Irrationality Criteria for Series by Erdős and Straus

Angeliki Koutsoukou-Argyraki and Wenda Li

March 17, 2025

Abstract

We formalise certain irrationality criteria for infinite series of the form:

$$\sum_n \frac{b_n}{\prod_{i \le n} a_i}$$

where b_n, a_i are integers. The result is due to P. Erdős and E.G. Straus [1], and in particular we formalise Theorem 2.1, Corollary 2.10 and Theorem 3.1. The latter is an application of Theorem 2.1 involving the prime numbers.

Contents

theory *Irrational-Series-Erdos-Straus* **imports**
  Prime-Number-Theorem.Prime-Number-Theorem
  Prime-Distribution-Elementary.PNT-Consequences
begin

1 Miscellaneous

lemma *suminf-comparison*:
  assumes *summable f* **and** *gf*: $\bigwedge n.$ *norm* $(g\ n) \le f\ n$
  shows *suminf g* $\le$ *suminf f*
proof (*rule suminf-le*)
  show $g\ n \le f\ n$ **for** n

using *gf*[*of n*] **by** *auto*
show *summable g*
using *assms summable-comparison-test'* **by** *blast*
show *summable f* **using** *assms*(*1*) **.**
qed

lemma *tendsto-of-int-diff-0*:
assumes $(\lambda n.\ f\ n - \textit{of-int}(g\ n)) \longrightarrow (0::\textit{real})$ $\forall_F$ *n in sequentially.* $f\ n > 0$
shows $\forall_F$ *n in sequentially.* $0 \leq g\ n$
proof −
have $\forall_F$ *n in sequentially.* $|f\ n - \textit{of-int}(g\ n)| < 1\ /\ 2$
using *assms*(*1*)[*unfolded tendsto-iff*,*rule-format*,*of 1/2*] **by** *auto*
then show *?thesis* **using** *assms*(*2*)
by *eventually-elim linarith*
qed

lemma *eventually-mono-sequentially*:
assumes *eventually P sequentially*
assumes $\bigwedge x.\ P\ (x+k) \Longrightarrow Q\ (x+k)$
shows *eventually Q sequentially*
using *sequentially-offset*[*OF assms*(*1*),*of k*]
apply (*subst eventually-sequentially-seg*[*symmetric*,*of - k*])
apply (*elim eventually-mono*)
by *fact*

lemma *frequently-eventually-at-top*:
fixes $P\ Q::'a::\textit{linorder} \Rightarrow \textit{bool}$
assumes *frequently P at-top eventually Q at-top*
shows *frequently* $(\lambda x.\ P\ x \wedge (\forall y \geq x.\ Q\ y))$ *at-top*
using *assms*
unfolding *frequently-def eventually-at-top-linorder*
by (*metis* (*mono-tags*, *opaque-lifting*) *le-cases order-trans*)

lemma *eventually-at-top-mono*:
fixes $P\ Q::'a::\textit{linorder} \Rightarrow \textit{bool}$
assumes *event-P*:*eventually P at-top*
assumes *PQ-imp*:$\bigwedge x.\ x \geq z \Longrightarrow \forall y \geq x.\ P\ y \Longrightarrow Q\ x$
shows *eventually Q at-top*
proof −
obtain *N* **where** $\forall n \geq N.\ P\ n$
by (*meson event-P eventually-at-top-linorder*)
then have *Q x* **when** $x \geq \textit{max}\ N\ z$ **for** *x*
using *PQ-imp that* **by** *auto*
then show *?thesis* **unfolding** *eventually-at-top-linorder*
by *blast*
qed

lemma *frequently-at-top-elim*:
fixes $P\ Q::'a::\textit{linorder} \Rightarrow \textit{bool}$

assumes $\exists_F x$ *in at-top*. $P\ x$
assumes $\bigwedge i.\ P\ i \Longrightarrow \exists j{>}i.\ Q\ j$
shows $\exists_F x$ *in at-top*. $Q\ x$
using *assms* **unfolding** *frequently-def eventually-at-top-linorder*
by (*meson leD le-cases less-le-trans*)

lemma *less-Liminf-iff*:
fixes X :: - $\Rightarrow$ - :: *complete-linorder*
shows $Liminf\ F\ X < C \longleftrightarrow (\exists y{<}C.\ frequently\ (\lambda x.\ y \geq X\ x)\ F)$
by (*force simp*: *not-less not-frequently not-le le-Liminf-iff simp flip*: *Not-eq-iff*)

lemma *sequentially-even-odd-imp*:
assumes $\forall_F\ N$ *in sequentially*. $P\ (2{*}N)$ $\forall_F\ N$ *in sequentially*. $P\ (2{*}N{+}1)$
shows $\forall_F\ n$ *in sequentially*. $P\ n$
proof $-$
obtain N **where** $N\text{-}P{:}\forall x{\geq}N.\ P\ (2 * x) \wedge P\ (2 * x + 1)$
using *eventually-conj*[*OF assms*]
unfolding *eventually-at-top-linorder* **by** *auto*
have $P\ n$ **when** $n \geq 2{*}N$ **for** n
proof $-$
define n' **where** $n' = n\ div\ 2$
then have $n' \geq N$ **using** *that* **by** *auto*
then have $P\ (2 * n') \wedge P\ (2 * n' + 1)$
using *N-P* **by** *auto*
then show *?thesis* **unfolding** n'-*def*
by (*cases even n*) *auto*
qed
then show *?thesis* **unfolding** *eventually-at-top-linorder* **by** *auto*
qed

2 Theorem 2.1 and Corollary 2.10

context
fixes $a\ b$ ::$nat{\Rightarrow}int$
assumes *a-pos*: $\forall\ n.\ a\ n > 0$ **and** *a-large*: $\forall_F\ n$ *in sequentially*. $a\ n > 1$
and *ab-tendsto*: $(\lambda n.\ |b\ n|\ /\ (a\ (n-1) * a\ n)) \longlonglongrightarrow 0$
begin

private lemma *aux-series-summable*: *summable* $(\lambda n.\ b\ n\ /\ (\prod k{\leq}n.\ a\ k))$
proof $-$
have $\bigwedge e.\ e{>}0 \Longrightarrow \forall_F\ x$ *in sequentially*. $|b\ x|\ /\ (a\ (x-1) * a\ x) < e$
using *ab-tendsto*[*unfolded tendsto-iff*]
apply (*simp add*: *abs-mult flip*: *of-int-abs*)
by (*subst* (*asm*) (*2*) *abs-of-pos*,*use* ‹$\forall\ n.\ a\ n > 0$› **in** *auto*)+
from *this*[*of 1*]
have $\forall_F\ x$ *in sequentially*. $|real\text{-}of\text{-}int(b\ x)| < (a\ (x-1) * a\ x)$
using ‹$\forall\ n.\ a\ n > 0$› **by** *auto*
moreover have $\forall n.\ (\prod k{\leq}n.\ real\text{-}of\text{-}int\ (a\ k)) > 0$
using *a-pos* **by** (*auto intro*!:*linordered-semidom-class.prod-pos*)

ultimately have $\forall_F$ *n in sequentially.* $|b\ n|\ /\ (\prod k{\le}n.\ a\ k)$
$< (a\ (n{-}1) * a\ n)\ /\ (\prod k{\le}n.\ a\ k)$
apply (*elim eventually-mono*)
by (*auto simp*:*field-simps*)
moreover have $|b\ n|\ /\ (\prod k{\le}n.\ a\ k) = norm\ (b\ n\ /\ (\prod k{\le}n.\ a\ k))$ **for** *n*
using ‹$\forall n.\ (\prod k{\le}n.\ real\text{-}of\text{-}int\ (a\ k)) > 0$›[*rule-format*,*of n*] **by** *auto*
ultimately have $\forall_F$ *n in sequentially.* $norm\ (b\ n\ /\ (\prod k{\le}n.\ a\ k))$
$< (a\ (n{-}1) * a\ n)\ /\ (\prod k{\le}n.\ a\ k)$
by *algebra*
moreover have *summable* $(\lambda n.\ (a\ (n{-}1) * a\ n)\ /\ (\prod k{\le}n.\ a\ k))$
proof −
obtain *s* **where** *a-gt-1*:$\forall\ n{\ge}s.\ a\ n > 1$
using *a-large*[*unfolded eventually-at-top-linorder*] **by** *auto*
define *cc* **where** $cc = (\prod k{<}s.\ a\ k)$
have $cc{>}0$
unfolding *cc-def* **by** (*meson a-pos prod-pos*)
have $(\prod k{\le}n{+}s.\ a\ k) \ge cc * 2\hat{}n$ **for** *n*
proof −
have $prod\ a\ \{s..{<}Suc\ (s + n)\} \ge 2\hat{}n$
proof (*induct n*)
case *0*
then show *?case* **using** *a-gt-1* **by** *auto*
next
case (*Suc n*)
moreover have $a\ (s + Suc\ n) \ge 2$
by (*smt* (*verit*, *ccfv-threshold*) *a-gt-1 le-add1*)
ultimately show *?case*
apply (*subst prod.atLeastLessThan-Suc*,*simp*)
using *mult-mono'*[*of* $2\ a\ (Suc\ (s + n))\ \ 2\ \hat{}\ n\ prod\ a\ \{s..{<}Suc\ (s + n)\}$]
by (*simp add*: *mult.commute*)
qed
moreover have $prod\ a\ \{0..(n + s)\} = prod\ a\ \{..{<}s\} * prod\ a\ \{s..{<}Suc\ (s + n)\}$
using *prod.atLeastLessThan-concat*[*of* $0\ s\ s{+}n{+}1\ a$]
by (*simp add*: *add.commute lessThan-atLeast0 prod.atLeastLessThan-concat prod.head-if*)
ultimately show *?thesis*
using ‹$cc{>}0$› **unfolding** *cc-def* **by** (*simp add*: *atLeast0AtMost*)
qed
then have $1/(\prod k{\le}n{+}s.\ a\ k) \le 1/(cc * 2\hat{}n)$ **for** *n*
proof −
assume *asm*:$\bigwedge n.\ cc * 2\ \hat{}\ n \le prod\ a\ \{..n + s\}$
then have $real\text{-}of\text{-}int\ (cc * 2\ \hat{}\ n) \le prod\ a\ \{..n + s\}$ **using** *of-int-le-iff* **by** *blast*
moreover have $prod\ a\ \{..n + s\} > 0$ **using** ‹$cc{>}0$› **by** (*simp add*: *a-pos prod-pos*)
ultimately show *?thesis* **using** ‹$cc{>}0$›
by (*auto simp*:*field-simps simp del*:*of-int-prod*)
qed

```
    moreover have summable (λn. 1/(cc * 2^n))
    proof -
      have summable (λn. 1/(2::int)^n)
        using summable-geometric[of 1/(2::int)] by (simp add:power-one-over)
      from summable-mult[OF this,of 1/cc] show ?thesis by auto
    qed
    ultimately have summable (λn. 1 / (∏ k≤n+s. a k))
      apply (elim summable-comparison-test'[where N=0])
      apply (unfold real-norm-def, subst abs-of-pos)
      by (auto simp: ‹∀ n. 0 < (∏ k≤n. real-of-int (a k))›)
    then have summable (λn. 1 / (∏ k≤n. a k))
      apply (subst summable-iff-shift[where k=s,symmetric])
      by simp
    then have summable (λn. (a (n+1) * a (n+2)) / (∏ k≤n+2. a k))
    proof -
      assume asm:summable (λn. 1 / real-of-int (prod a {..n}))
      have 1 / real-of-int (prod a {..n}) = (a (n+1) * a (n+2)) / (∏ k≤n+2. a
k) for n
      proof -
        have a (Suc (Suc n)) ≠ 0 a (Suc n) ≠0
          using a-pos by (metis less-irrefl)+
        then show ?thesis
          by (simp add: atLeast0-atMost-Suc atMost-atLeast0)
      qed
      then show ?thesis using asm by auto
    qed
    then show summable (λn. (a (n−1) * a n) / (∏ k≤n. a k))
      apply (subst summable-iff-shift[symmetric,of - 2])
      by auto
  qed
  ultimately show ?thesis
    apply (elim summable-comparison-test-ev[rotated])
    by (simp add: eventually-mono)
qed

private fun get-c::(nat ⇒ int) ⇒ (nat ⇒ int) ⇒ int ⇒ nat ⇒ (nat ⇒ int) where
  get-c a' b' B N 0 = round (B * b' N / a' N)|
  get-c a' b' B N (Suc n) = get-c a' b' B N n * a' (n+N) − B * b' (n+N)

lemma ab-rationality-imp:
  assumes ab-rational:(∑ n. (b n / (∏ i ≤ n. a i))) ∈ ℚ
  shows ∃ (B::int)>0. ∃ c::nat⇒ int.
          (∀F n in sequentially. B*b n = c n * a n − c(n+1) ∧ |c(n+1)|<a n/2)
           ∧ (λn. c (Suc n) / a n) ⟶ 0
proof -
  have [simp]:a n ≠ 0 for n using a-pos by (metis less-numeral-extra(3))
  obtain A::int and B::int where
    AB-eq:(∑ n. real-of-int (b n) / real-of-int (prod a {..n})) = A / B and B>0
  proof -
```

