

Roth’s Theorem on Arithmetic Progressions

Chelsea Edmonds, Angeliki Koutsoukou-Argyraki and Lawrence C. Paulson
Computer Laboratory, University of Cambridge CB3 0FD
{cle47,ak2110,lp15}@cam.ac.uk

May 14, 2024

Abstract

We formalise a proof of Roth’s Theorem on Arithmetic Progressions, a major result in additive combinatorics on the existence of 3-term arithmetic progressions in subsets of natural numbers. To this end, we follow a proof using graph regularity. We employ our recent formalisation of Szemerédi’s Regularity Lemma, a major result in extremal graph theory, which we use here to prove the Triangle Counting Lemma and the Triangle Removal Lemma. Our sources are Yufei Zhao’s MIT lecture notes “Graph Theory and Additive Combinatorics”¹ and W.T. Gowers’s Cambridge lecture notes “Topics in Combinatorics”.² We also refer to the University of Georgia notes by Stephanie Bell and Will Grodzicki “Using Szemerédi’s Regularity Lemma to Prove Roth’s Theorem”.³

Contents

1 Roth’s Theorem on Arithmetic Progressions	2
1.1 Miscellaneous Preliminaries	2
1.2 Preliminaries on Neighbors in Graphs	4
1.3 Preliminaries on Triangles in Graphs	4
1.4 The Triangle Counting Lemma and the Triangle Removal Lemma	6
1.5 Roth’s Theorem	21

Acknowledgements

The authors were supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council.

¹<https://yufeizhao.com/gtacbook/> and <https://yufeizhao.com/gtac/gtac.pdf>

²<https://www.dpmms.cam.ac.uk/~par31/notes/tic.pdf>

³<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.432.327>

1 Roth's Theorem on Arithmetic Progressions

theory *Roth-Arithmetic-Progressions*

imports *Szemerédi-Regularity.Szemerédi*
Random-Graph-Subgraph-Threshold.Subgraph-Threshold
Ergodic-Theory.Asymptotic-Density
HOL-Library.Ramsey HOL-Library.Nat-Bijection

begin

1.1 Miscellaneous Preliminaries

lemma *sum-prod-le-prod-sum*:

fixes $a :: 'a \Rightarrow 'b :: \text{linordered-idom}$

assumes $\bigwedge i. i \in I \implies a\ i \geq 0 \wedge b\ i \geq 0$

shows $(\sum i \in I. \sum j \in I. a\ i * b\ j) \leq (\sum i \in I. a\ i) * (\sum i \in I. b\ i)$

using *assms*

by (*induction I rule: infinite-finite-induct*) (*auto simp add: algebra-simps sum.distrib sum-distrib-left*)

lemma *real-mult-gt-cube*: $A \geq (X :: \text{real}) \implies B \geq X \implies C \geq X \implies X \geq 0 \implies A * B * C \geq X^3$

by (*simp add: mult-mono' power3-eq-cube*)

lemma *triple-sigma-rewrite-card*:

assumes *finite X finite Y finite Z*

shows $\text{card } \{(x,y,z) . x \in X \wedge (y,z) \in Y \times Z \wedge P\ x\ y\ z\} = (\sum x \in X . \text{card } \{(y,z) \in Y \times Z . P\ x\ y\ z\})$

proof –

define W **where** $W \equiv \lambda x. \{(y,z) \in Y \times Z . P\ x\ y\ z\}$

have $W\ x \subseteq Y \times Z$ **for** x

by (*auto simp: W-def*)

then have [*simp*]: *finite (W x)* **for** x

by (*meson assms finite-SigmaI infinite-super*)

have *eq*: $\{(x,y,z) . x \in X \wedge (y,z) \in Y \times Z \wedge P\ x\ y\ z\} = (\bigcup x \in X. \bigcup (y,z) \in W\ x. \{(x,y,z)\})$

by (*auto simp: W-def*)

show *?thesis*

unfolding *eq* **by** (*simp add: disjoint-iff assms card-UN-disjoint*) (*simp add: W-def*)

qed

lemma *all-edges-between-mono1*:

$Y \subseteq Z \implies \text{all-edges-between } Y\ X\ G \subseteq \text{all-edges-between } Z\ X\ G$

by (*auto simp: all-edges-between-def*)

lemma *all-edges-between-mono2*:

$Y \subseteq Z \implies \text{all-edges-between } X\ Y\ G \subseteq \text{all-edges-between } X\ Z\ G$

by (*auto simp: all-edges-between-def*)

lemma *uwellformed-alt-fst*:
assumes *uwellformed* G $\{x, y\} \in \text{uedges } G$
shows $x \in \text{uverts } G$
using *uwellformed-def* *assms* **by** *simp*

lemma *uwellformed-alt-snd*:
assumes *uwellformed* G $\{x, y\} \in \text{uedges } G$
shows $y \in \text{uverts } G$
using *uwellformed-def* *assms* **by** *simp*

lemma *all-edges-between-subset-times*: *all-edges-between* $X Y G \subseteq (X \cap \bigcup(\text{uedges } G)) \times (Y \cap \bigcup(\text{uedges } G))$
by (*auto simp: all-edges-between-def*)

lemma *finite-all-edges-between'*:
assumes *finite* ($\text{uverts } G$) *uwellformed* G
shows *finite* (*all-edges-between* $X Y G$)
proof –
have *finite* ($\bigcup(\text{uedges } G)$)
by (*meson Pow-iff all-edges-subset-Pow assms finite-Sup subsetD wellformed-all-edges*)
with *all-edges-between-subset-times* **show** *?thesis*
by (*metis finite-Int finite-SigmaI finite-subset*)
qed

lemma *all-edges-between-E-diff*:
all-edges-between $X Y (V, E - E') = \text{all-edges-between } X Y (V, E) - \text{all-edges-between } X Y (V, E')$
by (*auto simp: all-edges-between-def*)

lemma *all-edges-between-E-Un*:
all-edges-between $X Y (V, E \cup E') = \text{all-edges-between } X Y (V, E) \cup \text{all-edges-between } X Y (V, E')$
by (*auto simp: all-edges-between-def*)

lemma *all-edges-between-E-UN*:
all-edges-between $X Y (V, \bigcup_{i \in I} E i) = (\bigcup_{i \in I} \text{all-edges-between } X Y (V, E i))$
by (*auto simp: all-edges-between-def*)

lemma *all-edges-preserved*: $\llbracket \text{all-edges-between } A B G' = \text{all-edges-between } A B G; X \subseteq A; Y \subseteq B \rrbracket$
 $\implies \text{all-edges-between } X Y G' = \text{all-edges-between } X Y G$
by (*auto simp: all-edges-between-def*)

lemma *subgraph-edge-wf*:
assumes *uwellformed* G $\text{uverts } H = \text{uverts } G$ $\text{uedges } H \subseteq \text{uedges } G$
shows *uwellformed* H
by (*metis assms subsetD uwellformed-def*)

1.2 Preliminaries on Neighbors in Graphs

definition *neighbor-in-graph*:: $uvert \Rightarrow uvert \Rightarrow ugraph \Rightarrow bool$
where *neighbor-in-graph* $x\ y\ G \equiv (x \in (uverts\ G) \wedge y \in (uverts\ G) \wedge \{x,y\} \in (uedges\ G))$

definition *neighbors* :: $uvert \Rightarrow ugraph \Rightarrow uvert\ set$ **where**
neighbors $x\ G \equiv \{y \in uverts\ G \mid neighbor-in-graph\ x\ y\ G\}$

definition *neighbors-ss*:: $uvert\ set \Rightarrow ugraph \Rightarrow uvert\ set$ **where**
neighbors-ss $x\ Y\ G \equiv \{y \in Y \mid neighbor-in-graph\ x\ y\ G\}$

lemma *all-edges-betw-sigma-neighbor*:
 $uwellformed\ G \implies all-edges-between\ X\ Y\ G = (SIGMA\ x:X. neighbors-ss\ x\ Y\ G)$
by (*auto simp add: all-edges-between-def neighbors-ss-def neighbor-in-graph-def uwellformed-alt-fst uwellformed-alt-snd*)

lemma *card-all-edges-betw-neighbor*:
assumes *finite* X *finite* Y *uwellformed* G
shows $card\ (all-edges-between\ X\ Y\ G) = (\sum\ x \in X. card\ (neighbors-ss\ x\ Y\ G))$
using *all-edges-betw-sigma-neighbor* *assms* **by** (*simp add: neighbors-ss-def*)

1.3 Preliminaries on Triangles in Graphs

definition *triangle-in-graph*:: $uvert \Rightarrow uvert \Rightarrow uvert \Rightarrow ugraph \Rightarrow bool$
where *triangle-in-graph* $x\ y\ z\ G$
 $\equiv (\{x,y\} \in uedges\ G) \wedge (\{y,z\} \in uedges\ G) \wedge (\{x,z\} \in uedges\ G)$

definition *triangle-triples*
where *triangle-triples* $X\ Y\ Z\ G \equiv \{(x,y,z) \in X \times Y \times Z \mid triangle-in-graph\ x\ y\ z\ G\}$

lemma *triangle-commu1*:
assumes *triangle-in-graph* $x\ y\ z\ G$
shows *triangle-in-graph* $y\ x\ z\ G$
using *assms triangle-in-graph-def* **by** (*auto simp add: insert-commute*)

lemma *triangle-vertices-distinct1*:
assumes *wf*: *uwellformed* G
assumes *tri*: *triangle-in-graph* $x\ y\ z\ G$
shows $x \neq y$

proof (*rule ccontr*)
assume $a: \neg x \neq y$
have $card\ \{x, y\} = 2$ **using** *tri wf triangle-in-graph-def*
using *uwellformed-def* **by** *blast*
thus *False* **using** a **by** *simp*
qed

lemma *triangle-vertices-distinct2*:
assumes *uwellformed* G *triangle-in-graph* $x\ y\ z\ G$

shows $y \neq z$
by (*metis* *assms* *triangle-vertices-distinct1* *triangle-in-graph-def*)

lemma *triangle-in-graph-edge-point*:

assumes *uwellformed* G

shows $\text{triangle-in-graph } x \ y \ z \ G \longleftrightarrow \{y, z\} \in \text{uedges } G \wedge \text{neighbor-in-graph } x \ y \ G \wedge \text{neighbor-in-graph } x \ z \ G$

by (*auto* *simp* *add*: *triangle-in-graph-def* *neighbor-in-graph-def* *assms* *uwellformed-alt-fst* *uwellformed-alt-snd*)

definition

unique-triangles G

$\equiv \forall e \in \text{uedges } G. \exists ! T. \exists x \ y \ z. T = \{x, y, z\} \wedge \text{triangle-in-graph } x \ y \ z \ G \wedge e \subseteq T$

definition *triangle-free-graph*:: *ugraph* \Rightarrow *bool*

where *triangle-free-graph* $G \equiv \neg(\exists x \ y \ z. \text{triangle-in-graph } x \ y \ z \ G)$

