Arithmetic progressions and relative primes

José Manuel Rodríguez Caballero

December 14, 2021

Abstract

This article provides a formalization of the solution obtained by the author of the Problem "ARITHMETIC PROGRESSIONS" from the Putnam exam problems [1] of 2002. The statement of the problem is as follows: For which integers $n > 1$ does the set of positive integers less than and relatively prime to n constitute an arithmetic progression?

Contents

1 Problem ARITHMETIC PROGRESSIONS (Putnam exam problems 2002)

theory *Arith-Prog-Rel-Primes*
  imports
    Complex-Main
    HOL−Number-Theory.Number-Theory
begin

Statement of the problem (from [1]): For which integers $n > 1$ does the set of positive integers less than and relatively prime to n constitute an arithmetic progression?

The solution of the above problem is theorem *arith-prog-rel-primes-solution*.

First, we will require some auxiliary material before we get started with the actual solution.

1.1 Auxiliary results

lemma *even-and-odd-parts*:
  fixes *n*::*nat*

```
  assumes ‹n ≠ 0›
  shows ‹∃ k q::nat. n = (2::nat)^k*q ∧ odd q›
proof−
  have ‹prime (2::nat)›
    by simp
  thus ?thesis
    using prime-power-canonical[where p = 2 and m = n]
      assms semiring-normalization-rules(7) by auto
qed

lemma only-one-odd-div-power2:
  fixes n::nat
  assumes ‹n ≠ 0› and ‹⋀ x. x dvd n ⟹ odd x ⟹ x = 1›
  shows ‹∃ k. n = (2::nat)^k›
 by (metis even-and-odd-parts assms(1) assms(2) dvd-triv-left semiring-normalization-rules(11)
      semiring-normalization-rules(7))

lemma coprime-power2:
  fixes n::nat
  assumes ‹n ≠ 0› and ‹⋀ x. x < n ⟹ (coprime x n ⟷ odd x)›
  shows ‹∃ k. n = (2::nat)^k›
proof−
  have ‹x dvd n ⟹ odd x ⟹ x = 1›
    for x
   by (metis neq0-conv One-nat-def Suc-1 Suc-lessI assms(1) assms(2) coprime-left-2-iff-odd
        dvd-refl linorder-neqE-nat nat-dvd-1-iff-1 nat-dvd-not-less not-coprimeI)
  thus ?thesis
    using assms(1) only-one-odd-div-power2
    by auto
qed
```

1.2 Main result

The solution to the problem ARITHMETIC PROGRESSIONS (Putnam exam problems 2002)

```
theorem arith-prog-rel-primes-solution:
  fixes n :: nat
  assumes ‹n > 1›
  shows ‹(prime n ∨ (∃ k. n = 2^k) ∨ n = 6) ⟷
(∃ a b m. m ≠ 0 ∧ {x | x. x < n ∧ coprime x n} = {a+j*b| j::nat. j < m})›
proof−
  have ‹ (prime n ∨ (∃ k. n = 2^k) ∨  n = 6) ⟷
 (∃ b m. m ≠ 0 ∧ {x | x :: nat. x < n ∧ coprime x n} = {1+j*b| j::nat. j < m})›
  proof
    show ∃ b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}
      if prime n ∨ (∃ k. n = 2 ^ k) ∨ n = 6
    proof−
      have ∃ b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}
        if prime n
```