obtain *q*::*rat* **where** $(\sum n.\ \textit{real-of-int}\ (b\ n)\ /\ \textit{real-of-int}\ (\textit{prod}\ a\ \{..n\})) = \textit{real-of-rat}\ q$
using *ab-rational* **by** (*rule Rats-cases*) *simp*
moreover obtain *A*::*int* **and** *B*::*int* **where** $q = \textit{Rat.Fract}\ A\ B\ B > 0\ \textit{coprime}\ A\ B$
by (*rule Rat-cases*) *auto*
ultimately show *?thesis* **by** (*auto intro*!: *that*[*of A B*] *simp*:*of-rat-rat*)
qed
define *f* **where** $f \equiv (\lambda n.\ b\ n\ /\ \textit{real-of-int}\ (\textit{prod}\ a\ \{..n\}))$
define *R* **where** $R \equiv (\lambda N.\ (\sum n.\ B*b\ (n+N+1)\ /\ \textit{prod}\ a\ \{N..n+N+1\}))$
have *all-e-ubound*:$\forall e>0.\ \forall_F\ M\ \textit{in sequentially}.\ \forall n.\ |B*b\ (n+M+1)\ /\ \textit{prod}\ a\ \{M..n+M+1\}| < e/4 * 1/2\hat{}n$
proof *safe*
fix *e*::*real* **assume** $e>0$
obtain *N* **where** *N-a2*: $\forall n \geq N.\ a\ n \geq 2$
and *N-ba*: $\forall n \geq N.\ |b\ n|\ /\ (a\ (n-1) * a\ n) < e/(4*B)$
proof −
have $\forall_F\ n\ \textit{in sequentially}.\ |b\ n|\ /\ (a\ (n-1) * a\ n) < e/(4*B)$
using *order-topology-class.order-tendstoD*[*OF ab-tendsto*,*of* $e/(4*B)$] ‹$B>0$› ‹$e>0$›
by *auto*
moreover have $\forall_F\ n\ \textit{in sequentially}.\ a\ n \geq 2$
using *a-large* **by** (*auto elim*: *eventually-mono*)
ultimately have $\forall_F\ n\ \textit{in sequentially}.\ |b\ n|\ /\ (a\ (n-1) * a\ n) < e/(4*B) \wedge a\ n \geq 2$
by *eventually-elim auto*
then show *?thesis* **unfolding** *eventually-at-top-linorder* **using** *that*
by *auto*
qed
have *geq-N-bound*:$|B*b\ (n+M+1)\ /\ \textit{prod}\ a\ \{M..n+M+1\}| < e/4 * 1/2\hat{}n$ **when** $M \geq N$ **for** *n M*
proof −
define *D* **where** $D = B*b\ (n+M+1)/\ (a\ (n+M) * a\ (n+M+1))$
have $|B*b\ (n+M+1)\ /\ \textit{prod}\ a\ \{M..n+M+1\}| = |D\ /\ \textit{prod}\ a\ \{M..{<}n+M\}|$
proof −
have $\{M..n+M+1\} = \{M..{<}n+M\} \cup \{n+M,n+M+1\}$ **by** *auto*
then have $\textit{prod}\ a\ \{M..n+M+1\} = a\ (n+M) * a\ (n+M+1) * \textit{prod}\ a\ \{M..{<}n+M\}$ **by** *simp*
then show *?thesis* **unfolding** *D-def* **by** (*simp add*:*algebra-simps*)
qed
also have ... $< |e/4 * (1/\textit{prod}\ a\ \{M..{<}n+M\})|$
proof −
have $|D| < e/4$
unfolding *D-def* **using** *N-ba*[*rule-format*, *of* $n+M+1$] ‹$B>0$› ‹$M \geq N$› ‹$e>0$› *a-pos*
by (*auto simp*:*field-simps abs-mult abs-of-pos*)
from *mult-strict-right-mono*[*OF this*,*of* $1/\textit{prod}\ a\ \{M..{<}n+M\}$] *a-pos* ‹$e>0$›
show *?thesis*
apply (*auto simp*:*abs-prod abs-mult prod-pos*)

by (*subst* (*2*) *abs-of-pos*,*auto*)+
qed
also have ... $\leq$ *e*/*4* $*$ *1*/*2*^*n*
proof −
have *prod a* {*M*..<*n*+*M*} $\geq$ *2*^*n*
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
then show *?case*
using ‹*M*≥*N*› **by** (*simp add*: *N-a2 mult.commute mult-mono′ prod.atLeastLessThan-Suc*)
qed
then have *real-of-int* (*prod a* {*M*..<*n*+*M*}) $\geq$ *2*^*n*
using *numeral-power-le-of-int-cancel-iff* **by** *blast*
then show *?thesis* **using** ‹*e*>*0*› **by** (*auto simp*:*divide-simps*)
qed
finally show *?thesis* **.**
qed
show $\forall_F$ *M in sequentially.* $\forall$ *n.* |*real-of-int* (*B* $*$ *b* (*n* + *M* + *1*))
/ *real-of-int* (*prod a* {*M*..*n* + *M* + *1*})| < *e* / *4* $*$ *1* / *2* ^ *n*
apply (*rule eventually-sequentiallyI*[*of N*])
using *geq-N-bound* **by** *blast*
qed
have *R-tendsto-0*:*R* $\longrightarrow$ *0*
proof (*rule tendstoI*)
fix *e*::*real* **assume** *e*>*0*
show $\forall_F$ *x in sequentially. dist* (*R x*) *0* < *e* **using** *all-e-ubound*[*rule-format*,*OF*
‹*e*>*0*›]
proof *eventually-elim*
case (*elim M*)
define *g* **where** *g* = (λ*n. B*$*$*b* (*n*+*M*+*1*) / *prod a* {*M*..*n*+*M*+*1*})
have *g-lt*:|*g n*| < *e*/*4* $*$ *1*/*2*^*n* **for** *n*
using *elim* **unfolding** *g-def* **by** *auto*
have §: *summable* (λ*n.* (*e*/*4*) $*$ (*1*/*2*)^*n*)
by *simp*
then have *g-abs-summable*:*summable* (λ*n.* |*g n*|)
apply (*elim summable-comparison-test′*)
by (*metis abs-idempotent g-lt less-eq-real-def power-one-over real-norm-def
times-divide-eq-right*)
have |$\sum$*n. g n*| $\leq$ ($\sum$*n.* |*g n*|) **by** (*rule summable-rabs*[*OF g-abs-summable*])
also have ... $\leq$($\sum$*n. e*/*4* $*$ *1*/*2*^*n*)
proof (*rule suminf-comparison*)
show *summable* (λ*n. e*/*4* $*$ *1*/*2*^*n*)
using § **unfolding** *power-divide* **by** *simp*
show $\bigwedge$*n. norm* |*g n*| $\leq$ *e* / *4* $*$ *1* / *2* ^ *n* **using** *g-lt less-eq-real-def* **by**
auto
qed
also have ... = (*e*/*4*) $*$ ($\sum$*n.* (*1*/*2*)^*n*)

apply (*subst suminf-mult*[*symmetric*])
by (*auto simp*: *algebra-simps power-divide*)
also have ... = *e*/*2* **by** (*simp add*:*suminf-geometric*[*of 1*/*2*])
finally have $|\sum n.\ g\ n| \leq e\ /\ 2$ **.**
then show *dist* (*R M*) *0* < *e* **unfolding** *R-def g-def* **using** ‹*e*>*0*› **by** *auto*
qed
qed

obtain *N* **where** *R-N-bound*:$\forall M \geq N.\ |R\ M| \leq\ 1\ /\ 4$
and *N-geometric*:$\forall M{\geq}N.\ \forall n.\ |\mathit{real\text{-}of\text{-}int}\ (B * b\ (n + M + 1))\ /\ (\mathit{prod}\ a\ \{M..n + M + 1\})| < 1\ /\ 2\ \hat{}\ n$
proof −
obtain *N1* **where** *N1*:$\forall M \geq N1.\ |R\ M| \leq\ 1\ /\ 4$
using *metric-LIMSEQ-D*[*OF R-tendsto-0*,*of 1*/*4*] *all-e-ubound*[*rule-format*,*of 4*,*unfolded eventually-sequentially*]
by (*auto simp*:*less-eq-real-def*)
obtain *N2* **where** *N2*:$\forall M{\geq}N2.\ \forall n.\ |\mathit{real\text{-}of\text{-}int}\ (B * b\ (n + M + 1))$
$/\ (\mathit{prod}\ a\ \{M..n + M + 1\})| < 1\ /\ 2\ \hat{}\ n$
using *all-e-ubound*[*rule-format*,*of 4*,*unfolded eventually-sequentially*]
by (*auto simp*:*less-eq-real-def*)
define *N* **where** *N*=*max N1 N2*
show *?thesis* **using** *that*[*of N*] *N1 N2* **unfolding** *N-def* **by** *simp*
qed

define *C* **where** $C = B * \mathit{prod}\ a\ \{..{<}N\} * (\sum n{<}N.\ f\ n)$
have *summable f*
unfolding *f-def* **using** *aux-series-summable* **.**
have $A * \mathit{prod}\ a\ \{..{<}N\} = C + B * b\ N\ /\ a\ N\ + R\ N$
proof −
have $A * \mathit{prod}\ a\ \{..{<}N\} = B * \mathit{prod}\ a\ \{..{<}N\} * (\sum n.\ f\ n)$
unfolding *AB-eq f-def* **using** ‹*B*>*0*› **by** *auto*
also have ... $= B * \mathit{prod}\ a\ \{..{<}N\} * ((\sum n{<}N{+}1.\ f\ n) + (\sum n.\ f\ (n{+}N{+}1)))$
using *suminf-split-initial-segment*[*OF* ‹*summable f*›, *of N*+*1*] **by** *auto*
also have ... $= B * \mathit{prod}\ a\ \{..{<}N\} * ((\sum n{<}N.\ f\ n) + f\ N + (\sum n.\ f\ (n{+}N{+}1)))$
using *sum.atLeast0-lessThan-Suc* **by** *simp*
also have ... $= C + B * b\ N\ /\ a\ N + (\sum n.\ B{*}b\ (n{+}N{+}1)\ /\ \mathit{prod}\ a\ \{N..n{+}N{+}1\})$
proof −
have $B * \mathit{prod}\ a\ \{..{<}N\} * f\ N = B * b\ N\ /\ a\ N$
proof −
have $\{..N\} =\ \{..{<}N\} \cup \{N\}$ **using** *ivl-disj-un-singleton*(*2*) **by** *blast*
then show *?thesis* **unfolding** *f-def* **by** *auto*
qed
moreover have $B * \mathit{prod}\ a\ \{..{<}N\} * (\sum n.\ f\ (n{+}N{+}1)) = (\sum n.\ B{*}b\ (n{+}N{+}1)\ /\ \mathit{prod}\ a\ \{N..n{+}N{+}1\})$
proof −
have *summable* ($\lambda n.\ f\ (n + N + 1)$)
using ‹*summable f*› *summable-iff-shift*[*of f N*+*1*] **by** *auto*
moreover have $\mathit{prod}\ a\ \{..{<}N\} * f\ (n + N + 1) = b\ (n + N + 1)\ /\ \mathit{prod}$

$a\ \{N..n + N + 1\}$ **for** n

proof $-$

have $\{..n + N + 1\} = \{..<N\} \cup \{N..n + N + 1\}$ **by** *auto*

then show *?thesis*

unfolding *f-def*

apply *simp*

apply (*subst prod.union-disjoint*)

by *auto*

qed

ultimately show *?thesis*

apply (*subst suminf-mult*[*symmetric*])

by (*auto simp*: *mult.commute mult.left-commute*)

qed

ultimately show *?thesis* **unfolding** *C-def* **by** (*auto simp*:*algebra-simps*)

qed

also have $... = C + B * b\ N\ /\ a\ N\ + R\ N$

unfolding *R-def* **by** *simp*

finally show *?thesis* **.**

qed

have *R-bound*:$|R\ M| \leq 1\ /\ 4$ **and** *R-Suc*:$R\ (Suc\ M) = a\ M * R\ M - B * b\ (Suc\ M)\ /\ a\ (Suc\ M)$

when $M \geq N$ **for** M

proof $-$

define g **where** $g = (\lambda n.\ B{*}b\ (n{+}M{+}1)\ /\ prod\ a\ \{M..n{+}M{+}1\})$

have *g-abs-summable*:*summable* $(\lambda n.\ |g\ n|)$

proof $-$

have *summable* $(\lambda n.\ (1/2{::}real)\ \hat{}\ n)$

by *simp*

moreover have $|g\ n| < 1/2\hat{}n$ **for** n

using *N-geometric*[*rule-format*,*OF that*] **unfolding** *g-def* **by** *simp*

ultimately show *?thesis*

apply (*elim summable-comparison-test'*)

by (*simp add*: *less-eq-real-def power-one-over*)

qed

show $|R\ M| \leq 1\ /\ 4$ **using** *R-N-bound*[*rule-format*,*OF that*] **.**

have $R\ M = (\sum n.\ g\ n)$ **unfolding** *R-def g-def* **by** *simp*

also have $... = g\ 0 + (\sum n.\ g\ (Suc\ n))$

apply (*subst suminf-split-head*)

using *summable-rabs-cancel*[*OF g-abs-summable*] **by** *auto*

also have $... = g\ 0 + 1/a\ M * (\sum n.\ a\ M * g\ (Suc\ n))$

apply (*subst suminf-mult*)

by (*auto simp*: *g-abs-summable summable-Suc-iff summable-rabs-cancel*)

also have $... = g\ 0 + 1/a\ M * R\ (Suc\ M)$

proof $-$

have $a\ M * g\ (Suc\ n) = B * b\ (n + M + 2)\ /\ prod\ a\ \{Suc\ M..n + M + 2\}$ **for** n

