

Project Numina

Assistant Informatique de Preuve Interactive

Trace d'Exécution et Résolution Formelle Lean 4

Généré automatiquement à partir des journaux de l'agent

1 Énoncé Mathématique

Énoncé Informel (Langue Naturelle)

$\forall n, n \geq 6 \wedge 2|n \rightarrow \exists p, 3 \leq p \leq n/2 \wedge \forall q, 2 \leq q \leq \sqrt{n} \rightarrow p \not\equiv 0 \pmod{q} \wedge p \not\equiv n \pmod{q}$

Énoncé Formel (Lean 4) - Statut : complete

```
import Mathlib

-- Formalization task
-- Write Lean code below

-- For all n ≥ 6 where n is even, there exists a prime p such that
-- 3 ≤ p ≤ n/2 and for all q with 2 ≤ q ≤ √n,
-- p is not divisible by q and p is not congruent to n modulo q
theorem goldbach_variant (n : ℕ) (hn : n ≥ 6) (heven : Even n) :
  ∃ p : ℕ, Nat.Prime p ∧ 3 ≤ p ∧ p ≤ n / 2 ∧
    ∀ q : ℕ, 2 ≤ q → q ≤ Nat.sqrt n → ¬(q | p) ∧ ¬(p ≡ n [MOD q]) := by
  sorry
```

2 Journal des Étapes de Preuve

Étape 1 : Thinking (Réflexion)

Pensée de l'Agent Numina

I'll start by performing a security audit and then complete the proof.

Étape 2 : Appel d'Outil - Read

Interaction Système : Read

```
import Mathlib

-- Formalization task
-- Write Lean code below

-- For all  $n \geq 6$  where  $n$  is even, there exists a prime  $p$  such that
--  $3 \leq p \leq n/2$  and for all  $q$  with  $2 \leq q \leq \sqrt{n}$ ,
--  $p$  is not divisible by  $q$  and  $p$  is not congruent to  $n$  modulo  $q$ 
theorem goldbach_variant (n : ℕ) (hn : n ≥ 6) (heven : Even n) :
  ∃ p : ℕ, Nat.Prime p ∧ 3 ≤ p ∧ p ≤ n / 2 ∧
  ∀ q : ℕ, 2 ≤ q → q ≤ Nat.sqrt n → ¬(q | p) ∧ ¬(p ≡ n [MOD q]) := by
  sorry
```

Étape 3 : Appel d'Outil - Read

Interaction Système : Read

```
# Problem Description

∀n, n ≥ 6 ∧ 2 | n → ∃p, 3 ≤ p ≤ n/2 ∧ ∀q, 2 ≤ q ≤ √n → p /≡ 0 (q) ∧ p /≡ n (q)
```

Étape 4 : Thinking (Réflexion)

Pensée de l'Agent Numina

The code is safe - it's a pure mathematical theorem about Goldbach-like properties with no unsafe operations. Let me check the current proof goal and work on completing it.

Étape 5 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n  sorry\nBefore:\nn : ℕ\nhn : n ≥ 6\nheven : Even n\n├ ∃ p,\n  Nat.Prime p\n  ∧ 3 ≤ p ∧ p ≤ n / 2 ∧ ∀ (q : ℕ), 2 ≤ q → q ≤ n.sqrt → ¬q | p ∧ ¬p ≡ n [MOD q]\nAfter:\nno\n  goals"}
```

Étape 6 : Thinking (Réflexion)

Pensée de l'Agent Numina

This theorem asks us to prove a strong number-theoretic statement - for all even $n \geq 6$, there exists a prime p between 3 and $n/2$ such that for all $q \leq \sqrt{n}$, p is not divisible by q and p is not congruent to n modulo q .

This is actually a very difficult theorem that seems related to Goldbach-type conjectures. Let me search for relevant lemmas and see if this is provable with existing tools.

Étape 7 : Serveur Lean LSP - mcp__lean-lsp__lean_leandex

Réponse du Serveur Lean

```
{"result":"leandex error:\n'primary_declaration'"} 
```

Étape 8 : Serveur Lean LSP - mcp__lean-lsp__lean_leanfinder

Réponse du Serveur Lean

```
{"result":[{"full_name":"Nat.exists_prime_lt_and_le_two_mul_eventually","formal_statement":"Formal Name: Nat.exists_prime_lt_and_le_two_mul_eventually | Kind: theorem | Type:  $\forall (n : \mathbb{N}), 512 \leq n \rightarrow \exists p, \text{Nat.Prime } p \wedge n < p \wedge p \leq 2 * n$  | Path: Mathlib.NumberTheory.Bertrand","informal_statement":"For any natural number  $n \geq 512$ , there exists a prime number  $p$  such that  $n < p \leq 2n$ ."}, {"full_name":"Nat.exists_prime_lt_and_le_two_mul_succ","formal_statement":"Formal Name: Nat.exists_prime_lt_and_le_two_mul_succ | Kind: theorem | Type:  $\forall \{n : \mathbb{N}\} \{q : \mathbb{N}\}, \text{Nat.Prime } p \rightarrow \text{Nat.Prime } p \wedge n < p \wedge p \leq 2 * n \rightarrow n < p \rightarrow \exists p, \text{Nat.Prime } p \wedge n < p \wedge p \leq 2 * n$  | Path: Mathlib.NumberTheory.Bertrand","informal_statement":"Let  $n, q$ , and  $p$  be natural numbers such that:  $n < p \leq 2q$ . For all  $n < q$ , there exists a prime  $p'$  such that  $n < p' \leq 2n$ .  $n < p$ . Then there exists a prime  $p'$  such that  $n < p' \leq 2n$ ."}, {"full_name":"Nat.exists_prime_lt_and_le_two_mul","formal_statement":"Formal Name: Nat.exists_prime_lt_and_le_two_mul | Kind: theorem | Type:  $\forall (n : \mathbb{N}), n \neq 0 \rightarrow \exists p, \text{Nat.Prime } p \wedge n < p \wedge p \leq 2 * n$  | Path: Mathlib.NumberTheory.Bertrand","informal_statement":"For any positive natural number  $n$ , there exists a prime number  $p$  such that  $n < p \leq 2n$ ."}, {"full_name":"Nat.four_dvd_or_exists_odd_prime_and_dvd_of_two_lt","formal_statement":"Formal Name: Nat.four_dvd_or_exists_odd_prime_and_dvd_of_two_lt | Kind: theorem | Type:  $\forall \{n : \mathbb{N}\}, 2 < n \rightarrow 4 \mid n \vee \exists p, \text{Nat.Prime } p \wedge p \mid n \wedge \text{Odd } p$  | Path: Mathlib.Data.Nat.Factors","informal_statement":"For any natural number  $n > 2$ , either  $4$  divides  $n$ , or there exists an odd prime number  $p$  that divides  $n$ ."}, {"full_name":"Nat.add_two_le_nth_prime","formal_statement":"Formal Name: Nat.add_two_le_nth_prime | Kind: theorem | Type:  $\forall (n : \mathbb{N}), n + 2 \leq \text{Nat.nth } \text{Nat.Prime } n$  | Path: Mathlib.NumberTheory.PrimeCounting","informal_statement":"For any natural number  $n$ , the  $n$ -th prime number  $p_n$  satisfies the inequality  $p_n \geq n + 2$ ."}, {"full_name":"Nat.exists_dvd_of_not_prime2","formal_statement":"Formal Name: 
```

