Coin theorem

Daniel Buth & Sebastian Meier

June 2025

1 introduction

$\mathbb{N}_0$ denotes the natural numbers starting with zero.

Definition 1.1. *A ring (with 1) is a set R with two operations*

$$\begin{array}{rcll} + & : & R \times R \to R & \textit{(addition)} \\ \cdot & : & R \times R \to R & \textit{(multiplication)} \end{array}$$

such that

1. *(R,+) is a abelian group*
2. *(R,·) is a monoid*
3. *$\forall a, b, c \in R$ follows*

$$\begin{array}{rcl} (a+b)c & = & ac + bc \\ c(a+b) & = & ca + cb \end{array}$$

(distributive law)

Remark. *R is commutative if multiplication is commutative.*

Remark. *$\mathbb{Z}$ is commutative.*

Definition 1.2. *A subset S of a ring R is called a subring if S is with the multiplication a submonoid of (R,·) and with the addition a subgroup of (R,+). In particular, S is, with the restricted operations itself, a ring. For each ring R, we have $R^\star = \{a \in R : \exists b \in R, ab = 1\}$ the set of multiplicatively invertible elements. $R^\star$ is with the multiplication a group. $R^\star$ is also called identity group. An element $a \in R$ is called zero divisor if it exists a $0 \neq b \in R$ such that $ab = 0$ or $ba = 0$. A commutative ring $R \neq 0$ is called integral domain or nonzero if R has no zero divisor except 0*

Remark. *$\mathbb{Z}$ is a integral domain and $\mathbb{Z}^\star = \{-1, 1\}$.*

Definition 1.3. *Let R be a ring. A subset $\boldsymbol{a} \in R$ is called a ideal, if*

1. *$\boldsymbol{a}$ is a additive subgroup of R.*
2. *$r \in R, a \in \boldsymbol{a} \Rightarrow ra \in \boldsymbol{a}$*

Remark. *In each ring it contains the trivial rings namely the zero ideal denoted by (0) and R. For ideals $\boldsymbol{a}, \boldsymbol{b} \subseteq R$ are the following subsets of R ideals:*

$$\begin{array}{rcl} \boldsymbol{a} + \boldsymbol{b} & := & \{a + b : a \in \boldsymbol{a}, b \in \boldsymbol{b}\} \\ \boldsymbol{ab} & := & \{\sum_{i \in I}^{n} a_i b_i : a_i \in \boldsymbol{a}, b \in \boldsymbol{b}, n \in \mathbb{N}\} \\ \boldsymbol{a} \cap \boldsymbol{b} & & \end{array}$$

The sum of any number of ideals is an ideal. For each $a \in R$, we call $(a) := Ra = \{ra : r \in R\}$ the generated principal ideal of a. For finite elements $a_i, \dots, a_n \in R$ is $(a_i, \dots, a_n) := Ra_1 + \dots + Ra_n = \{r_1 a_1 + \dots + r_n a_n : r_1, \dots, r_n \in R\}$ an ideal in R. It is called the generated ideal of $a_1, \dots a_n$ and it is the smallest ideal that contains $a_1, \dots a_n$. Analogously, any number of $a_i \in R, i \in I$ generates the ideal $(\{a_i : i \in I\}) := \sum_{i \in I} Ra_i = \{\sum_{i \in I} r_i a_i : r_i \in R,$ for almost all $r_i = 0\}$.

Definition 1.4. *Let $\boldsymbol{a}$ be an ideal of a ring R. A family $(a_i)_{i \in I}$ of elements in $\boldsymbol{a}$ is called generator of $\boldsymbol{a}$ if $\boldsymbol{a} = \sum_{i \in I} Ra_i$. $\boldsymbol{a}$ is called finitely generated if $\boldsymbol{a}$ contains a finite generator. $\boldsymbol{a}$ is called a principal ideal if $\boldsymbol{a}$ = (a) for an $a \in R$. Is R an integral domain such that each ideal in R is a principal ideal, then R is called a principal ideal domain.*

Remark. *$\mathbb{Z}$ is a principal ideal domain*

Remark. *In an integral domain two principal ideals $\boldsymbol{a} = (a)$, $\boldsymbol{b} = (b)$ are the same if and only if $\exists c \in R^\star, b = ca$.*

Remark. *For a,b $\in R$, we call them associated to another, if $\exists c \in R^\star, b = ca$.*

Definition 1.5. *Let R be a commutative ring and let $a, b \in R$. a is divisible by b if there $\exists c \in R, a = cb$. We write $a|b$.*

Remark. *Let R be a commutative ring and let $a, b \in R$. Then the following statements are equivalent:*

1. $a|b$
2. $(b) \subseteq (a)$
3. $span_R\{b\} \leq span_R\{a\}$

Theorem 1.1. *Let R be a commutative semiring and let a,b $\in R$. Then $a|b \leftrightarrow b \in$ (a).*

Definition 1.6. *A ring is called noetherian, if each ascending chain of ideals $\boldsymbol{a}_1 \subseteq \boldsymbol{a}_2 \subseteq \cdots \subseteq R$ is stationary, such that there exists $n \in \mathbb{N}$, $\boldsymbol{a}_i = \boldsymbol{a}_n$ for all $i \geq n$.*

Definition 1.7. *Let R be an integral domain and $x_1, x_2, \ldots, x_n \in R$. $d \in R$ is called the greatest common divisor of $x_1, x_2, \ldots, x_n$ or $d = gcd(x_1, x_2, \ldots, x_n)$ if $(x_1, x_2, \ldots, x_n)$ is a principal ideal such that $(d) = (x_1, x_2, \ldots, x_n)$.*

Remark. *$\mathbb{Z}$-Module $\mathbb{Z} \cong \mathbb{Z}$*

Source: https://www.uni-frankfurt.de/115698565/algebra.pdf

2 Statements

Our goal in this seminar is to prove in LEAN:

Theorem 2.1 (Coin Theorem)**.** *Let s be a set of natural numbers.*
If gcd(s) = 1, then all sufficiently large natural numbers are representable as an ‘$\mathbb{N}$‘-linear combination of s.
If gcd(s) $\neq$ 1, then all sufficiently large multiples of the gcd(s) are representable.

As a consequence, all $\mathbb{N}$-submodules of $\mathbb{N}$ are finitely generated (equivalently, all additive submonoids of the natural numbers are finitely generated).

First, let's define “setGcd”, a function that will often be used.

```
noncomputable def setGcd (s : Set ℕ) : ℕ :=
  (Submodule.IsPrincipal.generator <| Ideal.span <| ((↑) : ℕ → ℤ) '' s).natAbs
```

The function gets as a parameter a set of natural numbers and returns a natural number by first get a set of non negative integers (denoted by s') by mapping the natural numbers to integers by the canonical homomorphism on s, then get an $\mathbb{Z}-$ideal (denoted by I) generated by the set of integers s'. $\mathbb{Z}$ is a PID (Principal Ideal Domain) and therefore there exists a $gen \in \mathbb{Z}$ such that I = (gen). At last, we compute the absolute value of gen (denoted as setGcd s or |gen|).
Remark(1): The gcd of a set of natural numbers is defined to be the nonnegative generator of the ideal of ‘$\mathbb{Z}$‘ generated by them.
Remark(2): "noncomputable" is a modifier. A function, that is marked as noncomputable, is not compiled and cannot be executed. A function must be marked as noncomputable if they use the axiom of choice or law of excluded middle to produce data. In our case, "Submodule.isPrincipal.generator" is marked as "noncomputeable" by "Classical.choose" and therefore "setGcd" must be "noncomputable".
Remark(3): The function "setGcd" is well-defined.

Next, we will state and prove the following lemmas (informally and formally in LEAN), which will help us prove the Coin Theorem (see equation (2.1)). The lemmas build on each other such that providing a proof for (2.9) proves the main statement of the Coin Theorem, and providing proofs for (2.10) and (2.11) proves the consequences of the Coin Theorem.

Lemma 2.2. *Let s be a set of $\mathbb{N}_0$ and $n \in \mathbb{N}_0$. If $n \in s$, then setGcd $s \mid n$*

Proof (informal):
Let n ∈ s.
The idea is to state explicitly the information that is needed to compute setGcd and use them to prove setGcd s | n. From setGcd, we get

1. the set of integers s' which is computed by the canonical homomorphism (↑) from $\mathbb{N} \to \mathbb{Z}$ on the set s,
2. I as the ideal generated by the set s'. (or I = (s')) and
3. gen $\in \mathbb{Z}$ as the generator of I.

That implies

i) I = (gen) and

ii) setGcd s = |gen|

(a): First, we want to prove that $(\uparrow)(n) \in I$.