proof $-$

have $\{M..Suc\ (Suc\ (M + n))\} = \{M\} \cup \{Suc\ M..Suc\ (Suc\ (M + n))\}$ **by** *auto*

```
      then show ?thesis
        unfolding g-def using ‹B>0› by (auto simp:algebra-simps)
    qed
    then have (∑ n. a M ∗ g (Suc n)) = R (Suc M)
      unfolding R-def by auto
    then show ?thesis by auto
  qed
  finally have R M = g 0 + 1 / a M ∗ R (Suc M) .
  then have R (Suc M) = a M ∗ R M − g 0 ∗ a M
    by (auto simp:algebra-simps)
  moreover have {M..Suc M} = {M,Suc M} by auto
  ultimately show R (Suc M) = a M ∗ R M − B ∗ b (Suc M) / a (Suc M)
    unfolding g-def by auto
qed

define c where c = (λn. if n≥N then get-c a b B N (n−N) else undefined)
have c-rec:c (n+1) = c n ∗ a n −  B ∗ b n when n ≥ N for n
  unfolding c-def using that by (auto simp:Suc-diff-le)
have c-R:c (Suc n) / a n = R n when n ≥ N for n
  using that
proof (induct rule:nat-induct-at-least)
  case base
  have | c (N+1) / a N | ≤ 1/2
  proof −
    have c N = round (B ∗ b N / a N) unfolding c-def by simp
    moreover have c (N+1) / a N = c N − B ∗ b N / a N
      using a-pos[rule-format,of N]
      by (auto simp:c-rec[of N,simplified] divide-simps)
    ultimately show ?thesis using of-int-round-abs-le by auto
  qed
  moreover have |R N| ≤ 1 / 4 using R-bound[of N] by simp
  ultimately have |c (N+1) / a N − R N | < 1 by linarith
  moreover have c (N+1) / a N − R N ∈ ℤ
  proof −
    have c (N+1) / a N = c N − B ∗ b N / a N
      using a-pos[rule-format,of N]
      by (auto simp:c-rec[of N,simplified] divide-simps)
    moreover have  B ∗ b N / a N + R N ∈ ℤ
    proof −
      have C = B ∗ (∑ n<N. prod a {..<N} ∗ (b n / prod a {..n}))
        unfolding C-def f-def by (simp add:sum-distrib-left algebra-simps)
      also have ... = B ∗ (∑ n<N. prod a {n<..<N} ∗ b n)
      proof −
        have {..<N} = {n<..<N} ∪ {..n} if n<N for n
          by (simp add: ivl-disj-un-one(1) sup-commute that)
        then show ?thesis
          using ‹B>0›
          apply simp
          apply (subst prod.union-disjoint)
```

```
          by auto
        qed
        finally have C = real-of-int (B * (∑n<N. prod a {n<..<N} * b n)) .
        then have C ∈ ℤ using Ints-of-int by blast
        moreover note ‹A * prod a {..<N} = C + B * b N / a N + R N›
        ultimately show ?thesis
          by (metis Ints-diff Ints-of-int add.assoc add-diff-cancel-left')
      qed
      ultimately show ?thesis by (simp add: diff-diff-add)
    qed
    ultimately have c (N+1) / a N − R N = 0
      by (metis Ints-cases less-irrefl of-int-0 of-int-lessD)
    then show ?case by simp
  next
    case (Suc n)
    have c (Suc (Suc n)) / a (Suc n) = c (Suc n) − B * b (Suc n) / a (Suc n)
      apply (subst c-rec[of Suc n,simplified])
      using ‹N ≤ n› by (auto simp: divide-simps)
    also have ... = a n * R n − B * b (Suc n) / a (Suc n)
      using Suc by (auto simp: divide-simps)
    also have ... = R (Suc n)
      using R-Suc[OF ‹N ≤ n›] by simp
    finally  show ?case .
  qed
  have ca-tendsto-zero:(λn. c (Suc n) / a n) ⟶ 0
    using R-tendsto-0
    apply (elim filterlim-mono-eventually)
    using c-R by (auto intro!:eventually-sequentiallyI[of N])
  have ca-bound:|c (n + 1)| < a n / 2 when n ≥ N for n
  proof −
     have |c (Suc n)| / a n  = |c (Suc n) / a n| using a-pos[rule-format,of n] by
auto
    also have ... = |R n| using c-R[OF that] by auto
    also have ... < 1/2 using R-bound[OF that] by auto
    finally have |c (Suc n)| / a n < 1/2 .
    then show ?thesis using a-pos[rule-format,of n] by auto
  qed


  show ∃B>0. ∃c. (∀F n in sequentially. B * b n = c n * a n − c (n + 1)
          ∧ real-of-int |c (n + 1)| < a n / 2) ∧ (λn. c (Suc n) / a n) ⟶ 0
    unfolding eventually-at-top-linorder
    apply (rule exI[of - B],use ‹B>0› in simp)
    apply (intro exI[of -c] exI[of - N])
    using c-rec ca-bound ca-tendsto-zero
    by fastforce
qed

private lemma imp-ab-rational:
```

```
  assumes ∃ (B::int)>0. ∃ c::nat⇒ int.
              (∀F n in sequentially. B*b n = c n * a n − c(n+1) ∧ |c(n+1)|<a
n/2)
  shows (∑ n. (b n / (∏ i ≤ n. a i))) ∈ ℚ
proof −
  obtain B::int and c::nat⇒int and N::nat where B>0 and
    large-n:∀ n≥N. B * b n = c n * a n − c (n + 1) ∧ real-of-int |c (n + 1)| < a
n / 2
                    ∧ a n≥2
  proof −
    obtain B c where B>0 and event1:∀F n in sequentially. B * b n = c n * a
n − c (n + 1)
            ∧ real-of-int |c (n + 1)| < real-of-int (a n) / 2
      using assms by auto
    from eventually-conj[OF event1 a-large,unfolded eventually-at-top-linorder]
    obtain N where ∀ n≥N. (B * b n = c n * a n − c (n + 1)
                        ∧ real-of-int |c (n + 1)| < real-of-int (a n) / 2) ∧ 2 ≤ a n
      by fastforce
    then show ?thesis using that[of B N c] ‹B>0› by auto
  qed
  define f where f=(λn. real-of-int (b n) / real-of-int (prod a {..n}))
  define S where S = (∑ n. f n)
  have summable f
    unfolding f-def by (rule aux-series-summable)
  define C where C=B*prod a {..<N} * (∑ n<N. f n)
  have B*prod a {..<N} * S = C + real-of-int (c N)
  proof −
    define h1 where h1 ≡ (λn. (c (n+N) * a (n+N)) / prod a {N..n+N})
    define h2 where h2 ≡ (λn. c (n+N+1) / prod a {N..n+N})
    have f-h12: B * prod a {..<N}*f (n+N) = h1 n − h2 n for n
    proof −
      define g1 where g1 ≡ (λn. B * b (n+N))
      define g2 where g2 ≡ (λn. prod a {..<N} / prod a {..n + N})
      have B * prod a {..<N}*f (n+N) = (g1 n * g2 n)
        unfolding f-def g1-def g2-def by (auto simp:algebra-simps)
      moreover have g1 n = c (n+N) * a (n+N) − c (n+N+1)
        using large-n[rule-format,of n+N] unfolding g1-def by auto
      moreover have g2 n = (1/prod a {N..n+N})
      proof −
         have prod a ({..<N} ∪ {N..n + N}) = prod a {..<N} * prod a {N..n +
N}
          apply (rule prod.union-disjoint[of {..<N} {N..n+N} a])
          by auto
        moreover have prod a ({..<N} ∪ {N..n + N}) = prod a {..n+N}
          by (simp add: ivl-disj-un-one(4))
        ultimately show ?thesis
          unfolding g2-def
          apply simp
          using a-pos by (metis less-irrefl)
```

qed
ultimately have *B∗prod a* $\{..<N\}$*∗f* $(n+N) = (c\ (n+N) * a\ (n+N) - c\ (n+N+1))$ / *prod a* $\{N..n+N\}$
by *auto*
also have ... = *h1 n* − *h2 n*
unfolding *h1-def h2-def* **by** (*auto simp*:*algebra-simps diff-divide-distrib*)
finally show *?thesis* **by** *simp*
qed
have *B∗prod a* $\{..<N\} * S =$ *B∗prod a* $\{..<N\} * ((\sum n<N.\ f\ n) + (\sum n.\ f\ (n+N)))$
using *suminf-split-initial-segment*[*OF* ‹*summable f*›,*of N*]
unfolding *S-def* **by** (*auto simp*:*algebra-simps*)
also have ... = $C +$ *B∗prod a* $\{..<N\}*(\sum n.\ f\ (n+N))$
unfolding *C-def* **by** (*auto simp*:*algebra-simps*)
also have ... = $C + (\sum n.\ h1\ n - h2\ n)$
apply (*subst suminf-mult*[*symmetric*])
using ‹*summable f*› *f-h12* **by** *auto*
also have ... = *C* + *h1 0*
proof −
have $(\lambda n.\ \sum i\le n.\ h1\ i - h2\ i) \longrightarrow (\sum i.\ h1\ i - h2\ i)$
proof (*rule summable-LIMSEQ′*)
have $(\lambda i.\ h1\ i - h2\ i) = (\lambda i.$ *real-of-int* (*B ∗ prod a* $\{..<N\}) * f\ (i + N))$
using *f-h12* **by** *auto*
then show *summable* $(\lambda i.\ h1\ i - h2\ i)$
using ‹*summable f*› **by** (*simp add*: *summable-mult*)
qed
moreover have $(\sum i\le n.\ h1\ i - h2\ i) = h1\ 0 - h2\ n$ **for** *n*
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
have $(\sum i\le Suc\ n.\ h1\ i - h2\ i) = (\sum i\le n.\ h1\ i - h2\ i) + h1\ (n+1) - h2\ (n+1)$
by *auto*
also have ... = $h1\ 0 - h2\ n + h1\ (n+1) - h2\ (n+1)$ **using** *Suc* **by** *auto*
also have ... = $h1\ 0 - h2\ (n+1)$
proof −
have $h2\ n = h1\ (n+1)$
unfolding *h2-def h1-def*
apply (*auto simp*:*prod.nat-ivl-Suc′*)
using *a-pos* **by** (*metis less-numeral-extra*(*3*))
then show *?thesis* **by** *auto*
qed
finally show *?case* **by** *simp*
qed
ultimately have $(\lambda n.\ h1\ 0 - h2\ n) \longrightarrow (\sum i.\ h1\ i - h2\ i)$ **by** *simp*
then have $h2 \longrightarrow (h1\ 0 - (\sum i.\ h1\ i - h2\ i))$
apply (*elim metric-tendsto-imp-tendsto*)

by (*auto intro*!:*eventuallyI simp add*:*dist-real-def*)
moreover have *h2* ⟶ *0*
proof −
have *h2-n*:|*h2 n*| < (*1* / *2*)^(*n*+*1*) **for** *n*
proof −
have |*h2 n*| = |*c* (*n* + *N* + *1*)| / *prod a* {*N*..*n* + *N*}
unfolding *h2-def abs-divide*
using *a-pos* **by** (*simp add*: *abs-of-pos prod-pos*)
also have ... < (*a* (*N*+*n*) / *2*) / *prod a* {*N*..*n* + *N*}
unfolding *h2-def*
apply (*rule divide-strict-right-mono*)
subgoal using *large-n*[*rule-format*,*of N*+*n*] **by** (*auto simp*:*algebra-simps*)
subgoal using *a-pos* **by** (*simp add*: *prod-pos*)
done
also have ... = *1* / (*2*∗*prod a* {*N*..<*n* + *N*})
apply (*subst ivl-disj-un*(*6*)[*of N n*+*N*,*symmetric*])
using *a-pos*[*rule-format*,*of N*+*n*] **by** (*auto simp*:*algebra-simps*)
also have ... ≤ (*1*/*2*)^(*n*+*1*)
proof (*induct n*)
case *0*
then show *?case* **by** *auto*
next
case (*Suc n*)
define *P* **where** *P*=*1* / *real-of-int* (*2* ∗ *prod a* {*N*..<*n* + *N*})
have *1* / *real-of-int* (*2* ∗ *prod a* {*N*..<*Suc n* + *N*}) = *P* / *a* (*n*+*N*)
unfolding *P-def* **by** (*auto simp*: *prod.atLeastLessThan-Suc*)
also have ... ≤ ((*1* / *2*) ^(*n* + *1*)) / *a* (*n*+*N*)
apply (*rule divide-right-mono*)
subgoal unfolding *P-def* **using** *Suc* **by** *auto*
subgoal by (*simp add*: *a-pos less-imp-le*)
done
also have ... ≤ ((*1* / *2*) ^(*n* + *1*)) / *2*
apply (*rule divide-left-mono*)
using *large-n*[*rule-format*,*of n*+*N*,*simplified*] **by** *auto*
also have ... = (*1* / *2*) ^(*n* + *2*) **by** *auto*
finally show *?case* **by** *simp*
qed
finally show *?thesis* **.**
qed
have (λ*n*. (*1* / *2*)^(*n*+*1*)) ⟶ (*0*::*real*)
using *tendsto-mult-right-zero*[*OF LIMSEQ-abs-realpow-zero2*[*of 1*/*2*,*simplified*],*of 1*/*2*]
by *auto*
then show *?thesis*
apply (*elim Lim-null-comparison*[*rotated*])
using *h2-n less-eq-real-def* **by** (*auto intro*!:*eventuallyI*)
qed
ultimately have (∑ *i*. *h1 i* − *h2 i*) = *h1 0*
using *LIMSEQ-unique* **by** *fastforce*

then show *?thesis* **by** *simp*
qed
also have ... = $C + c\ N$
unfolding *h1-def* **using** *a-pos*
by *auto* (*metis less-irrefl*)
finally show *?thesis* **.**
qed
then have $S = (C + \textit{real-of-int}\ (c\ N))\ /\ (B{*}\textit{prod}\ a\ \{..{<}N\})$
by (*metis* ‹$0 < B$› *a-pos less-irrefl mult.commute mult-pos-pos nonzero-mult-div-cancel-right of-int-eq-0-iff prod-pos*)
moreover have ... $\in \mathbb{Q}$
unfolding *C-def f-def* **by** (*intro Rats-divide Rats-add Rats-mult Rats-of-int Rats-sum*)
ultimately show $S \in \mathbb{Q}$ **by** *auto*
qed

theorem *theorem-2-1-Erdos-Straus* :
$(\sum n.\ (b\ n\ /\ (\prod i \le n.\ a\ i))) \in \mathbb{Q} \longleftrightarrow (\exists\ (B{::}int){>}0.\ \exists\ c{::}nat{\Rightarrow}\ int.$
$(\forall_F\ n\ in\ sequentially.\ B{*}b\ n = c\ n * a\ n - c(n{+}1) \wedge |c(n{+}1)|{<}a\ n/2))$
using *ab-rationality-imp imp-ab-rational* **by** *auto*

The following is a Corollary to Theorem 2.1.

corollary *corollary-2-10-Erdos-Straus*:
assumes *ab-event*:$\forall_F\ n\ in\ sequentially.\ b\ n > 0 \wedge a\ (n{+}1) \ge a\ n$
and *ba-lim-leq*:$lim\ (\lambda n.\ (b(n{+}1) - b\ n\ )/a\ n) \le 0$
and *ba-lim-exist*:$convergent\ (\lambda n.\ (b(n{+}1) - b\ n\ )/a\ n)$
and $liminf\ (\lambda n.\ a\ n\ /\ b\ n) = 0$
shows $(\sum n.\ (b\ n\ /\ (\prod i \le n.\ a\ i))) \notin \mathbb{Q}$
proof
assume $(\sum n.\ (b\ n\ /\ (\prod i \le n.\ a\ i))) \in \mathbb{Q}$
then obtain $B\ c$ **where** $B{>}0$ **and** *abc-event*:$\forall_F\ n\ in\ sequentially.\ B * b\ n = c\ n * a\ n - c\ (n + 1)$
$\wedge\ |c\ (n + 1)| < a\ n\ /\ 2$ **and** *ca-vanish*: $(\lambda n.\ c\ (Suc\ n)\ /\ a\ n) \longrightarrow 0$
using *ab-rationality-imp* **by** *auto*

have *bac-close*:$(\lambda n.\ B * b\ n\ /\ a\ n - c\ n) \longrightarrow 0$
proof −
have $\forall_F\ n\ in\ sequentially.\ B * b\ n - c\ n * a\ n + c\ (n + 1) = 0$
using *abc-event* **by** (*auto elim*!:*eventually-mono*)
then have $\forall_F\ n\ in\ sequentially.\ (B * b\ n - c\ n * a\ n + c\ (n{+}1))\ /\ a\ n = 0$
apply *eventually-elim*
by *auto*
then have $\forall_F\ n\ in\ sequentially.\ B * b\ n\ /\ a\ n - c\ n\ + c\ (n + 1)\ /\ a\ n = 0$
apply *eventually-elim*
using *a-pos* **by** (*auto simp*:*divide-simps*) (*metis less-irrefl*)
then have $(\lambda n.\ B * b\ n\ /\ a\ n - c\ n\ + c\ (n + 1)\ /\ a\ n) \longrightarrow 0$
by (*simp add*: *eventually-mono tendsto-iff*)
from *tendsto-diff*[*OF this ca-vanish*]
show *?thesis* **by** *auto*