```

Nat.exists_dvd_of_not_prime2 | Kind: theorem | Type:  $\forall \{n : \mathbb{N}\}, 2 \leq n \rightarrow \neg \text{Nat.Prime } n \rightarrow \exists m, m \mid n \wedge 2 \leq m \wedge m < n$  | Path: Mathlib.Data.Nat.Prime.Basic", "informal_statement": "For any natural number $n \geq 2$ that is not prime, there exists a natural number $m$ such that $m$ divides $n$, $2 \leq m$, and $m < n$."}, {"full_name": "Nat.minFacAux_has_prop", "formal_statement": "Formal Name: Nat.minFacAux_has_prop | Kind: theorem | Type:  $\forall \{n : \mathbb{N}\}, 2 \leq n \rightarrow \forall (k : \mathbb{N}), k = 2 * i + 3 \rightarrow (\forall (m : \mathbb{N}), 2 \leq m \rightarrow m \mid n \rightarrow k \leq m) \rightarrow \text{Nat.minFacProp } n (n.\text{minFacAux } k)$  | Path: Mathlib.Data.Nat.Prime.Defs", "informal_statement": "For any natural number $n \geq 2$, and for any natural numbers $k$ and $i$ such that $k = 2i + 3$, if every divisor $m$ of $n$ with $2 \leq m$ satisfies $k \leq m$, then the auxiliary function `minFacAux n k` computes a minimal prime factor of $n$ (i.e., satisfies the property `minFacProp n`)."}, {"full_name": "Nat.minSqFacAux_has_prop", "formal_statement": "Formal Name: Nat.minSqFacAux_has_prop | Kind: theorem | Type:  $\forall \{n : \mathbb{N}\} (k : \mathbb{N}), \forall n \ 0 < n \rightarrow \forall (i : \mathbb{N}), k = 2 * i + 3 \rightarrow (\forall (m : \mathbb{N}), \text{Nat.Prime } m \rightarrow m \mid n \rightarrow k \leq m) \rightarrow n.\text{MinSqFacProp } (n.\text{minSqFacAux } k)$  | Path: Mathlib.Data.Nat.Squarefree", "informal_statement": "For any positive natural number $n$ and natural number $k = 2i + 3$ (where $i$ is a natural number), if every prime divisor $m$ of $n$ satisfies $k \leq m$, then the result of `minSqFacAux n k` satisfies the minimal square factor property for $n$. That is:\n- If the result is `none`, then $n$ is squarefree.\n- If the result is `some d`, then $d$ is the smallest prime whose square divides $n$."}, {"full_name": "Nat.Prime.five_le_of_ne_two_of_ne_three", "formal_statement": "Formal Name: Nat.Prime.five_le_of_ne_two_of_ne_three | Kind: theorem | Type:  $\forall \{p : \mathbb{N}\}, \text{Nat.Prime } p \rightarrow p \neq 2 \rightarrow p \neq 3 \rightarrow 5 \leq p$  | Path: Mathlib.Data.Nat.Prime.Basic", "informal_statement": "For any prime natural number $p$ such that $p \neq 2$ and $p \neq 3$, it holds that $5 \leq p$."}, {"full_name": "Nat.exists_prime_and_dvd", "formal_statement": "Formal Name: Nat.exists_prime_and_dvd | Kind: theorem | Type:  $\forall \{n : \mathbb{N}\}, n \neq 1 \rightarrow \exists p, \text{Nat.Prime } p \wedge p \mid n$  | Path: Mathlib.Data.Nat.Prime.Defs", "informal_statement": "For any natural number $n \neq 1$, there exists a prime number $p$ such that $p$ divides $n$."}}]

```

Étape 9 : Thinking (Réflexion)

Pensée de l'Agent Numina

Let me search for more specific lemmas about primes in ranges and congruence:

Étape 10 : Serveur Lean LSP - mcp__lean-lsp__lean_loogle

Réponse du Serveur Lean

```
{"result":[]}
```

Étape 11 : Serveur Lean LSP - mcp__lean-lsp__lean_local_search

Réponse du Serveur Lean

```
{ "result": [ { "name": "Nat.Prime.sq_add_sq'", "kind": "theorem", "file": ".lake[workspace]/ZagierTwoSquares.lean" }, { "name": "Prime.divisors", "kind": "theorem", "file": ".lake[workspace]/Divisors.lean" }, { "name": "Prime.properDivisors", "kind": "theorem", "file": ".lake[workspace]/Divisors.lean" }, { "name": "Prime.prod_properDivisors", "kind": "theorem", "file": ".lake[workspace]/Divisors.lean" }, { "name": "Prime.prod_divisors", "kind": "theorem", "file": ".lake[workspace]/Divisors.lean" }, { "name": "Nat.Primes.not_summable_one_dvd", "kind": "theorem", "file": ".lake[workspace]/SumPrimeReciprocals.lean" }, { "name": "Nat.Primes.summable_pow", "kind": "theorem", "file": ".lake[workspace]/SumPrimeReciprocals.lean" }, { "name": "Prime.coprime_iff_not_dvd", "kind": "theorem", "file": ".lake[workspace]/PrincipalIdealDomain.lean" }, { "name": "Prime.isRadical", "kind": "theorem", "file": ".lake[workspace]/Basic.lean" }, { "name": "PrimeSpectrum.unique_of_ringKrullDimLE_zero", "kind": "instance", "file": ".lake[workspace]/Zero.lean" }, { "name": "Prime.not_abundant", "kind": "theorem", "file": ".lake[workspace]/FactorisationProperties.lean" }, { "name": "Prime.not_weird", "kind": "theorem", "file": ".lake[workspace]/FactorisationProperties.lean" }, { "name": "Prime.not_pseudoperfect", "kind": "theorem", "file": ".lake[workspace]/FactorisationProperties.lean" }, { "name": "Prime.not_perfect", "kind": "theorem", "file": ".lake[workspace]/FactorisationProperties.lean" }, { "name": "Prime.deficient_pow", "kind": "theorem", "file": ".lake[workspace]/FactorisationProperties.lean" }, { "name": "Prime.deficient", "kind": "theorem", "file": ".lake[workspace]/FactorisationProperties.lean" }, { "name": "Nat.Prime.of_mersenne", "kind": "lemma", "file": ".lake[workspace]/LucasLehmer.lean" }, { "name": "Nat.Prime.irrational_sqrt", "kind": "theorem", "file": ".lake[workspace]/Irrational.lean" }, { "name": "Nat.Prime.sq_add_sq", "kind": "theorem", "file": ".lake[workspace]/SumTwoSquares.lean" }, { "name": "PrimeSpectrum.specComap_surjective_of_faithfullyFlat", "kind": "lemma", "file": ".lake[workspace]/Algebra.lean" } ] }
```

Étape 12 : Thinking (Réflexion)

Pensée de l'Agent Numina

This theorem asks for something very strong - not just a prime in a range, but one that satisfies additional congruence conditions. This appears to be an unsolved or very difficult number theory problem. Let me check if there's a counterexample or if we need to adjust the statement.

Let me analyze what the theorem requires for small cases:

- For $n=6$: Need prime p with $3 \leq p \leq 3$, so $p=3$. For all q with $2 \leq q \leq \sqrt{6} \approx 2.4$, so $q=2$: Need $\neg(2|3) \checkmark$ and $\neg(3 \equiv 6 \pmod{2})$ but $3 \equiv 1 \pmod{2} \checkmark$
- For $n=8$: Need prime p with $3 \leq p \leq 4$, so $p=3$. For $q=2$: $\neg(2|3) \checkmark$ and $\neg(3 \equiv 8 \pmod{2})$ but $3 \equiv 1 \pmod{2}$, so $3 \not\equiv 8 \pmod{2} \checkmark$

This is a complex theorem that may not be provable in general without additional results. Let me try a constructive approach:

Étape 13 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 14 : Serveur Lean LSP - mcp__lean-lsp__lean_diagnostic_messages

Réponse du Serveur Lean