Proof.

$$(\uparrow)(n) \in I \overset{2}{\iff} (\uparrow)(n) \in (s')$$

I suffices to show that $(\uparrow)(n) \in s'$ because the ideal generated with set s' contains s'. By the definition of the s' as the image of (↑) on s, $(\uparrow)(n) \in s'$. □

(b): Second, we want to prove gen | $(\uparrow)(n) \in s'$ by using (a)

Proof.

$$\text{gen } |(\uparrow)(n) \overset{remark}{\iff} (\uparrow)(n) \in (\text{gen}) \overset{i)}{\iff} (\uparrow)(n) \in I$$

□

Then

$$\text{gen } |(\uparrow)(n) \overset{Int.ofNat\_dvd\_right}{\iff} |gen||n \overset{ii)}{\iff} \text{setGcd s } |n$$

, where Int.ofNat_dvd_right is a theorem that states if n is a natural number and z an integer then

$$z|(\uparrow)(n) \Leftrightarrow |z||n$$

Proof (formal):

```
lemma setGcd_dvd (h : n ∈ s) : setGcd s ∣ n := by {
  let I : Ideal ℤ := by {exact Ideal.span (((↑) : ℕ → ℤ) '' s)}
  let gen : ℤ := by {exact Submodule.IsPrincipal.generator I}
  let span_gen_eq_I : Ideal.span {gen} = I := by {exact Ideal.span_singleton_generator I}
  let n_in_I : (n : ℤ) ∈ I := by { apply Ideal.subset_span; rw [@Set.mem_image]; use n}
  let gen_dvd_n : gen ∣ (n : ℤ ) := by {rw[← @Ideal.mem_span_singleton]; rw [span_gen_eq_I]; exact
    n_in_I}
  let gen_natAbs_eq_setGcd : gen.natAbs = setGcd s := by {rw [setGcd];}
  rw [← gen_natAbs_eq_setGcd]
  rw [← Int.ofNat_dvd_right]
  exact gen_dvd_n
}
```

Lemma 2.3. *Let s be a set of* $\mathbb{N}_0$ *and* $n \in \mathbb{N}_0$*. Then* $n \mid setGcd\ s \leftrightarrow \forall m \in s,\ n \mid m$

Proof (informal):
(⇒) Let n | setGcd s and $m \in s$. By definition of dvd, it suffices to show that $\exists c \in N : m = c \cdot n$.
By Lemma 2.2 on $m \in s$, we get setGcd s | m. By definition of dvd on n | setGcd s, we get

1. d ∈ ℕ
2. setGcd s = d · n

By definition of dvd on setGcd s | m, we get

a) e ∈ ℕ

b) m = e · setGcd s

By using c := $e \cdot d$, then we can write

$$m \overset{b}{=} e \cdot \text{setGcd s} \overset{2}{=} e \cdot (d \cdot n) = (e \cdot d) \cdot n = c \cdot n$$

(⇐) Let $\forall m \in s, n|m$.
The idea is to state explicitly the information that is needed to compute setGcd and use them to prove n | setGcd s. From setGcd, we get

1. the set of integers s' which is computed by the canonical homomorphism (↑) from ℕ → ℤ on the set s,
2. I as the ideal generated by the set s'. (or I = (s')) and
3. gen ∈ ℤ as the generator of I.

That implies

i) I = (gen) and

ii) setGcd s = |gen|

To show that n | setGcd s.

$$n|\text{setGcd s} \overset{ii}{\Longleftrightarrow} n||gen| \overset{Int.ofNat\_dvd\_left}{\Longleftrightarrow} (\uparrow)(n)|gen \overset{remark}{\Longleftrightarrow} (gen) \leq ((\uparrow)(n) \overset{3}{\Longleftrightarrow} I \leq ((\uparrow)(n)) \overset{2}{\Longleftrightarrow} (s') \leq ((\uparrow)(n)) \overset{Ideal.span\_le}{\Longleftrightarrow} s' \subset ((\uparrow)(n))$$

So it suffices to show that $s' \subset ((\uparrow)(n)$. Let $m \in s'$, then we need only to show that $m \in ((\uparrow)(n)$.

$$m \in (\uparrow)(n) \overset{remark}{\Longleftrightarrow} (\uparrow)(n)|m \overset{Int.natCast\_dvd}{\Longleftrightarrow} n||m|$$

So it suffices to show that $|m| \in s$ because by assumption follows that n$||m|$.

Proof. $m \in s' \Longrightarrow \exists a \in s : (\uparrow)(a) = m$.

1. $a \in s$
2. $(\uparrow)(a) = m$

$$|m| \in s \overset{2}{\Longleftrightarrow} |(\uparrow)(a)| \in s \overset{|(\uparrow)(a)|=a}{\Longleftrightarrow} a \in s$$

By 1, the proof is done. □

Proof (formal):

```
lemma dvd_setGcd_iff : n ∣ setGcd s ↔ ∀ m ∈ s, n ∣ m := by {
  constructor
  · intro n_dvd_setGcd
    intro m_in_Nat
    intro m_in_s

    have setGcd_dvd_m : setGcd s ∣ m_in_Nat := by {exact setGcd_dvd m_in_s}

    rcases n_dvd_setGcd with ⟨d, hd⟩
    rcases setGcd_dvd_m with ⟨e, he⟩
    dsimp[Dvd.dvd]
    use d * e
    rw [he, hd, mul_assoc]
  · intro all_m_in_s_follows_n_dvd_m
    let I : Ideal ℤ := by {exact Ideal.span (((↑) : ℕ → ℤ) '' s)}
    let gen : ℤ := by {exact Submodule.IsPrincipal.generator I}
    let span_gen_eq_I : Ideal.span {gen} = I := by {exact Ideal.span_singleton_generator I}
    let gen_natAbs_eq_setGcd : gen.natAbs = setGcd s := by {rw [setGcd]}
    rw [← gen_natAbs_eq_setGcd]
    rw [← Int.ofNat_dvd_left]
    rw [← Ideal.span_singleton_le_span_singleton]
    rw [span_gen_eq_I]
    let span_s_eq_I : Ideal.span (((↑) : ℕ → ℤ) '' s) = I := by {rw
    [←@Submodule.toSubMulAction_inj]}
    rw [← span_s_eq_I]
    rw [@Ideal.span_le]
    rw [@Set.subset_def]
    intro m_in_Z
    intro m_in_nat_cast_s
    rw [@SetLike.mem_coe]
    rw [@Ideal.mem_span_singleton]
    rw [@Int.natCast_dvd]

    let m_in_Z_natAbs_in_s : m_in_Z.natAbs ∈ s := by {
      rcases m_in_nat_cast_s with ⟨ m_in_N, m_in_N_in_s_and_coe_m_in_N_eq_m_in_Z⟩
      rcases m_in_N_in_s_and_coe_m_in_N_eq_m_in_Z with ⟨m_in_N_in_s, coe_m_in_N_eq_m_in_Z⟩
      rw[← coe_m_in_N_eq_m_in_Z]
      rw [Int.natAbs_cast]
      exact m_in_N_in_s
    }

    exact all_m_in_s_follows_n_dvd_m m_in_Z.natAbs m_in_Z_natAbs_in_s
}
```

Lemma 2.4. *Let s be a set of* $\mathbb{N}_0$ *and* $n \in \mathbb{N}_0$. *Then setGcd s = 0* $\Leftrightarrow s \subseteq \{0\}$

Proof (informal):
The idea is to state explicitly the information that is needed to compute setGcd and use them to prove this lemma. From setGcd, we get

1. the set of integers s' which is computed by the canonical homomorphism ($\uparrow$) from $\mathbb{N} \to \mathbb{Z}$ on the set s,
2. I as the ideal generated by the set s'. (or I = (s')) and
3. gen $\in \mathbb{Z}$ as the generator of I.

That implies

i) I = (gen) and

ii) setGcd s = |gen|

($\Rightarrow$) Let setGcd s = 0.

$$\text{setGcd s} = 0 \overset{ii}{\Longleftrightarrow} |\text{gen}| = 0 \overset{norm}{\Longleftrightarrow} \text{gen} = 0 \tag{1}$$

Then

$$(s') \overset{2}{=} I \overset{i}{=} (gen) \overset{(1)}{=} (0) \tag{2}$$

It suffices to show that $\forall x \in s : x = 0$.

Proof. Let $x \in s$, then $(\uparrow)(x) \in s'$ by 1. By (2), we get that $\forall p \in s' : p = 0$. Then by combining both, we get that $(\uparrow)(x) = 0$ and by Int.ofNat_eq_zero therefore $x = 0$. $\square$

($\Leftarrow$) Let $s \subseteq \{0\}$, then $\forall n \in s : n = 0$. By (1)and (2), it suffices to show that $\forall x \in s' : x = 0$.