qed

have *c-pos*:∀$_F$ *n in sequentially. c n > 0*
proof −
from *bac-close* **have** ∗:∀$_F$ *n in sequentially. c n ≥ 0*
apply (*elim tendsto-of-int-diff-0*)
using *ab-event a-large* **apply** (*eventually-elim*)
using ‹*B>0*› **by** *auto*
show *?thesis*
proof (*rule ccontr*)
assume ¬ (∀$_F$ *n in sequentially. c n > 0*)
moreover have ∀$_F$ *n in sequentially. c* (*Suc n*) ≥ *0* ∧ *c n≥0*
using ∗ *eventually-sequentially-Suc*[*of λn. c n≥0*]
by (*metis* (*mono-tags*, *lifting*) *eventually-at-top-linorder le-Suc-eq*)
ultimately have ∃$_F$ *n in sequentially. c n = 0* ∧ *c* (*Suc n*) ≥ *0*
using *eventually-elim2 frequently-def* **by** *fastforce*
moreover have ∀$_F$ *n in sequentially. b n > 0* ∧ *B* ∗ *b n = c n* ∗ *a n* − *c* (*n* + *1*)
using *ab-event abc-event* **by** *eventually-elim auto*
ultimately have ∃$_F$ *n in sequentially. c n = 0* ∧ *c* (*Suc n*) ≥ *0* ∧ *b n > 0* ∧ *B* ∗ *b n = c n* ∗ *a n* − *c* (*n* + *1*)
using *frequently-eventually-frequently* **by** *fastforce*
from *frequently-ex*[*OF this*]
obtain *n* **where** *c n = 0 c* (*Suc n*) ≥ *0 b n > 0*
B ∗ *b n = c n* ∗ *a n* − *c* (*n* + *1*)
by *auto*
then have *B* ∗ *b n* ≤ *0* **by** *auto*
then show *False* **using** ‹*b n>0*› ‹*B > 0*› **using** *mult-pos-pos not-le* **by** *blast*
qed
qed

have *bc-epsilon*:∀$_F$ *n in sequentially. b* (*n+1*) / *b n >* (*c* (*n+1*) − *ε*) / *c n* **when** *ε>0 ε<1* **for** *ε*::*real*
proof −
have ∀$_F$ *x in sequentially.* |*c* (*Suc x*) / *a x*| < *ε* / *2*
using *ca-vanish*[*unfolded tendsto-iff*,*rule-format*, *of ε/2*] ‹*ε>0*› **by** *auto*
moreover then have ∀$_F$ *x in sequentially.* |*c* (*x+2*) / *a* (*x+1*)| < *ε* / *2*
apply (*subst* (*asm*) *eventually-sequentially-Suc*[*symmetric*])
by *simp*
moreover have ∀$_F$ *n in sequentially. B* ∗ *b* (*n+1*) = *c* (*n+1*) ∗ *a* (*n+1*) − *c* (*n* + *2*)
using *abc-event*
apply (*subst* (*asm*) *eventually-sequentially-Suc*[*symmetric*])
by (*auto elim*:*eventually-mono*)
moreover have ∀$_F$ *n in sequentially. c n > 0* ∧ *c* (*n+1*) > *0* ∧ *c* (*n+2*) > *0*
proof −
have ∀$_F$ *n in sequentially. 0 < c* (*Suc n*)
using *c-pos* **by** (*subst eventually-sequentially-Suc*) *simp*
moreover then have ∀$_F$ *n in sequentially. 0 < c* (*Suc* (*Suc n*))

using *c-pos* **by** (*subst eventually-sequentially-Suc*) *simp*
ultimately show *?thesis* **using** *c-pos* **by** *eventually-elim auto*
qed
ultimately show *?thesis* **using** *ab-event abc-event*
proof *eventually-elim*
case (*elim n*)
define ε_0 ε_1 **where** $\varepsilon_0 = c\ (n+1)\ /\ a\ n$ **and** $\varepsilon_1 = c\ (n+2)\ /\ a\ (n+1)$
have $\varepsilon_0 > 0$ $\varepsilon_1 > 0$ $\varepsilon_0 < \varepsilon/2$ $\varepsilon_1 < \varepsilon/2$ **using** *a-pos elim* **by** (*auto simp*: ε_0*-def* ε_1*-def*)
have $(\varepsilon - \varepsilon_1) * c\ n > 0$
using ‹$\varepsilon_1 < \varepsilon\ /\ 2$› *elim*(*4*) *that*(*1*) **by** *auto*
moreover have $\varepsilon_0 * (c\ (n+1) - \varepsilon) > 0$
using ‹$0 < \varepsilon_0$› *elim*(*4*) *that*(*2*) **by** *auto*
ultimately have $(\varepsilon - \varepsilon_1) * c\ n + \varepsilon_0 * (c\ (n+1) - \varepsilon) > 0$ **by** *auto*
moreover have *gt0*: $c\ n - \varepsilon_0 > 0$ **using** ‹$\varepsilon_0 < \varepsilon\ /\ 2$› *elim*(*4*) *that*(*2*) **by** *linarith*
moreover have $c\ n > 0$ **by** (*simp add*: *elim*(*4*))
ultimately have $(c\ (n+1) - \varepsilon)\ /\ c\ n < (c\ (n+1) - \varepsilon_1)\ /\ (c\ n - \varepsilon_0)$
by (*auto simp*: *field-simps*)
also have ... $\leq (c\ (n+1) - \varepsilon_1)\ /\ (c\ n - \varepsilon_0) * (a\ (n+1)\ /\ a\ n)$
proof −
have $(c\ (n+1) - \varepsilon_1)\ /\ (c\ n - \varepsilon_0) > 0$
using *gt0* ‹$\varepsilon_1 < \varepsilon\ /\ 2$› *elim*(*4*) *that*(*2*) **by** *force*
moreover have $(a\ (n+1)\ /\ a\ n) \geq 1$
using *a-pos elim*(*5*) **by** *auto*
ultimately show *?thesis* **by** (*metis mult-cancel-left1 mult-le-cancel-left-pos*)
qed
also have ... $= (B * b\ (n+1))\ /\ (B * b\ n)$
proof −
have $B * b\ n = c\ n * a\ n - c\ (n + 1)$
using *elim* **by** *auto*
also have ... $= a\ n * (c\ n - \varepsilon_0)$
using *a-pos*[*rule-format*,*of n*] **unfolding** ε_0*-def* **by** (*auto simp*:*field-simps*)
finally have $B * b\ n = a\ n * (c\ n - \varepsilon_0)$ **.**
moreover have $B * b\ (n+1) = a\ (n+1) * (c\ (n+1) - \varepsilon_1)$
unfolding ε_1*-def*
using *a-pos*[*rule-format*,*of n+1*]
apply (*subst* ‹$B * b\ (n + 1) = c\ (n + 1) * a\ (n + 1) - c\ (n + 2)$›)
by (*auto simp*:*field-simps*)
ultimately show *?thesis* **by** (*simp add*: *mult.commute*)
qed
also have ... $= b\ (n+1)\ /\ b\ n$
using ‹$B>0$› **by** *auto*
finally show *?case* **.**
qed
qed

have *eq-2-11*:$\exists_F$ *n in sequentially.* $b\ (n+1) > b\ n + (1 - \varepsilon)$^$2 * a\ n\ /\ B$
when $\varepsilon>0$ $\varepsilon<1$ $\neg$ ($\forall_F$ *n in sequentially.* $c\ (n+1) \leq c\ n$) **for** ε::*real*

proof −
have $\exists_F$ *x in sequentially. c x < c (Suc x)* **using** *that(3)*
by (*simp add:not-eventually frequently-elim1*)
moreover have $\forall_F$ *x in sequentially.* $|c\ (Suc\ x)\ /\ a\ x| < \varepsilon$
using *ca-vanish[unfolded tendsto-iff,rule-format, of* ε*]* ‹$\varepsilon>0$› **by** *auto*
moreover have $\forall_F$ *n in sequentially. c n > 0* $\wedge$ *c (n+1) > 0*
proof −
have $\forall_F$ *n in sequentially. 0 < c (Suc n)*
using *c-pos* **by** (*subst eventually-sequentially-Suc*) *simp*
then show *?thesis* **using** *c-pos* **by** *eventually-elim auto*
qed
ultimately show *?thesis* **using** *ab-event abc-event bc-epsilon[OF* ‹$\varepsilon>0$› ‹$\varepsilon<1$›*]*

proof (*elim frequently-rev-mp,eventually-elim*)
case (*elim n*)
then have *c (n+1) / a n <* ε
using *a-pos[rule-format,of n]* **by** *auto*
also have ... $\le \varepsilon *$ *c n* **using** *elim(7) that(1)* **by** *auto*
finally have *c (n+1) / a n <* $\varepsilon *$ *c n* **.**
then have *c (n+1) / c n <* $\varepsilon *$ *a n*
using *a-pos[rule-format,of n] elim* **by** (*auto simp:field-simps*)
then have $(1 - \varepsilon) *$ *a n < a n − c (n+1) / c n*
by (*auto simp:algebra-simps*)
then have $(1 - \varepsilon)$^$2 *$ *a n / B <* $(1 - \varepsilon) *$ *(a n − c (n+1) / c n) / B*
apply (*subst (asm) mult-less-cancel-right-pos[symmetric, of* $(1-\varepsilon)/B$*]*)
using ‹$\varepsilon<1$› ‹$B>0$› **by** (*auto simp: divide-simps power2-eq-square mult-less-cancel-right-pos*)
then have *b n +* $(1 - \varepsilon)$^$2 *$ *a n / B < b n +* $(1 - \varepsilon) *$ *(a n − c (n+1) /*
c n) / B
using ‹$B>0$› **by** *auto*
also have ... = *b n +* $(1 - \varepsilon) *$ *((c n *a n − c (n+1)) / c n) / B*
using *elim* **by** (*auto simp:field-simps*)
also have ... = *b n +* $(1 - \varepsilon) *$ *(b n / c n)*
proof −
have *B * b n = c n * a n − c (n + 1)* **using** *elim* **by** *auto*
from *this[symmetric]* **show** *?thesis*
using ‹$B>0$› **by** *simp*
qed
also have ... = $(1+(1-\varepsilon)/c\ n) * b\ n$
by (*auto simp:algebra-simps*)
also have ... = $((c\ n+1-\varepsilon)/c\ n) * b\ n$
using *elim* **by** (*auto simp:divide-simps*)
also have ... $\le ((c\ (n+1) -\varepsilon)/c\ n) * b\ n$
proof −
define *cp* **where** *cp = c n+1*
have *c (n+1)* $\ge$ *cp* **unfolding** *cp-def* **using** ‹*c n < c (Suc n)*› **by** *auto*
moreover have *c n>0 b n>0* **using** *elim* **by** *auto*
ultimately show *?thesis*
apply (*fold cp-def*)
by (*auto simp:divide-simps*)

```
      qed
      also have ... < b (n+1)
        using elim by (auto simp:divide-simps)
      finally show ?case .
    qed
  qed

  have ∀F n in sequentially. c (n+1) ≤ c n
  proof (rule ccontr)
    assume ¬ (∀F n in sequentially. c (n + 1) ≤ c n)
    from eq-2-11[OF - - this,of 1/2]
    have ∃F n in sequentially. b (n+1) > b n + 1/4 * a n / B
      by (auto simp:algebra-simps power2-eq-square)
    then have *:∃F n in sequentially. (b (n+1) − b n) / a n > 1 / (B * 4)
      apply (elim frequently-elim1)
      subgoal for n
        using a-pos[rule-format,of n] by (auto simp:field-simps)
      done
    define f where f = (λn. (b (n+1) − b n) / a n)
    have f ⟶ lim f
      using convergent-LIMSEQ-iff ba-lim-exist unfolding f-def by auto
    from this[unfolded tendsto-iff,rule-format, of 1 / (B*4)]
    have ∀F x in sequentially. |f x − lim f| < 1 / (B * 4)
      using ‹B>0› by (auto simp:dist-real-def)
    moreover have ∃F n in sequentially. f n > 1 / (B * 4)
      using * unfolding f-def by auto
    ultimately have ∃F n in sequentially. f n > 1 / (B * 4) ∧ |f n − lim f| < 1
/ (B * 4)
      by (auto elim:frequently-eventually-frequently[rotated])
    from frequently-ex[OF this]
    obtain n where f n > 1 / (B * 4) |f n − lim f| < 1 / (B * 4)
      by auto
    moreover have lim f ≤ 0 using ba-lim-leq unfolding f-def by auto
    ultimately show False by linarith
  qed
  then obtain N where N-dec:∀ n≥N. c (n+1) ≤ c n by (meson eventually-at-top-linorder)
  define max-c where max-c = (MAX n ∈ {..N}. c n)
  have max-c:c n ≤ max-c for n
  proof (cases n≤N)
    case True
    then show ?thesis unfolding max-c-def by simp
  next
    case False
    then have n≥N by auto
    then have c n≤c N
    proof (induct rule:nat-induct-at-least)
      case base
      then show ?case by simp
    next
```

case (*Suc n*)
then have *c* (*n*+*1*) ≤ *c n* **using** *N-dec* **by** *auto*
then show *?case* **using** ‹*c n* ≤ *c N*› **by** *auto*
qed
moreover have *c N* ≤ *max-c* **unfolding** *max-c-def* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed
have *max-c* > *0*
proof −
obtain *N* **where** ∀ *n*≥*N*. *0* < *c n*
using *c-pos*[*unfolded eventually-at-top-linorder*] **by** *auto*
then have *c N* > *0* **by** *auto*
then show *?thesis* **using** *max-c*[*of N*] **by** *simp*
qed
have *ba-limsup-bound*:*1*/(*B*∗(*B*+*1*)) ≤ *limsup* (λ*n*. *b n*/*a n*)
limsup (λ*n*. *b n*/*a n*) ≤ *max-c* / *B* + *1* / (*B*+*1*)
proof −
define *f* **where** *f* = (λ*n*. *b n*/*a n*)
from *tendsto-mult-right-zero*[*OF bac-close*,*of 1*/*B*]
have (λ*n*. *f n* − *c n* / *B*) ⟶ *0*
unfolding *f-def* **using** ‹*B*>*0*› **by** (*auto simp*:*algebra-simps*)
from *this*[*unfolded tendsto-iff*,*rule-format*,*of 1*/(*B*+*1*)]
have ∀$_F$ *x in sequentially*. |*f x* − *c x* / *B*| < *1* / (*B*+*1*)
using ‹*B*>*0*› **by** *auto*
then have ∗:∀$_F$ *n in sequentially*. *1*/(*B*∗(*B*+*1*)) ≤ *ereal* (*f n*) ∧ *ereal* (*f n*) ≤ *max-c* / *B* + *1* / (*B*+*1*)
using *c-pos*
proof *eventually-elim*
case (*elim n*)
then have *f n* − *c n* / *B* < *1* / (*B*+*1*) **by** *auto*
then have *f n* < *c n* / *B* + *1* / (*B*+*1*) **by** *simp*
also have ... ≤ *max-c* / *B* + *1* / (*B*+*1*)
using *max-c*[*of n*] **using** ‹*B*>*0*› **by** (*auto simp*:*divide-simps*)
finally have ∗:*f n* < *max-c* / *B* + *1* / (*B*+*1*) **.**

have *1*/(*B*∗(*B*+*1*)) = *1*/*B* − *1* / (*B*+*1*)
using ‹*B*>*0*› **by** (*auto simp*:*divide-simps*)
also have ... ≤ *c n*/*B* − *1* / (*B*+*1*)
using ‹*0* < *c n*› ‹*B*>*0*› **by** (*auto*,*auto simp*:*divide-simps*)
also have ... < *f n* **using** *elim* **by** *auto*
finally have *1*/(*B*∗(*B*+*1*)) < *f n* **.**
with ∗ **show** *?case* **by** *simp*
qed
show *limsup f* ≤ *max-c* / *B* + *1* / (*B*+*1*)
apply (*rule Limsup-bounded*)
using ∗ **by** (*auto elim*:*eventually-mono*)
have *1*/(*B*∗(*B*+*1*)) ≤ *liminf f*
apply (*rule Liminf-bounded*)
using ∗ **by** (*auto elim*:*eventually-mono*)