lemma *triangle-free-graph-empty*: $\text{uedges } G = \{\} \Longrightarrow \text{triangle-free-graph } G$

by (*simp* *add*: *triangle-free-graph-def* *triangle-in-graph-def*)

lemma *edge-vertices-not-equal*:

assumes *uwellformed* G $\{x, y\} \in \text{uedges } G$

shows $x \neq y$

using *assms* *triangle-in-graph-def* *triangle-vertices-distinct1* **by** *blast*

lemma *triangle-in-graph-verts*:

assumes *uwellformed* G *triangle-in-graph* $x \ y \ z \ G$

shows $x \in \text{uverts } G \ y \in \text{uverts } G \ z \in \text{uverts } G$

proof –

have 1: $\{x, y\} \in \text{uedges } G$ **using** *triangle-in-graph-def*

using *assms*(2) **by** *auto*

then show $x \in \text{uverts } G$ **using** *uwellformed-alt-fst* *assms* **by** *blast*

then show $y \in \text{uverts } G$ **using** 1 *uwellformed-alt-snd* *assms* **by** *blast*

have $\{x, z\} \in \text{uedges } G$ **using** *triangle-in-graph-def* *assms*(2) **by** *auto*

then show $z \in \text{uverts } G$ **using** *uwellformed-alt-snd* *assms* **by** *blast*

qed

definition *triangle-set* :: *ugraph* \Rightarrow *uvert set set*

where *triangle-set* $G \equiv \{ \{x, y, z\} \mid x \ y \ z. \text{triangle-in-graph } x \ y \ z \ G \}$

fun *mk-triangle-set* :: (*uvert* \times *uvert* \times *uvert*) \Rightarrow *uvert set*

where *mk-triangle-set* $(x, y, z) = \{x, y, z\}$

lemma *finite-triangle-set*:

assumes *fin*: *finite* (*uverts* G) **and** *wf*: *uwellformed* G

shows *finite* (*triangle-set* G)

proof –

```

have triangle-set  $G \subseteq \text{Pow}$  (uverts  $G$ )
  using PowI local.wf triangle-in-graph-def triangle-set-def uwellformed-def by
  auto
  then show ?thesis
  by (meson fin finite-Pow-iff infinite-super)
qed

```

```

lemma card-triangle-3:
  assumes  $t \in \text{triangle-set } G$  uwellformed  $G$ 
  shows  $\text{card } t = 3$ 
  using assms by (auto simp: triangle-set-def edge-vertices-not-equal triangle-in-graph-def)

```

```

lemma triangle-set-power-set-ss: uwellformed  $G \implies \text{triangle-set } G \subseteq \text{Pow}$  (uverts
 $G$ )
  by (auto simp add: triangle-set-def triangle-in-graph-def uwellformed-alt-fst uwell-
  formed-alt-snd)

```

```

lemma triangle-in-graph-ss:
  assumes  $\text{uedges } G_{\text{new}} \subseteq \text{uedges } G$ 
  assumes  $\text{triangle-in-graph } x \ y \ z \ G_{\text{new}}$ 
  shows  $\text{triangle-in-graph } x \ y \ z \ G$ 
using assms triangle-in-graph-def by auto

```

```

lemma triangle-set-graph-edge-ss:
  assumes  $\text{uedges } G_{\text{new}} \subseteq \text{uedges } G$ 
  assumes  $\text{uverts } G_{\text{new}} = \text{uverts } G$ 
  shows  $\text{triangle-set } G_{\text{new}} \subseteq \text{triangle-set } G$ 
  using assms unfolding triangle-set-def by (blast intro: triangle-in-graph-ss)

```

```

lemma triangle-set-graph-edge-ss-bound:
  fixes  $G :: \text{ugraph}$  and  $G_{\text{new}} :: \text{ugraph}$ 
  assumes uwellformed  $G$  finite (uverts  $G$ )  $\text{uedges } G_{\text{new}} \subseteq \text{uedges } G$   $\text{uverts } G_{\text{new}}$ 
  =  $\text{uverts } G$ 
  shows  $\text{card} (\text{triangle-set } G) \geq \text{card} (\text{triangle-set } G_{\text{new}})$ 
  by (simp add: assms card-mono finite-triangle-set triangle-set-graph-edge-ss)

```

1.4 The Triangle Counting Lemma and the Triangle Removal Lemma

We begin with some more auxiliary material to be used in the main lemmas.

```

lemma regular-pair-neighbor-bound:
  fixes  $\varepsilon :: \text{real}$ 
  assumes  $\text{fin } G$ : finite (uverts  $G$ )
  assumes  $xss$ :  $X \subseteq \text{uverts } G$  and  $yss$ :  $Y \subseteq \text{uverts } G$  and  $\text{card } X > 0$ 
  and  $wf$ : uwellformed  $G$ 
  and  $eg0$ :  $\varepsilon > 0$  and  $\varepsilon$ -regular-pair  $X \ Y \ G$  and  $ed$ :  $\text{edge-density } X \ Y \ G \geq 2 * \varepsilon$ 
  defines  $X' \equiv \{x \in X. \text{card} (\text{neighbors-ss } x \ Y \ G) < (\text{edge-density } X \ Y \ G - \varepsilon) * \text{card } (Y)\}$ 
  shows  $\text{card } X' < \varepsilon * \text{card } X$ 

```

(is card (?X') < ε * -)
proof (cases ?X' = {})

case False — Following Gowers's proof - more in depth with reasoning on contradiction

let ?rxy = 1/(card X' * card Y)

show ?thesis

proof (rule ccontr)

assume ¬ (card (X') < ε * card X)

then have a: (card(X') ≥ ε * card X) **by** simp

have fin: finite X finite Y **using** assms finite-subset **by** auto

have ebound: ε ≤ 1/2

by (metis ed edge-density-le1 le-divide-eq-numeral1(1) mult.commute order-trans)

have finx: finite X' **using** fin X'-def **by** simp

have ∧ x. x ∈ X' ⇒ (card (neighbors-ss x Y G)) < (edge-density X Y G - ε) * (card Y)

unfolding X'-def **by** blast

then have (∑ x∈X'. card (neighbors-ss x Y G)) < (∑ x∈X'. ((edge-density X Y G - ε) * (card Y)))

using False sum-strict-mono X'-def

by (smt (verit, del-insts) finx of-nat-sum)

then have upper: (∑ x∈X'. card (neighbors-ss x Y G)) < (card X') * ((edge-density X Y G - ε) * (card Y))

by (simp add: sum-bounded-above)

have yge0: card Y > 0

by (metis gr0I mult-eq-0-iff of-nat-0 of-nat-less-0-iff upper)

have ?rxy > 0

using card-0-eq finx False yge0 X'-def **by** fastforce

then have upper2: ?rxy * (∑ x∈X'. card (neighbors-ss x Y G)) < ?rxy * (card X') * ((edge-density X Y G - ε) * (card Y))

by (smt (verit) mult.assoc mult-le-cancel-left upper)

have ?rxy * (card X') * ((edge-density X Y G - ε) * (card Y)) = edge-density X Y G - ε

using False X'-def finx **by** force

with ⟨ε > 0⟩ upper2 **have** con: edge-density X Y G - ?rxy * (∑ x∈X'. card (neighbors-ss x Y G)) > ε

by linarith

have |edge-density X Y G - ?rxy * (∑ x∈X'. card (neighbors-ss x Y G))| = |?rxy * (card (all-edges-between X' Y G)) - edge-density X Y G|

using card-all-edges-betw-neighbor fin wf **by** (simp add: X'-def)

also have ... = |edge-density X' Y G - edge-density X Y G|

by (simp add: edge-density-def)

also have ... ≤ ε

using assms ebound yge0 a **by** (force simp add: X'-def regular-pair-def)

finally show False **using** con **by** linarith

qed

qed (simp add: ⟨card X > 0⟩ eg0)

lemma neighbor-set-meets-e-reg-cond:

fixes $\varepsilon::\text{real}$
assumes $\text{edge-density } X Y G \geq 2*\varepsilon$
and $\text{card } (\text{neighbors-ss } x Y G) \geq (\text{edge-density } X Y G - \varepsilon) * \text{card } Y$
shows $\text{card } (\text{neighbors-ss } x Y G) \geq \varepsilon * \text{card } (Y)$
by (*smt (verit) assms mult-right-mono of-nat-0-le-iff*)

lemma *all-edges-btwn-neighbor-sets-lower-bound:*

fixes $\varepsilon::\text{real}$
assumes $\text{rp2: } \varepsilon\text{-regular-pair } Y Z G$
and $\text{ed1: edge-density } X Y G \geq 2*\varepsilon$ **and** $\text{ed2: edge-density } X Z G \geq 2*\varepsilon$
and $\text{cond1: card } (\text{neighbors-ss } x Y G) \geq (\text{edge-density } X Y G - \varepsilon) * \text{card } Y$
and $\text{cond2: card } (\text{neighbors-ss } x Z G) \geq (\text{edge-density } X Z G - \varepsilon) * \text{card } Z$
shows $\text{card } (\text{all-edges-between } (\text{neighbors-ss } x Y G) (\text{neighbors-ss } x Z G) G)$
 $\geq (\text{edge-density } Y Z G - \varepsilon) * \text{card } (\text{neighbors-ss } x Y G) * \text{card } (\text{neighbors-ss } x Z G)$
(is $\text{card } (\text{all-edges-between } ?Y' ?Z' G) \geq (\text{edge-density } Y Z G - \varepsilon) * \text{card } ?Y'$
 $* \text{card } ?Z')$
proof –
have $\text{yss': } ?Y' \subseteq Y$ **using** *neighbors-ss-def* **by** *simp*
have $\text{zss': } ?Z' \subseteq Z$ **using** *neighbors-ss-def* **by** *simp*
have $\text{min-sizeY: card } ?Y' \geq \varepsilon * \text{card } Y$
using $\text{cond1 ed1 neighbor-set-meets-e-reg-cond}$ **by** *blast*
have $\text{min-sizeZ: card } ?Z' \geq \varepsilon * \text{card } Z$
using $\text{cond2 ed2 neighbor-set-meets-e-reg-cond}$ **by** *blast*
then have $|\text{edge-density } ?Y' ?Z' G - \text{edge-density } Y Z G| \leq \varepsilon$
using $\text{min-sizeY yss' zss' assms}$ **by** (*force simp add: regular-pair-def*)
then have $\text{edge-density } Y Z G - \varepsilon \leq (\text{card } (\text{all-edges-between } ?Y' ?Z' G)) / (\text{card } ?Y' * \text{card } ?Z')$
using *edge-density-def* **by** *simp*
then have $(\text{card } ?Y' * \text{card } ?Z') * (\text{edge-density } Y Z G - \varepsilon) \leq (\text{card } (\text{all-edges-between } ?Y' ?Z' G))$
by (*fastforce simp: divide-simps mult.commute simp flip: of-nat-mult split: if-split-asm*)
then show $?thesis$
by (*metis (no-types, lifting) mult.assoc mult-of-nat-commute of-nat-mult*)
qed