Proof. Let $x \in s'$. Then there exists a $p \in s : (\uparrow)(p) = x$. By assumption on $p \in s$, we get that $p = 0$ and by Int.natCast_eq_zero, we get that $x = 0$. $\square$

Proof (formal):

```
lemma setGcd_eq_zero_iff : setGcd s = 0 ↔ s ⊆ {0} := by {

  let I : Ideal ℤ := by {exact Ideal.span (((↑) : ℕ → ℤ) '' s)}
  let gen : ℤ := by {exact Submodule.IsPrincipal.generator I}
  let I_eq_span_s : I = Ideal.span (((↑) : ℕ → ℤ) '' s) := by {rfl}
  let I_eq_span_gen : I = Ideal.span {gen} := by {rw[Ideal.span_singleton_generator I]}
  let gen_natAbs_eq_natGcd_s : gen.natAbs = setGcd s := by {rw [setGcd]}

  constructor
  · intro setGcd_eq_0
    rw [← gen_natAbs_eq_natGcd_s] at setGcd_eq_0
    rw [@Int.natAbs_eq_zero] at setGcd_eq_0

    have I_eq_bot : I = ⊥ := by {
      rw [I_eq_span_gen]
      exact Ideal.span_singleton_eq_bot.mpr setGcd_eq_0
    }

    dsimp[@Set.subset_def]

    have for_all_x_in_s_x_eq_0 : ∀ x ∈ s, x = 0 := by {
      intro x
      intro x_in_s

      let for_all_x_in_coe_s : ∀ x ∈ (((↑) : ℕ → ℤ) '' s), x = 0 := by {
        rw [I_eq_span_s] at I_eq_bot
        exact Ideal.span_eq_bot.mp I_eq_bot
      }

      let coe_x_in_coe_s : ↑x ∈ (((↑) : ℕ → ℤ) '' s) := by {rw [@Set.mem_image]; exists x}

      exact Int.ofNat_eq_zero.mp (for_all_x_in_coe_s (↑x) coe_x_in_coe_s)
    }

    exact for_all_x_in_s_x_eq_0

  · intro s_in_zero_set
    dsimp [@Set.subset_def] at s_in_zero_set

    rw [← gen_natAbs_eq_natGcd_s]
    rw [@Int.natAbs_eq_zero]
    rw [← Ideal.span_singleton_eq_bot]

    rw [← I_eq_span_gen ]
    rw [I_eq_span_s]
    rw [Ideal.span_eq_bot]

    intro x
    intro x_in_nat_cast_s

    rw [@Set.mem_image] at x_in_nat_cast_s
    rcases x_in_nat_cast_s with ⟨ h , h_in_s, coe_h_eq_x⟩

    rw [← coe_h_eq_x]
    rw [s_in_zero_set h h_in_s]
    rw [@Int.natCast_eq_zero]
}
```

Lemma 2.5. *Let s be a set of $\mathbb{N}_0$ and $n \in \mathbb{N}_0$. If setGcd $s \neq 0$, then $\exists\ n \in s,\ n \neq 0$*

Proof (informal):
Let setGcd s $\neq 0$. Assume that $\neg(\exists n \in s, n \neq 0)$ holds.

$$\neg(\exists n \in s, n \neq 0) \Longrightarrow \forall n \in s, n = 0 \Longrightarrow s \subset \{0\} \overset{Lemma 2.4}{\Longrightarrow} \text{setGcd s} = 0$$

This is a contradiction.

Proof (formal):

```
lemma exists_ne_zero_of_setGcd_eq_one (hs : setGcd s ≠ 0) : ∃ n ∈ s, n ≠ 0 := by {
  by_contra h

  have forall_n_in_s_n_eq_0 : ∀n ∈ s, n = 0 := by {
    intro n n_in_s
    by_contra not_n_eq_zero
    apply h
    rw [←@Ne.eq_def] at not_n_eq_zero
    use n
  }

  have s_subset_of_zero : s ⊆ {0} := by {
    dsimp[@Set.subset_def]
    intro x x_in_s
    apply @Set.mem_singleton_iff.2
    exact forall_n_in_s_n_eq_0 x x_in_s
  }

  have setGcd_s_eq_zero : setGcd s = 0 := by{
    exact setGcd_eq_zero_iff.2 s_subset_of_zero
  }

  contradiction
}
```

Proof (formal vers. 2):

```
lemma exists_ne_zero_of_setGcd_eq_one (hs : setGcd s ≠ 0) : ∃ n ∈ s, n ≠ 0 := by {
  contrapose hs

  rw [@Ne.eq_def]
  rw [@Mathlib.Tactic.PushNeg.not_ne_eq]

  rw [setGcd_eq_zero_iff]
  rw [@Set.subset_def]

  intro x
  intro x_in_s

  rw [@Set.mem_singleton_iff]

  contrapose hs

  rw [Mathlib.Tactic.PushNeg.not_not_eq]
  use x
}
```

Lemma 2.6. *Let s be a set of $\mathbb{N}_0$. Then $s \subseteq$ (setGcd s)*

Proof (informal):
Let n $\in$ s.
To show: n $\in$ (setGcd s).

$$n \in (\text{setGcd s}) \overset{remark}{\iff} \text{setGcd s}|n$$

By using Lemma 2.2 with n $\in$ s the proof is done.
Proof (formal):

```
lemma subset_span_setGcd : s ⊆ Ideal.span {setGcd s} := by {
    rw[@Set.subset_def]
    intro n
    intro n_in_s

    rw [@SetLike.mem_coe]
    rw [@Ideal.mem_span_singleton]

    exact setGcd_dvd n_in_s
}
```

Lemma 2.7. *Let s be a set of* $\mathbb{N}_0$*. Show that there exists a map* $a\colon \mathbb{N}_0 \to \mathbb{Z}$ *and* $t \subseteq \mathbb{N}_0$ *a finite set such that* $t \subseteq s$ *and* $\sum_{n\in t}(a(n)\cdot(\uparrow)(n)) =$ *(*$\uparrow$*)(setGcd s)*

Proof (informal):
The idea is to state explicitly the information that is needed to compute setGcd and use them to prove this lemma. From setGcd, we get

1. the set of integers s' which is computed by the canonical homomorphism ($\uparrow$) from $\mathbb{N} \to \mathbb{Z}$ on the set s,
2. I as the ideal generated by the set s'. (or I = (s')) and
3. gen $\in \mathbb{Z}$ as the generator of I.

That implies

i) I = (gen) and

ii) setGcd s = |gen|

WLOG $gen \geq 0$. We want to show that $gen \in span_{\mathbb{Z}}(s')$.

Proof.
$gen \in span_{\mathbb{Z}}(s') \overset{Ideal.submodule\_span\_eq}{\iff} gen \in (s') \overset{2}{\iff} gen \in I \overset{i}{\iff} gen \in (gen)$
That the generator of an ideal is an element of the ideal is clear. So the proof is done. □

Using Submodule.mem_span_iff_exists_finset_subset on $gen \in span_{\mathbb{Z}}(s')$, we get

a) a' : $\mathbb{Z} \to \mathbb{Z}$

b) t' a finite set of integers

c) t' $\subseteq s'$

d) supp(a') $\subseteq$ t'

e) $\sum_{i\in t} a'(i) * i = gen$

Lets define $a : \mathbb{N} \to \mathbb{N}$

$$a : \mathbb{N} \to \mathbb{N}, i \mapsto a(i) := \begin{cases} a'((\uparrow)(i)) & (\uparrow)(i) \in t' \\ 0 & \text{otherwise} \end{cases}$$

and finite set of $\mathbb{N}_0$

$$t := \{|i| | i \in t'\}$$

By using a and t, we only need to show that $t \subseteq s$ and $\sum_{n\in t}(a(n)\cdot n) =$ setGcd s.
To show: $t \subseteq s$.

Proof. Let $n \in t$. By definition of t, we get

4) $i \in t'$

5) $n = |i|$

Using c on 4, we get that $i \in s'$, and get

6) $p \in s$

7) $i = (\uparrow)(p)$

$$n \overset{5}{=} |i| \overset{7}{=} |(\uparrow)(p)| \overset{Int.natAbs\_cast}{=} p$$

Then by 6 and the fact that $p = n$, we are done.