also have *liminf f* ≤ *limsup f* **by** (*simp add*: *Liminf-le-Limsup*)
finally show *1*/(*B*∗(*B*+*1*)) ≤ *limsup f* **.**
qed

have *0* < *inverse* (*ereal* (*max-c* / *B* + *1* / (*B*+*1*)))
using ‹*max-c* > *0*› ‹*B*>*0*›
by (*simp add*: *pos-add-strict*)
also have ... ≤ *inverse* (*limsup* (λ*n*. *b n*/*a n*))
proof (*rule ereal-inverse-antimono*[*OF* - *ba-limsup-bound*(*2*)])
have *0*<*1*/(*B*∗(*B*+*1*)) **using** ‹*B*>*0*› **by** *auto*
also have ... ≤ *limsup* (λ*n*. *b n*/*a n*) **using** *ba-limsup-bound*(*1*) **.**
finally show *0*≤*limsup* (λ*n*. *b n*/*a n*) **using** *zero-ereal-def* **by** *auto*
qed
also have ... = *liminf* (λ*n*. *inverse* (*ereal* (*b n*/*a n*)))
apply (*subst Liminf-inverse-ereal*[*symmetric*])
using *a-pos ab-event* **by** (*auto elim*!:*eventually-mono simp*:*divide-simps*)
also have ... = *liminf* (λ*n*. (*a n*/*b n*))
apply (*rule Liminf-eq*)
using *a-pos ab-event*
apply (*auto elim*!:*eventually-mono*)
by (*metis less-int-code*(*1*))
finally have *liminf* (λ*n*. (*a n*/*b n*)) > *0* **.**
then show *False* **using** ‹*liminf* (λ*n*. *a n* / *b n*) = *0*› **by** *simp*
qed

end

3 Some auxiliary results on the prime numbers.

lemma *nth-prime-nonzero*[*simp*]:*nth-prime n* ≠ *0*
by (*simp add*: *prime-gt-0-nat prime-nth-prime*)

lemma *nth-prime-gt-zero*[*simp*]:*nth-prime n* >*0*
by (*simp add*: *prime-gt-0-nat prime-nth-prime*)

lemma *ratio-of-consecutive-primes*:
(λ*n*. *nth-prime* (*n*+*1*)/*nth-prime n*) ⟶*1*
proof −
define *f* **where** *f*=(λ*x*. *real* (*nth-prime* (*Suc x*)) /*real* (*nth-prime x*))
define *g* **where** *g*=(λ*x*. (*real x* ∗ *ln* (*real x*))
/ (*real* (*Suc x*) ∗ *ln* (*real* (*Suc x*))))
have *p-n*:(λ*x*. *real* (*nth-prime x*) / (*real x* ∗ *ln* (*real x*))) ⟶ *1*
using *nth-prime-asymptotics*[*unfolded asymp-equiv-def*,*simplified*] **.**
moreover have *p-sn*:(λ*n*. *real* (*nth-prime* (*Suc n*))
/ (*real* (*Suc n*) ∗ *ln* (*real* (*Suc n*)))) ⟶ *1*
using *nth-prime-asymptotics*[*unfolded asymp-equiv-def*,*simplified*
,*THEN LIMSEQ-Suc*] **.**
ultimately have (λ*x*. *f x* ∗ *g x*) ⟶ *1*
using *tendsto-divide*[*OF p-sn p-n*]

```
    unfolding f-def g-def by (auto simp:algebra-simps)
  moreover have g ⟶ 1 unfolding g-def
    by real-asymp
  ultimately have (λx. if g x = 0 then 0 else f x) ⟶ 1
    apply (drule-tac tendsto-divide[OF - ‹g ⟶ 1›])
    by auto
  then have f ⟶ 1
  proof (elim filterlim-mono-eventually)
    have ∀F x in sequentially. (if g (x+3) = 0 then 0
              else f (x+3)) = f (x+3)
      unfolding g-def by auto
    then show ∀F x in sequentially. (if g x = 0 then 0 else f x) = f x
      apply (subst (asm) eventually-sequentially-seg)
      by simp
  qed auto
  then show ?thesis unfolding f-def by auto
qed

lemma nth-prime-double-sqrt-less:
  assumes ε > 0
  shows ∀F n in sequentially. (nth-prime (2*n) − nth-prime n)
          / sqrt (nth-prime n) < n powr (1/2+ε)
proof −
  define pp ll where
    pp=(λn. (nth-prime (2*n) − nth-prime n) / sqrt (nth-prime n)) and
    ll=(λx::nat. x * ln x)
  have pp-pos:pp (n+1) > 0 for n
    unfolding pp-def by simp

  have (λx. nth-prime (2 * x)) ∼[sequentially] (λx. (2 * x) * ln (2 * x))
    using nth-prime-asymptotics[THEN asymp-equiv-compose
        ,of (*) 2 sequentially,unfolded comp-def]
    using mult-nat-left-at-top pos2 by blast
  also have ... ∼[sequentially] (λx. 2 *x * ln x)
    by real-asymp
  finally have (λx. nth-prime (2 * x)) ∼[sequentially] (λx. 2 *x * ln x) .
  from this[unfolded asymp-equiv-def, THEN tendsto-mult-left,of 2]
  have (λx. nth-prime (2 * x) / (x * ln x)) ⟶ 2
    unfolding asymp-equiv-def by auto
  moreover have *:(λx. nth-prime x / (x * ln x)) ⟶ 1
    using nth-prime-asymptotics unfolding asymp-equiv-def by auto
  ultimately
  have (λx. (nth-prime (2 * x) − nth-prime x) / ll x) ⟶ 1
    unfolding ll-def
    apply −
    apply (drule (1) tendsto-diff)
    apply (subst of-nat-diff,simp)
    by (subst diff-divide-distrib,simp)
  moreover have (λx. sqrt (nth-prime x) / sqrt (ll x)) ⟶ 1
```

```
    unfolding ll-def
    using tendsto-real-sqrt[OF *]
    by (auto simp: real-sqrt-divide)
  ultimately have (λx. pp x * (sqrt (ll x) / (ll x))) ⟶ 1
    apply -
    apply (drule (1) tendsto-divide,simp)
    by (auto simp:field-simps of-nat-diff pp-def)
  moreover have ∀F x in sequentially. sqrt (ll x) / ll x = 1/sqrt (ll x)
    apply (subst eventually-sequentially-Suc[symmetric])
    by (auto intro!:eventuallyI simp:ll-def divide-simps)
  ultimately have (λx. pp x / sqrt (ll x)) ⟶ 1
    apply (elim filterlim-mono-eventually)
   by (auto elim!:eventually-mono) (metis mult.right-neutral times-divide-eq-right)
  moreover have (λx. sqrt (ll x) / x powr (1/2+ε)) ⟶ 0
    unfolding ll-def using ‹ε>0› by real-asymp
  ultimately have (λx. pp x / x powr (1/2+ε) *
                    (sqrt (ll x) / sqrt (ll x))) ⟶ 0
    apply -
    apply (drule (1) tendsto-mult)
    by (auto elim:filterlim-mono-eventually)
  moreover have ∀F x in sequentially. sqrt (ll x) / sqrt (ll x) = 1
    apply (subst eventually-sequentially-Suc[symmetric])
    by (auto intro!:eventuallyI simp:ll-def )
  ultimately have (λx. pp x / x powr (1/2+ε)) ⟶ 0
    apply (elim filterlim-mono-eventually)
    by (auto elim:eventually-mono)
  from tendstoD[OF this, of 1,simplified]
  show ∀F x in sequentially. pp x < x powr (1 / 2 + ε)
    apply (elim eventually-mono-sequentially[of - 1])
    using pp-pos by auto
qed
```

4 Theorem 3.1

Theorem 3.1 is an application of Theorem 2.1 with the sequences considered involving the prime numbers.

```
theorem theorem-3-10-Erdos-Straus:
  fixes a::nat ⇒ int
  assumes a-pos:∀ n. a n >0 and mono a
    and nth-1:(λn. nth-prime n / (a n)^2) ⟶ 0
    and nth-2:liminf (λn. a n / nth-prime n) = 0
  shows (∑n. (nth-prime n / (∏i ≤ n. a i))) ∉ ℚ
proof
  assume asm:(∑n. (nth-prime n / (∏i ≤ n. a i))) ∈ ℚ

  have a2-omega:(λn. (a n)^2) ∈ ω(λx. x * ln x)
  proof -
    have (λn. real (nth-prime n)) ∈ o(λn. real-of-int ((a n)²))
```

apply (*rule smalloI-tendsto*[*OF nth-1*])
using *a-pos* **by** (*metis* (*mono-tags*, *lifting*) *less-int-code*(*1*)
not-eventuallyD of-int-0-eq-iff zero-eq-power2)
moreover have $(\lambda x.\ real\ (nth\text{-}prime\ x)) \in \Omega(\lambda x.\ real\ x * ln\ (real\ x))$
using *nth-prime-bigtheta*
by *blast*
ultimately show *?thesis*
using *landau-omega.small-big-trans smallo-imp-smallomega* **by** *blast*
qed

have *a-gt-1*:$\forall_F$ *n in sequentially.* $1 < a\ n$
proof −
have $\forall_F$ *x in sequentially.* $|x * ln\ x| \le (a\ x)^2$
using *a2-omega*[*unfolded smallomega-def*,*simplified*,*rule-format*,*of 1*]
by *auto*
then have $\forall_F$ *x in sequentially.* $|(x+3) * ln\ (x+3)| \le (a\ (x+3))^2$
apply (*subst* (*asm*) *eventually-sequentially-seg*[*symmetric*, *of - 3*])
by *simp*
then have $\forall_F$ *n in sequentially.* $1 < a\ (\ n+3)$
proof (*elim eventually-mono*)
fix *x*
assume $|real\ (x + 3) * ln\ (real\ (x + 3))| \le real\text{-}of\text{-}int\ ((a\ (x + 3))^2)$
moreover have $|real\ (x + 3) * ln\ (real\ (x + 3))| > 3$
proof −
have $ln\ (real\ (x + 3)) > 1$
using *ln3-gt-1 ln-gt-1* **by** *force*
moreover have $real(x+3) \ge 3$ **by** *simp*
ultimately have $(x+3)*ln\ (real\ (x + 3)) > 3*1$
by (*smt* (*verit*, *best*) *mult-less-cancel-left1*)
then show *?thesis* **by** *auto*
qed
ultimately have $(a\ (x + 3))^2 > 3$
by *linarith*
then show $1 < a\ (x + 3)$
by (*smt* (*verit*) *assms*(*1*) *one-power2*)
qed
then show *?thesis*
using *eventually-sequentially-seg*[*symmetric*, *of - 3*]
by *blast*
qed

