square root of x . Ex: $\text{isqrt}(8) = 2$, $\text{isqrt}(9) = 3$, $\text{isqrt}(15) = 3$, $\text{isqrt}(16) = 4$.

Applying Modular Reflection

We now know that r_{max} sets the limit as to how large a *lhr* can be; an odd *lhr* value e.g. $r_i \leq r_{max} < n/2$. Because r_{max} is computed 2 *hhr* units below n , its *lhr* region high bound reflection must do so too:

$$\text{HighBound}_{lhr} = r_{max} + 2 \quad (9)$$

We now determine its *hhr* modular (negative) reflection. Starting at the top of the Figure 1. clock, the equivalent region extends down r_{max} CCW units. Again, because we computed the value from $n - 2$, and not n , the *hhr* region extends r_{max} units below it. Thus the *hhr* region low bound is:

$$\text{LowBound}_{hhr} = (n - 2) - r_{max} \quad (10)$$

Thus the smallest bounded *hhr* region which a prime residue exists within whose *lhr mc* forms a *pcp* is:

$$(n - 2) - r_{max} \leq p_r < n - 2 \quad (11)$$

We use $\leq p_r$ vs $< p_r$ because LowBound can be an odd|prime value, while odd $p_r < n - 2$ is always true. The equivalent *lhr* bounded region is then:

$$2 < p_r \leq r_{max} + 2 \quad (12)$$

We use $p_r \leq r_{max}$ vs $p_r < r_{max}$ here too, as r_{max} can be odd, making the right side odd.

These bounded regions produce a *pcp* prime pair for all even $n > 6$. As n increases, and its factorization includes more and larger primes, other *pcp* prime pairs will be created, whose total will greatly exceed those coming from them alone. However, the analysis of these regions alone proves the *GC's* validity. In fact, these optimally adjusting dynamic bounds have implications far beyond *Goldbach's Conjecture*.

Bounded Bonding of Primes

Shown again are the parameters of the *lhr* and *hhr* bounded regions that produce at least one *pcp*.

$$\begin{array}{ccc}
 r_{max} & hhr \text{ bounds} & lhr \text{ bounds} \\
 isqrt(n-2) & (n-2) - r_{max} \leq p_r < n-2 & 2 < p_r \leq r_{max} + 2
 \end{array} \quad (13)$$

The modular complement and symmetry relationships between the residues are inherent and immutable properties of \mathbb{Z}_n modular groups. These properties define the distribution of the residues, which means they define the distribution of the primes, whose values and relationships constitute them.

Let's start with the integers 4 and 6. They have $\phi(2||4) = 2$ residues, $\{1, 3\}$ and $\{1, 5\}$, which are their single *mcp*. In Goldbach's time they would have been considered a prime pair by some.

Looking at $8 = 2^3$, its residues are all the primes < 8 , $\{1, 3, 5, 7\}$, whose *mcp* are $\{1, 7\}$ and $\{3, 5\}$. Here $r_{max} = 2$, giving the bounded regions $4 \leq 5 < 6$ and $2 < 3 \leq 4$. For 10 it's, $6 \leq 7 < 8$ and $2 < 3 \leq 4$. And we see the bounds dynamically adjusting to give the smallest possible regions the *pcp* exist within.

The residues for $10 = 2 \cdot 5$ are $\{1, 3, 7, 9\}$, with *mcp* $\{1, 9 = 3^2\}$. But unlike for 4, 6, and 8, it wouldn't have been considered a prime pair even in Goldbach's time. But this explicitly illustrates why we set the *hhr* upper limit to $n - 2$ and not n . Integers 8, 10, 12, 14, and 38 are the only ones with one *pcp*.

Metaphorically, to understand the primes we must understand there is a clear deterministic chemistry that exists within the residue properties of modular groups that defines their behavior. Part of that chemistry tells us the primes bond in pairs, and with other relationships, that are built into the structure of modular groups. (For example, verification of *Polignac's Twin Primes* conjectures [2]).

The primes, as the primary residues of modular groups, are their fundamental building block elements. And their powers and cross-products are their molecules. And like with the chemical elements, there are defined ways we know they can and must move and interact, and ways they can't.

The *pcp* are the symmetric load bearing beams of the structure of these modular groups. They are the bonding forces that hold everything together. It is part of the chemistry of the primes that they bond in this manner. As n grows so do the *pcp*, in a structured way that is a function of the factorization of n .

We can conceptually determine the number of *pcp* bounded by r_{max} from the following formula:

$$\#pcp(r_{max}) = \#lhr(r_{max}) - \#hhr(r_{max})_{r_{cp}} - \#lhr(r_{max})_{r_{cp}} || p_r^e \quad (14)$$

However, the actual algorithm to determine the number and values of the *pcp*, within the r_{max} bounded regions and for all n , is quite simple and short, easy to do by hand, and translates easily into software.