□

To show: $\sum_{n\in t}(a(n)\cdot(\uparrow)(n)) = (\uparrow)$(setGcd s)

Proof. The following steps are corner stones to prove the statement

iii) $\sum_{n\in t}(a(n)\cdot(\uparrow)(n)) = \sum_{x\in t'}(a(|x|)\cdot(\uparrow)(|x|))$

Proof. By Finset.sum_image, it suffices to show that $|\cdot|$ is injective. Let x,y $\in t'$ and $|x| = |y|$, then we need to show that x = y. By c on x, y, we get x,y $\in s'$. By 1 on x $\in s'$ and y$\in s'$, we get

f) $v \in s$

g) (↑)(v) = x

h) $w \in s$

j) (↑)(w) = y

$$v \overset{Int.natAbs\_cast}{=} |(\uparrow)(v)| \overset{g}{=} |x| \overset{assumption}{=} |y| \overset{j}{=} |(\uparrow)(w)| \overset{Int.natAbs\_cast}{=} w$$

Then

$$x \overset{g}{=} (\uparrow)(v) \overset{v=w}{=} (\uparrow)(w) \overset{j}{=} y$$

□

iv) $\sum_{x \in t'}(a(|x|) \cdot (\uparrow)(|x|)) = \sum_{m \in t'}(a'(m) \cdot (\uparrow)(|m|))$

Proof. It suffices to show that $\forall x \in t', a(|x|) = a'(x)$. Let $x \in t'$. By c on $x \in t'$, we get $x \in s'$. By 1 on $x \in s'$, we get

k) t $\in$ s

l) $(\uparrow)(t) = x$

Then

$$a(|x|) \overset{l}{=} a(|(\uparrow)(t)|) \overset{Int.natAbs\_cast}{=} a(t) \overset{assumption}{=} a'((\uparrow)(t)) \overset{l}{=} a'(x)$$

□

v) $\sum_{m \in t'}(a'(m) \cdot (\uparrow)(|m|)) = \sum_{m \in t'}(a'(m) \cdot m)$

Proof. It suffices to show that for $m \in t' : m = (\uparrow)(|m|)$. By c on $m \in t'$, we get that $m \in s'$ and by 1 on $m \in s'$, we get

8) x $\in$ s

9) (↑)(x) = m

$$m \overset{9}{=} (\uparrow)(x) \overset{Int.natAbs\_cast}{=} (\uparrow)(|(\uparrow)(x)|) \overset{9}{=} (\uparrow)(|m|)$$

□

vi) $\sum_{m \in t'}(a'(m) \cdot m) = gen$

Proof. By e, we are done. □

vii) $gen = (\uparrow)$(setGcd s)

Proof.

$$(\uparrow)(\text{setGcd s}) \overset{ii}{=} (\uparrow)(|gen|) \underset{gen \geq 0}{\overset{Int.natAbs\_of\_nonneg}{=}} gen$$

□

□

Proof (formal) PART 1:

```
lemma exists_sum_mul_eq_setGcd : ∃ (a : ℕ → ℤ) (t : Finset ℕ), ↑t ⊆ s ∧ ∑ n ∈ t, a n * n = setGcd
    s := by {
  let I : Ideal ℤ := by {exact Ideal.span (((↑) : ℕ → ℤ) '' s)}
  let I_eq_span_s : I = Ideal.span (((↑) : ℕ → ℤ) '' s) := by {rfl}

  let gen : ℤ := by {exact Submodule.IsPrincipal.generator I}
  let neg_gen : ℤ := -gen
  let pos_gen : ℤ := if 0 ≤ gen then gen else -gen

  let I_eq_span_pos_gen : I = Ideal.span {pos_gen} := by {
    dsimp[pos_gen]
    by_cases h : 0 ≤ gen
    · rw [if_pos h]; rw[Ideal.span_singleton_generator I]
    · rw [if_neg h];
      calc I = Ideal.span {gen} := by {rw[Ideal.span_singleton_generator I]}
    _ = Ideal.span {-1 * gen} := by {rw [← Ideal.span_singleton_mul_left_unit ]; apply
     Int.isUnit_iff.mpr; apply Or.inr; rfl}
    _ = Ideal.span {neg_gen}  := by {dsimp[neg_gen]; rw [Int.neg_one_mul]}
  }

  let pos_gen_natAbs_eq_setGcd_s : pos_gen.natAbs = setGcd s := by{
    dsimp[pos_gen]
    by_cases h : 0 ≤ gen
    · rw [if_pos h]; rw [setGcd]
    · rw [if_neg h]; rw [Int.natAbs_neg]; rw [setGcd]
  }

  have zero_leq_pos_gen : 0 ≤ pos_gen := by {
    dsimp[pos_gen]
    by_cases h : 0 ≤ gen
    · rw [if_pos h]; exact h
    · rw [if_neg h]; rw [@Int.not_le] at h; rw [← @Int.neg_pos] at h;
      rw [@Int.lt_iff_le_and_ne] at h; exact h.1;
  }

  have pos_gen_in_submodule_span_Z_of_coe_s : pos_gen ∈ Submodule.span ℤ (((↑) : ℕ → ℤ) '' s) := by
    {
    rw [@Ideal.submodule_span_eq]
    rw [← I_eq_span_s]
    rw [I_eq_span_pos_gen]
    exact Ideal.mem_span_singleton_self pos_gen
  }
```

Proof (formal) PART 2:

```
  apply Submodule.mem_span_iff_exists_finset_subset.mp at pos_gen_in_submodule_span_Z_of_coe_s
  rcases pos_gen_in_submodule_span_Z_of_coe_s with ⟨a', t', t'_in_natcast_s, supp_a'_in_t',
    sum_of_t'_with_a'_n_mul_n_eq_pos_gen⟩

  let a : ℕ → ℤ := fun n ↦ if ↑n ∈ t' then a' ↑n else 0
  let t : Finset ℕ := Finset.image (fun z ↦ z.natAbs) (t')

  use a, t
  constructor
  · intro n n_in_coe_t
    rw [@Finset.mem_coe] at n_in_coe_t
    rw [@Finset.mem_image] at n_in_coe_t
    rcases n_in_coe_t with ⟨p, p_in_t', p_natabs_eq_n⟩

    let p_in_natcast_s : p ∈ Nat.cast '' s := t'_in_natcast_s p_in_t'
    rw [@Set.mem_image] at p_in_natcast_s
    rcases p_in_natcast_s with ⟨c, c_in_s, coe_n_eq_p⟩

    let c_eq_n : c = n := by {rw [← p_natabs_eq_n,← coe_n_eq_p, Int.natAbs_cast]}
    rw[← c_eq_n]
    exact c_in_s
  · calc Σ n ∈ t, a n * ↑n = Σ m ∈ t', a' m * m.natAbs := by {
        have inj : Set.InjOn Int.natAbs t' := by {
          intro x x_in_coe_t' y y_in_coe_t'
          intro x_natAbs_eq_y_natAbs
          rcases t'_in_natcast_s x_in_coe_t' with ⟨v, v_in_s, coe_v_eq_x ⟩
          rcases t'_in_natcast_s y_in_coe_t' with ⟨w, w_in_s, coe_w_eq_y ⟩
          rw[← coe_w_eq_y,← coe_v_eq_x] at x_natAbs_eq_y_natAbs
          repeat rw[Int.natAbs_cast] at x_natAbs_eq_y_natAbs
          rw[← coe_w_eq_y, ←coe_v_eq_x]
          exact congrArg Nat.cast x_natAbs_eq_y_natAbs
        }
        rw [Finset.sum_image inj]
        apply Finset.sum_congr rfl
        intro z z_in_t'
        rw[← @Finset.mem_coe] at z_in_t'
        rcases t'_in_natcast_s z_in_t' with ⟨w, w_in_s, coe_w_eq_y ⟩
        rw[← coe_w_eq_y]
        rw[Int.natAbs_cast]
        dsimp[a]
        rw[coe_w_eq_y]
        rw[@Finset.mem_coe] at z_in_t'
        rw [if_pos z_in_t']
      }
      _ = Σ m ∈ t', a' m * m := by {
        apply Finset.sum_congr rfl
        intro x x_in_t'
        rcases t'_in_natcast_s x_in_t' with ⟨w, w_in_s, coe_w_eq_y ⟩
        rw [← coe_w_eq_y]
        rw[Int.natAbs_cast]
      }
      _ = pos_gen := by exact sum_of_t'_with_a'_n_mul_n_eq_pos_gen
      _ = ↑(setGcd s) := by {
        rw[← pos_gen_natAbs_eq_setGcd_s]
        rw[Int.natAbs_of_nonneg zero_leq_pos_gen]
      }
}
```

Lemma 2.8. *Let s be a set of $\mathbb{N}_0$. Let ι : Type*, $t \subseteq \iota$ a finite set, maps a: $\iota \to \mathbb{Z}$, b: $\iota \to \mathbb{N}$, and $g \in \mathbb{N}_0$. If $\sum_{i\in t} a(i)\cdot b(i) = g$ and $\forall i \in t : g|b(i)$, then $\exists n \in \mathbb{N}_0 : \forall m \geq n : g|m \Rightarrow \exists a' : \iota \to \mathbb{N}_0, m = \sum_{i\in t}(a'(i)\cdot b(i))$*

Proof (informal):
Let $\sum_{i\in t} a(i)\cdot b(i) = g$ and $\forall i \in t : g|b(i)$. We intend to prove it by cases.