obtain *B*::*int* **and** *c* **where**
$B>0$ **and** *Bc-large*:$\forall_F$ *n in sequentially.* $B * nth\text{-}prime\ n$
$= c\ n * a\ n - c\ (n + 1) \wedge |c\ (n + 1)| < a\ n\ /\ 2$
and *ca-vanish*: $(\lambda n.\ c\ (Suc\ n)\ /\ real\text{-}of\text{-}int\ (a\ n)) \longrightarrow 0$
proof −
note *a-gt-1*
moreover have $(\lambda n.\ real\text{-}of\text{-}int\ |int\ (nth\text{-}prime\ n)|$
$/\ real\text{-}of\text{-}int\ (a\ (n - 1) * a\ n)) \longrightarrow 0$

```
    proof -
      define f where f=(λn. nth-prime (n+1) / (a n * a (n+1)))
      define g where g=(λn. 2*nth-prime n / (a n)^2)
      have ∀F x in sequentially. norm (f x) ≤ g x
      proof -
        have ∀F n in sequentially. nth-prime (n+1) < 2*nth-prime n
          using ratio-of-consecutive-primes[unfolded tendsto-iff
              ,rule-format,of 1,simplified]
          apply (elim eventually-mono)
          by (auto simp :divide-simps dist-norm)
        moreover have ∀F n in sequentially. real-of-int (a n * a (n+1))
                          ≥ (a n)^2
          apply (rule eventuallyI)
          using ‹mono a› by (auto simp:power2-eq-square a-pos incseq-SucD)
        ultimately show ?thesis unfolding f-def g-def
          apply eventually-elim
          apply (subst norm-divide)
          apply (rule-tac linordered-field-class.frac-le)
          using a-pos[rule-format, THEN order.strict-implies-not-eq ]
          by auto
      qed
      moreover have g ⟶ 0
        using nth-1[THEN tendsto-mult-right-zero,of 2] unfolding g-def
        by auto
      ultimately have f ⟶ 0
        using Lim-null-comparison[of f g sequentially]
        by auto
      then show ?thesis
        unfolding f-def
        by (rule-tac LIMSEQ-imp-Suc) auto
    qed
    moreover have (∑ n. real-of-int (int (nth-prime n))
                  / real-of-int (prod a {..n})) ∈ ℚ
      using asm by simp
    ultimately have ∃ B>0. ∃ c. (∀F n in sequentially.
              B * int (nth-prime n) = c n * a n - c (n + 1) ∧
              real-of-int |c (n + 1)| < real-of-int (a n) / 2) ∧
          (λn. real-of-int (c (Suc n)) / real-of-int (a n)) ⟶ 0
      using ab-rationality-imp[OF a-pos,of nth-prime] by fast
    then show thesis
      apply clarify
      apply (rule-tac c=c and B=B in that)
      by auto
  qed

  have bac-close:(λn. B * nth-prime n / a n - c n) ⟶ 0
  proof -
    have ∀F n in sequentially. B * nth-prime n - c n * a n + c (n + 1) = 0
      using Bc-large by (auto elim!:eventually-mono)
```

```
    then have ∀F n in sequentially. (B * nth-prime n − c n * a n + c (n+1)) /
a n = 0
      by eventually-elim auto
    then have ∀F n in sequentially. B * nth-prime n / a n − c n + c (n + 1) /
a n = 0
      apply eventually-elim
      using a-pos by (auto simp:divide-simps) (metis less-irrefl)
    then have (λn. B * nth-prime n / a n − c n + c (n + 1) / a n) ⟶ 0
      by (simp add: eventually-mono tendsto-iff)
    from tendsto-diff[OF this ca-vanish]
    show ?thesis by auto
  qed

  have c-pos:∀F n in sequentially. c n > 0
  proof −
    from bac-close have *:∀F n in sequentially. c n ≥ 0
      apply (elim tendsto-of-int-diff-0)
      using a-gt-1 apply (eventually-elim)
      using ‹B>0› by auto
    show ?thesis
    proof (rule ccontr)
      assume ¬ (∀F n in sequentially. c n > 0)
      moreover have ∀F n in sequentially. c (Suc n) ≥ 0 ∧ c n≥0
        using * eventually-sequentially-Suc[of λn. c n≥0]
        by (metis (mono-tags, lifting) eventually-at-top-linorder le-Suc-eq)
      ultimately have ∃F n in sequentially. c n = 0 ∧ c (Suc n) ≥ 0
        using eventually-elim2 frequently-def by fastforce
      moreover have ∀F n in sequentially. nth-prime n > 0
                  ∧ B * nth-prime n = c n * a n − c (n + 1)
        using Bc-large by eventually-elim auto
      ultimately have ∃F n in sequentially. c n = 0 ∧ c (Suc n) ≥ 0
          ∧ B * nth-prime n = c n * a n − c (n + 1)
        using frequently-eventually-frequently by fastforce
      from frequently-ex[OF this]
      obtain n where c n = 0 c (Suc n) ≥ 0
        B * nth-prime n = c n * a n − c (n + 1)
        by auto
      then have B * nth-prime n ≤ 0 by auto
      then show False using ‹B > 0›
        by (simp add: mult-le-0-iff)
    qed
  qed

  have B-nth-prime:∀F n in sequentially. nth-prime n > B
  proof −
    have ∀F x in sequentially. B+1 ≤ nth-prime x
      using nth-prime-at-top[unfolded filterlim-at-top-ge[where c=nat B+1]
          ,rule-format,of nat B + 1,simplified]
```

```
      apply (elim eventually-mono)
      using ‹B>0› by auto
    then show ?thesis
      by (auto elim: eventually-mono)
  qed

  have bc-epsilon:∀F n in sequentially. nth-prime (n+1)
          / nth-prime n >  (c (n+1) − ε) / c n when ε>0 ε<1 for ε::real
  proof −
    have ∀F x in sequentially. |c (Suc x) / a x| < ε / 2
      using ca-vanish[unfolded tendsto-iff,rule-format, of ε/2] ‹ε>0› by auto
    moreover then have ∀F x in sequentially. |c (x+2) / a (x+1)| < ε / 2
      apply (subst (asm) eventually-sequentially-Suc[symmetric])
      by simp
     moreover have ∀F n in sequentially. B ∗ nth-prime (n+1) = c (n+1) ∗ a
(n+1) − c (n + 2)
      using Bc-large
      apply (subst (asm) eventually-sequentially-Suc[symmetric])
      by (auto elim:eventually-mono)
    moreover have ∀F n in sequentially. c n > 0 ∧ c (n+1) > 0 ∧ c (n+2) > 0
    proof −
      have ∀F n in sequentially. 0 < c (Suc n)
        using c-pos by (subst eventually-sequentially-Suc) simp
      moreover then have ∀F n in sequentially. 0 < c (Suc (Suc n))
        using c-pos by (subst eventually-sequentially-Suc) simp
      ultimately show ?thesis using c-pos by eventually-elim auto
    qed
    ultimately show ?thesis using Bc-large
    proof eventually-elim
      case (elim n)
      define ε0 ε1 where ε0 = c (n+1) / a n and ε1 = c (n+2) / a (n+1)
      have ε0 > 0 ε1 > 0 ε0 < ε/2 ε1 < ε/2
        using a-pos elim ‹mono a›
        by (auto simp: ε0-def ε1-def abs-of-pos)
      have (ε − ε1) ∗ c n > 0
        using ‹ε1 > 0› ‹ε1 < ε/2› ‹ε>0› elim by auto
      moreover have A: ε0 ∗ (c (n+1) − ε) > 0
        using ‹ε0 > 0› elim(4) that(2) by force
      ultimately have (ε − ε1) ∗ c n + ε0 ∗ (c (n+1) − ε) > 0 by auto
       moreover have B: c n − ε0 > 0 using ‹ε0 < ε / 2› elim(4) that(2) by
linarith
      moreover have c n > 0 by (simp add: elim(4))
      ultimately have (c (n+1) − ε) / c n < (c (n+1) − ε1) / (c n −  ε0)
        by (auto simp:field-simps)
      also have ... ≤ (c (n+1) − ε1) / (c n −  ε0) ∗ (a (n+1) / a n)
      proof −
        have (c (n+1) − ε1) / (c n −  ε0) > 0
          using A ‹0 < ε0› B ‹ε1 < ε / 2› divide-pos-pos that(1) by force
        moreover have (a (n+1) / a n) ≥ 1
```

using *a-pos* ‹*mono a*› **by** (*simp add*: *mono-def*)
ultimately show *?thesis* **by** (*metis mult-cancel-left1 mult-le-cancel-left-pos*)
qed
also have ... = $(B * \textit{nth-prime}\ (n+1))\ /\ (B * \textit{nth-prime}\ n)$
proof −
have $B * \textit{nth-prime}\ n = c\ n * a\ n - c\ (n + 1)$
using *elim* **by** *auto*
also have ... = $a\ n * (c\ n - \varepsilon_0)$
using *a-pos*[*rule-format*,*of n*] **unfolding** ε_0*-def* **by** (*auto simp*:*field-simps*)
finally have $B * \textit{nth-prime}\ n = a\ n * (c\ n - \varepsilon_0)$ **.**
moreover have $B * \textit{nth-prime}\ (n+1) = a\ (n+1) * (c\ (n+1) - \varepsilon_1)$
unfolding ε_1*-def*
using *a-pos*[*rule-format*,*of n+1*]
apply (*subst* ‹$B * \textit{nth-prime}\ (n + 1) = c\ (n + 1) * a\ (n + 1) - c\ (n +$
$2)$›)
by (*auto simp*:*field-simps*)
ultimately show *?thesis* **by** (*simp add*: *mult.commute*)
qed
also have ... = *nth-prime* (*n+1*) / *nth-prime n*
using ‹$B>0$› **by** *auto*
finally show *?case* **.**
qed
qed

have *c-ubound*:$\forall x.\ \exists n.\ c\ n > x$
proof (*rule ccontr*)
assume $\neg\ (\forall x.\ \exists n.\ x < c\ n)$
then obtain *ub* **where** $\forall n.\ c\ n \leq ub\ ub > 0$
by (*meson dual-order.trans int-one-le-iff-zero-less le-cases not-le*)
define *pa* **where** $pa = (\lambda n.\ \textit{nth-prime}\ n\ /\ a\ n)$
have *pa-pos*:$\bigwedge n.\ pa\ n > 0$ **unfolding** *pa-def* **by** (*simp add*: *a-pos*)
have *liminf* $(\lambda n.\ 1\ /\ pa\ n) = 0$
using *nth-2* **unfolding** *pa-def* **by** *auto*
then have $(\exists y<\textit{ereal}\ (\textit{real-of-int}\ B\ /\ \textit{real-of-int}\ (ub + 1)).$
$\exists_F\ x\ \textit{in sequentially}.\ \textit{ereal}\ (1\ /\ pa\ x) \leq y)$
apply (*subst less-Liminf-iff*[*symmetric*])
using ‹$0 < B$› ‹$0 < ub$› **by** *auto*
then have $\exists_F\ x\ \textit{in sequentially}.\ 1\ /\ pa\ x < B/(ub+1)$
by (*meson frequently-mono le-less-trans less-ereal.simps*(*1*))
then have $\exists_F\ x\ \textit{in sequentially}.\ B{*}pa\ x > (ub+1)$
apply (*elim frequently-elim1*)
by (*metis* ‹$0 < ub$› *mult.left-neutral of-int-0-less-iff pa-pos pos-divide-less-eq pos-less-divide-eq times-divide-eq-left zless-add1-eq*)
moreover have $\forall_F\ x\ \textit{in sequentially}.\ c\ x \leq ub$
using ‹$\forall n.\ c\ n \leq ub$› **by** *simp*
ultimately have $\exists_F\ x\ \textit{in sequentially}.\ B{*}pa\ x - c\ x > 1$
by (*elim frequently-rev-mp eventually-mono*) *linarith*
moreover have $(\lambda n.\ B * pa\ n - c\ n) \longrightarrow 0$

unfolding *pa-def* **using** *bac-close* **by** *auto*
from *tendstoD*[*OF this*,*of 1*]
have $\forall_F$ *n in sequentially*. |*B* ∗ *pa n* − *c n*| < *1*
by *auto*
ultimately have $\exists_F$ *x in sequentially*. *B*∗*pa x* − *c x* > *1* ∧ |*B* ∗ *pa x* − *c x*| < *1*
using *frequently-eventually-frequently* **by** *blast*
then show *False*
by (*simp add*: *frequently-def*)
qed

have *eq-2-11*:$\forall_F$ *n in sequentially*. *c* (*n*+*1*)>*c n* ⟶
nth-prime (*n*+*1*) > *nth-prime n* + (*1* − ε)ˆ*2* ∗ *a n* / *B*
when ε>*0* ε<*1* **for** ε::*real*
proof −
have $\forall_F$ *x in sequentially*. |*c* (*Suc x*) / *a x*| < ε
using *ca-vanish*[*unfolded tendsto-iff*,*rule-format*, *of* ε] ‹ε>*0*› **by** *auto*
moreover have $\forall_F$ *n in sequentially*. *c n* > *0* ∧ *c* (*n*+*1*) > *0*
proof −
have $\forall_F$ *n in sequentially*. *0* < *c* (*Suc n*)
using *c-pos* **by** (*subst eventually-sequentially-Suc*) *simp*
then show *?thesis* **using** *c-pos* **by** *eventually-elim auto*
qed
ultimately show *?thesis* **using** *Bc-large bc-epsilon*[*OF* ‹ε>*0*› ‹ε<*1*›]
proof (*eventually-elim*, *rule-tac impI*)
case (*elim n*)
assume *c n* < *c* (*n* + *1*)
have *c* (*n*+*1*) / *a n* < ε
using *a-pos*[*rule-format*,*of n*] **using** *elim*(*1*,*2*) **by** *auto*
also have ... ≤ ε ∗ *c n* **using** *elim*(*2*) *that*(*1*) **by** *auto*
finally have *c* (*n*+*1*) / *a n* < ε ∗ *c n* **.**
then have *c* (*n*+*1*) / *c n* < ε ∗ *a n*
using *a-pos*[*rule-format*,*of n*] *elim* **by** (*auto simp*:*field-simps*)
then have (*1* − ε) ∗ *a n* < *a n* − *c* (*n*+*1*) / *c n*
by (*auto simp*:*algebra-simps*)
then have (*1* − ε)ˆ*2* ∗ *a n* / *B* < (*1* − ε) ∗ (*a n* − *c* (*n*+*1*) / *c n*) / *B*
apply (*subst* (*asm*) *mult-less-cancel-right-pos*[*symmetric*, *of* (*1*−ε)/*B*])
using ‹ε<*1*› ‹*B*>*0*› **by** (*auto simp*: *divide-simps power2-eq-square mult-less-cancel-right-pos*)
then have *nth-prime n* + (*1* − ε)ˆ*2* ∗ *a n* / *B* < *nth-prime n* + (*1* − ε) ∗ (*a n* − *c* (*n*+*1*) / *c n*) / *B*
using ‹*B*>*0*› **by** *auto*
also have ... = *nth-prime n* + (*1* − ε) ∗ ((*c n* ∗*a n* − *c* (*n*+*1*)) / *c n*) / *B*
using *elim* **by** (*auto simp*:*field-simps*)
also have ... = *nth-prime n* + (*1* − ε) ∗ (*nth-prime n* / *c n*)
proof −
have *B* ∗ *nth-prime n* = *c n* ∗ *a n* − *c* (*n* + *1*) **using** *elim* **by** *auto*
from *this*[*symmetric*] **show** *?thesis*
using ‹*B*>*0*› **by** *simp*
qed

also have ... = $(1+(1-\varepsilon)/c\ n) * nth\text{-}prime\ n$
by (*auto simp*:*algebra-simps*)
also have ... = $((c\ n+1-\varepsilon)/c\ n) * nth\text{-}prime\ n$
using *elim* **by** (*auto simp*:*divide-simps*)
also have ... $\leq ((c\ (n+1)\ -\varepsilon)/c\ n) * nth\text{-}prime\ n$
proof −
define *cp* **where** $cp = c\ n+1$
have $c\ (n+1) \geq cp$ **unfolding** *cp-def* **using** ‹$c\ n < c\ (n + 1)$› **by** *auto*
moreover have $c\ n>0$ $nth\text{-}prime\ n>0$ **using** *elim* **by** *auto*
ultimately show *?thesis*
apply (*fold cp-def*)
by (*auto simp*:*divide-simps*)
qed
also have ... $< nth\text{-}prime\ (n+1)$
using *elim* **by** (*auto simp*:*divide-simps*)
finally show $real\ (nth\text{-}prime\ n) + (1 - \varepsilon)^2 * real\text{-}of\text{-}int\ (a\ n)$
$/\ real\text{-}of\text{-}int\ B < real\ (nth\text{-}prime\ (n + 1))$ **.**
qed
qed