proof−
have ‹$\exists m.\ \ m \neq 0 \wedge \{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\} = \{1{+}j \mid j{::}nat.\ j < m\}$›
proof−
have ‹$\{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\} = \{x \mid x :: nat.\ x \neq 0 \wedge x < n\}$›
proof
show $\{x \mid x.\ x < n \wedge coprime\ x\ n\} \subseteq \{x \mid x.\ x \neq 0 \wedge x < n\}$
by (*smt Collect-mono not-le ord-0-nat ord-eq-0 order-refl prime-gt-1-nat that zero-neq-one*)
show $\{x \mid x.\ x \neq 0 \wedge x < n\} \subseteq \{x \mid x.\ x < n \wedge coprime\ x\ n\}$
using *coprime-commute prime-nat-iff'' that*
by *fastforce*
qed
obtain *m* **where** ‹$m{+}1 = n$›
using *add.commute assms less-imp-add-positive* **by** *blast*
have ‹$\{1{+}j \mid j{::}nat.\ j < (m{::}nat)\} = \{x \mid x :: nat.\ x \neq 0 \wedge x < m{+}1\}$›
by (*metis Suc-eq-plus1* ‹$m + 1 = n$› *gr0-implies-Suc le-simps*(*3*) *less-nat-zero-code linorder-not-less nat.simps*(*3*) *nat-neq-iff plus-1-eq-Suc*)
hence ‹$\{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\} = \{1{+}j \mid j{::}nat.\ j < (m{::}nat)\}$›
using ‹$\{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\} = \{x \mid x :: nat.\ x \neq 0 \wedge x < n\}$› ‹$m{+}1 = n$›
by *auto*
from ‹$n > 1$› **have** ‹$m \neq 0$›
using ‹$m + 1 = n$› **by** *linarith*
have ‹$\{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\} = \{1{+}j \mid j{::}nat.\ j < m\}$›
using *Suc-eq-plus1* ‹$1 < n$› ‹$\{x \mid x.\ x < n \wedge coprime\ x\ n\} = \{1 + j \mid j.\ j < m\}$›
by *auto*
hence ‹$(\exists\ m.\ \ m \neq 0 \wedge \{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\} = \{1{+}j \mid j{::}nat.\ j < m\})$›
using ‹$m \neq 0$›
by *blast*
thus *?thesis* **by** *blast*
qed
hence ‹$\exists m.\ \ m \neq 0 \wedge \{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\} = \{1{+}j{*}1 \mid j{::}nat.\ j < m\}$›
by *auto*
thus *?thesis*
by *blast*
qed
moreover have $\exists b\ m.\ m \neq 0 \wedge \{x \mid x.\ x < n \wedge coprime\ x\ n\} = \{1 + j * b \mid j.\ j < m\}$
if $\exists k.\ n = 2\ \hat{}\ k$
proof−
obtain *k* **where** ‹$n = 2\ \hat{}\ k$›
using ‹$\exists k.\ n = 2\ \hat{}\ k$› **by** *auto*
have ‹$k \neq 0$›
by (*metis* ‹$1 < n$› ‹$n = 2\ \hat{}\ k$› *nat-less-le power.simps*(*1*))

```
obtain t where ‹Suc t = k›
  by (metis ‹k ≠ 0› fib.cases)
have ‹n = 2^(Suc t)›
  by (simp add: ‹Suc t = k› ‹n = 2 ^ k›)
have ‹{x | x :: nat. x < n ∧ coprime x n} = {1+j*2| j::nat. j < 2^t}›
proof
  show {x |x. x < n ∧ coprime x n} ⊆ {1 + j * 2 |j. j < 2^t}
  proof
    fix x
    assume ‹x ∈ {x |x. x < n ∧ coprime x n}›
    hence ‹x < n›
      by blast
    have ‹coprime x n›
      using ‹x ∈ {x |x. x < n ∧ coprime x n}›
      by blast
    hence ‹coprime x (2^(Suc k))›
      by (simp add: ‹k ≠ 0› ‹n = 2 ^ k›)
    have ‹odd x›
      using ‹coprime x n› ‹k ≠ 0› ‹n = 2 ^ k›
      by auto
    then obtain j where ‹x = 1+j*2›
        by (metis add.commute add.left-commute left-add-twice mult-2-right
oddE)
    have ‹x < 2^k›
      using ‹n = 2 ^ k› ‹x < n› ‹x = 1+j*2›
      by linarith
    hence ‹1+j*2 < 2^k›
      using ‹x = 1+j*2›
      by blast
    hence ‹j < 2^t›
      using ‹Suc t = k› by auto
    thus ‹x ∈ {1 + j * 2 |j. j < 2^t}›
      using ‹x = 1+j*2›
      by blast
  qed
  show {1 + j * 2 |j. j < 2 ^ t} ⊆ {x |x. x < n ∧ coprime x n}
  proof
    fix x::nat
    assume ‹x ∈ {1 + j * 2 |j. j < 2 ^ t}›
    then obtain j where ‹x = 1 + j * 2› and ‹j < 2 ^ t›
      by blast
    have ‹x < 2*(2^t)›
      using ‹x = 1 + j * 2› ‹j < 2 ^ t›
      by linarith
    hence ‹x < n›
      by (simp add: ‹n = 2 ^ Suc t›)
    moreover have ‹coprime x n›
          by (metis (no-types) ‹⋀thesis. (⋀j. ⟦x = 1 + j * 2; j < 2 ^
t⟧ ⟹ thesis) ⟹ thesis› ‹n = 2 ^ k› coprime-Suc-left-nat coprime-mult-right-iff
```

```
coprime-power-right-iff plus-1-eq-Suc)
      ultimately show ‹x ∈ {x |x. x < n ∧ coprime x n}›
        by blast
    qed
  qed
  have ‹(2::nat)^(t::nat) ≠ 0›
    by simp
  obtain m where ‹m = (2::nat)^t› by blast
  have ‹m ≠ 0›
    using ‹m = 2 ^ t›
    by auto
  have ‹{x | x :: nat. x < n ∧ coprime x n} = {1+j*2| j::nat. j < m}›
    using ‹m = 2 ^ t› ‹{x |x. x < n ∧ coprime x n} = {1 + j * 2 |j. j < 2 ^