Again, the ability to determine the number and values of the *pcp*, and to create optimum dynamically adjusting boundary conditions, is based on the understanding of the properties of how the primes exist as the residues of these simple \mathbb{Z}_n modular groups. Thus the *GC* is really a (weak) statement on the distribution of the primes, whose structure requires they bond in the group patterns Goldbach imagined.

We will also see *Bertrand's Postulate* is another (weaker) statement on the distribution of the primes.

Bertrand's Postulate

Bertrand's Postulate (BP) states: **for each $n \geq 2$, there is a prime p such that $n < p < 2n$.**

For a *prime complement pairs (pcp)* to exist their must be a prime p_i e.g. $2 < p_i < n/2$ and another p_j e.g. $n/2 < p_j < n$. The later requirement is just a restatement of the *BP* that $n < p < 2n$. We previously established the optimum low bounds to identify at least one *pcp* prime within $\text{LowBound} \leq p < n$ to be:

$$\text{LowBound}_{(n-2)} = (n - 2) - \text{isqrt}(n - 2) \quad (15)$$

We can make the bound less strict to mimic the *BP*, to identify at least one prime, not necessarily a *pcp*.

$$\text{LowBound}_{(n)} = n - \text{isqrt}(n) \quad (16)$$

Let's compare the two bounds for the first few even values of n , and the *pcp* prime pairs identified.

n	$\text{LowBound}_{(n)} \leq p < n$	PCP Prime Pairs	$\text{LowBound}_{(n-2)} \leq p < n-2$
6	$4 \leq p < 6 : p = 5$	{3, 3}*	$2 \leq p < 4 : p = 3$
8	$6 \leq p < 8 : p = 7$	{3, 5}	$4 \leq p < 6 : p = 5$
10	$7 \leq p < 10 : p = 7$	{3, 7}	$6 \leq p < 8 : p = 7$
12	$9 \leq p < 12 : p = 11$	{5, 7}	$7 \leq p < 10 : p = 7$
14	$11 \leq p < 14 : p = \{11, 13\}$	{3, 11}	$9 \leq p < 12 : p = 11$
16	$12 \leq p < 16 : p = 13$	{5, 11}, {3, 13}	$11 \leq p < 14 : p = \{11, 13\}$
18	$16 \leq p < 18 : p = 17$	{5, 13}	$12 \leq p < 16 : p = 13$
20	$16 \leq p < 20 : p = \{17, 19\}$	{3, 17}	$14 \leq p < 18 : p = 17$
22	$18 \leq p < 22 : p = 19$	{5, 17}, {3, 19}	$16 \leq p < 20 : p = \{17, 19\}$
24	$20 \leq p < 24 : p = 23$	{5, 19}	$18 \leq p < 22 : p = 19$
26	$21 \leq p < 26 : p = 23$	{3, 23}	$20 \leq p < 24 : p = 23$
28	$23 \leq p < 28 : p = 23$	{5, 23}	$21 \leq p < 26 : p = 23$
30	$25 \leq p < 30 : p = 29$	{7, 23}	$23 \leq p < 28 : p = 23$
32	$27 \leq p < 32 : p = \{29, 31\}$	{3, 29}	$25 \leq p < 30 : p = 29$
34	$29 \leq p < 34 : p = \{29, 31\}$	{5, 29}, {3, 31}	$27 \leq p < 32 : p = \{29, 31\}$
36	$30 \leq p < 36 : p = 31$	{7, 29}, {5, 31}	$29 \leq p < 34 : p = \{29, 31\}$
38	$32 \leq p < 38 : p = 37$	{7, 31}	$30 \leq p < 36 : p = 31$
40	$34 \leq p < 40 : p = 37$	{3, 37}	$32 \leq p < 38 : p = 37$
42	$36 \leq p < 42 : p = \{37, 41\}$	{5, 37}	$34 \leq p < 40 : p = 37$
44	$38 \leq p < 44 : p = \{41, 43\}$	{7, 37}, {3, 41}	$36 \leq p < 42 : p = \{37, 41\}$
46	$40 \leq p < 46 : p = \{41, 43\}$	{5, 41}, {3, 43}	$38 \leq p < 44 : p = \{41, 43\}$
48	$42 \leq p < 48 : p = \{43, 47\}$	{7, 41}, {5, 43}	$40 \leq p < 46 : p = \{41, 43\}$
50	$43 \leq p < 50 : p = \{43, 47\}$	{7, 43}, {3, 47}	$42 \leq p < 48 : p = \{43, 47\}$

As n increases, the lower bounds dynamically grow, creating much tighter bounds on p than the *BP*.