have *c-neq-large*:$\forall_F\ n$ *in sequentially*. $c\ (n+1) \neq c\ n$
proof (*rule ccontr*)
assume $\neg\ (\forall_F\ n$ *in sequentially*. $c\ (n + 1) \neq c\ n)$
then have *that*:$\exists_F\ n$ *in sequentially*. $c\ (n + 1) = c\ n$
unfolding *frequently-def* **.**
have $\forall_F\ x$ *in sequentially*. $(B * int\ (nth\text{-}prime\ x) = c\ x * a\ x - c\ (x + 1)$
$\wedge\ |real\text{-}of\text{-}int\ (c\ (x + 1))| < real\text{-}of\text{-}int\ (a\ x)\ /\ 2) \wedge 0 < c\ x \wedge B < int$
$(nth\text{-}prime\ x)$
$\wedge\ (c\ (x+1)>c\ x \longrightarrow nth\text{-}prime\ (x+1) > nth\text{-}prime\ x + a\ x\ /\ (2* B))$
using *Bc-large c-pos B-nth-prime eq-2-11*[*of* $1-1/$ *sqrt* 2,*simplified*]
by *eventually-elim* (*auto simp*:*divide-simps*)
then have $\exists_F\ m$ *in sequentially*. $nth\text{-}prime\ (m+1) > (1+1/(2*B))*nth\text{-}prime$
m
proof (*elim frequently-eventually-at-top*[*OF that*, *THEN frequently-at-top-elim*])
fix *n*
assume $c\ (n + 1) = c\ n\ \wedge$
$(\forall\ y\geq n.\ (B * int\ (nth\text{-}prime\ y) = c\ y * a\ y - c\ (y + 1) \wedge$
$|real\text{-}of\text{-}int\ (c\ (y + 1))| < real\text{-}of\text{-}int\ (a\ y)\ /\ 2) \wedge$
$0 < c\ y \wedge B < int\ (nth\text{-}prime\ y) \wedge (c\ y < c\ (y + 1) \longrightarrow$
$real\ (nth\text{-}prime\ y) + real\text{-}of\text{-}int\ (a\ y)\ /\ real\text{-}of\text{-}int\ (2 * B)$
$< real\ (nth\text{-}prime\ (y + 1))))$
then have $c\ (n + 1) = c\ n$
and *Bc-eq*:$\forall\ y\geq n.\ B * int\ (nth\text{-}prime\ y) = c\ y * a\ y - c\ (y + 1) \wedge 0 < c\ y$
$\wedge\ |real\text{-}of\text{-}int\ (c\ (y + 1))| < real\text{-}of\text{-}int\ (a\ y)\ /\ 2$
$\wedge\ B < int\ (nth\text{-}prime\ y)$
$\wedge\ (c\ y < c\ (y + 1) \longrightarrow$
$real\ (nth\text{-}prime\ y) + real\text{-}of\text{-}int\ (a\ y)\ /\ real\text{-}of\text{-}int\ (2 * B)$
$< real\ (nth\text{-}prime\ (y + 1)))$
by *auto*

```
    obtain m where n<m c m ≤ c n c n<c (m+1)
    proof –
      have ∃N. N > n ∧ c N > c n
        using c-ubound[rule-format, of MAX x∈{..n}. c x]
          by (metis Max-ge atMost-iff dual-order.trans finite-atMost finite-imageI
image-eqI
              linorder-not-le order-refl)
      then obtain N where N>n c N>c n by auto
      define A m where A={m. n<m ∧ (m+1)≤N ∧ c (m+1) > c n} and m
= Min A
      have finite A unfolding A-def
            by (metis (no-types, lifting) A-def add-leE finite-nat-set-iff-bounded-le
mem-Collect-eq)
      moreover have N−1∈A unfolding A-def
        using ‹c n < c N› ‹n < N› ‹c (n + 1) = c n› nat-less-le by force
      ultimately have m∈A
        using Min-in unfolding m-def by auto
      then have n<m  c n<c (m+1) m>0
        unfolding m-def A-def by auto
      moreover have c m ≤ c n
      proof (rule ccontr)
        assume ¬ c m ≤ c n
        then have m−1∈A
        using ‹m∈A› ‹c (n + 1) = c n› le-eq-less-or-eq less-diff-conv by (fastforce
simp: A-def)
        from Min-le[OF ‹finite A› this,folded m-def] ‹m>0› show False by auto
      qed
      ultimately show ?thesis using that[of m] by auto
    qed
    have (1 + 1 / (2 ∗ B)) ∗ nth-prime m < nth-prime m + a m / (2∗B)
    proof –
      have nth-prime m < a m
      proof –
        have B ∗ int (nth-prime m) < c m ∗ (a m − 1)
          using Bc-eq[rule-format,of m] ‹c m ≤ c n› ‹c n < c (m + 1)› ‹n < m›
           by (auto simp:algebra-simps)
        also have ... ≤ c n ∗ (a m − 1)
           by (simp add: ‹c m ≤ c n› a-pos mult-right-mono)
        finally have B ∗ int (nth-prime m) < c n ∗ (a m − 1) .
        moreover have c n≤B
        proof –
         have B: B ∗ int (nth-prime n) = c n ∗ (a n − 1) B < int (nth-prime n)
            and c-a: |real-of-int (c (n + 1))| < real-of-int (a n) / 2
        using Bc-eq[rule-format,of n] ‹c (n + 1) = c n› by (auto simp:algebra-simps)
          from this(1) have  c n dvd (B ∗ int (nth-prime n))
            by simp
          moreover have coprime (c n) (int (nth-prime n))
          proof –
            have c n < int (nth-prime n)
```

```
            proof (rule ccontr)
              assume ¬ c n < int (nth-prime n)
              then have asm:c n ≥ int (nth-prime n) by auto
              then have a n > 2 ∗ nth-prime n
                using c-a ‹c (n + 1) = c n› by auto
              then have a n −1 ≥ 2 ∗ nth-prime n
                by simp
              then have a n − 1 > 2 ∗ B
                using ‹B < int (nth-prime n)› by auto
              from mult-le-less-imp-less[OF asm this] ‹B>0›
              have int (nth-prime n) ∗ (2 ∗ B) < c n ∗ (a n − 1)
                by auto
              then show False using B
                by (smt (verit, best) ‹0 < B› mult.commute mult-right-mono)
            qed
            then have ¬ nth-prime n dvd c n
              by (simp add: Bc-eq zdvd-not-zless)
            then have coprime (int (nth-prime n)) (c n)
              by (auto intro!:prime-imp-coprime-int)
            then show ?thesis using coprime-commute by blast
          qed
          ultimately have c n dvd B
            using coprime-dvd-mult-left-iff by auto
          then show ?thesis using ‹0 < B› zdvd-imp-le by blast
        qed
        moreover have c n > 0 using Bc-eq by blast
        ultimately show ?thesis
          using ‹B>0› by (smt (verit) a-pos mult-mono)
      qed
      then show ?thesis using ‹B>0› by (auto simp:field-simps)
    qed
    also have ... < nth-prime (m+1)
      using Bc-eq[rule-format, of m] ‹n<m› ‹c m ≤ c n› ‹c n < c (m+1)›
      by linarith
    finally show ∃j>n. (1 + 1 / real-of-int (2 ∗ B)) ∗ real (nth-prime j)
                  < real (nth-prime (j + 1)) using ‹m>n› by auto
  qed
  then have ∃F m in sequentially. nth-prime (m+1)/nth-prime m > (1+1/(2∗B))
    by (auto elim:frequently-elim1 simp:field-simps)
    moreover have ∀F m in sequentially. nth-prime (m+1)/nth-prime m <
(1+1/(2∗B))
    using ratio-of-consecutive-primes[unfolded tendsto-iff,rule-format,of 1/(2∗B)]
      ‹B>0›
    unfolding dist-real-def
    by (auto elim!:eventually-mono simp:algebra-simps)
  ultimately show False by (simp add: eventually-mono frequently-def)
qed

have c-gt-half:∀F N in sequentially. card {n∈{N..<2∗N}. c n > c (n+1)} >
```

```
N / 2
 proof -
  define h where h=(λn. (nth-prime (2*n) - nth-prime n)
                          /  sqrt (nth-prime n))
  have ∀F n in sequentially. h n < n / 2
  proof -
    have ∀F n in sequentially. h n < n powr (5/6)
      using nth-prime-double-sqrt-less[of 1/3]
      unfolding h-def by auto
    moreover have ∀F n in sequentially. n powr (5/6) < (n /2)
      by real-asymp
    ultimately show ?thesis
      by eventually-elim auto
  qed
  moreover have ∀F n in sequentially. sqrt (nth-prime n) / a n < 1 / (2*B)
    using nth-1[THEN tendsto-real-sqrt,unfolded tendsto-iff
        ,rule-format,of 1/(2*B)] ‹B>0› a-pos
    by (auto simp:real-sqrt-divide abs-of-pos)
  ultimately have ∀F x in sequentially. c (x+1) ≠ c x
      ∧ sqrt (nth-prime x) / a x < 1 / (2*B)
      ∧  h x < x / 2
      ∧ (c (x+1)>c x ⟶ nth-prime (x+1) > nth-prime x +  a x / (2* B))
    using c-neq-large B-nth-prime eq-2-11[of 1−1/ sqrt 2,simplified]
    by eventually-elim (auto simp:divide-simps)
  then show ?thesis
  proof (elim eventually-at-top-mono)
    fix N assume N≥1 and N-asm:∀y≥N. c (y + 1) ≠ c y ∧
            sqrt (real (nth-prime y)) / real-of-int (a y)
            < 1 / real-of-int (2 * B) ∧  h y < y / 2 ∧
            (c y < c (y + 1) ⟶
             real (nth-prime y) + real-of-int (a y) / real-of-int (2 * B)
             < real (nth-prime (y + 1)))

    define S where S={n ∈ {N..<2 * N}. c n < c (n + 1)}
    define g where g=(λn. (nth-prime (n+1) - nth-prime n)
                          /  sqrt (nth-prime n))
    define f where f=(λn. nth-prime (n+1) - nth-prime n)
    have g-gt-1:g n>1 when n≥N c n < c (n + 1) for n
    proof -
      have nth-prime n + sqrt (nth-prime n) < nth-prime (n+1)
      proof -
        have nth-prime n + sqrt (nth-prime n) < nth-prime n + a n / (2*B)
          using N-asm[rule-format,OF ‹n≥N›] a-pos
          by (auto simp:field-simps)
        also have ... < nth-prime (n+1)
          using N-asm[rule-format,OF ‹n≥N›] ‹c n < c (n + 1)› by auto
        finally show ?thesis .
      qed
      then show ?thesis unfolding g-def
```

```
    using ‹c n < c (n + 1)› by auto
qed
have g-geq-0:g n ≥ 0 for n
  unfolding g-def  by auto

have finite S ∀x∈S. x≥N ∧ c x<c (x+1)
  unfolding S-def by auto
then have card S ≤ sum g S
proof (induct S)
  case empty
  then show ?case by auto
next
  case (insert x F)
  moreover have g x>1
  proof −
    have c x < c (x+1) x≥N using insert(4) by auto
    then show ?thesis using g-gt-1 by auto
  qed
  ultimately show ?case by simp
qed
also have ... ≤ sum g {N..<2∗N}
  apply (rule sum-mono2)
  unfolding S-def using g-geq-0 by auto
also have ... ≤ sum (λn. f n/sqrt (nth-prime N)) {N..<2∗N}
  unfolding f-def g-def by (auto intro!:sum-mono divide-left-mono)
also have ... = sum f {N..<2∗N} / sqrt (nth-prime N)
  unfolding sum-divide-distrib[symmetric] by auto
also have ... = (nth-prime (2∗N) − nth-prime N) / sqrt (nth-prime N)
proof −
  have sum f {N..<2 ∗ N} = nth-prime (2 ∗ N) − nth-prime N
  proof (induct N)
    case 0
    then show ?case by simp
  next
    case (Suc N)
    have ?case if N=0
    proof −
      have sum f {Suc N..<2 ∗ Suc N} = sum f {1}
        using that by (simp add: numeral-2-eq-2)
      also have ... = nth-prime 2 − nth-prime 1
        unfolding f-def by (simp add:numeral-2-eq-2)
      also have ... = nth-prime (2 ∗ Suc N) − nth-prime (Suc N)
        using that by auto
      finally show ?thesis .
    qed
    moreover have ?case if N≠0
    proof −
      have sum f {Suc N..<2 ∗ Suc N} = sum f {N..<2 ∗ Suc N} − f N
        apply (subst (2) sum.atLeast-Suc-lessThan)
```

```
          using that by auto
        also have ... = sum f {N..<2 ∗ N}+ f (2∗N) + f(2∗N+1) − f N
          by auto
        also have ... = nth-prime (2 ∗ Suc N) − nth-prime (Suc N)
          using Suc unfolding f-def by auto
        finally show ?thesis .
      qed
      ultimately show ?case by blast
    qed
    then show ?thesis by auto
  qed
  also have ... = h N
    unfolding h-def by auto
  also have ... < N/2
    using N-asm by auto
  finally have card S < N/2 .

  define T where T={n ∈ {N..<2 ∗ N}. c n > c (n + 1)}
  have T ∪ S = {N..<2 ∗ N} T ∩ S = {} finite T
    unfolding T-def S-def using N-asm by fastforce+

  then have card T + card S = card {N..<2 ∗ N}
    using card-Un-disjoint ‹finite S› by metis
  also have ... = N
    by simp
  finally have card T + card S = N .
  with ‹card S < N/2›
  show card T > N/2 by linarith
 qed
qed
```

Inequality (3.5) in the original paper required a slight modification:

```
have a-gt-plus:∀F n in sequentially. c n > c (n+1) ⟶a (n+1) > a n + (a n
− c(n+1) − 1) / c (n+1)
proof −
 note a-gt-1[THEN eventually-all-ge-at-top] c-pos[THEN eventually-all-ge-at-top]
  moreover have ∀F n in sequentially.
          B ∗ int (nth-prime (n+1)) = c (n+1) ∗ a (n+1) − c (n + 2)
    using Bc-large
    apply (subst (asm) eventually-sequentially-Suc[symmetric])
    by (auto elim:eventually-mono)
  moreover have ∀F n in sequentially.
                 B ∗ int (nth-prime n) = c n ∗ a n − c (n + 1) ∧ |c (n + 1)|
<  a n / 2
    using Bc-large by (auto elim:eventually-mono)
  ultimately show ?thesis
    apply (eventually-elim)
  proof (rule impI)
    fix n
```

```
      assume ∀ y≥n. 1 < a y ∀ y≥n. 0 < c y
        and
        Suc-n-eq:B ∗ int (nth-prime (n + 1)) = c (n + 1) ∗ a (n + 1) − c (n +
2) and
        B ∗ int (nth-prime n) = c n ∗ a n − c (n + 1) ∧
              real-of-int |c (n + 1)| < real-of-int (a n) / 2
        and c (n + 1) < c n
      then have n-eq:B ∗ int (nth-prime n) = c n ∗ a n − c (n + 1) and
        c-less-a: real-of-int |c (n + 1)| < real-of-int (a n) / 2
        by auto
      from ‹∀ y≥n. 1 < a y› ‹∀ y≥n. 0 < c y›
      have ∗:a n>1 a (n+1) > 1 c n > 0
        c (n+1) > 0  c (n+2) > 0
        by auto
      then have (1+1/c (n+1))∗ (a n − 1)/a (n+1) = (c (n+1)+1) ∗ ((a n −
1) / (c (n+1) ∗ a (n+1)))
        by (auto simp:field-simps)
      also have ... ≤ c n ∗ ((a n − 1) / (c (n+1) ∗ a (n+1)))
      by (smt (verit) ∗(4) ‹c (n + 1) < c n› a-pos divide-nonneg-nonneg mult-mono
mult-nonneg-nonneg of-int-0-le-iff of-int-le-iff)
      also have ... = (c n ∗ (a n − 1)) / (c (n+1) ∗ a (n+1)) by auto
      also have ... < (c n ∗ (a n − 1)) / (c (n+1) ∗ a (n+1)  − c (n+2))
        apply (rule divide-strict-left-mono)
        subgoal using ‹c (n+2) > 0› by auto
        unfolding Suc-n-eq[symmetric] using ∗ ‹B>0› by auto
      also have ... < (c n ∗ a n − c (n+1)) / (c (n+1) ∗ a (n+1)  − c (n+2))
        apply (rule frac-less)
        unfolding Suc-n-eq[symmetric] using ∗ ‹B>0› ‹c (n + 1) < c n›
        by (auto simp:algebra-simps)
      also have ... = nth-prime n / nth-prime (n+1)
        unfolding Suc-n-eq[symmetric] n-eq[symmetric] using ‹B>0› by auto
      also have ... < 1 by auto
      finally have (1 + 1 / real-of-int (c (n + 1))) ∗ real-of-int (a n − 1)
        / real-of-int (a (n + 1)) < 1  .
      then show a n + (a n − c (n + 1) − 1) /  (c (n + 1)) < (a (n + 1))
        using ∗ by (auto simp:field-simps)
    qed
  qed
  have a-gt-1:∀F n in sequentially. c n > c (n+1) ⟶ a (n+1) > a n + 1
    using Bc-large a-gt-plus c-pos[THEN eventually-all-ge-at-top]
    apply eventually-elim
  proof (rule impI)
    fix n assume
      c (n + 1) < c n ⟶ a n +  (a n − c (n + 1) − 1) / c (n + 1) <  a (n +
1)
      c (n + 1) < c n and B-eq:B ∗ int (nth-prime n) = c n ∗ a n − c (n + 1) ∧
         |real-of-int (c (n + 1))| < real-of-int (a n) / 2 and c-pos:∀ y≥n. 0 < c y
    from this(1,2)
    have a n +  (a n − c (n + 1) − 1) / c (n + 1) <  a (n + 1) by auto
```

```
    moreover have a n − 2 ∗ c (n+1) > 0
      using B-eq c-pos[rule-format,of n+1] by auto
    then have a n − 2 ∗ c (n+1) ≥ 1 by simp
    then have (a n − c (n + 1) − 1) / c (n + 1) ≥ 1
      using c-pos[rule-format,of n+1] by (auto simp:field-simps)
    ultimately show a n + 1 < a (n + 1) by auto
  qed
```