We are now ready to show the Triangle Counting Lemma:

theorem *triangle-counting-lemma:*

fixes $\varepsilon::\text{real}$
assumes $\text{xss: } X \subseteq \text{uverts } G$ **and** $\text{yss: } Y \subseteq \text{uverts } G$ **and** $\text{zss: } Z \subseteq \text{uverts } G$ **and**
 $\text{en0: } \varepsilon > 0$
and $\text{finG: finite } (\text{uverts } G)$ **and** $\text{wf: uwellformed } G$
and $\text{rp1: } \varepsilon\text{-regular-pair } X Y G$ **and** $\text{rp2: } \varepsilon\text{-regular-pair } Y Z G$ **and** $\text{rp3: } \varepsilon\text{-regular-pair } X Z G$
and $\text{ed1: edge-density } X Y G \geq 2*\varepsilon$ **and** $\text{ed2: edge-density } X Z G \geq 2*\varepsilon$ **and**
 $\text{ed3: edge-density } Y Z G \geq 2*\varepsilon$
shows $\text{card } (\text{triangle-triples } X Y Z G)$
 $\geq (1-2*\varepsilon) * (\text{edge-density } X Y G - \varepsilon) * (\text{edge-density } X Z G - \varepsilon) *$

$(\text{edge-density } Y Z G - \varepsilon) * \text{card } X * \text{card } Y * \text{card } Z$

proof –

let $?T\text{-all} = \{(x,y,z) \in X \times Y \times Z. (\text{triangle-in-graph } x y z G)\}$
let $?ediff = \lambda X Y. \text{edge-density } X Y G - \varepsilon$
define XF **where** $XF \equiv \lambda Y. \{x \in X. \text{card}(\text{neighbors-ss } x Y G) < ?ediff X Y * \text{card } Y\}$
have fin : $\text{finite } X \text{ finite } Y \text{ finite } Z$ **using** $finG \text{ rev-finite-subset } xss yss zss$ **by** $auto$
then have $\text{card } X > 0$
using $\text{card-0-eq } ed1 \text{ edge-density-def } en0$ **by** fastforce

Obtain a subset of X where all elements meet minimum numbers for neighborhood size in Y and Z .

define $X2$ **where** $X2 \equiv X - (XF Y \cup XF Z)$
have xss : $X2 \subseteq X$ **and** $finx2$: $\text{finite } X2$
by $(\text{auto simp add: } X2\text{-def } fin)$

Reasoning on the minimum size of $X2$:

have $part1$: $(XF Y \cup XF Z) \cup X2 = X$
by $(\text{auto simp: } XF\text{-def } X2\text{-def})$
have $\text{card-}XFY$: $\text{card } (XF Y) < \varepsilon * \text{card } X$
using $\text{regular-pair-neighbor-bound } assms \langle \text{card } X > 0 \rangle$ **by** $(\text{simp add: } XF\text{-def})$

We now repeat the same argument as above to the regular pair $X Z$ in G .

have $\text{card-}XFZ$: $\text{card } (XF Z) < \varepsilon * \text{card } X$
using $\text{regular-pair-neighbor-bound } assms \langle \text{card } X > 0 \rangle$ **by** $(\text{simp add: } XF\text{-def})$
have $\text{card } (XF Y \cup XF Z) \leq 2 * \varepsilon * (\text{card } X)$
by $(\text{smt } (verit) \text{ card-}XFY \text{ card-}XFZ \text{ card-Un-le } \text{comm-semiring-class.distrib } \text{of-nat-add } \text{of-nat-mono})$
then have $\text{card } X2 \geq \text{card } X - 2 * \varepsilon * \text{card } X$
using $part1$ **by** $(\text{smt } (verit, \text{del-insts}) \text{ card-Un-le } \text{of-nat-add } \text{of-nat-mono})$
then have $minx2$: $\text{card } X2 \geq (1 - 2 * \varepsilon) * \text{card } X$
by $(\text{metis } \text{mult.commute } \text{mult-cancel-left2 } \text{right-diff-distrib})$

Reasoning on the minimum number of edges between neighborhoods of X in Y and Z .

have $edyzgt0$: $?ediff Y Z > 0$ **and** $edxygt0$: $?ediff X Y > 0$
using $ed1 ed3 \langle \varepsilon > 0 \rangle$ **by** linarith+
have card-y-bound : $\text{card } (\text{neighbors-ss } x Y G) \geq ?ediff X Y * \text{card } Y$
and card-z-bound : $\text{card } (\text{neighbors-ss } x Z G) \geq ?ediff X Z * \text{card } Z$
if $x \in X2$ **for** x
using $that$ **by** $(\text{auto simp: } XF\text{-def } X2\text{-def})$
have $\text{card-y-bound}'$:

$$\left(\sum_{x \in X2. ?ediff Y Z * (\text{card } (\text{neighbors-ss } x Y G)) * (\text{card } (\text{neighbors-ss } x Z G)) \right) \geq$$

$$\left(\sum_{x \in X2. ?ediff Y Z * ?ediff X Y * (\text{card } Y) * (\text{card } (\text{neighbors-ss } x Z G)) \right)$$

by (*rule sum-mono*) (*smt (verit, best) mult.left-commute card-y-bound edyzt0 mult.commute mult-right-mono of-nat-0-le-iff*)
have *card-z-bound'*:
 $(\sum_{x \in X2}. ?ediff\ Y\ Z * ?ediff\ X\ Y * (card\ Y) * (card\ (neighbors-ss\ x\ Z\ G))) \geq$
 $(\sum_{x \in X2}. ?ediff\ Y\ Z * ?ediff\ X\ Y * (card\ Y) * ?ediff\ X\ Z * (card\ Z))$
using *card-z-bound mult-left-mono edxygt0 edyzt0* **by** (*fastforce intro!: sum-mono*)
have *eq-set*: $\bigwedge x. \{ (y,z). y \in Y \wedge z \in Z \wedge \{y, z\} \in uedges\ G \wedge neighbor-in-graph\ x\ y\ G \wedge neighbor-in-graph\ x\ z\ G \} =$
 $\{ (y,z). y \in (neighbors-ss\ x\ Y\ G) \wedge z \in (neighbors-ss\ x\ Z\ G) \wedge \{y, z\} \in uedges\ G \}$
by (*auto simp: neighbors-ss-def*)
have *card ?T-all* = $(\sum_{x \in X}. card\ \{ (y,z) \in Y \times Z. triangle-in-graph\ x\ y\ z\ G \})$
using *triple-sigma-rewrite-card fin* **by** *force*
also have $\dots = (\sum_{x \in X}. card\ \{ (y,z). y \in Y \wedge z \in Z \wedge \{y, z\} \in uedges\ G \wedge neighbor-in-graph\ x\ y\ G \wedge neighbor-in-graph\ x\ z\ G \})$
using *triangle-in-graph-edge-point assms* **by** *auto*
also have $\dots = (\sum_{x \in X}. card\ (all-edges-between\ (neighbors-ss\ x\ Y\ G)\ (neighbors-ss\ x\ Z\ G)\ G))$
using *all-edges-between-def eq-set* **by** *presburger*
finally have *l*: $card\ ?T-all \geq (\sum_{x \in X2}. card\ (all-edges-between\ (neighbors-ss\ x\ Y\ G)\ (neighbors-ss\ x\ Z\ G)\ G))$
by (*simp add: fin xss sum-mono2*)
have $(\sum_{x \in X2}. ?ediff\ Y\ Z * (card\ (neighbors-ss\ x\ Y\ G)) * (card\ (neighbors-ss\ x\ Z\ G))) \leq$
 $(\sum_{x \in X2}. real\ (card\ (all-edges-between\ (neighbors-ss\ x\ Y\ G)\ (neighbors-ss\ x\ Z\ G)\ G)))$
(is sum ?F - ≤ sum ?G -)
proof (*rule sum-mono*)
show $\bigwedge x. x \in X2 \implies ?F\ x \leq ?G\ x$
using *all-edges-btwn-neighbor-sets-lower-bound card-y-bound card-z-bound ed1 ed2 rp2* **by** *blast*
qed
then have $card\ ?T-all \geq card\ X2 * ?ediff\ Y\ Z * ?ediff\ X\ Y * card\ Y * ?ediff\ X\ Z * card\ Z$
using *card-z-bound' card-y-bound' l of-nat-le-iff [symmetric, where 'a=real]*
by *force*
then have $real\ (card\ ?T-all) \geq ((1 - 2 * \epsilon) * card\ X) * ?ediff\ Y\ Z * ?ediff\ X\ Y * (card\ Y) * ?ediff\ X\ Z * (card\ Z)$
by (*smt (verit, best) ed2 edxygt0 edyzt0 en0 minx2 mult-right-mono of-nat-0-le-iff*)
then show *?thesis* **by** (*simp add: triangle-triples-def mult.commute mult.left-commute*)

qed

definition *regular-graph* :: $real \Rightarrow uvert\ set\ set \Rightarrow ugraph \Rightarrow bool$

(*--regular'-graph [999]1000*)

where ϵ -*regular-graph* $P\ G \equiv \forall R\ S. R \in P \longrightarrow S \in P \longrightarrow \epsilon$ -*regular-pair* $R\ S\ G$
for $\epsilon :: real$

A minimum density, but empty edge sets are excluded.

definition *edge-dense* :: *nat set* \Rightarrow *nat set* \Rightarrow *ugraph* \Rightarrow *real* \Rightarrow *bool*
where *edge-dense* *X Y G* $\varepsilon \equiv$ *all-edges-between* *X Y G* = $\{\}$ \vee *edge-density* *X Y G* $\geq \varepsilon$

definition *dense-graph* :: *uvert set set* \Rightarrow *ugraph* \Rightarrow *real* \Rightarrow *bool*
where *dense-graph* *P G* $\varepsilon \equiv \forall R S. R \in P \longrightarrow S \in P \longrightarrow$ *edge-dense* *R S G* ε **for** $\varepsilon :: \text{real}$

definition *decent* :: *nat set* \Rightarrow *nat set* \Rightarrow *ugraph* \Rightarrow *real* \Rightarrow *bool*
where *decent* *X Y G* $\eta \equiv$ *all-edges-between* *X Y G* = $\{\}$ \vee (*card* *X* $\geq \eta \wedge$ *card* *Y* $\geq \eta$) **for** $\eta :: \text{real}$

definition *decent-graph* :: *uvert set set* \Rightarrow *ugraph* \Rightarrow *real* \Rightarrow *bool*
where *decent-graph* *P G* $\eta \equiv \forall R S. R \in P \longrightarrow S \in P \longrightarrow$ *decent* *R S G* η