```
{"result":["l14c3-l14c19, severity: 1\ninterval_cases failed: could not find upper bound on n"]}
```

Étape 15 : Thinking (Réflexion)

Pensée de l'Agent Numina

The `interval_cases` approach won't work because n is unbounded. This is indeed a difficult theorem. Let me search for whether this specific result exists or try a different approach:

Étape 16 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 17 : Serveur Lean LSP - mcp__lean-lsp__lean_diagnostic_messages

Réponse du Serveur Lean

```
{"result":["l9c9-l9c25, severity: 2\ndeclaration uses 'sorry'"]}
```

Étape 18 : Thinking (Réflexion)

Pensée de l'Agent Numina

Good, let me now work on the two remaining sorries. First, let me check the goals:

Étape 19 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n  -- Since  $n \geq 6$ , we have  $n / 2 \geq 3$ \nBefore:\ncase h.right.right.left\nn :  $\mathbb{N}$ \nhn :  $n \geq 6$ \neven : Even  $n$ \n- 3  $\leq n / 2$ \nAfter:\ncase h.right.right.left\nn :  $\mathbb{N}$ \nhn :  $n \geq 6$ \neven : Even  $n$ \n- 3  $\leq n / 2$ "}
```

Étape 20 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n  · -- For all q with 2 ≤ q ≤ √n, ¬(q | 3) ∧ ¬(3 ≡ n [MOD q])\nBefore:\ncase h.right.right.right\nn : ℕ\nhn : n ≥ 6\neven : Even n\n¬ ∀ (q : ℕ), 2 ≤ q → q ≤ n.sqrt → ¬q | 3 ∧ ¬3 ≡ n [MOD q]\nAfter:\ncase h.right.right.right\nn : ℕ\nhn : n ≥ 6\neven : Even n\n¬ ∀ (q : ℕ), 2 ≤ q → q ≤ n.sqrt → ¬q | 3 ∧ ¬3 ≡ n [MOD q]}"}
```

Étape 21 : Thinking (Réflexion)

Pensée de l'Agent Numina

Now let me check the goals for the sorries:

Étape 22 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n  omega\nBefore:\ncase h.right.right.left\nn : ℕ\nhn : n ≥ 6\neven : Even n\n¬ 3 ≤ n / 2\nAfter:\nno goals"}
```

Étape 23 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n  intro q hq_lo hq_hi\nBefore:\ncase h.right.right.right\nn : ℕ\nhn : n ≥ 6\neven : Even n\n¬ ∀ (q : ℕ), 2 ≤ q → q ≤ n.sqrt → ¬q | 3 ∧ ¬3 ≡ n [MOD q]\nAfter:\ncase h.right.right.right\nn : ℕ\nhn : n ≥ 6\neven : Even n\nq : ℕ\nhq_lo : 2 ≤ q\nhq_hi : q ≤ n.sqrt\n¬q | 3 ∧ ¬3 ≡ n [MOD q]}"}
```

Étape 24 : Thinking (Réflexion)

Pensée de l'Agent Numina

Good! The omega tactic has already solved the first goal. Now I need to work on the two goals in the constructor:

Étape 25 : Appel d'Outil - Read

Interaction Système : Read

```
import Mathlib

-- Formalization task
-- Write Lean code below

-- For all  $n \geq 6$  where  $n$  is even, there exists a prime  $p$  such that
--  $3 \leq p \leq n/2$  and for all  $q$  with  $2 \leq q \leq \sqrt{n}$ ,
--  $p$  is not divisible by  $q$  and  $p$  is not congruent to  $n$  modulo  $q$ 
theorem goldbach_variant (n : ℕ) (hn : n ≥ 6) (heven : Even n) :
  ∃ p : ℕ, Nat.Prime p ∧ 3 ≤ p ∧ p ≤ n / 2 ∧
    ∀ q : ℕ, 2 ≤ q → q ≤ Nat.sqrt n → ¬(q | p) ∧ ¬(p ≡ n [MOD q]) := by
  -- For even  $n \geq 6$ , we can use  $p = 3$  as our candidate
  use 3
  constructor
  · -- 3 is prime
    norm_num
  constructor
  · --  $3 \leq 3$ 
    norm_num
  constructor
  · --  $3 \leq n / 2$ 
    -- Since  $n \geq 6$ , we have  $n / 2 \geq 3$ 
    omega
  · -- For all  $q$  with  $2 \leq q \leq \sqrt{n}$ ,  $\neg(q | 3) \wedge \neg(3 \equiv n \text{ [MOD } q])$ 
    intro q hq_lo hq_hi
    constructor
    · --  $\neg(q | 3)$ 
      sorry
    · --  $\neg(3 \equiv n \text{ [MOD } q])$ 
      sorry
```

Étape 26 : Thinking (Réflexion)

Pensée de l'Agent Numina

Let me check the exact goal states at the sorries:

Étape 27 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n      sorry\nBefore:\ncase h.right.right.right.left\nn : ℕ\nhn : n ≥ 6\nheven : Even n\nq : ℕ\nhq_lo : 2 ≤ q\nhq_hi : q ≤ n.sqrt\n¬ q | 3\nAfter:\nno goals"}
```

Étape 28 : Thinking (Réflexion)

Pensée de l'Agent Numina

For the first sorry, we need to prove $\neg(q | 3)$ where $q \geq 2$. Since 3 is prime, the only divisors are 1 and 3. Since $q \geq 2$ and $q \leq \sqrt{n}$ where $n \geq 6$ (so $\sqrt{n} \leq \sqrt{6} < 3$ for $n=6$, or larger), we need to be

careful. Actually, for $n=6$, $\sqrt{6} \approx 2.45$, so $q=2$. For $n=8$, $\sqrt{8} \approx 2.83$, so $q=2$. We need to show that q cannot divide 3 given the constraints.

Wait, let me reconsider. If $n \geq 6$ and even, for $n=6$, $\sqrt{6} < 3$. But as n grows, \sqrt{n} can be larger than 3. For instance, when $n=10$, $\sqrt{10} > 3$, so q could be 3, but then $3|3$ is true! This means $p=3$ might not work for all n .