100	$90 \leq p < 100 : p = 97$	{11, 89}, {3, 97}	$89 \leq p < 98 : p = \{89, 97\}$
256	$240 \leq p < 256 : p = \{241, 151\}$	{17, 239}, {5, 251}	$239 \leq p < 254 : p = \{239, 241, 251\}$

For 256, the bounded primes are {239, 241, 251}, and cross-products {243, 245, 247, 249, 253}. The *mc* for 241 is $15 = 3 \cdot 5$, but 5 forms the *pcp* with 251. The *mc* for 3 is $253 = 11 \cdot 23$, *mc* to 245 and 233, and the *mc* for 247 is $9 = 3^2$. Now $243 = 3^5$, but $3|3^2 < (r_{\max}=15)$, and $3^3|3^4$ are residues too, so 243 really is $3 \cdot 81$ and $9 \cdot 27$. For $245 = 5 \cdot 7^2$, then $5|7 < r_{\max}$ multiply residues $49|35$, and $249 = 3 \cdot 83$, a product of primes. Also, 256 has 6 other *pcp* below this region. In comparison, the *BP* bound is: $128 < p < 256$.

Thus the *BP* structurally falls out of the *GC*, as its necessarily true for the *GC* to be true. But the *GC* doesn't need it, as seen here, as much tighter bounds on p can be created using the various properties of modular groups. The *BP* is a mere consequence of the *GC* being true, and states a weaker bound on p .

Prime Pairs Algorithm

An algorithm to identify all the *pcp* prime pairs is very short and simple, which is demonstrated here. It merely removes (filters out) the composite *non-pcp* residues, leaving only the prime residues of *pcp*.

1) For even n treat as Z_n , and identify its **low-half-residues (lhr)** $< n/2$.
The residue values r are odd integers **coprime** to n , e.g. $\gcd(r, n) = 1$.
The **1st canonical residue** is 1, but 1 is not prime, so it can't be part of a prime pair, so test the odds numbers from $3 < n/2$ to identify the **pcp lhr**.

2) Store in **lhr_mults** the powers of r_i in $lhr < n-2$; these are composite residues.
We test up to $n-2$ because $n-1$ is the **mcp** for 1, which can't be part of a **pcp**.
Once the square of an r_i (i.e. r_i^2) $> n-2$, exit process, as all other r_j are $> n-2$.
We test against the n th roots of $n-2$ to prevent arithmetic overflow possibilities.

3) Store in **lhr_mults** the cross-products $r_i * r_j < n-2$.
Starting with smallest r_i **lhr**, test cross-products with all larger **lhr**.
If $r_i > \sqrt{n-2}$, exit process, as r_i^2 , and all other r_i **lhr** cross-products, are $> n-2$.
If for next larger residue r_j , $r_i * r_j > n-2$, exit process, as no others are $< n-2$.
Otherwise save in **lhr_mults** cross-product $r_i * r_j < n-2$, repeat for the next larger r_j **lhr**.

lhr_mults now has all the non-prime composite residue values $< n-2$.

4) Remove from the **lhr** the **non-pcp lhr_mults** values. The **pcp** prime pairs remain.
a) For **lhr_mults** values $r > n/2$, convert to their **mcp** values $n-r$.
b) All now are **non-pcp** values $< n/2$; remove their values from **lhr** list.

The **lhr** list now contains all prime residues whose **mcp** make a **pcp** prime pair.
For the remaining primes p_n , the **pcp** for n are the prime pairs $[p_n, n - p_n]$.

Let's now do a non-trivial example that performs all the algorithmic parts.

Example: Find the **pcp** prime pairs for $n = 50 = 2 * 5 * 5$.

1) Identify the **lhr** values $< n/2$.

Write out list of odd numbers from 3 to $< 25 = 50/2$

[3 5 7 9 11 13 15 17 19 21 23]

The **lhr** are values r coprime (share no factors) to n ; i.e $\gcd(r, 50) = 1$.

[3 7 9 11 13 17 19 21 23]

Thus: $lhr = [3, 7, 9, 11, 13, 17, 19, 21, 23]$

2) Store in **lhr_mults** the powers of the **lhr** $< 48 = 50-2$.

a) for 3: **lhr_mults** = []
3 * 3: **lhr_mults** = [9], as 9 < 48
3 * 9: **lhr_mults** = [9, 27], as 27 < 48
3 * 27: stop powers for 3, as 81 > 48
b) for 7: exit powers process; as 7 * 7 = 49 > 48; no other **lhr** power can be < 48.