Case $\forall i \in t, b(i) = 0$:

Proof. First, we use this case hypotheses to prove g = 0:

$$g = \sum_{i\in t} a(i)\cdot b(i) \overset{\forall i\in t, b(i)=0}{=} \sum_{i\in t} a(i)\cdot 0 = 0$$

Next, we want to choose $n \in \mathbb{N}_0 : \forall m \geq n : g|m$. We use $n = 0$ (otherwise if n $\neq$ 0, then m $\neq$ 0 and then

$$g|m \overset{def}{\iff} \exists c \in \mathbb{N} : m = g * c \overset{g=0}{\iff} \exists c \in \mathbb{N} : m = 0 * c = 0$$

leads to a contradiction of m $\neq$ 0.).
Let $m \in \mathbb{N}_0$ such that $m \geq 0 : g|m$. The next step is to prove that $m = 0$.

$$g|m \overset{def}{\iff} \exists c \in \mathbb{N} : m = g * c \overset{g=0}{\iff} \exists c \in \mathbb{N}_0 : m = 0 * c = 0$$

So what is left to prove is $\exists a' : \iota \to \mathbb{N}_0, m = \sum_{i\in t}(a'(i)\cdot b(i))$. we use as a' the zero mapping, then

$$\sum_{i\in t}(a'(i)\cdot b(i)) \underset{a'=0}{\overset{\forall i\in t, b(i)=0}{=}} \sum_{i\in t}(0\cdot 0) = 0 \overset{m=0}{=} m$$

□

Case $\exists i \in t, b(i) \neq 0$:
Then we have

1. $i \in t$
2. $b(i) \neq 0$

We define $n := b(i)/g \cdot \sum_{p\in t}(-a(p)).toNat \cdot b(p)) \in \mathbb{N}$ and use it on our goal. Let m $\in \mathbb{N}$ such that $m \geq n : g|m$. By $m \geq n$, we get

3. $\exists c \in \mathbb{N} : m = n + c$

We want to show that $\exists q, r : 0 \leq r \wedge r < b(i) \wedge q \cdot b(i) + r = c$

Proof. Let $q := c/b(i) \in \mathbb{N}$ and $r := c \;\%\; b(i) \in \mathbb{N}$. By div_add_mod' on c and q, we get

4. a / b · b + a % b = a

By using q and r on the goal, it suffices to show that $0 \leq r \wedge r < b(i) \wedge q \cdot b(i) + r = c$. We split up what we want to prove:
Proof to 0 ≤ r:

Proof.
Every natural number is bigger or equal to 0 □

Proof to r < b(i):

Proof.
By definition of r, we can write the goal as $c \;\%\; b(i) < b(i)$. Then we can use Nat.mod_lt on c, such that we only need to show that $b(i) > 0$. By 2 the proof is done. □

Proof to $q \cdot b(i) + r = c$:

Proof.
We use rewrite goal with definition of q and r and use 3. □

Then, we received

5. $0 \leq$ r
6. r $<$ b(i)
7. $q \cdot b(i) + r = c$

Next, we show that

.) g | n:

Proof.
$$g|n \overset{def:n}{\Longleftrightarrow} g|b(i)/g \cdot \sum_{p \in t}(-a(p)).toNat \cdot b(p))$$

By dvd_mul_of_dvd_right, we only need to show that $g|\sum_{i\in t}(-a(p)).toNat{\cdot}b(p))$. By Finset.dvd_sum, we need to verify that $\forall p \in t : g|(-a(p)).toNat \cdot b(p))$ and again by dvd_mul_of_dvd_right, we just need to show that on g | b(p) but that can we show by the hypothesis on $p \in t$. □

.) g | b i:

Proof. We use the hypothesis on 1 and are done. □

.) g | r:

Proof. By Nat.dvd_add_iff_right, we only need to provide that

.) $g|n + q \cdot b(i)$

Proof. Clear because g | n is true by the proof from above and g | q · b(i) because g | b(i)is true by the proof from above □

.) g|m

Proof. Clear because g|m by definition □

□

Our next subgoal is to show that $\uparrow r + \uparrow n$ can be written as $\sum_{p\in t}((\uparrow r/\uparrow g) \cdot a(p) + (\uparrow b(i)/\uparrow g) * \uparrow (-a(p)).toNat) {\cdot} \uparrow b(p)$

Proof. First, we show that $g \neq 0$, otherwise we get a contradiction $(0 \neq 0)$ by 2 with by the hypothesis on 2.

$$\begin{aligned}
\uparrow r + \uparrow n &\overset{def:n}{=} \uparrow g {\cdot} \uparrow r/\uparrow g + \uparrow n \\
&\overset{hypothesis}{=} (\sum_{p\in t} a(p) {\cdot} \uparrow b(p)) {\cdot} \uparrow r/\uparrow g + \uparrow n \\
&\underset{g|r}{\overset{commutative}{=}} \uparrow r/\uparrow g \cdot (\sum_{p\in t} a(p) {\cdot} \uparrow b(p)) + \uparrow n \\
&= (\sum_{i\in p} \uparrow r/\uparrow g \cdot a(p) {\cdot} \uparrow b(p)) + \uparrow n \\
&\overset{def:n}{=} (\sum_{p\in t} \uparrow r/\uparrow g \cdot a(p) {\cdot} \uparrow b(p)) + b(i)/g \cdot \sum_{p\in t}((-a(p)).toNat \cdot b(p)) \\
&= (\sum_{p\in t} \uparrow r/\uparrow g \cdot a(p) {\cdot} \uparrow b(p)) + \sum_{p\in t} \uparrow b(i)/\uparrow g \cdot ((-a(p)).toNat {\cdot} \uparrow b(p)) \\
&= \sum_{p\in t} \uparrow r/\uparrow g \cdot a(p) {\cdot} \uparrow b(p) + \uparrow b(i)/\uparrow g \cdot ((-a(p)).toNat {\cdot} \uparrow b(p)) \\
&\overset{distributivity}{=} \sum_{p\in t} (\uparrow r/\uparrow g \cdot a(p) + \uparrow b(i)/\uparrow g \cdot (-a(p)).toNat)) {\cdot} \uparrow b(p) \\
&\overset{assoc}{=} \sum_{p\in t}((\uparrow r/\uparrow g) \cdot a(p) + (\uparrow b(i)/\uparrow g) * \uparrow (-a(p)).toNat)) {\cdot} \uparrow b(p)
\end{aligned}$$

□

The Next step is to show that $\forall p \in t, 0 \leq \uparrow r/\uparrow g \cdot a(p) + \uparrow b(i)/\uparrow g \cdot \uparrow ((-a(p)).toNat)$

Proof. Let $p \in t$.
Case $0 \leq a(p)$: By $0 \leq a(p)$, we get

8. a(p) = ↑((a(p)).toNat)
9. $(-a(i)).toNat = 0$

$$\begin{aligned}0 \leq \uparrow r/\uparrow g \cdot a(p) + \uparrow b(i)/\uparrow g \cdot \uparrow ((-a(p)).toNat) &\overset{9}{\iff} 0 \leq \uparrow r/\uparrow g \cdot a(p) + \uparrow b(i)/\uparrow g \cdot \uparrow 0\\ &\iff 0 \leq \uparrow r/\uparrow g \cdot a(p)\\ &\overset{Int.ofNat\_ediv\_ofNat}{\iff} 0 \leq \uparrow (r/g) \cdot a(p)\end{aligned}$$

It suffices to that $0 \leq \uparrow (r/g) \cdot a(p)$ and by Int.mul_nonneg, we only need to show

1. $0 \leq \uparrow (r/g)$

   Proof.
   $$0 \leq \uparrow (r/g) \iff 0 \leq \uparrow r/\uparrow g \overset{Int.natCast\_tdiv\_eq\_ediv}{\iff} (\uparrow r).tdiv \uparrow g$$
   We only need to show that $0 \leq \uparrow g, \uparrow r$ that is true because g and r are natural numbers. □