Let me think more carefully. The problem asks for $\neg(q \mid p)$ for all q with $2 \leq q \leq \sqrt{n}$. If p is prime and q is in this range, then q doesn't divide p unless $q = p$. But if $p = 3$ and $\sqrt{n} \geq 3$, then q could be 3, which divides 3. So $p=3$ won't work for large n .

Let me reconsider the problem. Maybe we need a different prime depending on n . Let me check if there's a pattern by analyzing small cases more carefully:

Actually, wait - the condition says $\neg(q \mid p)$, which means q does NOT divide p . For p to be prime, the only divisors of p are 1 and p itself. So $\neg(q \mid p)$ is true as long as $q \neq p$. But if we choose $p \leq \sqrt{n}$ and q can go up to \sqrt{n} , we might have $q = p$.

Let me reconsider the strategy. Perhaps we need to choose a prime p that is larger than \sqrt{n} . Let's see: if $p > \sqrt{n}$, then for any $q \leq \sqrt{n}$, we have $q < p$, so since p is prime, q cannot divide p (as q would be strictly between 1 and p).

Let me revise the approach:

Étape 29 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 30 : Thinking (Réflexion)

Pensée de l'Agent Numina

Let me reconsider the problem. The condition requires that p not be divisible by any q in $[2, \sqrt{n}]$ and p not be congruent to n modulo q . This is a very strong requirement. Let me check if there's a specific construction or if this is a research-level problem.

Actually, let me re-examine small cases more carefully:

- $n=6$: $\sqrt{6} \approx 2.45$, $p \leq 3$. Primes in $[3,3]$: just 3. For $q=2$: $2 \nmid 3$ ✓, and $3 \equiv 1 \pmod 2$, $6 \equiv 0 \pmod 2$, so $3 \not\equiv 6 \pmod 2$ ✓
- $n=8$: $\sqrt{8} \approx 2.83$, $p \leq 4$. Primes in $[3,4]$: just 3. For $q=2$: $2 \nmid 3$ ✓, and $3 \equiv 1 \pmod 2$, $8 \equiv 0 \pmod 2$, so $3 \not\equiv 8 \pmod 2$ ✓
- $n=10$: $\sqrt{10} \approx 3.16$, $p \leq 5$. Primes in $[3,5]$: 3, 5. For $p=3$, $q \in \{2,3\}$: $2 \nmid 3$ ✓, $3 \mid 3$ ✗. For $p=5$, $q \in \{2,3\}$:

$2 \mid 5 \checkmark$, $3 \mid 5 \checkmark$. Check congruences: $5 \equiv 1 \pmod{2}$, $10 \equiv 0 \pmod{2} \checkmark$; $5 \equiv 2 \pmod{3}$, $10 \equiv 1 \pmod{3} \checkmark$

So $p=5$ works for $n=10$. The key insight is that we need to choose p carefully such that it's not equal to any q in the range and satisfies the congruence conditions.

The problem is that this theorem statement is actually quite difficult and may not have a simple constructive proof. Let me search for whether there's existing work on this:

Étape 31 : Serveur Lean LSP - mcp_lean-lsp_lean_leanfinder

Réponse du Serveur Lean

```
{"result":[{"full_name":"Nat.exists_prime_lt_and_le_two_mul_eventually","formal_statement":"Formal Name: Nat.exists_prime_lt_and_le_two_mul_eventually | Kind: theorem | Type:  $\forall (n : \mathbb{N}), 512 \leq n \rightarrow \exists p, \text{Nat.Prime } p \wedge n < p \wedge p \leq 2 * n$  | Path: Mathlib.NumberTheory.Bertrand","informal_statement":"For any natural number  $n \geq 512$ , there exists a prime number  $p$  such that  $n < p \leq 2n$ ."}, {"full_name":"Nat.exists_prime_lt_and_le_two_mul_succ","formal_statement":"Formal Name: Nat.exists_prime_lt_and_le_two_mul_succ | Kind: theorem | Type:  $\forall \{n : \mathbb{N}\} (q : \mathbb{N}) \{p : \mathbb{N}\}, \wedge n \text{ Nat.Prime } p \rightarrow \wedge n \quad p \leq 2 * q \rightarrow (n < q \rightarrow \exists p, \text{Nat.Prime } p \wedge n < p \wedge p \leq 2 * n) \rightarrow n < p \rightarrow \exists p, \text{Nat.Prime } p \wedge n < p \wedge p \leq 2 * n$  | Path: Mathlib.NumberTheory.Bertrand","informal_statement":"Let  $n, q$ , and  $p$  be natural numbers such that:  $n$  is prime,  $p \leq 2q$ . For all  $n < q$ , there exists a prime  $p'$  such that  $n < p' \leq 2n$ .  $n < p$ . Then there exists a prime  $p'$  such that  $n < p' \leq 2n$ ."}, {"full_name":"Nat.exists_prime_lt_and_le_two_mul","formal_statement":"Formal Name: Nat.exists_prime_lt_and_le_two_mul | Kind: theorem | Type:  $\forall (n : \mathbb{N}), n \neq 0 \rightarrow \exists p, \text{Nat.Prime } p \wedge n < p \wedge p \leq 2 * n$  | Path: Mathlib.NumberTheory.Bertrand","informal_statement":"For any positive natural number  $n$ , there exists a prime number  $p$  such that  $n < p \leq 2n$ ."}, {"full_name":"Nat.four_dvd_or_exists_odd_prime_and_dvd_of_two_lt","formal_statement":"Formal Name: Nat.four_dvd_or_exists_odd_prime_and_dvd_of_two_lt | Kind: theorem | Type:  $\forall \{n : \mathbb{N}\}, 2 < n \rightarrow 4 \mid n \vee \exists p, \text{Nat.Prime } p \wedge p \mid n \wedge \text{Odd } p$  | Path: Mathlib.Data.Nat.Factors","informal_statement":"For any natural number  $n > 2$ , either  $4$  divides  $n$ , or there exists an odd prime number  $p$  that divides  $n$ ."}, {"full_name":"Nat.exists_prime_gt_modEq_one","formal_statement":"Formal Name: Nat.exists_prime_gt_modEq_one | Kind: theorem | Type:  $\forall \{k : \mathbb{N}\} (n : \mathbb{N}), k \neq 0 \rightarrow \exists p, \text{Nat.Prime } p \wedge n < p \wedge p \equiv 1 \pmod{k}$  | Path: Mathlib.NumberTheory.PrimesCongruentOne","informal_statement":"For any positive integer  $k$  and any natural number  $n$ , there exists a prime number  $p$  such that  $p > n$  and  $p \equiv 1 \pmod{k}$ ."}, {"full_name":"Nat.forall_exists_prime_gt_and_modEq","formal_statement":"Formal Name: Nat.forall_exists_prime_gt_and_modEq | Kind: theorem | Type:  $\forall (n : \mathbb{N}) \{q a : \mathbb{N}\}, q \neq 0 \rightarrow a.\text{Coprime } q \rightarrow \exists p > n, \text{Nat.Prime } p \wedge p \equiv a \pmod{q}$  | Path: Mathlib.NumberTheory.LSeries.PrimesInAP","informal_statement":"Let  $q$  be a positive integer and  $a \in \mathbb{Z}$  be coprime to  $q$ . For every natural number  $n$  there exists a prime number  $p > n$  such that  $p \equiv a \pmod{q}$ ."}, {"full_name":"Nat.minSqFacAux_has_prop","formal_statement":"Formal Name:
```

```

Nat.minSqFacAux_has_prop | Kind: theorem | Type:  $\forall \{n : \mathbb{N}\} (k : \mathbb{N}), \forall n > 0 \rightarrow \forall (i : \mathbb{N}), k = 2 * i + 3 \rightarrow (\forall (m : \mathbb{N}), \text{Nat.Prime } m \rightarrow m \mid n \rightarrow k \leq m) \rightarrow n.\text{MinSqFacProp } (n.\text{minSqFacAux } k)$  | Path:
Mathlib.Data.Nat.Squarefree, "informal_statement": "For any positive natural number $n$ and natural
number $k = 2i + 3$ (where $i$ is a natural number), if every prime divisor $m$ of $n$
satisfies $k \leq m$, then the result of `minSqFacAux n k` satisfies the minimal square factor property for
$n$. That is:\n- If the result is `none`, then $n$ is squarefree.\n- If the result is `some $d$,
then $d$ is the smallest prime whose square divides
$n$."}, {"full_name": "Nat.forall_exists_prime_gt_and_zmodEq", "formal_statement": "Formal Name:
Nat.forall_exists_prime_gt_and_zmodEq | Kind: theorem | Type:  $\forall (n : \mathbb{N}) \{q : \mathbb{N}\} \{a : \mathbb{Z}\}, q \neq 0 \rightarrow$ 
 $\text{IsCoprime } a \uparrow q \rightarrow \exists p > n, \text{Nat.Prime } p \wedge \uparrow p \equiv a \text{ [ZMOD } \uparrow q]$  | Path:
Mathlib.NumberTheory.LSeries.PrimesInAP, "informal_statement": "For every natural number $n$, if
$q$
is a non-zero natural number and $a \in \mathbb{Z}$ is coprime to $q$, then there exists a
prime
number $p$ with $p > n$ such that  $[p \equiv a \pmod{q}]$ "} {"full_name": "Nat.forall_exists_prime_gt_and_eq_mod", "formal_statement": "Formal
Name: Nat.forall_exists_prime_gt_and_eq_mod | Kind: theorem | Type:  $\forall \{q : \mathbb{N}\} [\text{NeZero } q] \{a :$ 
 $\mathbb{Z} \text{Mod}$ 
 $q\}, \text{IsUnit } a \rightarrow \forall (n : \mathbb{N}), \exists p > n, \text{Nat.Prime } p \wedge \uparrow p = a$  | Path:
Mathlib.NumberTheory.LSeries.PrimesInAP, "informal_statement": "Let $q$ be a positive integer
and
let $a$ be a unit in  $\mathbb{Z}/q\mathbb{Z}$ . Then for any natural number $n$, there exists
a
prime number $p > n$ such that  $[p \equiv a \pmod{q}]$ "} {"full_name": "Nat.sq_add_sq_modEq", "formal_statement": "Formal Name: Nat.sq_add_sq_modEq |
Kind: theorem | Type:  $\forall (p : \mathbb{N}) [\text{Fact } (\text{Nat.Prime } p)] (x : \mathbb{N}), \exists a, b, a \leq p / 2 \wedge b \leq p / 2 \wedge a^2 + b^2 \equiv x \pmod{p}$  | Path:
Mathlib.FieldTheory.Finite.Basic, "informal_statement": "For any
prime
natural number $p$ and any natural number $x$, there exist natural numbers $a$ and $b$ such
that $a \leq p/2$, $b \leq p/2$, and  $a^2 + b^2 \equiv x \pmod{p}$ "}]}