3) Store in **lhr_mults** the **lhr** cross-products < 48.

a) for 3: **lhr_mults** = [9, 27]
3 * 7: **lhr_mults** = [9, 27, 21], as 21 < 48
3 * 9: **lhr_mults** = [9, 27, 21, 27], as 27 < 48
3 * 11: **lhr_mults** = [9, 27, 21, 27, 33], as 33 < 48
3 * 13: **lhr_mults** = [9, 27, 21, 27, 33, 39], as 39 < 48
3 * 17: stop cross-products process for 3, as 3 * 17 = 51 > 48

b) for 7: $\text{lhr_mults} = [9, 27, 21, 27, 33, 39]$
 $7 * 9: \text{exit total cross-product process, as } 7 * 9 = 63 > 48;$

4) Remove from the **lhr** the **lhr_mults** values

$\text{lhr} = [3, 7, 9, 11, 13, 17, 19, 21, 23]$

$\text{lhr_mults} = [9, 27, 21, 27, 33, 39]$

a) Convert **lhr_mults** values $> 25 = 50/2$ to their modular complement value $50-r$

$\text{lhr_mults} = [9, 23, 21, 23, 17, 11]$

b) Remove from **lhr** values in **lhr_mults**

$\text{lhr} = [3, 7, 9, 11, 13, 17, 19, 21, 23]$

$\text{lhr_mults} = [9, 23, 21, 23, 17, 11]$

b1) For **lhr_mults** val 9; remove from **lhr**:

$\text{lhr_mults} = [9, 23, 21, 23, 17, 11]$

$\text{lhr} = [3, 7, 11, 13, 17, 19, 21, 23]$

b2) For **lhr_mults** val 23; remove from **lhr**:

$\text{lhr_mults} = [9, 23, 21, 23, 17, 11]$

$\text{lhr} = [3, 7, 11, 13, 17, 19, 21]$

b3) For **lhr_mults** val 21; remove from **lhr**:

$\text{lhr_mults} = [9, 23, 21, 23, 17, 11]$

$\text{lhr} = [3, 7, 11, 13, 19]$

b4) For **lhr_mults** val 17; remove from **lhr**:

$\text{lhr_mults} = [9, 23, 21, 23, 17, 11]$

$\text{lhr} = [3, 7, 11, 13, 19]$

b5) For **lhr_mults** val 11; remove from **lhr**:

$\text{lhr_mults} = [9, 23, 21, 23, 17, 11]$

$\text{lhr} = [3, 7, 13, 19]$

lhr list of only primes now exists; as original **lhr** composite residues have been removed.

Thus **lhr** = **[3, 7, 13, 19]** now contains 4 prime **pcp** residues for $n = 50$.

Their 4 **pcp** prime pairs values for 50 are:

pcp for 3: $[3, 50 - 3] = [3, 47]$

pcp for 7: $[7, 50 - 7] = [7, 43]$

pcp for 13: $[13, 50 - 13] = [13, 37]$

pcp for 19: $[19, 50 - 19] = [19, 31]$

Every even integer $n > 6$ has at least one **pcp** prime pair.
This is a property of **modular groups Z_n** over even integers n .

Prime Pairs Code

The code below is a Ruby language version of the prime pairs algorithm. Ruby is a dynamic language that has various VM (Virtual Machine) implementations, with JRuby and Truffleruby prominent others. Ruby's robust language constructs allows for a short|efficient translation of the algorithm into code.

```
# Enable YJIT if using CRuby >= 3.3"
RubyVM::YJIT.enable if RUBY_ENGINE == "ruby" and RUBY_VERSION.to_f >= 3.3

def prime_pairs_lohi(n)
  return puts "Input not even n > 2" unless n.even? && n > 2
  return (pp [n, 1]; pp [n/2, n/2]; pp [n/2, n/2]) if n <= 6

  # generate the low-half-residues (lhr) r < n/2
  lhr = 3.step(n/2, 2).select { |r| r if r.gcd(n) == 1 }
  ndiv2, rhi = n/2, n-2          # lhr:hhr midpoint, max residue limit
  lhr_mults = []                 # for lhr values not part of a pcp

  # store all the powers of the lhr members < n-2
  lhr.each do |r|
    # step thru the lhr members
    r_pwr = r                    # set to first power of r
    break if r > rhi / r_pwr     # exit if r^2 > n-2, as all others are too
    while r < rhi / r_pwr       # while r^e < n-2
      lhr_mults << (r_pwr *= r)  # store its current power of r
    end
  end

  # store all the cross-products of the lhr members < n-2
  lhr_dup = lhr.dup             # make copy of the lhr members list
  while (r = lhr_dup.shift) && !lhr_dup.empty? # do mults of 1st list r w/others
    ri_max = rhi / r           # ri can't multiply r with values > this
    break if lhr_dup[0] > ri_max # exit if product of consecutive r's > n-2
    lhr_dup.each do |ri|
      # for each residue in reduced list
      break if ri > ri_max     # exit for r if cross-product with ri > n-2
      lhr_mults << r * ri      # store value if < n-2
    end
    # check cross-products of next lhr member
  end

  # convert lhr_mults vals > n/2 to their lhr complements n-r,
  # store them, those < n/2, in lhr_del; it now holds non-pcp lhr values
  lhr_del = lhr_mults.map { |r_del| (r_del > ndiv2 ? n - r_del : r_del) }

  pp [n, (lhr -= lhr_del).size] # show n and pcp prime pairs count
  pp [lhr.first, n-lhr.first]   # show first pcp prime pair of n
  pp [lhr.last, n-lhr.last]    # show last pcp prime pair of n
end

def tm; t = Time.now; yield; Time.now - t end # to time runtime execution

n = ARGV[0].to_i                # get n input from terminal
puts tm { prime_pairs_lohi(n) } # show execution runtime as last output
```

By placing the code a file named `prime_pairs_lohi.rb`, it can be run as shown.

```
$ ruby primes_pairs_lohi.rb 12_345_678          $ ruby primes_pairs_lohi.rb 123_456_780
[12345678, 71169]                               [123456780, 717906]
[31, 12345647]                                  [19, 123456761]
[6172799, 6172879]                             [61728367, 61728413]
1.854008033                                    15.993202831
```

Empirical Data Table 1.