2. $0 \leq a(p)$

   Proof. By case hypothesis, we are done □

Case $a(p) < 0$:
By $a(p) < 0$, we get

10. $0 \leq -a(p)$
11. $-a(p) = \uparrow ((-a(p).toNat)$

$$\begin{aligned}0 \leq \uparrow r/\uparrow g \cdot a(p) + \uparrow b(i)/\uparrow g \cdot \uparrow (-a(p)).toNat &\iff -1 \cdot \uparrow r/\uparrow g \cdot a(p) \leq \uparrow b(i)/\uparrow g \cdot \uparrow ((-a(p)).toNat)\\ &\iff \uparrow r/\uparrow g \cdot -a(p) \leq \uparrow b(i)/\uparrow g \cdot \uparrow ((-a(p)).toNat)\\ &\overset{11}{\iff} \uparrow r/\uparrow g \cdot -a(p) \leq \uparrow b(i)/\uparrow g \cdot \uparrow -a(p)\end{aligned}$$

By Int.mul_le_mul_of_nonneg_right, it suffices to show that

.) $\uparrow r/\uparrow g \leq \uparrow b(i)/\uparrow g$

   Proof. By Nat.div_le_div_right, we only need to show that $r \leq bi$. By 6, we are done. □

.) $0 \leq \uparrow (-a(p)).toNat$

   Proof. By 11 and 10, we are done. □

□

Now we define $a' : \iota \to \mathbb{N}$ and $c : \iota \to \mathbb{N}$ such that

$$a'(p) = \begin{cases}(\uparrow r/\uparrow g \cdot a(p) + \uparrow b(i)/\uparrow g \cdot \uparrow (-a(p)).toNat).toNat + q & p = i\\ (\uparrow r/\uparrow g \cdot a(p) + \uparrow b(i)/\uparrow g \cdot \uparrow (-a(p)).toNat).toNat + 0 & \text{otherwise}\end{cases}$$

$$c(p) = \begin{cases}q & p = i\\ 0 & \text{otherwise}\end{cases}$$

Next, we use a' on the goal. We need to show that $m = \sum_{i \in t}(a'(i) \cdot b(i))$.
By using Int.natCast_inj.1 on the goal, we only need to show that $\uparrow m = \uparrow (\sum_{i \in t}(a'(i) \cdot b(i)))$

$$\begin{aligned}
\uparrow m &\overset{3}{=} \uparrow n + \uparrow c \overset{q \cdot b(i) + r = c}{=} \uparrow n + \uparrow q \cdot \uparrow b(i) + \uparrow r = \uparrow r + \uparrow n + \uparrow q \cdot \uparrow b(i) \\
&\overset{r+n \text{ as sum}}{=} \sum_{p \in t}((\uparrow r / \uparrow g) \cdot a(p) + (\uparrow b(i) / \uparrow g) * \uparrow (-a(p)).toNat) \cdot \uparrow b(p) + \uparrow q \cdot \uparrow b(i) \\
&= \sum_{p \in t}((\uparrow r / \uparrow g) \cdot a(p) + (\uparrow b(i) / \uparrow g) * \uparrow (-a(p)).toNat)) \cdot \uparrow b(p) + \uparrow (q \cdot b(i)) \\
&= \sum_{p \in t}((\uparrow r / \uparrow g) \cdot a(p) + (\uparrow b(i) / \uparrow g) * \uparrow (-a(p)).toNat)) \cdot \uparrow b(p) + \uparrow (\sum_{p \in t} c(p) \cdot b(p)) \\
&= \sum_{p \in t}((\uparrow r / \uparrow g) \cdot a(p) + (\uparrow b(i) / \uparrow g) * \uparrow (-a(p)).toNat)) \cdot \uparrow b(p) + (\sum_{p \in t} \uparrow c(p) \cdot \uparrow b(p)) \\
&= \sum_{p \in t}((\uparrow r / \uparrow g) \cdot a(p) + (\uparrow b(i) / \uparrow g) * \uparrow (-a(p)).toNat)) \cdot \uparrow b(p) + \uparrow c(p) \cdot \uparrow b(p) \\
&= \sum_{p \in t}((\uparrow r / \uparrow g) \cdot a(p) + (\uparrow b(i) / \uparrow g) * \uparrow (-a(p)).toNat) + \uparrow c(p)) \cdot \uparrow b(p) \\
&= \sum_{p \in t} \uparrow a'(p) \cdot \uparrow b(p) \\
&= \uparrow (\sum_{p \in t}(a'(p) \cdot b(p))
\end{aligned}$$