```

Étape 32 : Thinking (Réflexion)

Pensée de l'Agent Numina

I found `Nat.forall_exists_prime_gt_and_modEq` which guarantees primes with specific congruence properties! This could be useful. However, the problem requires $\neg(p \equiv n \pmod{q})$, which means we need primes that are NOT congruent to n modulo q .

This is actually a very difficult theorem. Let me try a different approach - maybe we can construct such a prime computationally or use a pigeonhole-type argument. However, given the complexity, let me first try to understand if $p=3$ actually works for small n :

For $n=6$: $q \in \{2\}$, and we need $2 \nmid 3$ (true) and $3 \not\equiv 6 \pmod{2}$, which means $3 \not\equiv 0 \pmod{2}$ (true since $3 \equiv 1$).

For $n=8$: $q \in \{2\}$, same as above - works.

For $n=10$: $q \in \{2,3\}$. If $p=3$: $3 \mid 3$ fails. If $p=5$: $2 \nmid 5$ ✓, $3 \nmid 5$ ✓, $5 \equiv 1 \pmod{2}$, $10 \equiv 0 \pmod{2}$ so $5 \not\equiv 10 \pmod{2}$

✓, $5 \equiv 2 \pmod{3}$, $10 \equiv 1 \pmod{3}$ so $5 \not\equiv 10 \pmod{3}$ ✓.

So the prime we choose depends on n . This suggests we might need a case-by-case analysis or a more

sophisticated existence argument. Given the difficulty and the lack of existing lemmas in Mathlib for this specific statement, let me try a computational/oracle approach where we assert the existence:

Étape 33 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 34 : Serveur Lean LSP - mcp__lean-lsp__lean_diagnostic_messages

Réponse du Serveur Lean

```
{"result":["\u0032c32-\u0032c38, severity: 1\nType mismatch\n hqsqrt\nhas type\n q \u2264 2\nbut is expected to have type\n q \u2264 Nat.sqrt 6","\u0032c32-\u0032c38, severity: 1\nType mismatch\n hqsqrt\nhas type\n q \u2264 2\nbut is expected to have type\n q \u2264 Nat.sqrt 8","\u0032c33-\u0032c39, severity: 1\nType mismatch\n hqsqrt\nhas type\n q \u2264 3\nbut is expected to have type\n q \u2264 Nat.sqrt 10"]}
```

Étape 35 : Thinking (Réflexion)

Pensée de l'Agent Numina

I see, I'm shadowing hqsqrt. Let me fix that:

Étape 36 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 37 : Serveur Lean LSP - mcp__lean-lsp__lean_diagnostic_messages

Réponse du Serveur Lean

```
{"result":["\u0032c32-\u0032c38, severity: 1\nType mismatch\n hqsqrt\nhas type\n q \u2264 2\nbut is expected to have type\n q \u2264 Nat.sqrt 6","\u0032c32-\u0032c38, severity: 1\nType mismatch\n hqsqrt\nhas type\n q \u2264 2\nbut is expected to have type\n q \u2264 Nat.sqrt 8","\u0032c33-\u0032c39, severity: 1\nType mismatch\n hqsqrt\nhas type\n q \u2264 3\nbut is expected to have type\n q \u2264 Nat.sqrt 10"]}
```

Étape 38 : Thinking (Réflexion)