The data shown here is: # residues < $n/2$, primes cnt < $n/2|n$, $pcps$, $pcps$ % of primes, and 1st|last $pcps$.

n	$\varphi(n)/2$	$\pi(n/2)$	$\pi(n)$	PCP	% PCP($n/2 n$)		First Prime Pair	Last Prime Pair
1,000,000	200,000	41,538	78,498	5,402	13.00	13.76	[17, 999983]	[499943, 500057]
2,000,000	400,000	78,498	148,933	9,720	12.38	13.05	[7, 1999993]	[999961, 1000039]
3,000,000	400,000	114,155	216,816	27,502	24.09	25.37	[43, 2999957]	[1499857, 1500143]
4,000,000	800,000	148,933	283,146	17,630	11.84	12.45	[29, 3999971]	[1999853, 2000147]
5,000,000	1,000,000	183,072	348,513	21,290	11.63	12.22	[37, 4999963]	[2499949, 2500051]
6,000,000	800,000	216,816	412,849	49,783	22.96	24.12	[7, 5999993]	[2999911, 3000089]
7,000,000	1,200,000	250,150	476,648	34,284	13.71	14.39	[3, 6999997]	[3499967, 3500033]
8,000,000	1,600,000	283,146	539,777	31,753	11.21	11.77	[7, 7999993]	[3999763, 4000237]
9,000,000	1,200,000	315,948	602,489	70,619	22.35	23.44	[7, 8999993]	[4499953, 4500047]
10,000,000	2,000,000	348,513	664,579	38,807	11.14	11.68	[29, 9999971]	[4999913, 5000087]
11,000,000	2,000,000	380,800	726,517	46,812	12.29	12.89	[3, 10999997]	[5499979, 5500021]
12,000,000	1,600,000	412,849	788,060	90,877	22.01	23.06	[11, 11999989]	[5999947, 6000053]
13,000,000	2,400,000	444,757	849,252	53,398	12.01	12.58	[3, 12999997]	[6499841, 6500159]
14,000,000	2,400,000	476,648	910,077	62,026	13.01	13.63	[19, 13999981]	[6999997, 7000003]
15,000,000	2,000,000	508,261	970,704	110,140	21.67	22.69	[19, 14999981]	[7499939, 7500061]
16,000,000	3,200,000	539,777	1,031,130	58,383	10.82	11.32	[11, 15999989]	[7999913, 8000087]
17,000,000	3,200,000	571,119	1,091,314	65,592	11.48	12.02	[37, 16999963]	[8499979, 8500021]
18,000,000	2,400,000	602,489	1,151,367	129,501	21.49	22.49	[13, 17999987]	[8999777, 9000223]
19,000,000	3,600,000	633,578	1,211,050	71,656	11.31	11.83	[3, 18999997]	[9499811, 9500189]
20,000,000	4,000,000	664,579	1,270,607	70,730	10.64	11.13	[19, 19999981]	[9999739, 10000261]
21,000,000	2,400,000	695,609	1,329,943	177,440	25.51	26.68	[23, 20999977]	[10499963, 10500037]
22,000,000	4,000,000	726,517	1,389,261	85,476	11.77	12.31	[23, 21999977]	[10999811, 11000189]
23,000,000	4,400,000	757,288	1,448,221	83,727	11.06	11.56	[7, 22999993]	[11499949, 11500051]
24,000,000	3,200,000	788,060	1,507,122	165,922	21.05	22.02	[19, 23999981]	[11999927, 12000073]
25,000,000	5,000,000	818,703	1,565,927	85,838	10.48	10.96	[17, 24999983]	[12499481, 12500519]
26,000,000	4,800,000	849,252	1,624,527	97,209	11.45	11.97	[67, 25999933]	[12999919, 13000081]
27,000,000	3,600,000	879,640	1,683,065	184,050	20.92	21.87	[19, 26999981]	[13499939, 13500061]
28,000,000	4,800,000	910,077	1,741,430	113,922	12.52	13.08	[101, 27999899]	[13999691, 14000309]
29,000,000	5,600,000	940,455	1,799,676	101,387	10.78	11.27	[61, 28999939]	[14499973, 14500027]
30,000,000	4,000,000	970,704	1,857,859	202,166	20.83	21.76	[11, 29999989]	[14999969, 15000031]
31,000,000	6,000,000	1,000,862	1,915,979	107,710	10.76	11.24	[11, 30999989]	[15499943, 15500057]
32,000,000	6,400,000	1,031,130	1,973,815	106,627	10.34	10.80	[61, 31999939]	[15999871, 16000129]
33,000,000	4,000,000	1,061,198	2,031,667	243,780	22.97	23.99	[17, 32999983]	[16499939, 16500061]
34,000,000	6,400,000	1,091,314	2,089,379	120,272	11.02	11.51	[107, 33999893]	[16999859, 17000141]
35,000,000	6,000,000	1,121,389	2,146,775	138,452	12.35	12.89	[31, 34999969]	[17499793, 17500207]