t}›
    by auto
  from  ‹m ≠ 0›  ‹{x | x :: nat. x < n ∧ coprime x n} = {1+j*2| j::nat. j
< m}›
  show ?thesis by blast
qed
moreover have ∃ b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {1 + j * b
|j. j < m}
  if n = 6
proof−
  have  ‹{x | x. x < 6 ∧ coprime x 6} = {1+j*4| j::nat. j < 2}›
  proof−
    have ‹{x | x::nat. x < 6 ∧ coprime x 6} = {1, 5}›
    proof
      show {u. ∃x. u = (x::nat) ∧ x < 6 ∧ coprime x 6} ⊆ {1, 5}
      proof
        fix u::nat
        assume ‹u ∈ {u. ∃x. u = x ∧ x < 6 ∧ coprime x 6}›
        hence ‹coprime u 6›
          by blast
        have ‹u < 6›
          using ‹u ∈ {u. ∃x. u = x ∧ x < 6 ∧ coprime x 6}›
          by blast
        moreover have ‹u ≠ 0›
          using ‹coprime u 6› ord-eq-0
          by fastforce
        moreover have ‹u ≠ 2›
          using ‹coprime u 6›
          by auto
        moreover have ‹u ≠ 3›
        proof−
          have ‹gcd (3::nat) 6 = 3›
            by auto
          thus ?thesis
          by (metis (no-types) ‹coprime u 6› ‹gcd 3 6 = 3› coprime-iff-gcd-eq-1
```

```
                numeral-eq-one-iff semiring-norm(86))
            qed
            moreover have ‹u ≠ 4›
            proof−
              have ‹gcd (4::nat) 6 = 2›
                   by (metis (no-types, lifting) add-numeral-left gcd-add1 gcd-add2
gcd-nat.idem
                    numeral-Bit0 numeral-One one-plus-numeral semiring-norm(4)
semiring-norm(5))
              thus ?thesis
                using ‹coprime u 6› coprime-iff-gcd-eq-1
                by auto
            qed
            ultimately have ‹u = 1 ∨ u = 5›
              by auto
            thus ‹u ∈ {1, 5}›
              by blast
          qed
          show {1::nat, 5} ⊆ {x |x. x < 6 ∧ coprime x 6}
          proof−
            have ‹(1::nat) ∈ {x |x. x < 6 ∧ coprime x 6}›
              by simp
            moreover have ‹(5::nat) ∈ {x |x. x < 6 ∧ coprime x 6}›
          by (metis Suc-numeral coprime-Suc-right-nat less-add-one mem-Collect-eq
               numeral-plus-one semiring-norm(5) semiring-norm(8))
            ultimately show ?thesis
              by blast
          qed
        qed
        moreover have ‹{1+j*4| j::nat. j < 2} = {1, 5}›
          by auto
        ultimately show ?thesis by auto
      qed
      moreover have ‹(2::nat) ≠ 0›
        by simp
      ultimately have ‹∃ m. m ≠ 0 ∧ {x | x :: nat. x < 6 ∧ coprime x 6} =
{1+j*4| j::nat. j < m}›
        by blast
      thus ?thesis
        using that
        by auto
    qed
    ultimately show ?thesis
      using that
      by blast
  qed
  show prime n ∨ (∃ k. n = 2 ^ k) ∨ n = 6
    if ∃ b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}
  proof−
```

```
    obtain b m where ‹m ≠ 0› and ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗
b |j. j < m}›
      using ‹∃ b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j <
m}›
      by auto
    show ?thesis
    proof(cases ‹n = 2›)
      case True
      thus ?thesis
        by auto
    next
      case False
      have ‹b ≤ 4›
      proof(cases ‹odd b›)
        case True
        show ?thesis
        proof(rule classical)
          assume ‹¬(b ≤ 4)›
          hence ‹b > 4›
            using le-less-linear
            by blast
          obtain m where ‹m ≠ 0›
            and ‹{x | x :: nat. x < n ∧ coprime x n} = {1+j∗b| j::nat. j < m}›
           using ‹m ≠ 0› ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
            by blast
          have ‹b ≠ 0›
            using ‹4 < b›
            by linarith
          have ‹n = 2 + (m−1)∗b›
          proof−
            have ‹n−1 ∈ {x | x :: nat. x < n ∧ coprime x n}›
              using ‹1 < n› coprime-diff-one-left-nat
              by auto
            have ‹n−1 ∈ {1+j∗b| j::nat. j < m}›
              using ‹n − 1 ∈ {x |x. x < n ∧ coprime x n}›
                ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
              by blast
            then obtain i::nat where ‹n−1 = 1+i∗b› and ‹i < m›
              by blast
            have ‹i ≤ m−1›
              using ‹i < m›
              by linarith
            have ‹1 + (m−1)∗b ∈ {1+j∗b| j::nat. j < m}›
              using ‹m ≠ 0›
              by auto