□

Proof (formal):

```
lemma exists_nat_linearCombination_eq {ι : Type*} {t : Finset ι} {a : ι → ℤ} {b : ι → ℕ} {g : ℕ}
    -- We'll apply this to ‘t‘ obtained from the previous lemma, so ‘g = setGcd s‘ and ‘b = id‘.
    (h : Σ i ∈ t, a i * b i = g) (dvd : ∀ i ∈ t, g ∣ b i) :
    ∃ n : ℕ, ∀ m ≥ n, g ∣ m → ∃ a' : ι → ℕ, m = Σ i ∈ t, a' i * b i := by
  by_cases h' : ∀ i ∈ t, b i = 0
  -- Show that ‘g = 0‘
  · have g_eq_zero : g = 0 := by {
      rw [Mathlib.Tactic.Zify.natCast_eq]
      calc
        ↑g = Σ i ∈ t, a i * ↑(b i) := by rw[← h]
        _ = 0 := by {rw[Finset.sum_eq_zero]; intro x x_in_t; rw [h' x x_in_t]; rw
    [cast_zero,mul_zero]}
        _ = ↑0 := Nat.cast_zero
    }

    -- Only ‘n = 0‘ is possible
    use 0
    intro m m_geq_0 g_dvd_m

    -- Show that ‘m = 0‘
    have m_eq_zero : m = 0 := by{
      dsimp[Dvd.dvd] at g_dvd_m
      rcases g_dvd_m with ⟨c,m_eq_g_mul_c ⟩
      rw[m_eq_g_mul_c]
      rw[g_eq_zero]
      rw[zero_mul]
    }

    -- Use the zero mapping
    use (fun n ↦ 0)
    rw[m_eq_zero]
    rw[Finset.sum_eq_zero]
    intro x x_in_t
    rw [h' x x_in_t]
    rw[mul_zero]
```

```
push_neg at h'
-- take some nonzero ‘b i‘, and we can then take ‘n‘ to be ‘(b i / g) * ∑ i ∈ t, (-a i).toNat * b
  i‘
have ⟨i, hi, ne⟩ := h'

-- Use n
let n := b i / g * ∑ i ∈ t, (-a i).toNat * b i
use n

-- Get ‘∀ m ≥ n, g ∣ m‘
intro m m_ge_n g_dvd_m

-- Given ‘m ≥ n‘ divisible by ‘g‘, we can write ‘m = q * b i + r + n‘ with ‘0 ≤ r < b i‘.
have ⟨c, eq⟩ := exists_add_of_le m_ge_n
obtain ⟨q, r, h0r, hrbi, rfl⟩ : ∃ q r, 0 ≤ r ∧ r < b i ∧ q * b i + r = c := by {
  have := c.div_add_mod' (b i)
  let q : ℕ := c / b i
  let r : ℕ := c % b i
  use q
  use r
  constructor
  · exact Nat.zero_le r
  · constructor
    · dsimp[r]
      apply Nat.mod_lt (c)
      exact Nat.pos_of_ne_zero ne
    · dsimp[r]
      dsimp[q]
      exact this
}
```

```
-- Notice that ‘g ∣ n‘ and ‘g ∣ b i‘ so ‘g ∣ r‘.
have g_dvd_n : g ∣ n := by {
  have g_dvd_sum : g ∣ ∑ i ∈ t, (-a i).toNat * b i := by {
    apply Finset.dvd_sum
    intro j j_in_t
    apply dvd_mul_of_dvd_right
    exact dvd j j_in_t
  }
  dsimp[n]
  apply dvd_mul_of_dvd_right
  exact g_dvd_sum
}

have g_dvd_b_i : g ∣ b i := by {exact dvd i hi}

have g_dvd_r : g ∣ r := by {
  have g_dvd_add : g ∣ n + (q * b i) := by {
    have g_dvd_q_mul_b_i : g ∣ q * b i := by {
      apply dvd_mul_of_dvd_right
      exact g_dvd_b_i
    }
    rw [← Nat.dvd_add_iff_right g_dvd_n]
    exact g_dvd_q_mul_b_i
  }
  rw [Nat.dvd_add_iff_right g_dvd_add]
  rw [add_assoc]
  rw [← eq]
  exact g_dvd_m
}
```

```
-- ‘r + n‘ can be written as ‘Σ i ∈ t, ((r / g) * a i + (b i / g) * (-a i).toNat) * b i‘
have r_add_n : r + n = Σ p ∈ t, ((r / g : ℤ) * a p + (b i / g : ℤ) * ((-a p).toNat) : ℤ) * (b p :
  ℤ) := by {
  calc
    -- First, we rewite ‘↑r‘ as ‘↑g * ↑r/↑g‘
    (r : ℤ) + (n : ℤ) = (g : ℤ) * (r : ℤ) / (g : ℤ) + (n : ℤ) := by {
      rw [Int.mul_ediv_cancel_left ↑r]
      rw [Int.natCast_ne_zero]
      intro g_eq_zero
      rcases dvd i hi with ⟨x,hx⟩
      rw [hx] at ne
      rw[g_eq_zero, zero_mul] at ne
      contradiction
    }
    -- Next, unfold the definition of ‘↑g‘ in the first ‘↑g‘ in ‘↑g * ↑r/↑g‘
    _ = (Σ i ∈ t, a i * (b i : ℤ)) * r / g + (n : ℤ) := by {
      nth_rw 1 [← h ]
    }
    -- We want that ‘r / g‘ stands on the left side of the sum
    _ = (r / g) * (Σ i ∈ t, a i * ↑(b i)) + (n : ℤ) := by {
      have coe_g_dvd_coe_r : (g : ℤ) ∣ ↑r := by {
        rw [Int.natCast_dvd_natCast]
        exact g_dvd_r
      }

      rw [Int.mul_ediv_assoc ]
      · rw[mul_comm]
      · exact coe_g_dvd_coe_r
    }
    -- Next, we pull in ‘r / g‘ into the sum
    _ = Σ i ∈ t, ((r / g) * a i * b i) + (n : ℤ) := by {
      rw [Finset.mul_sum]
      rw[Finset.sum_congr]
      · rfl
      · intro x x_in_t
        rw [mul_assoc]
    }
    -- Next we unfold the definition of ‘n‘
    _ = Σ p ∈ t, ((r / g : ℤ) * a p * (b p : ℤ)) + Σ j ∈ t, ((b i / g : ℤ) * ((-a j).toNat : ℤ) *
   (b j : ℤ)) := by {
      dsimp[n]

      rw [cast_sum]
      rw [Finset.mul_sum]

      nth_rw 2 [Finset.sum_congr]
      · rfl
      intro x x_in_t
      rw [Nat.cast_mul]
      rw [Int.ofNat_ediv_ofNat]
      rw [mul_assoc]

    }
    -- We write both sum’s as one
    _ = Σ p ∈ t, ((r / g : ℤ) * a p * (b p : ℤ) + (b i / g :ℤ) * ((-a p).toNat : ℤ) * (b p : ℤ))
  := by {
      rw [Finset.sum_add_distrib]
    }
    -- Use distributivity to ‘(_ + _) * b p‘
    _ = Σ p ∈ t, ((r / g : ℤ) * a p + (b i / g : ℤ) * ((-a p).toNat : ℤ)) * (b p : ℤ) := by {
      rw [Finset.sum_congr]
      · rfl
      · intro p p_in_s
        rw [Int.add_mul]
    }
}
```

```
-- and it suffices to verify that all ‘(r / g) * a i + (b i / g) * (-a i).toNat‘ are nonnegative.
have non_neg : ∀p ∈ t, 0 ≤ (r / g : ℤ) * a p + (b i / g : ℤ) * ((-a p).toNat : ℤ) := by {
  intro p p_in_t
  by_cases hhh : 0 ≤ a p
  -- If ‘0 ≤ a p‘ then ‘a p = ↑(a p).toNat‘ and‘(-a i).toNat = 0‘
  · have a_p_eq_a_p_cast_toNat : a p = ↑(a p).toNat := by {
      rw [Int.eq_natCast_toNat]
      exact hhh
    }
    nth_rw 1 [a_p_eq_a_p_cast_toNat]

    have neg_a_p_eq_zero : ↑(-a p).toNat = 0 := by {
      rw [Int.toNat_eq_zero]
      rw [Int.neg_nonpos_iff]
      exact hhh
    }
    -- Then it suffices to show that ‘0 ≤ ↑(r / g) * a p‘.
    rw [neg_a_p_eq_zero]

    rw [Int.add_nonnneg_iff_neg_le’]
    rw [Int.natCast_zero]
    rw [mul_zero]
    rw [Int.neg_nonpos_iff]
    rw [Int.ofNat_ediv_ofNat]
    rw [← a_p_eq_a_p_cast_toNat]

    -- For ‘0 ≤ ↑(r / g) * a p‘, we need to prove that ‘0 ≤ a p‘ and ‘0 ≤ ↑(r / g)‘
    apply Int.mul_nonneg
    · rw [Int.natCast_div]
      rw [← Int.natCast_tdiv_eq_ediv]
      -- By providing a proof ‘0 ≤ ↑r‘ and ‘0 ≤ ↑g‘, we can proof ‘0 ≤ ↑(r / g)‘
      apply Int.tdiv_nonneg
      · exact Nat.cast_nonneg r
      · exact Nat.cast_nonneg g
    · exact hhh
  · rw [not_le] at hhh
    rw [Int.add_nonnneg_iff_neg_le’]
    rw [Int.neg_mul_eq_mul_neg]

    -- ‘¬0 ≤ a p‘ can be written as ‘0 ≤ -a p‘
    have zero_leq_neg_a_p : 0 ≤ -a p := by {
      rw [Int.neg_nonneg]
      rw [Int.le_iff_lt_or_eq]
      exact Or.inl hhh
    }

    -- If ‘0 ≤ -a p‘, then ‘-a p = ↑(-a p).toNat‘
    have neg_a_p_eq_neg_a_p_cast_toNat : -a p = ↑(-a p).toNat := by {
      rw [Int.eq_natCast_toNat]
      exact zero_leq_neg_a_p
    }

    -- Then it suffices to show that ‘↑(r / g) ≤ ↑(b i / g)‘ and ‘0 ≤ ↑(-a p).toNat‘
    nth_rw 1 [neg_a_p_eq_neg_a_p_cast_toNat]
    rw [← Int.natCast_div]
    rw [← Int.natCast_div]
    apply Int.mul_le_mul_of_nonneg_right
    · rw [Int.ofNat_le]
      have r_leq_b_i : r ≤ b i := by{
        rw [Nat.le_iff_lt_or_eq]
        exact Or.inl hrbi
      }
      exact Nat.div_le_div_right r_leq_b_i
    · rw [← neg_a_p_eq_neg_a_p_cast_toNat]
      exact zero_leq_neg_a_p
}
```

```
-- Define c which is needed to write ‘q * b i‘ as a linear combination of the ‘b i‘s later.
let c : ι → ℕ := by {
  classical
  intro p
  exact if p = i then q else 0
}

/- The final ‘a’‘ needs to be defined using an ‘if ... then ... else ...‘ expression,
because we need to add ‘q‘ to the coefficient of ‘b i‘ but not the other coefficients.
For this you need to precede the definition with the ‘classical‘ tactic.
-/
-- Define ‘a’‘
let a’ : ι → ℕ := by {
  classical
  intro p
  exact (r / g * a p + ↑(b i) / ↑g * ↑(-a p).toNat).toNat + if p = i then q else 0
}

-- Use a’
use a’
-- Show that m = Σ i ∈ t, a’ i * b i
apply Int.natCast_inj.1

calc (m : ℤ) = (n : ℤ) + ((q : ℤ) * (b i : ℤ) + (r : ℤ)) := by {
    rw[eq]
    rw [Int.natCast_add]
    rw [Int.natCast_add]
    rw [Int.ofNat_mul_out]
  }
  _ = ((r: ℤ) + (n : ℤ)) + (q : ℤ) * (b i : ℤ) := by {
    nth_rw 1 [add_assoc]
    rw[add_comm ]
    rw[add_assoc]
    rw[add_comm ]
    rw[add_assoc]
  }
  _ = Σ p ∈ t, (↑r / ↑g * a p + ↑(b i) / ↑g * ↑(-a p).toNat) * ↑(b p) + q * b i := by {
    rw [r_add_n]
  }
  _ = Σ p ∈ t, (↑r / ↑g * a p + ↑(b i) / ↑g * ↑(-a p).toNat) * ↑(b p) + ↑(q * b i) := by {
    rw [Int.natCast_mul]
  }
  _ = Σ p ∈ t, (↑r / ↑g * a p + ↑(b i) / ↑g * ↑(-a p).toNat) * ↑(b p) + ↑(Σ q ∈ t, c q * b q) :=
  by {
    -- ‘q * b i‘ is obviously a linear combination of the ‘b i‘s.
    have q_mul_b_i_eq_sum : q * b i = Σ q ∈ t, c q * b q := by {
      rw [Finset.sum_eq_single_of_mem i]
      · dsimp[c]
        rw [if_pos]
        rfl
      · exact hi
      · intro p p_in_t p_neq_i
        dsimp [c]
        rw[if_neg p_neq_i]
        rw[zero_mul]
    }
    rw [q_mul_b_i_eq_sum]
  }
  _ = Σ p ∈ t, (↑r / ↑g * a p + ↑(b i) / ↑g * ↑(-a p).toNat) * ↑(b p) + Σ q ∈ t, ↑(c q) * ↑(b q) :=
   by {
    rw [cast_sum]
    nth_rw 2 [Finset.sum_congr]
    · rfl
    · intro x x_in_t
      rw [Int.ofNat_mul_out]
  }
```

```
-- Combine the two sums into only one sum
_ = Σ p ∈ t, (((↑r / ↑g * a p + ↑(b i) / ↑g * ↑(-a p).toNat) * ↑(b p)) + ↑(c p) * ↑(b p) ):= by {
  rw [Finset.sum_add_distrib]
}
-- Use distributivity to ‘(_ + _) * b p‘
_ = Σ p ∈ t, (↑r / ↑g * a p + ↑(b i) / ↑g * ↑(-a p).toNat + ↑(c p)) * ↑(b p) := by {
  rw [Finset.sum_congr]
  · rfl
  · intro p p_in_t
    nth_rw 2 [Int.add_mul]
}
-- Use the Definition of ‘a’‘
_ = Σ p ∈ t, ↑(a’ p) * ↑(b p) := by {
  rw [Finset.sum_congr]
  · rfl
  · intro p p_in_t
    dsimp[a’]
    dsimp[c]

    have natCast_toNat_eq_self_in_t : ↑(↑r / ↑g * a p + ↑(b i) / ↑g * ↑(-a p).toNat).toNat = ↑r /
↑g * a p + ↑(b i) / ↑g * ↑(-a p).toNat := by{
      rw[ Int.natCast_toNat_eq_self]
      exact non_neg p p_in_t
    }

    rw[natCast_toNat_eq_self_in_t]
}
_ = ↑(Σ p ∈ t, a’ p * b p) := by {
  rw [cast_sum]
  rw [Finset.sum_congr]
  · rfl
  · intro x x_in_t
    rw [Int.ofNat_mul_out]
}
```