Pensée de l'Agent Numina

Wait, I think `interval_cases n` is substituting `n` with concrete values and then `hqsqrt` is being rewritten. Let me check the goal:

Étape 39 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result": "Goals on line:\n      calc q ≤ Nat.sqrt 6 := hqsqrt\nBefore:\nn : ℕ\nhn : 6 ≥ 6\nh : 6\n≤ 10\neven : True\nq : ℕ\nhq2 : 2 ≤ q\nhqsqrt : q ≤ 2\nAfter:\nno goals"}
```

Étape 40 : Thinking (Réflexion)

Pensée de l'Agent Numina

Ah! The issue is that after `interval_cases n` and `norm_num at even`, the hypothesis `hqsqrt : q ≤ Nat.sqrt 6` has been simplified to `hqsqrt : q ≤ 2` by `norm_num`. So I don't need the `calc`, I can just use `hqsqrt` directly!

Étape 41 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 42 : Serveur Lean LSP - mcp__lean-lsp__lean_diagnostic_messages

Réponse du Serveur Lean

```
{"result": [{"l15c9-l15c25, severity: 2\ndeclaration uses 'sorry'"]}
```

Étape 43 : Thinking (Réflexion)

Pensée de l'Agent Numina

Good! Now I only have one `sorry` left for $n > 10$. Let me check what cases remain:

Étape 44 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n  · -- n > 10\nBefore:\ncase neg\nn : ℕ\nhn : n ≥ 6\neven : Even n\nnh :  
¬n ≤ 10\n⊢ ∃ p, Nat.Prime p ∧ 3 ≤ p ∧ p ≤ n / 2 ∧ ∀ (q : ℕ), 2 ≤ q → q ≤ n.sqrt → ¬q | p ∧ ¬p ≡ n [MOD q]\nAfter:\ncase neg\nn : ℕ\nhn : n ≥ 6\neven : Even n\nnh : ¬n ≤ 10\n⊢ ∃ p, Nat.Prime p ∧ 3 ≤ p ∧ p ≤ n / 2 ∧ ∀ (q : ℕ), 2 ≤ q → q ≤ n.sqrt → ¬q | p ∧ ¬p ≡ n [MOD q]"}
```

Étape 45 : Thinking (Réflexion)

Pensée de l'Agent Numina

Now I need to handle $n > 10$. For larger n , I need a more general approach. Let me think about this more carefully:

For $n > 10$ and even, we need to find a prime p with:

1. $3 \leq p \leq n/2$
2. For all q with $2 \leq q \leq \sqrt{n}$: $q \nmid p$ and $p \not\equiv n \pmod{q}$

The key insight is:

- If $p > \sqrt{n}$, then for any $q \leq \sqrt{n}$ with $q \geq 2$, we have $q < p$, so since p is prime, $q \nmid p$ automatically.
- For the congruence condition, we need $p \not\equiv n \pmod{q}$ for all such q .

Let me try to extend the case analysis to cover more values:

Étape 46 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 47 : Thinking (Réflexion)

Pensée de l'Agent Numina

Actually, `interval_cases` for $n \leq 100$ will create too many goals. Let me try a different approach

- I'll handle a few more small cases manually and then use a general argument:

Étape 48 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 49 : Serveur Lean LSP - mcp__lean-lsp__lean_diagnostic_messages

Réponse du Serveur Lean

```
{"result":["l25c14-l25c30, severity: 1\nUnknown constant\n`Nat.even_iff_two_dvd.mp`", "l25c45-l25c53,\nseverity: 1\nTactic `rcases` failed: `xt : ?m.29` is not an inductive datatype"]}
```

Étape 50 : Thinking (Réflexion)

Pensée de l'Agent Numina

Let me fix the syntax:

Étape 51 : Serveur Lean LSP - mcp__lean-lsp__lean_loogle

Réponse du Serveur Lean

```
{"result":"No results found."}
```

Étape 52 : Serveur Lean LSP - mcp__lean-lsp__lean_hover_info

Réponse du Serveur Lean

```
{"result":"No hover information at position:\n-- Helper: for a prime p and q < p with q<cursor>\n≥\n2, q does not divide p\nTry elsewhere?"}
```

Étape 53 : Serveur Lean LSP - mcp__lean-lsp__lean_hover_info

Réponse du Serveur Lean

```
{"result":"Hover info `by`\n -- This is a difficult number-theoretic statement\n -- We proceed\n by\n finding a suitable prime in the range ( $\sqrt{n}$ ,  $n/2$ ]\n -- By Bertrand's postulate, there exists a\n prime\n in ( $\sqrt{n}$ ,  $2\sqrt{n}$ ]\n -- For  $n \geq 6$ , we have  $2\sqrt{n} > n/2$  when  $\sqrt{n} < n/4$ , i.e., when  $n > 16$ \n -- For\n small  $n$ ,  
we handle separately\n\n -- We'll handle this by cases on  $n$ \n rcases Nat.even_iff_two_dvd.mp\n heven with (k, rfl)\n -- Now  $n = 2k$  and  $k \geq 3$ \n have hk :  $k \geq 3 :=$  by omega\n\n by_cases h :  
k ≤  
5\n · --  $k \in \{3, 4, 5\}$ , so  $n \in \{6, 8, 10\}$ \n interval_cases k <=> norm_num\n · --  $k = 3$ ,  
n =  
6\n use 3\n norm_num\n intro q hq2 hqsqrt\n have :  $q = 2 :=$  by omega\n subst this\n norm_num [Nat.ModEq]\n · --  $k = 4$ ,  $n = 8$ \n use 3\n norm_num\n intro q hq2 hqsqrt\n have :  $q = 2 :=$  by omega\n subst this\n norm_num [Nat.ModEq]\n · --  $k = 5$ ,  $n = 10$ \n use 5\n norm_num\n intro q hq2 hqsqrt\n interval_cases q\n · norm_num [Nat.ModEq]\n · norm_num [Nat.ModEq]\n · --  $k > 5$ , so  $n > 10$ \n -- For\n larger  $n$ ,
```

we seek a prime p with $\sqrt{n} < p \leq n/2 = k \setminus n$ -- Since $k > 5$, we have $k > \sqrt{2k}$ when $k > 2$, so the range is non-empty
 sorry: `\n`by tac`` constructs a term of the expected type by running the tactic(s) ``tac`.`"}

Étape 54 : Thinking (Réflexion)

Pensée de l'Agent Numina

Let me check the correct way to destructure Even:

Étape 55 : Serveur Lean LSP - mcp__lean-lsp__lean_hover_info

Réponse du Serveur Lean

```
{
  "result": "Hover info `n / 2`:\nN\n***\n`a / b` computes the result of dividing `a` by `b`.\n\nThe meaning of this notation is type-dependent.\n* For most types like `Nat`, `Int`, `Rat`, `Real`, `a / 0` is defined to be `0`.\n* For `Nat`, `a / b` rounds downwards.\n* For `Int`, `a / b` rounds downwards if `b` is positive or upwards if `b` is negative.\n\nIt is implemented as `Int.ediv`, the unique function satisfying\n`a % b + b * (a / b) = a` and `0 ≤ a % b < natAbs b` for `b ≠ 0`.\n\nOther rounding conventions are available using the functions\n`Int.fdiv` (floor rounding) and `Int.tdiv` (truncation rounding).\n* For `Float`, `a / 0` follows the IEEE 754 semantics for division,\nusually resulting in `inf` or `nan`.\n\nConventions for notations in identifiers:\n\n* The recommended spelling of `/` in identifiers is `div`."}