The proof of the triangle counting lemma requires ordered triples. For each unordered triple there are six permutations, hence the factor of 1/6 here.

lemma *card-convert-triangle-rep*:

fixes *G* :: *ugraph*

assumes *X* \subseteq *uverts G* **and** *Y* \subseteq *uverts G* **and** *Z* \subseteq *uverts G* **and** *fin*: *finite* (*uverts G*)

and *wf*: *wellformed G*

shows *card* (*triangle-set G*) $\geq 1/6 * \text{card} \{(x,y,z) \in X \times Y \times Z . (\text{triangle-in-graph } x y z G)\}$

(**is** $\geq 1/6 * \text{card } ?TT$)

proof –

define *tofl* **where** *tofl* $\equiv \lambda l :: \text{nat list. } (\text{hd } l, \text{hd}(tl l), \text{hd}(tl(tl l)))$

have *in-tofl*: $(x,y,z) \in \text{tofl } \text{'permutations-of-set } \{x,y,z\}$ **if** $x \neq y \neq z \neq x$ **for** *x y z*

proof –

have *distinct*[*x,y,z*]

using *that* **by** *simp*

then show *?thesis*

unfolding *tofl-def* *image-iff*

by (*smt* (*verit*, *best*) *list.sel*(1) *list.sel*(3) *set-simps* *permutations-of-set1* *set-empty*)

qed

have *?TT* $\subseteq \{(x,y,z). (\text{triangle-in-graph } x y z G)\}$

by *auto*

also have $\dots \subseteq (\bigcup t \in \text{triangle-set } G. \text{tofl } \text{'permutations-of-set } t)$

using *edge-vertices-not-equal* [*OF wf*] *in-tofl*

by (*clarsimp* *simp* *add*: *triangle-set-def* *triangle-in-graph-def*) *metis*

finally have *?TT* $\subseteq (\bigcup t \in \text{triangle-set } G. \text{tofl } \text{'permutations-of-set } t)$.

then have *card* *?TT* $\leq \text{card}(\bigcup t \in \text{triangle-set } G. \text{tofl } \text{'permutations-of-set } t)$

by (*intro* *card-mono* *finite-UN-I* *finite-triangle-set*) (*auto* *simp*: *assms*)

also have $\dots \leq (\sum t \in \text{triangle-set } G. \text{card} (\text{tofl } \text{'permutations-of-set } t))$

using *card-UN-le* *fin* *finite-triangle-set* *local.wf* **by** *blast*

also have $\dots \leq (\sum t \in \text{triangle-set } G. \text{card } (\text{permutations-of-set } t))$
by (*meson card-image-le finite-permutations-of-set sum-mono*)
also have $\dots \leq (\sum t \in \text{triangle-set } G. \text{fact } 3)$
by (*rule sum-mono*) (*metis card.infinite card-permutations-of-set card-triangle-3*
eq-refl local.wf nat.case numeral-3-eq-3)
also have $\dots = 6 * \text{card } (\text{triangle-set } G)$
by (*simp add: eval-nat-numeral*)
finally have $\text{card } ?TT \leq 6 * \text{card } (\text{triangle-set } G)$.
then show *?thesis*
by (*simp add: divide-simps*)
qed

lemma *card-convert-triangle-rep-bound*:
fixes $G :: \text{ugraph}$ **and** $t :: \text{real}$
assumes $X \subseteq \text{uverts } G$ **and** $Y \subseteq \text{uverts } G$ **and** $Z \subseteq \text{uverts } G$ **and** *fin: finite*
(uverts G)
and *wf: wellformed G*
assumes $\text{card } \{(x,y,z) \in X \times Y \times Z . (\text{triangle-in-graph } x \ y \ z \ G)\} \geq t$
shows $\text{card } (\text{triangle-set } G) \geq 1/6 * t$
proof –
define t' **where** $t' \equiv \text{card } \{(x,y,z) \in X \times Y \times Z . (\text{triangle-in-graph } x \ y \ z \ G)\}$
have $t' \geq t$ **using** *assms t'-def* **by** *simp*
then have *tgt: 1/6 * t' ≥ 1/6 * t* **by** *simp*
have $\text{card } (\text{triangle-set } G) \geq 1/6 * t'$ **using** *t'-def card-convert-triangle-rep assms*
by *simp*
thus *?thesis* **using** *tgt* **by** *linarith*
qed

lemma *edge-density-eq0*:
assumes *all-edges-between A B G = {}* **and** $X \subseteq A$ $Y \subseteq B$
shows *edge-density X Y G = 0*
proof –
have *all-edges-between X Y G = {}*
by (*metis all-edges-between-mono1 all-edges-between-mono2 assms subset-empty*)
then show *?thesis*
by (*auto simp: edge-density-def*)
qed

The following is the Triangle Removal Lemma.

theorem *triangle-removal-lemma*:
fixes $\varepsilon :: \text{real}$
assumes *egt: $\varepsilon > 0$*
shows $\exists \delta :: \text{real} > 0. \forall G. \text{card}(\text{uverts } G) > 0 \longrightarrow \text{wellformed } G \longrightarrow$
 $\text{card } (\text{triangle-set } G) \leq \delta * \text{card}(\text{uverts } G) ^ 3 \longrightarrow$
 $(\exists G'. \text{triangle-free-graph } G' \wedge \text{uverts } G' = \text{uverts } G \wedge \text{uedges } G' \subseteq \text{uedges}$
 $G \wedge$
 $\text{card } (\text{uedges } G - \text{uedges } G') \leq \varepsilon * (\text{card } (\text{uverts } G))^2)$
(is $\exists \delta :: \text{real} > 0. \forall G. - \longrightarrow - \longrightarrow - \longrightarrow (\exists G_{\text{new}}. ?\Phi \ G \ G_{\text{new}}))$
proof (*cases $\varepsilon < 1$*)

```

case False
show ?thesis
proof (intro exI conjI strip)
  fix G
  define Gnew where  $Gnew \equiv ((uverts\ G), \{\}::uedge\ set)$ 
  assume G: uwellformed G  $card(uverts\ G) > 0$ 
  then show triangle-free-graph Gnew  $uverts\ Gnew = uverts\ G$   $uedges\ Gnew \subseteq$ 
uedges G
    by (auto simp: Gnew-def triangle-free-graph-empty)
  have  $real\ (card\ (uedges\ G)) \leq (card\ (uverts\ G))^2$ 
    by (meson G card-gt-0-iff max-edges-graph of-nat-le-iff)
  also have  $\dots \leq \varepsilon * (card\ (uverts\ G))^2$ 
    using False mult-le-cancel-right1 by fastforce
  finally show  $real\ (card\ (uedges\ G - uedges\ Gnew)) \leq \varepsilon * ((card\ (uverts\ G))^2)$ 
    by (simp add: Gnew-def)
  qed (rule zero-less-one)
next
case True
have e4gt:  $\varepsilon/4 > 0$  using  $\langle \varepsilon > 0 \rangle$  by auto
then obtain M0 where
  M0:  $\bigwedge G. card\ (uverts\ G) > 0 \implies \exists P. regular-partition\ (\varepsilon/4)\ G\ P \wedge card\ P$ 
 $\leq M0$ 
  and  $M0 > 0$ 
  by (metis Szemerédi-Regularity-Lemma le0 neq0-conv not-le not-numeral-le-zero)
define D0 where  $D0 \equiv 1/6 * (1 - (\varepsilon/2)) * ((\varepsilon/4) ^ 3) * ((\varepsilon / (4 * M0)) ^ 3)$ 
have  $D0 > 0$ 
  using  $\langle 0 < \varepsilon \rangle \langle \varepsilon < 1 \rangle \langle M0 > 0 \rangle$  by (simp add: D0-def zero-less-mult-iff)
then obtain  $\delta:: real$  where  $0 < \delta \wedge \delta < D0$ 
  by (meson dense)
show ?thesis
proof (rule exI, intro conjI strip)
  fix G
  assume  $card(uverts\ G) > 0$  and wf: uwellformed G
  then have fin: finite  $(uverts\ G)$ 
    by (simp add: card-gt-0-iff)

  Assume that, for a yet to be determined  $\delta$ , we have:
  assume ineq:  $real\ (card\ (triangle-set\ G)) \leq \delta * card\ (uverts\ G) ^ 3$ 

  Step 1: Partition: Using Szemerédi's Regularity Lemma, we get an  $\varepsilon/4$ 
  partition.
  let  $?n = card\ (uverts\ G)$ 
  have vne:  $uverts\ G \neq \{\}$ 
    using  $\langle 0 < card\ (uverts\ G) \rangle$  by force
  then have ngt0:  $?n > 0$ 
    by (simp add: fin card-gt-0-iff)
  with M0 obtain P where M: regular-partition  $(\varepsilon/4)\ G\ P$  and  $card\ P \leq M0$ 
    by blast
  define M where  $M \equiv card\ P$ 

```

```

have finite P
  by (meson M fin finite-elements regular-partition-def)
with M0 have  $M > 0$ 
  unfolding M-def
  by (metis M card-gt-0-iff partition-onD1 partition-on-empty regular-partition-def
vne)
  let  $?e_4M = \varepsilon / (4 * \text{real } M)$ 
  define D where  $D \equiv 1/6 * (1 - (\varepsilon/2)) * ((\varepsilon/4)^3) * ?e_4M^3$ 
  have  $D > 0$ 
    using  $\langle 0 < \varepsilon \rangle \langle \varepsilon < 1 \rangle \langle M > 0 \rangle$  by (simp add: D-def zero-less-mult-iff)
  have  $D0 \leq D$ 
    unfolding D0-def D-def using  $\langle 0 < \varepsilon \rangle \langle \varepsilon < 1 \rangle \langle \text{card } P \leq M0 \rangle \langle M > 0 \rangle$ 
    by (intro mult-mono) (auto simp: frac-le M-def)
  have fin-part: finite-graph-partition (uverts G) P M
    using M unfolding regular-partition-def finite-graph-partition-def
    by (metis M-def  $\langle 0 < M \rangle$  card-gt-0-iff)
  then have fin-P: finite R and card-P-gt0: card R > 0 if  $R \in P$  for R
    using fin finite-graph-partition-finite finite-graph-partition-gt0 that by auto
  have card-P-le: card R ≤ ?n if  $R \in P$  for R
    by (meson card-mono fin fin-part finite-graph-partition-subset that)
  have P-disjnt:  $\bigwedge R S. \llbracket R \neq S; R \in P; S \in P \rrbracket \implies R \cap S = \{\}$ 
    using fin-part
    by (metis disjnt-def finite-graph-partition-def insert-absorb pairwise-insert
partition-on-def)
  have sum-card-P:  $(\sum R \in P. \text{card } R) = ?n$ 
    using card-finite-graph-partition fin fin-part by meson

