theory *Problem-1*
imports *Complex-Main*
begin

Consider a set S of $n \geq 3$ positive real numbers. Show that at most $n - 2$ distinct integer powers of 3 can be expressed as a sum of three distinct elements of S.

definition *intpow3* **where**
$intpow3\ x \longleftrightarrow (\exists k{::}int.\ 3\ powr\ k = x)$

lemma *card3-distinct*[*elim*]:
$card\ \{a,\ b,\ c\} = 3 \Longrightarrow a \neq b \wedge b \neq c \wedge c \neq a$
by (*metis One-nat-def add-Suc-right add-cancel-left-right card.empty card.insert card-2-iff equalityI finite.intros*(*1*) *insert-subset nat.simps*(*3*) *numeral-eq-Suc one-add-one order-refl pred-numeral-simps*(*3*))+

definition *threepows* **where**
$threepows\ S = \{a + b + c \mid a\ b\ c.\ \{a,\ b,\ c\} \subseteq S \wedge card\ \{a,\ b,\ c\} = 3 \wedge intpow3\ (a + b + c)\}$

lemma *threepows-mono*:
$A \subseteq B \Longrightarrow threepows\ A \subseteq threepows\ B$
by (*auto simp*: *threepows-def*)

lemma *threepows-intpow3*:
$x \in threepows\ S \Longrightarrow intpow3\ x$
by (*auto simp*: *threepows-def*)

lemma *threepowsI*[*intro*]:
assumes $\{a,\ b,\ c\} \subseteq S$ **and** $card\ \{a,\ b,\ c\} = 3$ **and** $intpow3\ (a + b + c)$
shows $a + b + c \in threepows\ S$
using *assms* **by** (*auto simp*: *threepows-def*)

lemma *threepowsE*[*elim*]:
assumes $x \in threepows\ S$
assumes $\bigwedge a\ b\ c.\ x = a + b + c \Longrightarrow \{a,\ b,\ c\} \subseteq S \Longrightarrow card\ \{a,\ b,\ c\} = 3 \Longrightarrow intpow3\ (a + b + c) \Longrightarrow P\ x$
shows $P\ x$
using *assms* **by** (*auto simp*: *threepows-def*)

lemma *threepows-trivial*:
assumes $card\ S < 3$ **and** $finite\ S$
shows $threepows\ S = \{\}$
proof (*rule ccontr*)
assume $threepows\ S \neq \{\}$
then obtain x **where** $x \in threepows\ S$ **by** *auto*
thus *False*
proof (*rule threepowsE*)
fix $a\ b\ c$
assume $\{a,\ b,\ c\} \subseteq S$ $card\ \{a,\ b,\ c\} = 3$
have $card\ \{a,\ b,\ c\} \leq card\ S$
using ‹*finite S*› ‹$\{a,\ b,\ c\} \subseteq S$› **by** (*intro card-mono*)
thus *False*
using ‹$card\ \{a,\ b,\ c\} = 3$› ‹$card\ S < 3$› **by** *simp*
qed
qed