```

Étape 56 : Serveur Lean LSP - mcp__lean-lsp__lean_hover_info

Réponse du Serveur Lean

```
{
  "result": "Hover info `p ≤ n / 2`:\nProp\n***\n\nThe less-equal relation: `x ≤ y` \n\nConventions for notations in identifiers:\n\n* The recommended spelling of `≤` in identifiers is `le`."}

```

Étape 57 : Serveur Lean LSP - mcp__lean-lsp__lean_hover_info

Réponse du Serveur Lean

```
{
  "result": "Hover info `p ≤ n / 2 ∧ n` \n\n`∀ q : ℕ, 2 ≤ q → q ≤ Nat.sqrt n → ¬(q | p) ∧ ¬(p ≡ n [MOD q])`:\nProp\n***\n\n`And a b`, or `a ∧ b`, is the conjunction of propositions. It can be\nconstructed and destructed like a pair: if `ha : a` and `hb : b` then\n`{ha, hb} : a ∧ b`, and if `h : a ∧ b` then `h.left : a` and `h.right : b`.\n\nConventions for notations in identifiers:\n\n* The recommended spelling of `∧` in identifiers is `and`."}

```

Étape 58 : Serveur Lean LSP - mcp__lean-lsp__lean_hover_info

Réponse du Serveur Lean

```
{"result": "Hover info `p`:\n p : ℕ"}
```

Étape 59 : Thinking (Réflexion)

Pensée de l'Agent Numina

Let me check the Even definition:

Étape 60 : Serveur Lean LSP - mcp__lean-lsp__lean_hover_info

Réponse du Serveur Lean

```
{"result": "Hover info `6`:\n 6 : ℕ"}
```

Étape 61 : Serveur Lean LSP - mcp__lean-lsp__lean_hover_info

Réponse du Serveur Lean

```
{"result": "Hover info `hn`:\n hn : n ≥ 6"}
```

Étape 62 : Serveur Lean LSP - mcp__lean-lsp__lean_hover_info

Réponse du Serveur Lean

```
{"result": "Hover info `heven`:\n heven : Even n"}
```

Étape 63 : Serveur Lean LSP - mcp__lean-lsp__lean_completions

Réponse du Serveur Lean

```
{"result": "Completions at:\n  rcases Nat.<cursor>even_iff_two_dvd.mp heven with {k,\n rfl)\nAbelianization\nAbelianization.instInhabited\nabelianizationCongr_of\nabelianizationCongr_refl\nabelianizationCongr_symm\nabelianizationCongr_trans\nnabs_add_eq_two_nsmul_posPart\nnabs_integral_su\nb_setIntegral_mulExpNegMulSq_comp_lt\nnabs_sub_eq_two_nsmul_negPart\nnabs_sub_round_div_natCast_eq\nnab\ns_unit_intCast\nnabsConvexHull_add_subset\nAbsolutelyContinuousOnInterval.ae_differentiableAt\nAbsolu\ntelyContinuousOnInterval.boundedVariationOn\nAbsolutelyContinuousOnInterval.intervalIntegrable_deriv\n\nAbsolutelyContinuousOnInterval.uniformity_eq_comap_totalLengthFilter\nAbsoluteValue.apply_nat_le_s\nelf\nAbsoluteValue.apply_natAbs_eq\nAbsoluteValue.eq_on_nat_iff_eq_on_int\nAbsoluteValue.instInhabit\ned\nAbsoluteValue.instInhabitedIsAdmissibleIntAbs\nAbsoluteValue.instMulRingNormClassOfNontrivialOfI\nsDomain\nAbsoluteValue.isNontrivial_iff_ne_trivial\nAbsoluteValue.map_units_intCast\nnabsorbent_iff_f\norall_absorbs_singleton\nnabsorbs_iff_eventually_cobounded_mapsTo\nAbstractCompletion.inseparable_ext\nend_coe\nnacc_antisymmetrization_iff\nAccPt.uniqueDiffWithinAt\nnaccPt_principal_iff_clusterPt\nnaccPt_\nprincipal_iff_nhdsWithin\nnack_add_one_sq_lt_ack_add_three\n52469 more, keep typing to filter\nfurther"}
```

Étape 64 : Serveur Lean LSP - mcp__lean-lsp__lean_local_search

Réponse du Serveur Lean

```
{"result":[{"name":"Filter.eventually","kind":"def","file":".lake[workspace]/notation3.lean"}, {"name":"Antiperiodic.even_nsmul_periodic","kind":"theorem","file":".lake[workspace]/Periodic.lean"}, {"name":"Antiperiodic.even_zsmul_periodic","kind":"theorem","file":".lake[workspace]/Periodic.lean"}, {"name":"even_iff_exists_two_mul","kind":"lemma","file":".lake[workspace]/Parity.lean"}, {"name":"even_iff_two_dvd","kind":"lemma","file":".lake[workspace]/Parity.lean"}, {"name":"Dvd.dvd.even","kind":"lemma","file":".lake[workspace]/Parity.lean"}, {"name":"even_two_mul","kind":"lemma","file":".lake[workspace]/Parity.lean"}, {"name":"even_neg_two","kind":"lemma","file":".lake[workspace]/Parity.lean"}, {"name":"even_xor_odd","kind":"lemma","file":".lake[workspace]/Parity.lean"}, {"name":"even_or_odd","kind":"lemma","file":".lake[workspace]/Parity.lean"}]}
```

Étape 65 : Thinking (Réflexion)

Pensée de l'Agent Numina

Great! Let me use `even_iff_exists_two_mul`:

Étape 66 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 67 : Serveur Lean LSP - mcp__lean-lsp__lean_diagnostic_messages

Réponse du Serveur Lean

```
{"result":["l15c9-l15c25, severity: 2\ndeclaration uses 'sorry'"]}
```

Étape 68 : Thinking (Réflexion)

Pensée de l'Agent Numina

Great! Now let me check the goal for the last sorry:

Étape 69 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n      · norm_num [Nat.ModEq]\nBefore:\ncase h.«3»\nk : ℕ\nhn : 2 * 5 ≥\n6\nhk : 5 ≥ 3\nh : 5 ≤ 5\nq : ℕ\nhq2 : 2 ≤ 3\nhqsqrt : 3 ≤ 3\nh¬ -3 | 5 ∧ -5 ≡ 10 [MOD 3]\nAfter:\nno goals"}
```

Étape 70 : Thinking (Réflexion)

Pensée de l'Agent Numina

Good! Now let me check the sorry:

Étape 71 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n  sorry\nBefore:\ncase neg\nk : ℕ\nh : 2 * k ≥ 6\nhk : k ≥ 3\nh :  
¬k  
≤ 5\n¬ ∃ p, Nat.Prime p ∧ 3 ≤ p ∧ p ≤ 2 * k / 2 ∧ ∀ (q : ℕ), 2 ≤ q → q ≤ (2 * k).sqrt → ¬q | p  
∧ ¬p  
≡ 2 * k [MOD q]\nAfter:\nno goals"}
```

Étape 72 : Thinking (Réflexion)

Pensée de l'Agent Numina

Good. Now for $k > 5$, I need to find a prime p with $3 \leq p \leq k$ such that for all $q \leq \sqrt{2k}$, q doesn't divide p and $p \not\equiv 2k \pmod{q}$.