            hence ‹1 + (m−1)∗b ∈ {x | x::nat. x < n ∧ coprime x n}›
              using ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
              by blast
            hence ‹1 + (m−1)∗b < n›
```

```
      by blast
    hence ‹1 + (m−1)∗b ≤ n−1›
      by linarith
    hence ‹1 + (m−1)∗b ≤ 1+i∗b›
      using ‹n − 1 = 1 + i ∗ b›
      by linarith
    hence ‹(m−1)∗b ≤ i∗b›
      by linarith
    hence ‹m−1 ≤ i›
      using ‹b ≠ 0›
      by auto
    hence ‹m−1 = i›
      using ‹i ≤ m − 1› le-antisym
      by blast
    thus ?thesis
      using ‹m ≠ 0› ‹n − 1 = 1 + i ∗ b›
      by auto
  qed
  have ‹m ≥ 2›
    using ‹n = 2 + (m − 1)∗b› ‹n ≠ 2›
    by auto
  hence ‹1+b ∈ {1+j∗b| j. j < m}›
    by fastforce
  hence ‹1+b ∈ {x | x::nat. x < n ∧ coprime x n}›
    using ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
    by blast
  hence ‹coprime (1+b) n›
    by blast
  have ‹(2::nat) dvd (1+b)›
    using ‹odd b›
    by simp
  hence ‹coprime (2::nat) n›
 using ‹coprime (1 + b) n› coprime-common-divisor coprime-left-2-iff-odd
odd-one
    by blast
  have ‹(2::nat) < n›
    using ‹1 < n› ‹n ≠ 2›
    by linarith
  have ‹2 ∈ {x | x :: nat. x < n ∧ coprime x n}›
    using ‹2 < n› ‹coprime 2 n›
    by blast
  hence ‹2 ∈ {1+j∗b| j::nat. j < m}›
    using ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
    by blast
  then obtain j::nat where ‹2 = 1+j∗b›
    by blast
  have ‹1 = j∗b›
    using ‹2 = 1+j∗b›
    by linarith
```

```
    thus ?thesis
      by simp
  qed
next
  case False
  hence ‹even b›
    by simp
  show ?thesis
  proof(rule classical)
    assume ‹¬(b ≤ 4)›
    hence ‹b > 4›
      using le-less-linear
      by blast
    obtain m where  ‹ m ≠ 0›
      and ‹{x | x::nat. x < n ∧ coprime x n} = {1+j*b| j::nat. j < m}›
     using ‹m ≠ 0› ‹{x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}›
      by blast
    have ‹b ≠ 0›
      using ‹4 < b›
      by linarith
    have ‹n = 2 + (m−1)*b›
    proof−
      have ‹n−1 ∈ {x | x::nat. x < n ∧ coprime x n}›
        using ‹1 < n› coprime-diff-one-left-nat
        by auto
      have ‹n−1 ∈ {1+j*b| j::nat. j < m}›
        using ‹n − 1 ∈ {x |x. x < n ∧ coprime x n}›
          ‹{x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}›
        by blast
      then obtain i::nat where ‹n−1 = 1+i*b› and ‹i < m›
        by blast
      have ‹i ≤ m−1›
        using ‹i < m›
        by linarith
      have ‹1 + (m−1)*b ∈ {1+j*b| j::nat. j < m}›
        using ‹m ≠ 0›
        by auto
      hence ‹1 + (m−1)*b ∈ {x | x :: nat. x < n ∧ coprime x n}›
        using ‹{x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}›
        by blast
      hence ‹1 + (m−1)*b < n›
        by blast
      hence ‹1 + (m−1)*b ≤ n−1›
        by linarith
      hence ‹1 + (m−1)*b ≤ 1+i*b›
        using ‹n − 1 = 1 + i * b›
        by linarith
      hence ‹(m−1)*b ≤ i*b›
        by linarith
```

```
      hence ‹m−1 ≤ i›
        using ‹b ≠ 0›
        by auto
      hence ‹m−1 = i›
        using ‹i ≤ m − 1› le-antisym
        by blast
      thus ?thesis
        using ‹m ≠ 0› ‹n − 1 = 1 + i ∗ b›
        by auto
    qed
    obtain k :: nat where ‹b = 2∗k›
      using ‹even b›
      by blast
    have ‹n = 2∗(1 + (m−1)∗k)›
      using ‹n = 2 + (m−1)∗b› ‹b = 2∗k›
      by simp
    show ?thesis
    proof (cases ‹odd m›)
      case True
      hence ‹odd m› by blast
      then obtain t::nat where ‹m−1 = 2∗t›
        by (metis odd-two-times-div-two-nat)
      have ‹n = 2∗(1 + b∗t)›
        using ‹m − 1 = 2 ∗ t› ‹n = 2 + (m − 1) ∗ b›
        by auto
      have ‹t < m›
        using ‹m − 1 = 2 ∗ t› ‹m ≠ 0›
        by linarith
      have ‹1 + b∗t ∈ {1+j∗b| j::nat. j < m}›
        using ‹t < m›
        by auto
      hence ‹1 + b∗t ∈ {x | x::nat. x < n ∧ coprime x n}›
        using ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
        by blast
      hence ‹coprime (1 + b∗t) n›
        by auto
      thus ?thesis
         by (metis (no-types, lifting) ‹b = 2 ∗ k› ‹n = 2 ∗ (1 + (m − 1) ∗
k)› ‹n = 2 ∗ (1 + b ∗ t)› ‹n = 2 + (m − 1) ∗ b› ‹n ≠ 2› add-cancel-right-right
coprime-mult-right-iff coprime-self mult-cancel-left mult-is-0 nat-dvd-1-iff-1)
    next
      case False