```

Step 2. Cleaning. For each ordered pair of parts (P_i, P_j) , remove all edges between P_i and P_j if (a) it is an irregular pair, (b) its edge density $< \varepsilon/2$, (c) either P_i or P_j is small ($\leq (\varepsilon/4M)n$) Process (a) removes at most $(\varepsilon/4)n^2$ edges. Process (b) removes at most $(\varepsilon/2)n^2$ edges. Process (c) removes at most $(\varepsilon/4)n^2$ edges. The remaining graph is triangle-free for some choice of δ .

```

define edge where  $edge \equiv \lambda R S. \text{mk-uedge } \langle \text{all-edges-between } R S G \rangle$ 
have edge-commute: edge R S = edge S R for R S
  by (force simp add: edge-def all-edges-between-swap [of S] split: prod.split)
have card-edge-le-card: card (edge R S) ≤ card (all-edges-between R S G) for
R S
  by (simp add: card-image-le edge-def fin finite-all-edges-between' local.wf)
have card-edge-le: card (edge R S) ≤ card R * card S if  $R \in P$   $S \in P$  for R S
  by (meson card-edge-le-card fin-P le-trans max-all-edges-between that)

```

Obtain the set of edges meeting condition (a).

```

define irreg-pairs where  $irreg-pairs \equiv \{(R,S). R \in P \wedge S \in P \wedge \neg (\varepsilon/4)\text{-regular-pair}$ 
R S G\}

```

```

define Ea where  $Ea \equiv (\bigcup (R,S) \in irreg-pairs. \text{edge } R S)$ 

```

Obtain the set of edges meeting condition (b).

```

define low-density-pairs

```

where *low-density-pairs* $\equiv \{(R,S). R \in P \wedge S \in P \wedge \neg \text{edge-dense } R S G (\varepsilon/2)\}$
define *Eb* **where** *Eb* $\equiv (\bigcup (i,j) \in \text{low-density-pairs}. \text{edge } i j)$
Obtain the set of edges meeting condition (c).
define *small* **where** *small* $\equiv \lambda R. R \in P \wedge \text{card } R \leq ?e4M * ?n$
let *?SMALL* = *Collect small*
define *small-pairs* **where** *small-pairs* $\equiv \{(R,S). R \in P \wedge S \in P \wedge (\text{small } R \vee \text{small } S)\}$
define *Ec* **where** *Ec* $\equiv (\bigcup R \in ?SMALL. \bigcup S \in P. \text{edge } R S)$
have *Ec-def'*: *Ec* = $(\bigcup (i,j) \in \text{small-pairs}. \text{edge } i j)$
by (*force simp: edge-commute small-pairs-def small-def Ec-def*)
have *eabound*: *card Ea* $\leq (\varepsilon/4) * ?n^2$ — Count the edge bound for *Ea*
proof –
have $\S: \bigwedge R S. \llbracket R \in P; S \in P \rrbracket \implies \text{card } (\text{edge } R S) \leq \text{card } R * \text{card } S$
unfolding *edge-def*
by (*meson card-image-le fin-P finite-all-edges-between max-all-edges-between order-trans*)
have *irreg-pairs* $\subseteq P \times P$
by (*auto simp: irreg-pairs-def*)
then have *finite irreg-pairs*
by (*meson <finite P> finite-SigmaI finite-subset*)
have *card Ea* $\leq (\sum (R,S) \in \text{irreg-pairs}. \text{card } (\text{edge } R S))$
by (*simp add: Ea-def card-UN-le [OF <finite irreg-pairs>] case-prod-unfold*)
also have $\dots \leq (\sum (R,S) \in \{(R,S). R \in P \wedge S \in P \wedge \neg (\varepsilon/4)\text{-regular-pair } R S G\}. \text{card } R * \text{card } S)$
unfolding *irreg-pairs-def* **using** \S **by** (*force intro: sum-mono*)
also have $\dots = (\sum (R,S) \in \text{irregular-set } (\varepsilon/4) G P. \text{card } R * \text{card } S)$
by (*simp add: irregular-set-def*)
finally have *card Ea* $\leq (\sum (R,S) \in \text{irregular-set } (\varepsilon/4) G P. \text{card } R * \text{card } S)$.
with *M* **show** *?thesis*
unfolding *regular-partition-def* **by** *linarith*
qed
have *ebbound*: *card Eb* $\leq (\varepsilon/2) * (?n^2)$ — Count the edge bound for *Eb*.
proof –
have $\S: \bigwedge R S. \llbracket R \in P; S \in P; \neg \text{edge-dense } R S G (\varepsilon / 2) \rrbracket$
 $\implies \text{real } (\text{card } (\text{edge } R S)) * 2 \leq \varepsilon * \text{real } (\text{card } R) * \text{real } (\text{card } S)$
by (*simp add: divide-simps edge-dense-def edge-density-def card-P-gt0*)
(smt (verit, best) card-edge-le-card of-nat-le-iff mult.assoc)
have *subs*: *low-density-pairs* $\subseteq P \times P$
by (*auto simp: low-density-pairs-def*)
then have *finite low-density-pairs*
by (*metis <finite P> finite-SigmaI finite-subset*)
have $\text{real } (\text{card } Eb) \leq (\sum (i,j) \in \text{low-density-pairs}. \text{real } (\text{card } (\text{edge } i j)))$
unfolding *Eb-def*
by (*smt (verit, ccfv-SIG) <finite low-density-pairs> card-UN-le of-nat-mono of-nat-sum case-prod-unfold sum-mono*)

also have $\dots \leq (\sum (R,S) \in \text{low-density-pairs}. \varepsilon/2 * \text{card } R * \text{card } S)$
unfolding *low-density-pairs-def* **by** (*force intro: sum-mono §*)
also have $\dots \leq (\sum (R,S) \in P \times P. \varepsilon/2 * \text{card } R * \text{card } S)$
using *subs* $\langle \varepsilon > 0 \rangle$ **by** (*intro sum-mono2*) (*auto simp: finite P*)
also have $\dots = \varepsilon/2 * (\sum (R,S) \in P \times P. \text{card } R * \text{card } S)$
by (*simp add: sum-distrib-left case-prod-unfold mult-ac*)
also have $\dots \leq (\varepsilon/2) * (?n^2)$
using $\langle \varepsilon > 0 \rangle$ *sum-prod-le-prod-sum*
by (*simp add: power2-eq-square sum-product flip: sum.cartesian-product*
sum-card-P)
finally show *?thesis* .
qed
have *ecbound: card Ec* $\leq (\varepsilon/4) * (?n^2)$ — Count the edge bound for *Ec*.
proof –
have *edge-bound: (card (edge R S))* $\leq ?e4M * ?n^2$
if $S \in P$ **small R for R S**
proof –
have *real (card R)* $\leq \varepsilon * ?n / (4 * \text{real } M)$
using *that* **by** (*simp add: small-def*)
with *card-P-le* [*OF* $\langle S \in P \rangle$]
have *: *real (card R) * real (card S)* $\leq \varepsilon * \text{card } (\text{uverts } G) / (4 * \text{real } M)$
* *?n*
by (*meson mult-mono of-nat-0-le-iff of-nat-mono order.trans*)
also have $\dots = ?e4M * ?n^2$
by (*simp add: power2-eq-square*)
finally show *?thesis*
by (*smt (verit) card-edge-le of-nat-mono of-nat-mult small-def that*)
qed
have *subs: ?SMALL* $\subseteq P$
by (*auto simp: small-def*)
then obtain *card-sp: card (?SMALL)* $\leq M$ **and** *finite ?SMALL*
using *M-def* $\langle \text{finite } P \rangle$ *card-mono* **by** (*metis finite-subset*)
have *real (card Ec)* $\leq (\sum R \in ?SMALL. \text{real } (\text{card } (\bigcup S \in P. \text{edge } R S)))$
unfolding *Ec-def*
by (*smt (verit, ccfv-SIG) finite ?SMALL card-UN-le of-nat-mono of-nat-sum*
case-prod-unfold sum-mono)
also have $\dots \leq (\sum R \in ?SMALL. ?e4M * ?n^2)$
proof (*intro sum-mono*)
fix *R* **assume** *i: R* $\in \text{Collect small}$
then have $R \in P$ **and** *card-Pi: card R* $\leq ?e4M * ?n$
by (*auto simp: small-def*)
let *?UE* $= \bigcup (\text{edge } R \text{ ' } (P))$
have *: *real (card ?UE)* $\leq \text{real } (\text{card } R * ?n)$
proof –
have *?UE* $\subseteq \text{mk-uedge ' } (\text{all-edges-between } R (\text{uverts } G) G)$
apply (*simp add: edge-def UN-subset-iff Ball-def*)
by (*meson all-edges-between-mono2 fin-part finite-graph-partition-subset*
image-mono)
then have *card ?UE* $\leq \text{card } (\text{all-edges-between } R (\text{uverts } G) G)$

by (*meson card-image-le card-mono fin finite-all-edges-between' fi-
nite-imageI wf le-trans*)
then show *?thesis*
by (*meson of-nat-mono fin fin-P max-all-edges-between order.trans <R∈P>*)
qed
also have $\dots \leq ?e_4M * \text{real } (?n^2)$
using *card-Pi <M > 0> <?n > 0>* **by** (*force simp add: divide-simps
power2-eq-square*)
finally show $\text{real } (\text{card } ?UE) \leq ?e_4M * \text{real } (?n^2)$.
qed
also have $\dots \leq \text{card } ?SMALL * (?e_4M * ?n^2)$
by *simp*
also have $\dots \leq M * (?e_4M * ?n^2)$
using *egt by (intro mult-right-mono) (auto simp add: card-sp)*
also have $\dots \leq (\varepsilon/4) * (?n^2)$
using *<M > 0>* **by** *simp*
finally show *?thesis* .
qed
— total count
have *prev1: card (Ea ∪ Eb ∪ Ec) ≤ card (Ea ∪ Eb) + card Ec* **by** (*simp add:
card-Un-le*)
also have $\dots \leq \text{card } Ea + \text{card } Eb + \text{card } Ec$ **by** (*simp add: card-Un-le*)
also have *prev: ... ≤ (ε/4)*(?n^2) + (ε/2)*(?n^2) + (ε/4)*(?n^2)*
using *eabound ebbound ecbound* **by** *linarith*
finally have *cutedgesbound: card (Ea ∪ Eb ∪ Ec) ≤ ε * (?n^2)* **by** *simp*

define *Gnew* **where** $Gnew \equiv (\text{uverts } G, \text{uedges } G - (Ea \cup Eb \cup Ec))$
show $\exists Gnew. ?\Phi G Gnew$
proof (*intro exI conjI*)
show *verts: uverts Gnew = uverts G* **by** (*simp add: Gnew-def*)
have *difffedges: (Ea ∪ Eb ∪ Ec) ⊆ uedges G*
by (*auto simp: Ea-def Eb-def Ec-def all-edges-between-def edge-def*)
then show *edges: uedges Gnew ⊆ uedges G*
by (*simp add: Gnew-def*)
then have $\text{uedges } G - (\text{uedges } Gnew) = \text{uedges } G \cap (Ea \cup Eb \cup Ec)$
by (*simp add: Gnew-def Diff-Diff-Int*)
then have $\text{uedges } G - (\text{uedges } Gnew) = (Ea \cup Eb \cup Ec)$ **using** *difffedges*
by (*simp add: Int-absorb1*)
then have *cardbound: card (uedges G - uedges Gnew) ≤ ε * (?n^2)*
using *cutedgesbound* **by** *simp*
have *graph-partition-new: finite-graph-partition (uverts Gnew) P M* **using**
verts
by (*simp add: fin-part*)
have *new-wf: uwellformed Gnew* **using** *subgraph-edge-wf verts edges wf* **by**
simp
have *new-fin: finite (uverts Gnew)* **using** *verts fin* **by** *simp*