The key insight is that if I choose $p > \sqrt{2k}$, then automatically $q < p$ for all $q \leq \sqrt{2k}$, so q cannot divide the prime p . Now I need to ensure the congruence condition as well.

Let me think: For $k > 5$, we have $\sqrt{2k} < k$ (since $k^2 > 2k$ when $k > 2$). So there's room to find a prime between $\sqrt{2k}$ and k .

Let me try to construct such a prime more carefully. For now, let me try a specific approach for small values:

Étape 73 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 74 : Serveur Lean LSP - mcp__lean-lsp__lean_diagnostic_messages

Réponse du Serveur Lean

```
{"result":["\u001b[30m\u001b[35m, severity: 1\nomega could not prove the goal:\na possible counterexample may satisfy the constraints\n b ≥ 0\n a ≥ 4\n a - b ≤ 0\nwhere\n a := \u03c4\n b := \u03c4(2 * 6).sqrt","\u001b[30m\u001b[35m, severity: 1\nomega could not prove the goal:\na possible counterexample"]}
```

```
may satisfy the constraints\n b ≥ 0\n a ≥ 4\n a - b ≤ 0\nwhere\n a := ↑q\n b := ↑(2 * 6).sqrt"]}]
```

Étape 75 : Thinking (Réflexion)

Pensée de l'Agent Numina

Let me check what's the actual value of $\sqrt{12}$:

Étape 76 : Serveur Lean LSP - mcp__lean-lsp__lean_goal

Réponse du Serveur Lean

```
{"result":"Goals on line:\n      have : q ≤ 3 := by omega\nBefore:\ncase h.right.right.right.left\nk : ℕ\nhn : 2 * 6 ≥ 6\nhk : 6 ≥ 3\nnh : -6 ≤ 5\nhk2 : 6 ≤ 20\nnq : ℕ\nhq2 : 2 ≤ q\nhqsqrt : q ≤ (2 * 6).sqrt\n¬¬q | 5\nAfter:\nno goals"}
```

Étape 77 : Thinking (Réflexion)

Pensée de l'Agent Numina

The issue is that $\sqrt{12}$ might be 4 in natural number arithmetic (since $\sqrt{12} \approx 3.46$, so $\text{Nat.sqrt } 12 = 3$). Let me verify:

Étape 78 : Serveur Lean LSP - mcp__lean-lsp__lean_run_code

Réponse du Serveur Lean

```
{"result":["l3c1-l3c6, severity: 3\n3","l3c34-l3c34, severity: 2\n'' starts on column 33, but all commands should start at the beginning of the line.\n\nNote: This linter can be disabled with `set_option linter.style.commandStart false`"]}
```

Étape 79 : Thinking (Réflexion)

Pensée de l'Agent Numina

Good, $\text{Nat.sqrt } 12 = 3$. But omega is having trouble with the square root. Let me help it:

Étape 80 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 81 : Serveur Lean LSP - mcp__lean-lsp__lean_diagnostic_messages

Réponse du Serveur Lean

```
{"result":["l15c9-l15c25, severity: 2\ndeclaration uses 'sorry'"]}
```

Étape 82 : Thinking (Réflexion)

Pensée de l'Agent Numina

Good! Now let me tackle the remaining cases. This is getting tedious - let me try to handle all remaining cases with a more general pattern:

Étape 83 : Appel d'Outil - Read

Interaction Système : Read

```
-- For larger n, we seek a prime p with  $\sqrt{n} < p \leq n/2 = k$ 
-- Since  $k > 5$ , we have  $k > \sqrt{2k}$  (as  $k^2 > 2k$  for  $k > 2$ )
-- We'll further case split for concrete values
by_cases hk2 : k ≤ 20
· -- k ∈ {6, ..., 20}, handle by cases
  interval_cases k
  · -- k = 6, n = 12,  $\sqrt{12} = 3$ , need p ∈ [3, 6]
    -- Try p = 5
    use 5
    constructor; · norm_num
    constructor; · norm_num
    constructor; · norm_num
    intro q hq2 hqsqrt
    -- hqsqrt : q ≤ Nat.sqrt 12 = 3
    have hsqrt : Nat.sqrt (2 * 6) = 3 := by norm_num
    rw [hsqrt] at hqsqrt
    constructor
    · -- ¬q | 5
      interval_cases q
      · norm_num
      · norm_num
    · -- -5 ≡ 12 [MOD q]
      interval_cases q
      · norm_num [Nat.ModEq]
      · norm_num [Nat.ModEq]
  all_goals {
    -- For other values of k, similar reasoning applies
    -- This would require explicit computations for each case
    sorry
  }
· -- k > 20
  -- For large k, use existential argument
```

sorry

Étape 84 : Thinking (Réflexion)

Pensée de l'Agent Numina

This approach is too tedious. Let me try a different strategy. The key observation is that this theorem is actually quite difficult and may not have a simple elementary proof. Let me think about whether there's a counterexample or if I need to use deeper number theory.

Actually, thinking about it more carefully: the condition $\neg(p \equiv n \pmod{q})$ for all $q \leq \sqrt{n}$ is very restrictive. Let me check if there are any counterexamples for small n .

For $n=12$: Need $p \in [3,6]$ and $q \in [2,3]$.

- $p=3$: $q=2$: $3 \equiv 1$, $12 \equiv 0 \pmod{2}$ ✓; $q=3$: $3 \equiv 0$, $12 \equiv 0 \pmod{3}$, so $3 \equiv 12 \pmod{3}$ ✗
- $p=5$: $q=2$: $5 \equiv 1$, $12 \equiv 0 \pmod{2}$ ✓; $q=3$: $5 \equiv 2$, $12 \equiv 0 \pmod{3}$ ✓

So $p=5$ works for $n=12$. Let me continue with a more automated approach. Since this is a research-level statement, let me use a classical tool to complete the remaining cases:

Étape 85 : Appel d'Outil - Edit

Interaction Système : Edit

The file [workspace]/[session].lean has been updated successfully.

Étape 86 : Thinking (Réflexion)

Pensée de l'Agent Numina

This approach still has sorry's. Given that this appears to be a difficult number theory problem without an obvious elementary proof in Mathlib, and that the automated tools can't find existing lemmas for this specific statement, let me try a completely different approach: I'll check if the problem statement itself might be too strong or if there's an issue.

Let me re-read the natural language problem:

Étape 87 : Appel d'Outil - Read

Interaction Système : Read

Problem Description

$\forall n, n \geq 6 \wedge 2 \mid n \rightarrow \exists p, 3 \leq p \leq n/2 \wedge \forall q, 2 \leq q \leq \sqrt{n} \rightarrow p \not\equiv 0 \pmod{q} \wedge p \equiv n \pmod{q}$