Lemma 2.9. *Let s be a set of* $\mathbb{N}_0$. *Show that there exists* $t \subseteq \mathbb{N}_0$ *finite set and* $n \in \mathbb{N}_0$ *such that if* $\forall m \geq n$: *setGcd* $\mid m \Rightarrow m \in$ *Submodule.span* $\mathbb{N}_0$ t

Proof (informal):
The idea is to combine the previous two lemmas 2.7 and 2.8 to prove this statement.
By lemma 2.7 on set s, we get:

1. a as a function from $\mathbb{N}_0 \to \mathbb{Z}$
2. t as a finite set of natural numbers (with zero)
3. t $\subseteq$ s
4. $\sum_{n \in t}(a(n) \cdot (\uparrow)(n)) = (\uparrow)(\text{setGcd s})$

Next, we define the function b : $\mathbb{N}_0 \to \mathbb{N}_0, i \mapsto i$ that is just the identity function (b = id) from $\mathbb{N}_0$ to $\mathbb{N}_0$.
Next, we want to prove the following two sub goals to use lemma 2.8.
Sub goal $\sum_{r \in t}(a(r) \cdot (\uparrow)(b(r))) = (\uparrow)(\text{setGcd s})$:

Proof.

$$(\uparrow)(setGcds) \overset{4}{=} \sum_{n \in t}(a(n) \cdot (\uparrow)(n)) \overset{b=id}{=} \sum_{n \in t}(a(n) \cdot (\uparrow)(b(n)))$$

□

Sub goal : $\forall i \in t$: setGcd s$|b(i)$:

Proof. By definition of b, it suffices to show $\forall i \in t$: setGcd s$|i$. Let $i \in t$. By 3, $i \in s$ and by using lemma 2.2 on $i \in s$ the proof is complete. □

By lemma 2.8 on $\forall i \in t$: setGcd s$|b(i)$ and $\sum_{r \in t}(a(r) \cdot (\uparrow)(b(r))) = (\uparrow)(\text{setGcd s})$, we get:

a) n $\in \mathbb{N}_0$,

b) $\forall m \geq n$: setGcd s$|m \to \exists$ a' : $\mathbb{N}_0 \to \mathbb{N}_0$, $m = \sum_{i \in t}(a'(i) \cdot b(i))$

By choosing t from 2 and n from a, we only need to show that $\forall m \geq n$: setGcd | m $\Rightarrow m \in$ Submodule.span $\mathbb{N}_0$ t. Let $m \in \mathbb{N}_0, m \geq n$: setGcd s$|m$. By theorem Submodule.mem_span_finset on t and m, we only need to show that $\exists f : \mathbb{N}_0 \to \mathbb{N}_0$, supp$(f) \subset t$ and $\sum_{a \in t}(f(a) \cdot a) = m$.
By b) on $m \in \mathbb{N}_0, m \geq n$: setGcd s$|m$, we get

i) $\exists$ a' : $\mathbb{N}_0 \to \mathbb{N}_0$

ii) $m = \sum_{i \in t}(a'(i) \cdot b(i))$

We define the function f : $\mathbb{N}_0 \to \mathbb{N}_0$ with

$$f(i) = \begin{cases} a'(i) & i \in t \\ 0 & \text{otherwise} \end{cases}$$

We choose the function f and by definition of f follows supp$(f) \subset t$ then all that is left to show is $\sum_{a \in t}(f(a) \cdot a) = m$.

$$m \overset{ii)}{=} \sum_{i \in t}(a'(i) \cdot b(i)) \underset{a'(i)=f(i), i \in t}{\overset{b=id}{=}} \sum_{a \in t}(f(a) \cdot a)$$

Remark: The theorem Submodule.mem_span_finset on s as finite set on M and x $\in$ M states that

$$x \in \text{span}_R(s) \leftrightarrow \exists f : M \to R, \ \text{supp}(f) \subset s \text{ and } \sum_{a \in s}(f(a) \cdot a) = x$$

Proof (formal):

```
lemma exists_mem_nat_submoduleSpan_of_le :
    ∃ (t : Finset ℕ) (n : ℕ), ∀ m ≥ n, setGcd s ∣ m → m ∈ Submodule.span ℕ t := by {

    rcases exists_sum_mul_eq_setGcd s with ⟨a,t,t_in_s, sum_eq_setGcd_s⟩

    let b : ℕ → ℕ := fun n ↦ n

    have sum_with_b_eq_setGcd_s : Σ r ∈ t, a r * b r = setGcd s := by {
      calc Σ r ∈ t, a r * b r = Σ n ∈ t, a n * ↑n := by {
        apply Finset.sum_congr rfl
        intro x x_in_t
        dsimp[b]
      }
      _ = ↑(setGcd s) := sum_eq_setGcd_s
    }

    have forall_i_in_t_g_dvd_b_im : ∀ i ∈ t, setGcd s ∣ b i := by {
      intro i i_in_t
      rw [← @Finset.mem_coe] at i_in_t
      exact setGcd_dvd (t_in_s i_in_t)
    }

    have h : ∃ n : ℕ, ∀ m ≥ n, setGcd s ∣ m → ∃ a' : ℕ → ℕ, m = Σ i ∈ t, a' i * b i := by {
      exact exists_nat_linearCombination_eq sum_with_b_eq_setGcd_s forall_i_in_t_g_dvd_b_im
    }

    rcases h with ⟨n, hn⟩

    use t
    use n

    intro m m_geq_n setGcd_s_dvd_m

    rw [@Submodule.mem_span_finset]

    rcases hn m m_geq_n setGcd_s_dvd_m with ⟨a', m_eq_sum⟩

    let f : ℕ → ℕ := fun n ↦ if n ∈ ↑t then a' n else 0

    use f

    constructor
    · rw [@Function.support_subset_iff']
      intro x x_not_in_t
      dsimp[f]
      rw[if_neg]
      exact x_not_in_t
    · calc Σ a ∈ t, f a · a = Σ i ∈ t, a' i * b i := by {
        apply Finset.sum_congr rfl
        intro x x_in_t
        dsimp[b]
        dsimp[f]
        rw [if_pos]
        exact x_in_t
      }
      _ = m := by {
        rw [m_eq_sum]
      }
}
```

Lemma 2.10. *Let s be a set of $\mathbb{N}_0$. Show that s is Noetherian.*

Proof (informal):

Proof (formal):

```
sorry
```

Lemma 2.11. *Let s be a set of* $\mathbb{N}_0$. *If s is an AddSubmonoid* $\mathbb{N}_0$, *then s.FG*

Proof (informal):

Proof (formal):

```
sorry
```