Table 1.

The Tale of Table 1.

The variance seen in the *pcp* numbers is due to the number and characteristics of the residues $\varphi(n)$, which is due to the number, size, and values of the prime factors for each n , as shown here.

```
$ ruby prime_pairs_lohi.rb 148_000_006          $ ruby prime_pairs_lohi.rb 150_000_006
[1480000006, 436612]                          [1500000006, 708844]
[3, 148000003]                                [5, 150000001]
[73999643, 74000363]                         [74999959, 75000047]
32.511435635                                  24.097201574
```

While close in size, 148,000,006 has substantially fewer *pcp*, which take 33% more time to process. That's because its residues produce more prime powers and cross-products that have to be filtered out. And that's due to the difference in their prime factors, which determine their \mathbb{Z}_n groups residue values.

```
148_000_006.factors                          150_000_006.factors
[[2, 1], [7, 1], [11, 1], [19, 1], [50581, 1]]  [[2, 1], [3, 1], [13, 2], [29, 1], [5101, 1]]
 $\varphi(148,000,006) = (2 - 1)(7 - 1)(11 - 1)(19 - 1)(50581 - 1) = 54,626,400$ 
 $\varphi(150,000,006) = 13(2 - 1)(3 - 1)(13 - 1)(29 - 1)(5101 - 1) = 44,553,600$ 
```

So n values with relatively larger *pcp* counts have fewer residues, with relatively larger values. They produce relatively fewer prime factors and cross-products, leaving more *lhr* primes to form *pcp* with.

However, the much more interesting phenomena is what the data reveals in the two % PCP columns. These are the percentages of all the primes $\pi(n)$ less than n , and in low half $\pi(n/2)$, that form the *pcp*.

$$\% \text{PCP}(n/2) = (pcp / \pi(n/2)) \cdot 100 \quad \% \text{PCP}(n) = (2 \cdot pcp / \pi(n)) \cdot 100 \quad (17)$$

The interesting phenomena that's revealed is, the *pcp* appear to bond in quantized ratios! And the affect is stronger among all the primes $< n$ versus just those $< n/2$. So as the primes density decreases as n grows, the bonding between all the possible $p_r < n$ is slightly stronger|larger than just for those $< n/2$.

The n values in Table 1. only end in '0', so the table below shows results for larger consecutive even n that end in each even digit, and have the same number of $\pi(n)$ primes for those values.

n	$\varphi(n)/2$	$\pi(n)$	PCP	% PCP	First Prime Pair	Last Prime Pair
900,000,000	120,000,000	46,009,215	4,132,595	17.96	[37, 899999963]	[449999993, 450000007]
900,000,002	204,211,920	46,009,215	1,724,113	7.49	[73, 899999929]	[449999981, 450000121]
900,000,004	224,969,472	46,009,215	1,550,567	6.74	[41, 899999963]	[449999887, 450000117]
900,000,006	150,000,000	46,009,215	3,099,095	13.47	[43, 899999963]	[449999783, 450000223]
900,000,008	224,956,800	46,009,215	1,550,273	6.74	[79, 899999929]	[449999731, 450000277]

Table 1a.

We still see this quantizing phenomena, and that the largest number of *pcp* for n have the highest ratios. Now that it's been revealed, hopefully more research will be pursued to study and characterize it.

What is seen here is a deterministic pattern in the primes distribution, that facilitates they form prime pairs in such numbers, and in such ratios. This would not be possible if the primes existed randomly. Thus the data also illustrates again from this property, **the primes do not exist randomly!**

Empirical Data Table 2.