lemma *threepows-largest*:
fixes $k{::}int$
assumes $3\ powr\ k \in threepows\ S$
obtains a **where** $a \in S$ **and** $a \geq 3\ powr\ (k - 1)$
proof −
{
assume $\nexists a.\ a \in S \wedge a \geq 3\ powr\ (k - 1)$
then have $*$: $a \in S \Longrightarrow a < 3\ powr\ (k - 1)$ **for** a **by** *auto*

```
    have x ∈ threepows S ⟹ x < 3 ∗ 3 powr (k − 1) for x
    proof (erule threepowsE)
      fix a b c
      assume {a, b, c} ⊆ S and x = a + b + c
      with ∗ have a < 3 powr (k − 1) and b < 3 powr (k − 1) and c < 3 powr (k − 1)
        by auto
      thus x < 3 ∗ 3 powr (k − 1)
        unfolding ‹x = a + b + c› by simp
    qed

    with assms have 3 powr k < 3 ∗ 3 powr (k − 1)
      by simp
    hence False by (simp add: powr-mult-base)
  }
  thus thesis using that by blast
qed

theorem problem1:
  fixes S :: real set
  assumes card S ≥ 3
  assumes ⋀x. x ∈ S ⟹ x > 0
  shows card (threepows S) ≤ card S − 2
  using assms
proof (induction card S arbitrary: S rule: less-induct)
  case less
  then have finite S
    using card.infinite by fastforce
  show ?case
  proof (cases card (threepows S) = 0)
    case False
    then have finite (threepows S) and threepows S ≠ {}
      by (auto intro: card-ge-0-finite)
    then have Max (threepows S) ∈ threepows S
      by (intro Max-in)
    with threepows-intpow3 intpow3-def obtain k::int where k: 3 powr k = Max (threepows S)
      by blast
    let ?discard = {x ∈ S. x ≥ 3 powr (k − 1)}
    have threepows S − {Max (threepows S)} = threepows (S − ?discard)
    proof −
      have Max (threepows S) ∉ threepows (S − ?discard)
      proof
        assume Max (threepows S) ∈ threepows (S − ?discard)
        then obtain a where a ∈ S − ?discard and a ≥ 3 powr (k − 1)
          unfolding k[symmetric] by (rule threepows-largest)
        thus False by simp
      qed
      moreover have threepows (S − ?discard) ⊆ threepows S
        by (intro threepows-mono; auto)
      moreover have x = Max (threepows S)
        if x ∈ threepows S and not-in-discard: x ∉ threepows (S − ?discard) for x
        using ‹x ∈ threepows S›
      proof (rule threepowsE)
        fix a b c
        assume x = a + b + c  {a, b, c} ⊆ S  card {a, b, c} = 3  intpow3 (a + b + c)
        have {a, b, c} ∩ ?discard ≠ {}
        proof
          assume {a, b, c} ∩ ?discard = {}
          with ‹{a, b, c} ⊆ S› have {a, b, c} ⊆ S − ?discard by simp
          hence a + b + c ∈ threepows (S − ?discard)
            using ‹card {a, b, c} = 3› ‹intpow3 (a + b + c)› by (intro threepowsI)
          with ‹x = a + b + c› and not-in-discard show False by simp
        qed
        then have a ≥ 3 powr (k − 1) ∨ b ≥ 3 powr (k − 1) ∨ c ≥ 3 powr (k − 1)
```

```
          by auto
        moreover from less.prems and ‹{a, b, c} ⊆ S› have a > 0 ∧ b > 0 ∧ c > 0
          by simp
        ultimately have ∗: a + b + c > 3 powr (k − 1)
          by linarith
        have a + b + c ≥ 3 powr k
        proof (intro leI notI)
          assume ∗∗: a + b + c < 3 powr k
          moreover from ‹intpow3 (a + b + c)› obtain k'::int where a + b + c = 3 powr k'
            unfolding intpow3-def by metis
          with ∗ and ∗∗ have 3 powr (k − 1) < 3 powr k' and 3 powr k' < 3 powr k
            by auto
          hence k − 1 < k' and k' < k
            by auto
          thus False by auto
        qed
        moreover from ‹x ∈ threepows S› have x ≤ Max (threepows S)
          using ‹finite (threepows S)› by (intro Max.coboundedI; auto)
        ultimately show x = Max (threepows S)
          using ‹x = a + b + c› and k by simp
      qed
      ultimately show ?thesis
        by auto
    qed
    hence card-threepows: card (threepows S) = Suc (card (threepows (S − ?discard)))
      by (metis ‹Max (threepows S) ∈ threepows S› ‹finite (threepows S)› card-Suc-Diff1)

    have ?discard ≠ {}
    proof −
      from k ‹Max (threepows S) ∈ threepows S› threepows-largest
      obtain a where a ∈ S and a ≥ 3 powr (k − 1)
        by metis
      hence a ∈ ?discard by simp
      thus ?discard ≠ {} by auto
    qed
    hence discard-strict: card (S − ?discard) < card S
      using ‹finite S› by (intro psubset-card-mono; auto)
    show ?thesis
    proof (cases card (S − ?discard) < 3)
      case True
      then have threepows (S − ?discard) = {}
        using ‹finite S› by (intro threepows-trivial; auto)
      hence card (threepows S) = 1
        by (simp add: card-threepows)
      with ‹card S ≥ 3› show ?thesis by simp
    next
      case False
      hence card (threepows (S − ?discard)) ≤ card (S − ?discard) − 2
        using discard-strict ‹finite S› by (intro less; auto)
      hence card (threepows S) ≤ Suc (card (S − ?discard) − 2)
        unfolding card-threepows by simp
      then show ?thesis
        using discard-strict False by simp
    qed
  qed simp
qed
end
```