The notes by Bell and Grodzicki are quite useful for understanding the lines below. See pg 4 in the middle after the summary of the min edge counts.

```

have irreg-pairs-swap:  $(R,S) \in \text{irreg-pairs} \longleftrightarrow (S,R) \in \text{irreg-pairs}$  for  $R S$ 
  by (auto simp: irreg-pairs-def regular-pair-commute)
have low-density-pairs-swap:  $(R,S) \in \text{low-density-pairs} \longleftrightarrow (S,R) \in \text{low-density-pairs}$ 
for  $R S$ 
  by (simp add: low-density-pairs-def edge-density-commute edge-dense-def)
    (use all-edges-between-swap in blast)
have small-pairs-swap:  $(R,S) \in \text{small-pairs} \longleftrightarrow (S,R) \in \text{small-pairs}$  for  $R S$ 
  by (auto simp: small-pairs-def)

have all-edges-if:
  all-edges-between  $R S G_{\text{new}}$ 
  = (if  $(R,S) \in \text{irreg-pairs} \cup \text{low-density-pairs} \cup \text{small-pairs}$  then {}
    else all-edges-between  $R S G$ )
  (is ?lhs = ?rhs)
  if  $ij: R \in P S \in P$  for  $R S$ 
proof
  show ?lhs  $\subseteq$  ?rhs
    using that fin-part unfolding  $G_{\text{new-def}} Ea\text{-def} Eb\text{-def} Ec\text{-def}'$ 
  apply (simp add: all-edges-between-E-diff all-edges-between-E-Un all-edges-between-E-UN)
  apply (auto simp: edge-def in-mk-uedge-img-iff all-edges-between-def)
  done
next
  have  $Ea: \text{all-edges-between } R S (V, Ea) = \{\}$ 
  if  $(R,S) \notin \text{irreg-pairs}$  for  $V$ 
  using  $ij$  that  $P\text{-disjnt}$ 
  by (auto simp:  $Ea\text{-def}$  doubleton-eq-iff edge-def all-edges-between-def irreg-pairs-def;
   metis regular-pair-commute disjoint-iff-not-equal)
  have  $Eb: \text{all-edges-between } R S (V, Eb) = \{\}$ 
  if  $(R,S) \notin \text{low-density-pairs}$  for  $V$ 
  using  $ij$  that
  apply (auto simp:  $Eb\text{-def}$  edge-def all-edges-between-def low-density-pairs-def edge-dense-def)
  apply metis
  by (metis IntI  $P\text{-disjnt}$  doubleton-eq-iff edge-density-commute equals0D)
  have  $Ec: \text{all-edges-between } R S (V, Ec) = \{\}$ 
  if  $(R,S) \notin \text{small-pairs}$  for  $V$ 
  using  $ij$  that
  by (auto simp:  $Ec\text{-def}'$  doubleton-eq-iff edge-def all-edges-between-def small-pairs-def;
   metis  $P\text{-disjnt}$  disjoint-iff)
  show ?rhs  $\subseteq$  ?lhs
  by (auto simp add:  $G_{\text{new-def}} Ea Eb Ec$  all-edges-between-E-diff all-edges-between-E-Un)
qed

have  $rp: (\varepsilon/4)\text{-regular-pair } R S G_{\text{new}}$  if  $ij: R \in P S \in P$  for  $R S$ 
proof (cases  $(R,S) \in \text{irreg-pairs}$ )
  case False
  have  $ed: \text{edge-density } X Y G_{\text{new}} =$ 

```

```

      (if (R,S) ∈ irreg-pairs ∪ low-density-pairs ∪ small-pairs then 0
        else edge-density X Y G)
    if X ⊆ R Y ⊆ S for X Y
    using all-edges-if that ij False
    by (smt (verit) all-edges-preserved edge-density-eq0 edge-density-def)
  show ?thesis
    using that False ⟨ε > 0⟩
    by (auto simp add: irreg-pairs-def regular-pair-def less-le ed)
next
case True
then have ed: edge-density X Y Gnew = 0 if X ⊆ R Y ⊆ S for X Y
  by (meson edge-density-eq0 all-edges-if that ⟨R ∈ P⟩ ⟨S ∈ P⟩ UnCI)
with egt that show ?thesis
  by (auto simp: regular-pair-def ed)
qed
then have reg-pairs: (ε/4)-regular-graph P Gnew
  by (meson regular-graph-def)

have edge-dense R S Gnew (ε/2)
  if R ∈ P S ∈ P for R S
proof (cases (R,S) ∈ low-density-pairs)
case False
have ed: edge-density R S Gnew =
  (if (R,S) ∈ irreg-pairs ∪ low-density-pairs ∪ small-pairs then 0
    else edge-density R S G)
  using all-edges-if that that by (simp add: edge-density-def)
with that ⟨ε > 0⟩ False show ?thesis
  by (auto simp: low-density-pairs-def edge-dense-def all-edges-if)
next
case True
then have edge-density R S Gnew = 0
  by (simp add: all-edges-if edge-density-def that)
with ⟨ε > 0⟩ that show ?thesis
  by (simp add: True all-edges-if edge-dense-def)
qed
then have density-bound: dense-graph P Gnew (ε/2)
  by (meson dense-graph-def)

have min-subset-size: decent-graph P Gnew (?e4M * ?n)
  using ⟨ε > 0⟩
by (auto simp: decent-graph-def small-pairs-def small-def decent-def all-edges-if)
show triangle-free-graph Gnew
proof (rule ccontr)
  assume non: ¬?thesis
  then obtain x y z where trig-ex: triangle-in-graph x y z Gnew
    using triangle-free-graph-def non by auto
  then have xin: x ∈ (uverts Gnew) and yin: y ∈ (uverts Gnew) and zin: z
    ∈ (uverts Gnew)
    using triangle-in-graph-verts new-wf by auto

```

then obtain $R S T$ **where** $xinp: x \in R$ **and** $ilt: R \in P$ **and** $yinp: y \in S$ **and**
 $jlt: S \in P$

and $zinp: z \in T$ **and** $klt: T \in P$

by (*metis graph-partition-new xin Union-iff finite-graph-partition-equals*)

then have $finitesubsets: finite R finite S finite T$

using *new-fin fin-part finite-graph-partition-finite fin* **by** *auto*

have $subsets: R \subseteq uverts Gnew S \subseteq uverts Gnew T \subseteq uverts Gnew$

using *finite-graph-partition-subset ilt jlt klt graph-partition-new* **by** *auto*

have $min-sizes: card R \geq ?e4M * ?n card S \geq ?e4M * ?n card T \geq ?e4M * ?n$

using *trig-ex min-subset-size xinp yinp zinp ilt jlt klt*

by (*auto simp: triangle-in-graph-def decent-graph-def decent-def all-edges-between-def*)

have $min-dens: edge-density R S Gnew \geq \varepsilon/2 edge-density R T Gnew \geq$
 $\varepsilon/2 edge-density S T Gnew \geq \varepsilon/2$

using *density-bound ilt jlt klt xinp yinp zinp trig-ex*

by (*auto simp: dense-graph-def edge-dense-def all-edges-between-def triangle-in-graph-def*)

then have $min-dens-diff:$
 $edge-density R S Gnew - \varepsilon/4 \geq \varepsilon/4 edge-density R T Gnew - \varepsilon/4 \geq \varepsilon/4$
 $edge-density S T Gnew - \varepsilon/4 \geq \varepsilon/4$

by *auto*

have $mincard0: (card R) * (card S) * (card T) \geq 0$ **by** *simp*

have $gtcube: ((edge-density R S Gnew) - \varepsilon/4) * ((edge-density R T Gnew) - \varepsilon/4) * ((edge-density S T Gnew) - \varepsilon/4) \geq (\varepsilon/4)^3$

using *min-dens-diff e4gt real-mult-gt-cube* **by** *auto*

then have $c1: ((edge-density R S Gnew) - \varepsilon/4) * ((edge-density R T Gnew) - \varepsilon/4) * ((edge-density S T Gnew) - \varepsilon/4) \geq 0$

by (*smt (verit) e4gt zero-less-power*)

have $?e4M * ?n \geq 0$

using *egt by force*

then have $card R * card S * card T \geq (?e4M * ?n)^3$

by (*metis min-sizes of-nat-mult real-mult-gt-cube*)

then have $cardgtbound: card R * card S * card T \geq ?e4M^3 * ?n^3$

by (*metis of-nat-power power-mult-distrib*)

have $(1 - \varepsilon/2) * (\varepsilon/4)^3 * (\varepsilon/(4 * M))^3 * ?n^3 \leq (1 - \varepsilon/2) * (\varepsilon/4)^3 * card R * card S * card T$

using *cardgtbound ordered-comm-semiring-class.comm-mult-left-mono True e4gt by fastforce*

also have $... \leq (1 - 2 * (\varepsilon/4)) * (edge-density R S Gnew - \varepsilon/4) * (edge-density R T Gnew - \varepsilon/4) * (edge-density S T Gnew - \varepsilon/4) * card R * card S * card T$

using *gtcube c1 <\varepsilon < 1> mincard0* **by** (*simp add: mult.commute mult.left-commute mult-left-mono*)

also have $... \leq card (triangle-triples R S T Gnew)$

by (*smt (verit, best) e4gt ilt jlt klt min-dens-diff new-fin new-wf rp subsets triangle-counting-lemma*)

finally have $card (triangle-set Gnew) \geq D * ?n^3$

using *card-convert-triangle-rep-bound new-wf new-fin subsets*

by (*auto simp: triangle-triples-def D-def*)

```

then have g-tset-bound:  $\text{card} (\text{triangle-set } G) \geq D * ?n^3$ 
  using triangle-set-graph-edge-ss-bound by (smt (verit) edges fin local.wf
of-nat-mono verts)
have  $\text{card} (\text{triangle-set } G) > \delta * ?n^3$ 
proof –
  have  $?n^3 > 0$ 
  by (simp add: ⟨uverts G ≠ {}⟩ card-gt-0-iff fin)
with  $\delta \langle D0 \leq D \rangle$  have  $D * ?n^3 > \delta * ?n^3$ 
  by force
thus  $\text{card} (\text{triangle-set } G) > \delta * ?n^3$ 
  using g-tset-bound unfolding D-def by linarith
qed
thus False
  using ineq by linarith
qed
show  $\text{real} (\text{card} (\text{uedges } G - \text{uedges } G_{\text{new}})) \leq \varepsilon * \text{real} ((\text{card} (\text{uverts } G))^2)$ 
  using cardbound edges verts by blast
qed
qed (rule ⟨0 < δ⟩)
qed

```

1.5 Roth's Theorem

We will first need the following corollary of the Triangle Removal Lemma.