Here we see the lhr data variance in the region bounded by $r_{max}+2$. While r_{max} linearly increases with n , the number of residues and pcp prime pairs is still predominantly a function of the factorization of n .

n	$r_{max} + 2$	$lhr(r_{max+})$	$\pi(r_{max+})$	$PCP(r_{max+})$	% $PCP(r_{max+})$	First Prime Pair	Last Prime Pair(r_{max+})
1,000,000	1,001	400	168	20	11.90	[17, 999983]	[977, 999023]
2,000,000	1,416	565	223	24	10.76	[7, 1999993]	[1321, 1998679]
3,000,000	1,734	462	270	63	23.33	[43, 2999957]	[1721, 2998279]
4,000,000	2,001	800	303	34	11.22	[29, 3999971]	[1997, 3998003]
5,000,000	2,238	894	332	39	11.75	[37, 4999963]	[2113, 4997887]
6,000,000	2,451	653	363	84	23.14	[7, 5999993]	[2447, 5997553]
7,000,000	2,647	907	383	54	14.09	[3, 6999997]	[2621, 6997379]
8,000,000	2,830	1,131	410	45	10.98	[7, 7999993]	[2797, 7997203]
9,000,000	3,001	800	431	96	22.27	[7, 8999993]	[2917, 8997083]
10,000,000	3,164	1,265	447	44	9.84	[29, 9999971]	[2633, 9997367]
11,000,000	3,318	1,205	466	51	10.94	[3, 10999997]	[3271, 10996729]
12,000,000	3,466	923	485	100	10.62	[11, 11999989]	[3461, 11996539]
13,000,000	3,607	1,331	504	68	13.39	[3, 12999997]	[3461, 12996539]
14,000,000	3,743	1,283	522	82	15.71	[19, 13999981]	[3739, 13996261]
15,000,000	3,874	1,032	536	111	20.71	[19, 14999981]	[3847, 14996153]
16,000,000	4,001	1,600	551	61	11.07	[11, 15999989]	[3989, 15996011]
17,000,000	4,125	1,552	566	68	12.01	[37, 16999963]	[4093, 16995907]
18,000,000	4,244	1,131	582	112	19.24	[13, 17999987]	[4241, 17995759]
19,000,000	4,360	1,651	595	66	11.09	[3, 18999997]	[4271, 18995729]
20,000,000	4,474	1,789	607	62	10.21	[19, 19999981]	[4363, 19995637]
21,000,000	4,584	1,047	620	141	22.74	[23, 20999977]	[4561, 20995439]
22,000,000	4,692	1,706	634	70	11.04	[23, 21999977]	[4547, 21995453]
23,000,000	4,797	1,835	645	63	9.77	[7, 22999993]	[4723, 22995277]
24,000,000	4,900	1,305	654	129	19.72	[19, 23999981]	[4703, 23995297]
25,000,000	5,001	2,000	669	71	10.61	[17, 24999983]	[4889, 24995111]
26,000,000	5,101	1,883	682	76	11.14	[67, 25999933]	[4999, 25995001]
27,000,000	5,198	1,385	692	153	22.11	[19, 26999981]	[5171, 26994829]
28,000,000	5,293	1,815	701	91	12.98	[101, 27999899]	[5279, 27994721]
29,000,000	5,387	2,080	710	75	10.56	[61, 28999939]	[5347, 28994653]
30,000,000	5,479	1,461	724	148	20.44	[11, 29999989]	[5443, 29994557]
31,000,000	5,569	2,155	735	83	11.29	[11, 30999989]	[5501, 30994499]
32,000,000	5,658	2,262	745	73	9.79	[61, 31999939]	[5647, 31994353]
33,000,000	5,746	1,392	756	173	22.88	[17, 32999983]	[5693, 32994307]
34,000,000	5,832	2,194	765	85	11.11	[107, 33999893]	[5813, 33994187]
35,000,000	5,918	2,028	777	81	10.42	[31, 34999969]	[5791, 34994209]

Table 2.

The Tale of Table 2.