      thus ?thesis
      proof(cases ‹odd k›)
        case True
        hence ‹odd k›
          by blast
        have ‹even (1 + (m − 1) ∗ k)›
          by (simp add: False True ‹m ≠ 0›)
```

```
        have ‹coprime (2 + (m − 1) ∗ k) (1 + (m − 1) ∗ k)›
          by simp
        have ‹coprime (2 + (m − 1) ∗ k) n›
         using ‹coprime (2 + (m − 1) ∗ k) (1 + (m − 1) ∗ k)› ‹even (1 +
(m − 1) ∗ k)›
              ‹n = 2 ∗ (1 + (m − 1) ∗ k)› coprime-common-divisor
coprime-mult-right-iff
           coprime-right-2-iff-odd odd-one
          by blast
        have ‹2 + (m − 1) ∗ k < n›
         by (metis (no-types, lifting) ‹even (1 + (m − 1) ∗ k)› ‹n = 2 ∗ (1
+ (m − 1) ∗ k)›
            add-gr-0 add-mono-thms-linordered-semiring(1) dvd-add-left-iff
dvd-add-triv-left-iff dvd-imp-le le-add2 le-neq-implies-less less-numeral-extra(1) mult-2
odd-one)
        have ‹2 + (m − 1) ∗ k ∈ {x | x :: nat. x < n ∧ coprime x n}›
          using ‹2 + (m − 1) ∗ k < n› ‹coprime (2 + (m − 1) ∗ k) n›
          by blast
        hence ‹2 + (m − 1) ∗ k ∈ {1 + j ∗ b |j. j < m}›
          using ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
          by blast
        then obtain j::nat where ‹2 + (m − 1) ∗ k = 1 + j ∗ b› and ‹j
< m›
          by blast
        have ‹1 + (m − 1) ∗ k = j ∗ b›
          using ‹2 + (m − 1) ∗ k = 1 + j ∗ b›
          by simp
        hence ‹1 + (m − 1) ∗ k = j ∗ (2∗k)›
          using ‹b = 2 ∗ k› by blast
        thus ?thesis
         by (metis ‹b = 2 ∗ k› ‹even b› ‹n = 2 ∗ (1 + (m − 1) ∗ k)› ‹n = 2
+ (m − 1) ∗ b› dvd-add-times-triv-right-iff dvd-antisym dvd-imp-le dvd-triv-right
even-numeral mult-2 zero-less-numeral)
      next
        case False
        hence ‹even k› by auto
        have ‹odd (1 + (m − 1) ∗ k)›
          by (simp add: ‹even k› )
        hence ‹coprime (3 + (m − 1) ∗ k) (1 + (m − 1) ∗ k)›
      by (smt add-numeral-left coprime-common-divisor coprime-right-2-iff-odd
dvd-add-left-iff not-coprimeE numeral-Bit1 numeral-One numeral-plus-one one-add-one)
        hence ‹coprime (3 + (m − 1) ∗ k) n›
       by (metis ‹even k› ‹n = 2 ∗ (1 + (m − 1) ∗ k)› coprime-mult-right-iff
coprime-right-2-iff-odd even-add even-mult-iff odd-numeral)
        have ‹3 + (m − 1) ∗ k < n›
              by (smt Groups.add-ac(2) ‹even k› ‹n = 2 ∗ (1 + (m −
1) ∗ k)› ‹n = 2 + (m − 1) ∗ b› ‹n ≠ 2› add-Suc-right add-cancel-right-right
add-mono-thms-linordered-semiring(1) dvd-imp-le even-add even-mult-iff le-add2
le-neq-implies-less left-add-twice mult-2 neq0-conv numeral-Bit1 numeral-One odd-numeral
```

```
one-add-one plus-1-eq-Suc)
        have ‹3 + (m − 1) ∗ k ∈ {x |x. x < n ∧ coprime x n}›
          using ‹3 + (m − 1) ∗ k < n› ‹coprime (3 + (m − 1) ∗ k) n›
          by blast
        hence ‹3 + (m − 1) ∗ k ∈ {1 + j ∗ b |j. j < m}›
          using ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
          by blast
        then obtain j::nat where ‹3 + (m − 1) ∗ k = 1 + j ∗ b›
          by blast
        have ‹2 + (m − 1) ∗ k = j ∗ b›
          using ‹3 + (m − 1) ∗ k = 1 + j ∗ b›
          by simp
        hence  ‹2 + (m − 1) ∗ k = j ∗ 2∗k›
          by (simp add: ‹b = 2 ∗ k›)
        thus ?thesis
          by (metis ‹4 < b› ‹b = 2 ∗ k› ‹even k› dvd-add-times-triv-right-iff
dvd-antisym
              dvd-triv-right mult-2 nat-neq-iff numeral-Bit0)
      qed
    qed
  qed
qed
moreover have ‹b ≠ 3›
proof (rule classical)
  assume ‹¬ (b ≠ 3)›
  hence ‹b = 3›
    by blast
  obtain m where  ‹m ≠ 0› and
    ‹{x | x::nat. x < n ∧ coprime x n} = {1+j∗b| j::nat. j < m}›
    using ‹m ≠ 0› ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
    by blast
  have ‹b ≠ 0›
    by (simp add: ‹b = 3›)
  have ‹n = 2 + (m−1)∗b›
  proof−
    have ‹n−1 ∈ {x | x::nat. x < n ∧ coprime x n}›
      using ‹1 < n› coprime-diff-one-left-nat
      by auto
    have ‹n−1 ∈ {1+j∗b| j::nat. j < m}›
      using ‹n − 1 ∈ {x |x. x < n ∧ coprime x n}›
        ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
      by blast
    then obtain i::nat where ‹n−1 = 1+i∗b› and ‹i < m›
      by blast
    have ‹i ≤ m−1›
      using ‹i < m›
      by linarith
    have ‹1 + (m−1)∗b ∈ {1+j∗b| j::nat. j < m}›
      using ‹m ≠ 0›
```