See https://en.wikipedia.org/wiki/Ruzsa--Szemerédi_problem. Suggested by Yaël Dillies

corollary *Diamond-free*:

```

fixes  $\varepsilon :: \text{real}$ 
assumes  $0 < \varepsilon$ 
shows  $\exists N > 0. \forall G. \text{card}(\text{uverts } G) > N \longrightarrow \text{uwellformed } G \longrightarrow \text{unique-triangles } G \longrightarrow$ 
   $\text{card} (\text{uedges } G) \leq \varepsilon * (\text{card} (\text{uverts } G))^2$ 
proof –
  have  $\varepsilon/3 > 0$ 
  using assms by auto
then obtain  $\delta :: \text{real}$  where  $\delta > 0$ 
  and  $\delta: \bigwedge G. [\text{card}(\text{uverts } G) > 0; \text{uwellformed } G; \text{card} (\text{triangle-set } G) \leq \delta * \text{card}(\text{uverts } G)^3]$ 
   $\implies \exists G'. \text{triangle-free-graph } G' \wedge \text{uverts } G' = \text{uverts } G \wedge (\text{uedges } G' \subseteq \text{uedges } G) \wedge$ 
   $\text{card} (\text{uedges } G - \text{uedges } G') \leq \varepsilon/3 * (\text{card} (\text{uverts } G))^2$ 
  using triangle-removal-lemma by metis
obtain  $N :: \text{nat}$  where  $N: \text{real } N \geq 1 / (3*\delta)$ 
  by (meson real-arch-simple)
show ?thesis
proof (intro exI conjI strip)
  show  $N > 0$ 
  using  $N \langle 0 < \delta \rangle$  zero-less-iff-neq-zero by fastforce

```

```

fix G
let ?n = card (uverts G)
assume G-gt-N: N < ?n
  and wf: uwellformed G
  and uniq: unique-triangles G
have G-ne: ?n > 0
  using G-gt-N by linarith
let ?TWO = (λt. [t]2)
have tri-nsets-2: [{x,y,z}]2 = {{x,y},{y,z},{x,z}} if triangle-in-graph x y z G
for x y z
  using that unfolding nsets-def triangle-in-graph-def card-2-iff doubleton-eq-iff
  by (blast dest!: edge-vertices-not-equal [OF wf])
have tri-nsets-3: {{x,y},{y,z},{x,z}} ∈ [uedges G]3 if triangle-in-graph x y z G
for x y z
  using that by (simp add: nsets-def card-3-iff triangle-in-graph-def)
  (metis doubleton-eq-iff edge-vertices-not-equal [OF wf])
have sub: ?TWO ‘ triangle-set G ⊆ [uedges G]3
  using tri-nsets-2 tri-nsets-3 triangle-set-def by auto
have ∧i. i ∈ triangle-set G ⇒ ?TWO i ≠ {}
  using tri-nsets-2 triangle-set-def by auto
moreover have dfam: disjoint-family-on ?TWO (triangle-set G)
  using sub [unfolded image-subset-iff] uniq
  unfolding disjoint-family-on-def triangle-set-def nsets-def unique-triangles-def
  by (smt (verit) disjoint-iff-not-equal insert-subset mem-Collect-eq mk-disjoint-insert
)
ultimately have inj: inj-on ?TWO (triangle-set G)
  by (simp add: disjoint-family-on-iff-disjoint-image)
have §: ∃ T ∈ triangle-set G. e ∈ [T]2 if e ∈ uedges G for e
  using uniq [unfolded unique-triangles-def] that local.wf
apply (simp add: triangle-set-def triangle-in-graph-def nsets-def uwellformed-def)
  by (metis (mono-tags, lifting) finite.emptyI finite.insertI finite-subset)
with sub have ∪(?TWO ‘ triangle-set G) = uedges G
  by (auto simp: image-subset-iff nsets-def)
then have card (∪(?TWO ‘ triangle-set G)) = card (uedges G)
  by simp
moreover have card (∪(?TWO ‘ triangle-set G)) = 3 * card (triangle-set G)
proof (subst card-UN-disjoint! [OF dfam])
  show finite ([i]2) if i ∈ triangle-set G for i
    using that tri-nsets-2 triangle-set-def by fastforce
  show finite (triangle-set G)
    by (meson G-ne card-gt-0-iff local.wf finite-triangle-set)
  have card ([i]2) = 3 if i ∈ triangle-set G for i
    using that wf tri-nsets-2 tri-nsets-3 by (force simp add: nsets-def trian-
gle-set-def)
  then show (∑ i ∈ triangle-set G. card ([i]2)) = 3 * card (triangle-set G)
    by simp
qed
ultimately have 3: 3 * card (triangle-set G) = card (uedges G)
  by auto

```

have $\text{card}(\text{uedges } G) \leq \text{card}(\text{all-edges}(\text{uverts } G))$
by (*meson G-ne all-edges-finite card-gt-0-iff card-mono local.wf wellformed-all-edges*)
also have $\dots = ?n$ *choose 2*
by (*metis G-ne card-all-edges card-eq-0-iff not-less0*)
also have $\dots \leq ?n^2$
by (*metis binomial-eq-0-iff binomial-le-pow linorder-not-le zero-le*)
finally have $\text{card}(\text{uedges } G) \leq ?n^2$.
with β **have** $\text{card}(\text{triangle-set } G) \leq ?n^2 / \beta$ **by** *linarith*
also have $\dots \leq \delta * ?n^{\wedge} \beta$
proof –
have $1 \leq \beta * \delta * N$
using $N \langle \delta > 0 \rangle$ **by** (*simp add: field-simps*)
also have $\dots \leq \beta * \delta * ?n$
using $G\text{-gt-}N \langle 0 < \delta \rangle$ **by** *force*
finally have $1 * ?n^{\wedge} 2 \leq (\beta * \delta * ?n) * ?n^{\wedge} 2$
by (*simp add: G-ne*)
then show *?thesis*
by (*simp add: eval-nat-numeral mult-ac*)
qed
finally have $\text{card}(\text{triangle-set } G) \leq \delta * ?n^{\wedge} \beta$.
then obtain G_{new} **where** G_{new} : *triangle-free-graph Gnew uverts Gnew =*
uverts G
 $\text{uedges } G_{\text{new}} \subseteq \text{uedges } G$ **and** *card-edge-diff: card(uedges G – uedges Gnew)*
 $\leq \varepsilon / \beta * ?n^2$
using $G\text{-ne } \delta$ *local.wf* **by** *meson*

Deleting an edge removes at most one triangle from the graph by assumption, so the number of edges removed in this process is at least the number of triangles.

obtain TF **where** $TF: \bigwedge e. e \in \text{uedges } G \implies \exists x y z. TF\ e = \{x,y,z\} \wedge$
triangle-in-graph x y z G $\wedge e \subseteq TF\ e$
using *uniq unfolding unique-triangles-def* **by** *metis*
have *False*
if $\text{non}: \bigwedge e. e \in \text{uedges } G - \text{uedges } G_{\text{new}} \implies \{x,y,z\} \neq TF\ e$
and $\text{tri}: \text{triangle-in-graph } x\ y\ z\ G$ **for** $x\ y\ z$
proof –
have $\neg \text{triangle-in-graph } x\ y\ z\ G_{\text{new}}$
using G_{new} *triangle-free-graph-def* **by** *blast*
with tri **obtain** e **where** $eG: e \in \text{uedges } G - \text{uedges } G_{\text{new}}$ **and** $e_{\text{sub}}: e \subseteq$
 $\{x,y,z\}$
using *insert-commute triangle-in-graph-def* **by** *auto*
then show *False*
by (*metis DiffD1 TF tri uniq unique-triangles-def non [OF eG]*)
qed
then have $\text{triangle-set } G \subseteq TF\ '(\text{uedges } G - \text{uedges } G_{\text{new}})$
unfolding *triangle-set-def* **by** *blast*
moreover have *finite(uedges G – uedges Gnew)*
by (*meson G-ne card-gt-0-iff finite-Diff finite-graph-def wf wellformed-finite*)
ultimately have $\text{card}(\text{triangle-set } G) \leq \text{card}(\text{uedges } G - \text{uedges } G_{\text{new}})$

by (meson surj-card-le)
 then show $\text{card}(\text{uedges } G) \leq \varepsilon * ?n^2$
 using 3 card-edge-diff by linarith
 qed
 qed

We are now ready to proceed to the proof of Roth's Theorem for Arithmetic Progressions.

definition *progression3* :: 'a::comm-monoid-add \Rightarrow 'a \Rightarrow 'a set
 where *progression3* k $d \equiv \{k, k+d, k+d+d\}$

lemma *p3-int-iff*: *progression3* (int k) (int d) \subseteq int ' $A \iff$ *progression3* k $d \subseteq A$

apply (simp add: *progression3-def image-iff*)
 by (smt (verit, best) int-plus of-nat-eq-iff)

We assume that a set of naturals $A \subseteq \{\dots < N\}$ does not have any arithmetic progression. We will then show that A is of cardinality $o(N)$.

lemma *RothArithmeticProgressions-aux*:

fixes $\varepsilon::\text{real}$

assumes $\varepsilon > 0$

obtains M where $\forall N \geq M. \forall A \subseteq \{\dots < N\}. (\nexists k d. d > 0 \wedge \text{progression3 } k d \subseteq A) \longrightarrow \text{card } A < \varepsilon * \text{real } N$

proof –

obtain L where $L > 0$

and $L: \bigwedge G. [\text{card}(\text{uverts } G) > L; \text{uwellformed } G; \text{unique-triangles } G] \implies \text{card}(\text{uedges } G) \leq \varepsilon/12 * (\text{card}(\text{uverts } G))^2$

by (metis assms Diamond-free less-divide-eq-numeral1(1) mult-eq-0-iff)

show *thesis*

proof (intro strip that)

fix $N A$

assume $L \leq N$ and $A: A \subseteq \{\dots < N\}$

and non: $\nexists k d. 0 < d \wedge \text{progression3 } k d \subseteq A$

then have $N > 0$ using $\langle 0 < L \rangle$ by linarith

define M where $M \equiv \text{Suc } (2*N)$

have *M-mod-bound*[simp]: $x \bmod M < M$ for x

by (simp add: *M-def*)

have *odd M M > 0 N < M* by (auto simp: *M-def*)

have *coprime M (Suc N)*

unfolding *M-def*

by (metis add-2-eq-Suc coprime-Suc-right-nat coprime-mult-right-iff mult-Suc-right)

then have *cop*: *coprime M (1 + int N)*

by (metis coprime-int-iff of-nat-Suc)

have *A-sub-M*: $\text{int } ' A \subseteq \{\dots < M\}$

using A by (force simp: *M-def*)

have *non-img-A*: $\nexists k d. d > 0 \wedge \text{progression3 } k d \subseteq \text{int } ' A$

by (metis imageE insert-subset non p3-int-iff pos-int-cases *progression3-def*)