The following corresponds to inequality (3.6) in the paper, which had to be slightly corrected:

```
  have a-gt-sqrt:∀F n in sequentially. c n > c (n+1) ⟶ a (n+1) > a n + (sqrt
n − 2)
  proof −
    have a-2N:∀F N in sequentially. a (2∗N) ≥ N /2 +1
      using c-gt-half a-gt-1[THEN eventually-all-ge-at-top]
    proof eventually-elim
      case (elim N)
      define S where S={n ∈ {N..<2 ∗ N}. c (n + 1) < c n}
      define f where f = (λn. a (Suc n) − a n)

      have f-1:∀ x∈S. f x≥1 and f-0:∀ x. f x≥0
        subgoal using elim unfolding S-def f-def by auto
        subgoal using ‹mono a›[THEN incseq-SucD] unfolding f-def by auto
        done
      have N / 2 < card S
        using elim unfolding S-def by auto
      also have ... ≤ sum f S
        unfolding of-int-sum
        apply (rule sum-bounded-below[of - 1,simplified])
        using f-1 by auto
      also have ... ≤ sum f {N..<2 ∗ N}
        unfolding of-int-sum
        apply (rule sum-mono2)
        unfolding S-def using f-0 by auto
      also have ... = a (2∗N) − a N
        unfolding of-int-sum f-def of-int-diff
        apply (rule sum-Suc-diff')
        by auto
      finally have N / 2 < a (2∗N) − a N .
      then show ?case using a-pos[rule-format,of N] by linarith
    qed

    have a-n4:∀F n in sequentially. a n > n/4
    proof −
      obtain N where a-N:∀ n≥N. a (2∗n) ≥ n /2+1
        using a-2N unfolding eventually-at-top-linorder by auto
      have a n>n/4 when n≥2∗N for n
      proof −
        define n' where n'=n div 2
```

have $n' \geq N$ **unfolding** *n′-def* **using** *that* **by** *auto*
have $n/4 < n' /2+1$
unfolding *n′-def* **by** *auto*
also have ... $\leq a\ (2*n')$
using *a-N* ‹$n' \geq N$› **by** *auto*
also have ... $\leq a\ n$ **unfolding** *n′-def*
apply (*cases even n*)
subgoal by *simp*
subgoal by (*simp add*: *assms*(*2*) *incseqD*)
done
finally show *?thesis* **.**
qed
then show *?thesis*
unfolding *eventually-at-top-linorder* **by** *auto*
qed

have *c-sqrt*:$\forall_F$ *n in sequentially. c n < sqrt n / 4*
proof −
have $\forall_F$ *x in sequentially. x>1* **by** *simp*
moreover have $\forall_F$ *x in sequentially. real* (*nth-prime x*) / (*real x* ∗ *ln* (*real x*)) < *2*
using *nth-prime-asymptotics*[*unfolded asymp-equiv-def*,*THEN order-tendstoD*(*2*),*of 2*]
by *simp*
ultimately have $\forall_F$ *n in sequentially. c n < B*∗*8* ∗*ln n + 1* **using** *a-n4 Bc-large*
proof *eventually-elim*
case (*elim n*)
from *this*(*4*) **have** *c n*=(*B*∗*nth-prime n*+*c* (*n*+*1*))/*a n*
using *a-pos*[*rule-format*,*of n*]
by (*auto simp*:*divide-simps*)
also have ... = (*B*∗*nth-prime n*)/*a n*+*c* (*n*+*1*)/*a n*
by (*auto simp*:*divide-simps*)
also have ... < (*B*∗*nth-prime n*)/*a n* + *1*
proof −
have *c* (*n*+*1*)/*a n* < *1* **using** *elim*(*4*) **by** *auto*
then show *?thesis* **by** *auto*
qed
also have ... < *B*∗*8* ∗ *ln n* + *1*
proof −
have *B*∗*nth-prime n* < *2*∗*B*∗*n*∗*ln n*
using ‹*real* (*nth-prime n*) / (*real n* ∗ *ln* (*real n*)) < *2*› ‹*B*>*0*› ‹ *1* < *n*›
by (*auto simp*:*divide-simps*)
moreover have *real n* / *4* < *real-of-int* (*a n*) **by** *fact*
ultimately have (*B*∗*nth-prime n*) / *a n* < (*2*∗*B*∗*n*∗*ln n*) / (*n*/*4*)
apply (*rule-tac frac-less*)
using ‹*B*>*0*› ‹ *1* < *n*› **by** *auto*
also have ... = *B*∗*8* ∗ *ln n*
using ‹ *1* < *n*› **by** *auto*

finally show *?thesis* **by** *auto*
qed
finally show *?case* **.**
qed
moreover have $\forall_F$ *n in sequentially. B*8 *ln n + 1 < sqrt n / 4*
by *real-asymp*
ultimately show *?thesis*
by *eventually-elim auto*
qed

have
$\forall_F$ *n in sequentially. 0 < c (n+1)*
$\forall_F$ *n in sequentially. c (n+1) < sqrt (n+1) / 4*
$\forall_F$ *n in sequentially. n > 4*
$\forall_F$ *n in sequentially. (n − 4) / sqrt (n + 1) + 1 > sqrt n*
subgoal using *c-pos*[*THEN eventually-all-ge-at-top*]
by *eventually-elim auto*
subgoal using *c-sqrt*[*THEN eventually-all-ge-at-top*]
by *eventually-elim* (*use le-add1* **in** *blast*)
subgoal by *simp*
subgoal
by *real-asymp*
done
then show *?thesis* **using** *a-gt-plus a-n4*
apply *eventually-elim*
proof (*rule impI*)
fix *n* **assume** *asm*:*0 < c (n + 1) c (n + 1) < sqrt (real (n + 1)) / 4* **and**
a-ineq:*c (n + 1) < c n ⟶ a n + (a n − c (n + 1) − 1) / c (n + 1) < a (n + 1)*
c (n + 1) < c n **and** *n / 4 < a n n > 4*
and *n-neq*: *sqrt (real n) < real (n − 4) / sqrt (real (n + 1)) + 1*

have *(n−4) / sqrt(n+1) = (n/4 − 1)/ (sqrt (real (n + 1)) / 4)*
using ‹*n>4*› **by** (*auto simp*:*divide-simps*)
also have ... *< (a n − 1) / c (n + 1)*
apply (*rule frac-less*)
using ‹*n > 4*› ‹*n / 4 < a n*› ‹*0 < c (n + 1)*› ‹*c (n + 1) < sqrt (real (n + 1)) / 4*›
by *auto*
also have ... *− 1 = (a n − c (n + 1) − 1) / c (n + 1)*
using ‹*0 < c (n + 1)*› **by** (*auto simp*:*field-simps*)
also have *a n +* ... *< a (n+1)*
using *a-ineq* **by** *auto*
finally have *a n + ((n − 4) / sqrt (n + 1) − 1) < a (n + 1)* **by** *simp*
moreover have *(n − 4) / sqrt (n + 1) − 1 > sqrt n − 2*
using *n-neq*[*THEN diff-strict-right-mono*,*of 2*] ‹*n>4*›
by (*auto simp*:*algebra-simps of-nat-diff*)
ultimately show *real-of-int (a n) + (sqrt (real n) − 2) < real-of-int (a (n + 1))*

```
      by argo
    qed
  qed
```

The following corresponds to inequality $a_{2N} > N^{3/2}/2$ in the paper, which had to be slightly corrected:

```
  have a-2N-sqrt:∀F N in sequentially. a (2*N) > real N * (sqrt (real N)/2 − 1)
    using c-gt-half a-gt-sqrt[THEN eventually-all-ge-at-top] eventually-gt-at-top[of 4]
  proof eventually-elim
    case (elim N)
    define S where S={n ∈ {N..<2 * N}. c (n + 1) < c n}
    define f where f = (λn. a (Suc n) − a n)

    have f-N:∀x∈S. f x≥sqrt N − 2
    proof
      fix x assume x∈S
      then have sqrt (real x) − 2 < f x x≥N
        using elim unfolding S-def f-def by auto
      moreover have sqrt x − 2 ≥ sqrt N − 2
        using ‹x≥N› by simp
      ultimately show sqrt (real N) − 2 ≤ real-of-int (f x) by argo
    qed
    have f-0:∀x. f x≥0
      using ‹mono a›[THEN incseq-SucD] unfolding f-def by auto

    have (N / 2) * (sqrt N − 2) < card S * (sqrt N − 2)
      apply (rule mult-strict-right-mono)
      subgoal using elim unfolding S-def by auto
      subgoal using ‹N>4›
      by (metis diff-gt-0-iff-gt numeral-less-real-of-nat-iff real-sqrt-four real-sqrt-less-iff)
      done
    also have ... ≤ sum f S
      unfolding of-int-sum
      apply (rule sum-bounded-below)
      using f-N by auto
    also have ... ≤ sum f {N..<2 * N}
      unfolding of-int-sum
      apply (rule sum-mono2)
      unfolding S-def using f-0 by auto
    also have ... = a (2*N) − a N
      unfolding of-int-sum f-def of-int-diff
      apply (rule sum-Suc-diff')
      by auto
    finally have real N / 2 * (sqrt (real N) − 2) < real-of-int (a (2 * N) − a N)
      .
    then have real N / 2 * (sqrt (real N) − 2) < a (2 * N)
      using a-pos[rule-format,of N] by linarith
```

then show *?case* **by** (*auto simp*:*field-simps*)
qed

The following part is required to derive the final contradiction of the proof.

have *a-n-sqrt*:$\forall_F$ *n in sequentially. a n* > (((*n*−*1*)/*2*) *powr* (*3*/*2*) − (*n*−*1*)) /*2*
proof (*rule sequentially-even-odd-imp*)
define *f* **where** *f*=(λ*N*. ((*real* (*2* ∗ *N* − *1*) / *2*) *powr* (*3* / *2*) − *real* (*2* ∗ *N* − *1*)) / *2*)
define *g* **where** *g*=(λ*N*. *real N* ∗ (*sqrt* (*real N*) / *2* − *1*))
have $\forall_F$ *N in sequentially. g N* > *f N*
unfolding *f-def g-def*
by *real-asymp*
moreover have $\forall_F$ *N in sequentially. a* (*2* ∗ *N*) > *g N*
unfolding *g-def* **using** *a-2N-sqrt* **.**
ultimately show $\forall_F$ *N in sequentially. f N* < *a* (*2* ∗ *N*)
by *eventually-elim auto*
next
define *f* **where** *f*=(λ*N*. ((*real* (*2* ∗ *N* + *1* − *1*) / *2*) *powr* (*3* / *2*)
− *real* (*2* ∗ *N* + *1* − *1*)) / *2*)
define *g* **where** *g*=(λ*N*. *real N* ∗ (*sqrt* (*real N*) / *2* − *1*))
have $\forall_F$ *N in sequentially. g N* = *f N*
using *eventually-gt-at-top*[*of 0*]
apply *eventually-elim*
unfolding *f-def g-def*
by (*auto simp*:*algebra-simps powr-half-sqrt*[*symmetric*] *powr-mult-base*)
moreover have $\forall_F$ *N in sequentially. a* (*2* ∗ *N*) > *g N*
unfolding *g-def* **using** *a-2N-sqrt* **.**
moreover have $\forall_F$ *N in sequentially. a* (*2* ∗ *N* + *1*) ≥ *a* (*2*∗*N*)
apply (*rule eventuallyI*)
using ‹*mono a*› **by** (*simp add*: *incseqD*)
ultimately show $\forall_F$ *N in sequentially. f N* < (*a* (*2* ∗ *N* + *1*))
by *eventually-elim auto*
qed

have *a-nth-prime-gt*:$\forall_F$ *n in sequentially. a n* / *nth-prime n* > *1*
proof −
define *f* **where** *f*=(λ*n*::*nat*. (((*n*−*1*)/*2*) *powr* (*3*/*2*) − (*n*−*1*)) /*2*)
have $\forall_F$ *x in sequentially. real* (*nth-prime x*) / (*real x* ∗ *ln* (*real x*)) < *2*
using *nth-prime-asymptotics*[*unfolded asymp-equiv-def*,*THEN order-tendstoD*(*2*),*of 2*]
by *simp*
from *this eventually-gt-at-top*[*of 1*]
have $\forall_F$ *n in sequentially. real* (*nth-prime n*) < *2*∗(*real n* ∗ *ln n*)
by *eventually-elim* (*auto simp*:*field-simps*)
moreover have ∗:$\forall_F$ *N in sequentially. f N* >*0*
unfolding *f-def*
by *real-asymp*
moreover have $\forall_F$ *n in sequentially. f n* < *a n*

using *a-n-sqrt* **unfolding** *f-def* **.**
ultimately have $\forall_F$ *n in sequentially. a n / nth-prime n > f n / (2∗(real n ∗ ln n))*
proof *eventually-elim*
case (*elim n*)
then show *?case*
by (*auto intro*: *frac-less2*)
qed
moreover have $\forall_F$ *n in sequentially.* (*f n*)/ (*2∗*(*real n ∗ ln n*)) > *1*
unfolding *f-def* **by** *real-asymp*
ultimately show *?thesis*
by *eventually-elim argo*
qed

have *a-nth-prime-lt*:$\exists_F$ *n in sequentially. a n / nth-prime n < 1*
proof −
have *liminf* (*λx. a x / nth-prime x*) < *1*
using *nth-2* **by** *auto*
from *this*[*unfolded less-Liminf-iff*]
show *?thesis*
by (*smt* (*verit*) *ereal-less*(*3*) *frequently-elim1 le-less-trans*)
qed

from *a-nth-prime-gt a-nth-prime-lt* **show** *False*
by (*simp add*: *eventually-mono frequently-def*)
qed

5 Acknowledgements

A.K.-A. and W.L. were supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council and led by Professor Lawrence Paulson at the University of Cambridge, UK.

end

References

[1] P. Erdős and E. Straus. On the irrationality of certain series. *Pacific journal of mathematics*, 55(1):85–92, 1974.