    by *auto*
  hence ‹$1 + (m-1)*b \in \{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\}$›
    using ‹$\{x \mid x.\ x < n \wedge coprime\ x\ n\} = \{1 + j * b \mid j.\ j < m\}$›
    by *blast*
  hence ‹$1 + (m-1)*b < n$›
    by *blast*
  hence ‹$1 + (m-1)*b \leq n-1$›
    by *linarith*
  hence ‹$1 + (m-1)*b \leq 1+i*b$›
    using ‹$n - 1 = 1 + i * b$›
    by *linarith*
  hence ‹$(m-1)*b \leq i*b$›
    by *linarith*
  hence ‹$m-1 \leq i$›
    using ‹$b \neq 0$›
    by *auto*
  hence ‹$m-1 = i$›
    using ‹$i \leq m - 1$› *le-antisym*
    by *blast*
  thus *?thesis*
    using ‹$m \neq 0$› ‹$n - 1 = 1 + i * b$›
    by *auto*
qed
have ‹$n > 2$›
  using ‹$1 < n$› ‹$n \neq 2$›
  by *linarith*
hence ‹ $m \geq 2$ › **using** ‹$n = 2 + (m-1)*b$› ‹$b = 3$›
  by *simp*
have ‹$4 \in \{1+j*(b::nat) \mid j::nat.\ j < m\}$›
  using ‹$2 \leq m$› ‹$b = 3$›
  by *force*
hence ‹$(4::nat) \in \{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\}$›
  using ‹$\{x \mid x.\ x < n \wedge coprime\ x\ n\} = \{1 + j * b \mid j.\ j < m\}$›
  by *auto*
hence ‹$coprime\ (4::nat)\ n$›
  by *blast*
have ‹$(2::nat)\ dvd\ 4$›
  by *auto*
hence ‹$coprime\ (2::nat)\ n$›
  using ‹$coprime\ (4::nat)\ n$› *coprime-divisors dvd-refl*
  by *blast*
have ‹$4 < n$›
  using ‹$4 \in \{x \mid x.\ x < n \wedge coprime\ x\ n\}$›
  by *blast*
have ‹$2 < (4::nat)$›
  by *auto*
have ‹$2 < n$›
  by (*simp add*: ‹$2 < n$›)
hence ‹$2 \in \{x \mid x :: nat.\ x < n \wedge coprime\ x\ n\}$›

```
      using ‹coprime (2::nat) n›
      by blast
    hence ‹2 ∈ {1+j*(b::nat)| j::nat. j < m}›
      using ‹{x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}›
      by blast
    then obtain j::nat where ‹2 = 1+j*3›
      using ‹b = 3›
      by blast
    from ‹2 = 1+j*3›
    have ‹1 = j*3›
      by auto
    hence ‹3 dvd 1›
      by auto
    thus ?thesis
      using nat-dvd-1-iff-1 numeral-eq-one-iff
      by blast
  qed
  ultimately have ‹b = 0 ∨ b = 1 ∨ b = 2 ∨ b = 4›
    by auto
  moreover have ‹b = 0 ⟹ ∃ k. n = 2^k›
  proof−
    assume ‹b = 0›
    have ‹{1 + j * b |j. j < m} = {1}›
      using ‹b = 0› ‹m ≠ 0›
      by auto
    hence ‹{x |x. x < n ∧ coprime x n} = {1}›
      using ‹{x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}›
      by blast
    hence ‹n = 2›
    proof−
      have ‹n−1 ∈  {x | x :: nat. x < n ∧ coprime x n}›
        using ‹1 < n› coprime-diff-one-left-nat
        by auto
      have ‹n−1 ∈ {1}›
       using ‹n − 1 ∈ {x |x. x < n ∧ coprime x n}› ‹{x |x. x < n ∧ coprime
x n} = {1}›
        by blast
      hence ‹n−1 = 1›
        by blast
      hence ‹n = 2›
        by simp
      thus ?thesis
        by blast
    qed
    hence ‹n = 2^1›
      by auto
    thus ?thesis
      by blast
  qed
```

```
    moreover have ‹b = 1 ⟹ prime n›
    proof−
      assume ‹b = 1›
      have ‹x < n ⟹ x ≠ 0 ⟹ coprime x n›
        for x
      proof−
        assume ‹x < n› and ‹x ≠ 0›
        have ‹{1+j| j::nat. j < m} = {x | x::nat. x < m+1 ∧ x ≠ 0}›
              by (metis (full-types) Suc-eq-plus1  add-mono1 less-Suc-eq-0-disj
nat.simps(3) plus-1-eq-Suc )
        hence ‹{x | x :: nat. x < n ∧ coprime x n} = {x | x :: nat. x < m+1 ∧
x ≠ 0}›
          using ‹b = 1› ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
          by auto
        have ‹coprime (n−1) n›
          using ‹1 < n› coprime-diff-one-left-nat
          by auto
        have ‹n−1 < n›
          using ‹1 < n›
          by auto
        have ‹n−1 ∈ {x |x. x < n ∧ coprime x n}›
          using ‹coprime (n − 1) n› ‹n − 1 < n›
          by blast
        have ‹n−1 ≤ m›
          by (metis (no-types, lifting) CollectD Suc-eq-plus1 Suc-less-eq2 ‹n − 1
∈ {x |x. x < n ∧ coprime x n}› ‹{x |x. x < n ∧ coprime x n} = {x |x. x < m +
1 ∧ x ≠ 0}›  leD  le-less-linear not-less-eq-eq )
        have ‹m ∈  {x | x :: nat. x < m+1 ∧ x ≠ 0}›
          using ‹m ≠ 0›
          by auto
        have ‹m ∈ {x |x. x < n ∧ coprime x n} ›
          using ‹m ∈ {x |x. x < m + 1 ∧ x ≠ 0}›
            ‹{x |x. x < n ∧ coprime x n} = {x |x. x < m + 1 ∧ x ≠ 0}›
          by blast
        have ‹m < n›
          using ‹m ∈ {x |x. x < n ∧ coprime x n}›
          by blast
        have ‹m+1 = n›
          using ‹m < n› ‹n − 1 ≤ m›
          by linarith
        have ‹x ∈  {x | x :: nat. x < m+1 ∧ x ≠ 0}›
          using ‹m + 1 = n› ‹x < n› ‹x ≠ 0›
          by blast
        hence ‹x ∈ {x |x. x < n ∧ coprime x n}›
          using ‹{x |x. x < n ∧ coprime x n} = {x |x. x < m + 1 ∧ x ≠ 0}›
          by blast
        thus ?thesis
          by blast
      qed
```

```
      thus ?thesis
        using assms coprime-commute nat-neq-iff prime-nat-iff'' by auto
    qed
    moreover have ‹b = 2 ⟹ ∃ k. n = 2^k›
    proof−
      assume ‹b = 2›
      have ‹{x | x :: nat. x < n ∧ coprime x n} = {1+j∗2| j::nat. j < m}›
        using ‹b = 2› ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
        by auto
      have ‹x < n ⟹ coprime x n ⟷ odd x›
        for x::nat
      proof−
        assume ‹x < n›
        have ‹coprime x n ⟹ odd x›
        proof−
          assume ‹coprime x n›
          have ‹x ∈ {x | x :: nat. x < n ∧ coprime x n}›
            by (simp add: ‹coprime x n› ‹x < n›)
          hence ‹x ∈ {1+j∗2| j::nat. j < m}›
            using ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ 2 |j. j < m}›
            by blast
          then obtain j where ‹x = 1+j∗2›
            by blast
          thus ?thesis
            by simp
        qed
        moreover have ‹odd x ⟹ coprime x n›
        proof−
          assume ‹odd x›
          obtain j::nat where ‹x = 1+j∗2›
          by (metis ‹odd x› add.commute mult-2-right odd-two-times-div-two-succ
one-add-one semiring-normalization-rules(16))
          have ‹j < (n−1)/2›
            using ‹x < n› ‹x = 1 + j ∗ 2›
            by linarith
          have ‹n = 2∗m›
          proof−
            have  ‹(2::nat) ≠ 0›
              by auto
            have ‹n = 2+(m−1)∗2›
            proof−
              have ‹n−1 ∈ {x | x :: nat. x < n ∧ coprime x n}›
                using ‹1 < n› coprime-diff-one-left-nat
                by auto
              have ‹n−1 ∈ {1+j∗b| j::nat. j < m}›
                using ‹n − 1 ∈ {x |x. x < n ∧ coprime x n}›
                  ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