Here $\% \text{PCP}(r_{\max+}) = (pcp / \pi(r_{\max+})) \cdot 100$ is the percentage of primes as $pcp \leq r_{\max}+2$, and other parameters $r_{\max+}$ for $lhr \leq r_{\max}+2$. We see the $\% \text{PCP}$ values for n in both tables closely match, though the bounded regions become a smaller percentage of $n/2$ as n grows. Code to generate the r_{\max} data is given below.

```
# Enable YJIT if using CRuby >= 3.3"
RubyVM::YJIT.enable if RUBY_ENGINE == "ruby" and RUBY_VERSION.to_f >= 3.3

def prime_pairs_rmax(n)
  return puts "Input not even n > 6" unless n.even? && n > 6

  # generate the low-half-residues (lhr) r < n/2
  lhr = 3.step(n/2, 2).select { |r| r if r.gcd(n) == 1 }
  ndiv2, rhi = n/2, n-2          # lhr:hhr midpoint, max residue limit
  lhr_mults = []                 # for lhr values not part of a pcp

  # store all the powers of the lhr members < n-2
  lhr.each do |r|                # step thru the lhr members
    r_pwr = r                     # set to first power of r
    break if r > rhi / r_pwr      # exit if r^2 > n-2, as all others are too
    while r < rhi / r_pwr        # while r^e < n-2
      lhr_mults << (r_pwr *= r)  # store its current power of r
    end
  end

  # store all the cross-products of the lhr members < n-2
  lhr_dup = lhr.dup              # make copy of the lhr members list
  while (r = lhr_dup.shift) && !lhr_dup.empty? # do mults of 1st list r w/others
    ri_max = rhi / r             # ri can't multiply r with values > this
    break if lhr_dup[0] > ri_max # exit if product of consecutive r's > n-2
    lhr_dup.each do |ri|        # for each residue in reduced list
      break if ri > ri_max      # exit for r if cross-product with ri > n-2
      lhr_mults << r * ri       # store value if < n-2
    end
  end

  # convert lhr_mults vals > n/2 to their lhr complements n-r,
  # store them, those < n/2, in lhr_del; it now holds non-pcp lhr vals
  lhr_del = lhr_mults.map { |r_del| (r_del > ndiv2 ? n - r_del : r_del) }

  lhr_rmax = Integer.sqrt(n-2) + 2 # the lhr_rmax high bound
  lhr_rmax_cnt = lhr.count { |r| r <= lhr_rmax } # count of lhr <= lhr_rmax
  lhr -= lhr_del                    # lhr pcp prime residues
  pcp_rmax = lhr.select { |r| r if r <= lhr_rmax } # lhr pcp prime residues <= lhr_rmax

  pp [n, lhr_rmax, lhr_rmax_cnt, pcp_rmax.size] # show n, lhr_rmax, lhr_max_cnt, pcp count
  pp [pcp_rmax.first, n-pcp_rmax.first]        # show first pcp prime pair of n
  pp [pcp_rmax.last, n-pcp_rmax.last]          # show last pcp prime pair <= rmax
end

def tm; t = Time.now; yield; Time.now - t end # to time runtime execution

n = ARGV[0].to_i # get n value from terminal
puts tm { prime_pairs_rmax(n) } # show execution runtime as last output
```

\$ ruby primes_pairs_rmax.rb 12_345_678	\$ ruby primes_pairs_rmax.rb 123_456_780
[12345678, 3515, 1146, 86]	[123456780, 11113, 2900, 268]
[31, 12345647]	[19, 123456761]
[3511, 12342167]	[11113, 123445667]
1.929410993	17.027832421

Conclusion

Prime Generator Theory (PGT) provides the most powerful and conceptually simple mathematical framework for understanding the nature, characteristics, and distribution of the primes. By treating even values of n as modular groups \mathbb{Z}_n , we apply the properties of residues to easily identify the prime pairs of n , which deterministically exist as the *modular complement pairs (mcp)* of strictly primes, i.e. they exist as the *prime complement pair (pcp)* residues of \mathbb{Z}_n .

From this framework we establish every even integer $n > 6$ will (must) have at least one *pcp* prime pair. The properties of this framework can then be algorithmically constructed to identify all the *pcp* prime pairs. This simple algorithm is then translated into software that provides the total count of *pcp* prime pairs of n , and displays their values as desired. This is done without the need to perform any explicit prime searches or tests. The structure of the \mathbb{Z}_n residues guarantees identification of the primes.

We also see *Bertrand's Postulate* is just a necessary condition that falls out of the modular symmetry of the *pcp*. There always must be a prime in the intervals $n/2 < p < n$ in order for there to be prime pairs. As an extra bonus, we're able to create much smaller (tighter) dynamically adjusting bounds for p , compared to the wider static *BP* bounds, which become relatively larger as n increases. Thus we now have a much better understanding of the distribution of the primes.

Finally, empirical data is provided for an increasing range of n values, showing their total *pcp* counts, and their first and last prime pairs values. It shows large variances in the *pcp* counts for close n values, as a functions of their factorization profiles. We also see the ratios of the *pcp* primes as a percentage of all the primes $< n$ occur in quantized bands. Thus the *pcp* increase as n does, verifying that *Goldbach's Conjecture* is correct, but also revealing so much more to the story than was previously known.

References

- [1] Oliveira e Silva, Herzog, and Pardi – Empirical Verification Of The Even Goldbach Conjecture And Computation Of Prime Gaps Up To $4 \cdot 10^{18}$, Journal: Math. Comp. **83** (2014), 2033-2060.
<https://www.ams.org/journals/mcom/2014-83-288/S0025-5718-2013-02787-1/S0025-5718-2013-02787-1.pdf>
- [2] Jabari Zakiya – On The Infinity of Twin Primes and other K-tuples, International Journal of Mathematics and Computer Research (IJMCR), Vol 13 No 1 (2025), 4739-4761.
<https://ijmcr.in/index.php/ijmcr/article/view/867/678> (pdf)
- [3] Jabari Zakiya – Twin Primes Segmented Sieve of Zakiya (SSoZ) Explained, J Curr Trends Comp Sci Res 2(2), 119 - 147, 2023.
<https://www.opastpublishers.com/open-access-articles/twin-primes-segmented-sieve-of-zakiya-ssoz-explained.pdf>