                by blast
              then obtain i::nat where ‹n−1 = 1+i∗b› and ‹i < m›
```

```
        by blast
      have ‹i ≤ m−1›
        using ‹i < m›
        by linarith
      have ‹1 + (m−1)*b ∈ {1+j*b| j::nat. j < m}›
        using ‹m ≠ 0› by auto
      hence ‹1 + (m−1)*b ∈ {x | x :: nat. x < n ∧ coprime x n}›
        using ‹{x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}›
        by blast
      hence ‹1 + (m−1)*b < n›
        by blast
      hence ‹1 + (m−1)*b ≤ n−1›
        by linarith
      hence ‹1 + (m−1)*b ≤ 1+i*b›
        using ‹n − 1 = 1 + i * b›
        by linarith
      hence ‹(m−1)*b ≤ i*b›
        by linarith
      hence ‹m−1 ≤ i›
      proof−
        have ‹b ≠ 0›
          using ‹b = 2›
          by simp
        thus ?thesis
          using ‹(m − 1) * b ≤ i * b› mult-le-cancel2
          by blast
      qed
      hence ‹m−1 = i›
        using ‹i ≤ m − 1› le-antisym
        by blast
      thus ?thesis
        using ‹m ≠ 0› ‹n − 1 = 1 + i * b›
        by (simp add: ‹b = 2›)
    qed
    thus  ?thesis
         by (simp add: ‹m ≠ 0› ‹n = 2 + (m − 1) * 2› mult.commute
mult-eq-if)
  qed
  hence ‹j < m›
    using ‹x < n› ‹x = 1 + j * 2›
    by linarith
  hence ‹x ∈ {1+j*2| j::nat. j < m}›
    using ‹x = 1 + j * 2›
    by blast
  hence ‹x ∈ {x | x :: nat. x < n ∧ coprime x n}›
    using ‹{x |x. x < n ∧ coprime x n} = {1 + j * 2 |j. j < m}›
    by blast
  thus ?thesis
    by blast
```

```
    qed
    ultimately show ?thesis
      by blast
  qed
  thus ?thesis
    using coprime-power2 assms
    by auto
qed
moreover have ‹b = 4 ⟹ n = 6›
proof−
  assume ‹b = 4›
  have ‹n = 2 ∨ n = 6›
  proof(rule classical)
    assume ‹¬ (n = 2 ∨ n = 6)›
    have  ‹(4::nat) ≠ 0›
      by auto
    have ‹n =  2+(m−1)∗4›
    proof−
      have ‹n−1 ∈ {x | x :: nat. x < n ∧ coprime x n}›
        using ‹1 < n› coprime-diff-one-left-nat
        by auto
      have ‹n−1 ∈ {1+j∗b| j::nat. j < m}›
        using ‹n − 1 ∈ {x |x. x < n ∧ coprime x n}›
          ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
        by blast
      then obtain i::nat where ‹n−1 = 1+i∗b› and ‹i < m›
        by blast
      have ‹i ≤ m−1›
        using ‹i < m›
        by linarith
      have ‹1 + (m−1)∗b ∈ {1+j∗b| j::nat. j < m}›
        using ‹m ≠ 0›
        by auto
      hence ‹1 + (m−1)∗b ∈ {x | x :: nat. x < n ∧ coprime x n}›
        using ‹{x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}›
        by blast
      hence ‹1 + (m−1)∗b < n›
        by blast
      hence ‹1 + (m−1)∗b ≤ n−1›
        by linarith
      hence ‹1 + (m−1)∗b ≤ 1+i∗b›
        using ‹n − 1 = 1 + i ∗ b›
        by linarith
      hence ‹(m−1)∗b ≤ i∗b›
        by linarith
      hence ‹m−1 ≤ i›
      proof−
        have ‹b ≠ 0›
          using ‹b = 4› by auto
```

```
          thus ?thesis
            using ‹(m - 1) * b ≤ i * b› mult-le-cancel2
            by blast
        qed
        hence ‹m−1 = i›
          using ‹i ≤ m - 1› le-antisym
          by blast
        thus ?thesis
          using ‹m ≠ 0› ‹n - 1 = 1 + i * b›
          by (simp add: ‹b = 4›)
      qed
      hence ‹n = 4*m - 2›
        by (simp add: ‹m ≠ 0› mult.commute mult-eq-if)
      have ‹m ≥ 3›
        using ‹¬ (n = 2 ∨ n = 6)› ‹n = 2 + (m - 1) * 4›
        by auto
      hence ‹ {1+j*4| j::nat. j < 3} ⊆ {1+j*4| j::nat. j < m}›
        by force
      hence ‹9 ∈ {1+j*4| j::nat. j < 3}›
        by force
      hence ‹9 ∈ {1+j*4| j::nat. j < m}›
        using ‹ {1+j*4| j::nat. j < 3} ⊆ {1+j*4| j::nat. j < m}›
        by blast
      hence ‹9 ∈ {x | x :: nat. x < n ∧ coprime x n}›
       using ‹b = 4› ‹{x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}›
        by auto
      hence ‹coprime (9::nat) n›
        by blast
      have ‹(3::nat) dvd 9›
        by auto
     have ‹coprime (3::nat) n› using ‹coprime (9::nat) n› ‹(3::nat) dvd 9›
        by (metis coprime-commute coprime-mult-right-iff dvd-def)
      have ‹(3::nat) < n›
       by (metis One-nat-def Suc-lessI ‹1 < n› ‹¬ (n = 2 ∨ n = 6)› ‹coprime
3 n›
              coprime-self numeral-2-eq-2 numeral-3-eq-3 less-numeral-extra(1)
nat-dvd-not-less)
      have ‹3 ∈ {x | x :: nat. x < n ∧ coprime x n}›
        using ‹3 < n› ‹coprime 3 n›
        by blast
      hence ‹(3::nat) ∈ {1+j*4| j::nat. j < m}›
       using ‹b = 4› ‹{x |x. x < n ∧ coprime x n} = {1 + j * b |j. j < m}›
        by blast
      then obtain j::nat where ‹(3::nat) = 1 + j*4›
        by blast
      have ‹2 = j*4›
        using numeral-3-eq-3 ‹(3::nat) = 1 + j*4›
        by linarith
      hence ‹1 = j*2›
```

```
            by linarith
          hence ‹even 1›
            by simp
          thus ?thesis
            using odd-one
            by blast
        qed
        thus ?thesis
          by (simp add: False)
      qed
      ultimately show ?thesis
        by blast
    qed
  qed
qed
 moreover have ‹(∃ b m. m ≠ 0 ∧ {x | x :: nat. x < n ∧ coprime x n} =
{1+j∗b| j::nat. j < m})
 ⟷ (∃ a b m. m ≠ 0 ∧ {x | x :: nat. x < n ∧ coprime x n} = {a+j∗b| j::nat. j
< m})›
 proof
   show ∃a b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {a + j ∗ b |j. j < m}
     if ∃b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}
     using that
     by blast
   show ∃b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {1 + j ∗ b |j. j < m}
     if ∃a b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {a + j ∗ b |j. j < m}
   proof−
     obtain a b m::nat where ‹m ≠ 0›
       and ‹{x | x :: nat. x < n ∧ coprime x n} = {a+j∗b| j::nat. j < m}›
       using ‹∃a b m. m ≠ 0 ∧ {x |x. x < n ∧ coprime x n} = {a + j ∗ b |j. j <
m}›
       by auto
     have ‹a = 1›
     proof−
       have ‹{x | x :: nat. x < n ∧ coprime x n} = {(a::nat)+j∗(b::nat)| j::nat. j
< m} ⟹ a = 1›
       proof−
         have ‹Min {x | x :: nat. x < n ∧ coprime x n} = Min {a+j∗b| j::nat. j
< m}›
           using ‹{x |x. x < n ∧ coprime x n} = {a + j ∗ b |j. j < m}›
           by auto
         have ‹Min {x | x :: nat. x < n ∧ coprime x n} = 1›
         proof−
           have ‹finite {x | x :: nat. x < n ∧ coprime x n}›
             by auto
           have ‹{x | x :: nat. x < n ∧ coprime x n} ≠ {}›
             using ‹1 < n› by auto
           have ‹1 ∈ {x | x :: nat. x < n ∧ coprime x n}›
             using ‹1 < n›
```

by *auto*
have ‹∀ *x*. *coprime x n* ⟶ *x* ≥ *1*›
using ‹*1* < *n*› *le-less-linear*
by *fastforce*
hence ‹∀ *x*. *x* < *n* ∧ *coprime x n* ⟶ *x* ≥ *1*›
by *blast*
hence ‹∀ *x* ∈ {*x* | *x* :: *nat*. *x* < *n* ∧ *coprime x n*}. *x* ≥ *1*›
by *blast*
hence ‹*Min* {*x* | *x* :: *nat*. *x* < *n* ∧ *coprime x n*} ≥ *1*›
using ‹*finite* {*x* | *x* :: *nat*. *x* < *n* ∧ *coprime x n*}› ‹{*x* |*x*. *x* < *n* ∧ *coprime x n*} ≠ {}›
by *auto*
thus *?thesis*
using *Min-le* ‹*1* ∈ {*x* |*x*. *x* < *n* ∧ *coprime x n*}› ‹*finite* {*x* |*x*. *x* < *n* ∧ *coprime x n*}›
antisym **by** *blast*
qed
have ‹*Min* {*a*+*j*∗*b*| *j*::*nat*. *j* < *m*} = *a*›
proof −
have *f1*: ∃ *n*. *a* = *a* + *n* ∗ *b* ∧ *n* < *m*
using ‹*m* ≠ *0*›
by *auto*
have *f2*: ∃ *n*. *1* = *a* + *n* ∗ *b* ∧ *n* < *m*
using ‹{*x* |*x*. *x* < *n* ∧ *coprime x n*} = {*a* + *j* ∗ *b* |*j*. *j* < *m*}› *assms coprime-1-left*
by *blast*
have *f3*: ∃ *na*. *a* = *na* ∧ *na* < *n* ∧ *coprime na n*
using *f1* ‹{*x* |*x*. *x* < *n* ∧ *coprime x n*} = {*a* + *j* ∗ *b* |*j*. *j* < *m*}› **by** *blast*
have *n* ≠ *1*
by (*metis* (*lifting*) *assms less-irrefl-nat*)
then have ¬ *coprime 0 n*
by *simp*
then show *?thesis*
using *f3 f2* **by** (*metis* ‹*Min* {*x* |*x*. *x* < *n* ∧ *coprime x n*} = *1*› ‹{*x* |*x*. *x* < *n* ∧ *coprime x n*} = {*a* + *j* ∗ *b* |*j*. *j* < *m*}› *less-one linorder-neqE-nat not-add-less1*)
qed
hence ‹*Min* {*a*+*j*∗*b*| *j*::*nat*. *j* < *m*} = *a*› **by** *blast*
thus *?thesis*
using ‹*Min* {*x* | *x* :: *nat*. *x* < *n* ∧ *coprime x n*} = *1*› ‹*Min* {*x* | *x* :: *nat*. *x* < *n* ∧ *coprime x n*} = *Min* {*a*+*j*∗*b*| *j*::*nat*. *j* < *m*}›
by *linarith*
qed
thus *?thesis*
using ‹{*x* |*x*. *x* < *n* ∧ *coprime x n*} = {*a* + *j* ∗ *b* |*j*. *j* < *m*}›
by *blast*
qed
thus *?thesis* **using** ‹*m* ≠ *0*› ‹{*x* | *x*. *x* < *n* ∧ *coprime x n*} = {*a*+*j*∗*b*| *j*::*nat*.

$j < m\}$›
        by *auto*
    qed
  qed
  ultimately show *?thesis*
    by *simp*
qed

end

References

[1] Problem "ARITHMETIC PROGRESSIONS", from Putnam exam problems 2002, https://www.ocf.berkeley.edu/ wwu/riddles/putnam.shtml.