Construct a tripartite graph G whose three parts are copies of $\mathbb{Z}/M\mathbb{Z}$.


```

define part-of where part-of  $\equiv \lambda \xi. (\lambda i. \text{prod-encode } (\xi, i)) \text{ '}\{..<M\}$ 
define label-of-part where label-of-part  $\equiv \lambda p. \text{fst } (\text{prod-decode } p)$ 
define from-part where from-part  $\equiv \lambda p. \text{snd } (\text{prod-decode } p)$ 
have enc-iff [simp]:  $\text{prod-encode } (a, i) \in \text{part-of } a' \iff a'=a \wedge i < M$  for  $a \ a' \ i$ 
  using  $\langle 0 < M \rangle$  by (clarsimp simp: part-of-def image-iff Bex-def) presburger
have part-of-M:  $p \in \text{part-of } a \implies \text{from-part } p < M$  for  $a \ p$ 
  using from-part-def part-of-def by fastforce
have disjnt-part-of:  $a \neq b \implies \text{disjnt } (\text{part-of } a) (\text{part-of } b)$  for  $a \ b$ 
  by (auto simp: part-of-def disjnt-iff)
have from-enc [simp]:  $\text{from-part } (\text{prod-encode } (a, i)) = i$  for  $a \ i$ 
  by (simp add: from-part-def)
have finpart [iff]:  $\text{finite } (\text{part-of } a)$  for  $a$ 
  by (simp add: part-of-def \langle 0 < M \rangle)
have cardpart [simp]:  $\text{card } (\text{part-of } a) = M$  for  $a$ 
  using  $\langle 0 < M \rangle$ 
  by (simp add: part-of-def eq-nat-nat-iff inj-on-def card-image)
let  $?X = \text{part-of } 0$ 
let  $?Y = \text{part-of } (\text{Suc } 0)$ 
let  $?Z = \text{part-of } (\text{Suc } (\text{Suc } 0))$ 
define diff where diff  $\equiv \lambda a \ b. (\text{int } a - \text{int } b) \bmod (\text{int } M)$ 
have inj-on-diff:  $\text{inj-on } (\lambda x. \text{diff } x \ a) \{..<M\}$  for  $a$ 
  apply (clarsimp simp: diff-def inj-on-def)
  by (metis diff-add-cancel mod-add-left-eq mod-less nat-int of-nat-mod)
have eq-mod-M:  $(x - y) \bmod \text{int } M = (x' - y) \bmod \text{int } M \implies x \bmod \text{int } M = x' \bmod \text{int } M$  for  $x \ x' \ y$ 
  by (simp add: mod-eq-dvd-iff)

have diff-invert:  $\text{diff } y \ x = \text{int } a \iff y = (x + a) \bmod M$  if  $y < M \ a \in A$  for
 $x \ y \ a$ 
proof –
  have  $a < M$ 
    using  $A \ \langle N < M \rangle$  that by auto
  show ?thesis
proof
  assume  $\text{diff } y \ x = \text{int } a$ 
  with that  $\langle a < M \rangle$  have  $\text{int } y = \text{int } (x+a) \bmod \text{int } M$ 
    by (smt (verit) diff-def eq-mod-M mod-less of-nat-add zmod-int)
  with that show  $y = (x + a) \bmod M$ 
    by (metis nat-int zmod-int)
qed (simp add: \langle a < M \rangle diff-def mod-diff-left-eq zmod-int)
qed

define diff2 where diff2  $\equiv \lambda a \ b. ((\text{int } a - \text{int } b) * \text{int}(\text{Suc } N)) \bmod (\text{int } M)$ 
have inj-on-diff2:  $\text{inj-on } (\lambda x. \text{diff2 } x \ a) \{..<M\}$  for  $a$ 
  apply (clarsimp simp: diff2-def inj-on-def)
  by (metis eq-mod-M mult-mod-cancel-right [OF - cop] int-int-eq mod-less zmod-int)
have [simp]:  $(1 + \text{int } N) \bmod \text{int } M = 1 + \text{int } N$ 
  using M-def  $\langle 0 < N \rangle$  by auto

```

```

have diff2-by2: (diff2 a b * 2) mod M = diff a b for a b
proof -
  have int M dvd ((int a - int b) * int M)
  by simp
  then have int M dvd ((int a - int b) * int (Suc N) * 2 - (int a - int b))
  by (auto simp: M-def algebra-simps)
  then show ?thesis
  by (metis diff2-def diff-def mod-eq-dvd-iff mod-mult-left-eq)
qed
have diff2-invert: diff2 (((x + a) mod M + a) mod M) x = int a if a ∈ A for
x a
proof -
  have 1: ((x + a) mod M + a) mod M = (x + 2*a) mod M
  by (metis group-cancel.add1 mod-add-left-eq mult-2)
  have (int ((x + 2*a) mod M) - int x) * (1 + int N) mod int M
  = (int (x + 2*a) - int x) * (1 + int N) mod int M
  by (metis mod-diff-left-eq mod-mult-cong of-nat-mod)
  also have ... = int (a * (Suc M)) mod int M
  by (simp add: algebra-simps M-def)
  also have ... = int a mod int M
  by simp
  also have ... = int a
  using A M-def subsetD that by auto
  finally show ?thesis
  using that by (auto simp: 1 diff2-def)
qed

define Edges where Edges ≡ λX Y df. {{x,y} | x y. x ∈ X ∧ y ∈ Y ∧
df(from-part y) (from-part x) ∈ int ' A}
have Edges-subset: Edges X Y df ⊆ Pow (X ∪ Y) for X Y df
by (auto simp: Edges-def)
define XY where XY ≡ Edges ?X ?Y diff
define YZ where YZ ≡ Edges ?Y ?Z diff
define XZ where XZ ≡ Edges ?X ?Z diff2
obtain [simp]: finite XY finite YZ finite XZ
using Edges-subset unfolding XY-def YZ-def XZ-def
by (metis finite-Pow-iff finite-UnI finite-subset finpart)
define G where G ≡ (?X ∪ ?Y ∪ ?Z, XY ∪ YZ ∪ XZ)
have finG: finite (uverts G) and cardG: card (uverts G) = 3*M
by (simp-all add: G-def card-Un-disjnt disjnt-part-of)
then have card(uverts G) > L
using M-def ⟨L ≤ N⟩ by linarith
have uwellformed G
by (fastforce simp: card-insert-if part-of-def G-def XY-def YZ-def XZ-def
Edges-def uwellformed-def)
have [simp]: {prod-encode (ξ,x), prod-encode (ξ,y)} ∉ XY
{prod-encode (ξ,x), prod-encode (ξ,y)} ∉ YZ
{prod-encode (ξ,x), prod-encode (ξ,y)} ∉ XZ for x y ξ
by (auto simp: XY-def YZ-def XZ-def Edges-def doubleton-eq-iff)

```

have *label-ne-XY* [*simp*]: *label-of-part p* \neq *label-of-part q* **if** $\{p,q\} \in XY$ **for** p
 q
using that by (*auto simp add: XY-def part-of-def Edges-def doubleton-eq-iff label-of-part-def*)
then have [*simp*]: $\{p\} \notin XY$ **for** p
by (*metis insert-absorb2*)
have *label-ne-YZ* [*simp*]: *label-of-part p* \neq *label-of-part q* **if** $\{p,q\} \in YZ$ **for** p q
using that by (*auto simp add: YZ-def part-of-def Edges-def doubleton-eq-iff label-of-part-def*)
then have [*simp*]: $\{p\} \notin YZ$ **for** p
by (*metis insert-absorb2*)
have *label-ne-XZ* [*simp*]: *label-of-part p* \neq *label-of-part q* **if** $\{p,q\} \in XZ$ **for** p q
using that by (*auto simp add: XZ-def part-of-def Edges-def doubleton-eq-iff label-of-part-def*)
then have [*simp*]: $\{p\} \notin XZ$ **for** p
by (*metis insert-absorb2*)
have *label012*: *label-of-part v* $<$ 3 **if** $v \in \text{uverts } G$ **for** v
using that by (*auto simp add: G-def eval-nat-numeral part-of-def label-of-part-def*)

have *Edges-distinct*: $\bigwedge p q r \xi \zeta \gamma \beta$ *df df'*. $\llbracket \{p,q\} \in \text{Edges (part-of } \xi) \text{ (part-of } \zeta) \text{ df};$
 $\{q,r\} \in \text{Edges (part-of } \xi) \text{ (part-of } \zeta) \text{ df};$
 $\{p,r\} \in \text{Edges (part-of } \gamma) \text{ (part-of } \beta) \text{ df}'; \xi \neq \zeta; \gamma \neq \beta \rrbracket \implies \text{False}$
apply (*auto simp: disjnt-iff Edges-def doubleton-eq-iff conj-disj-distribR ex-disj-distrib*)
apply (*metis disjnt-iff disjnt-part-of*)+
done

have *uniq*: $\exists i < M. \exists d \in A. \exists x \in \{p,q,r\}. \exists y \in \{p,q,r\}. \exists z \in \{p,q,r\}.$
 $x = \text{prod-encode}(0, i)$
 $\wedge y = \text{prod-encode}(1, (i+d) \bmod M)$
 $\wedge z = \text{prod-encode}(2, (i+d+d) \bmod M)$
if T : *triangle-in-graph p q r G* **for** $p q r$
proof –
obtain $x y z$ **where** $xy: \{x,y\} \in XY$ **and** $yz: \{y,z\} \in YZ$ **and** $xz: \{x,z\} \in XZ$
and $x: x \in \{p,q,r\}$ **and** $y: y \in \{p,q,r\}$ **and** $z: z \in \{p,q,r\}$
using T **apply** (*simp add: triangle-in-graph-def G-def XY-def YZ-def XZ-def*)
by (*smt (verit, ccfv-SIG) Edges-distinct Zero-not-Suc insert-commute n-not-Suc-n*)
then have $x \in ?X$ $y \in ?Y$ $z \in ?Z$
by (*auto simp: XY-def YZ-def XZ-def Edges-def doubleton-eq-iff; metis disjnt-iff disjnt-part-of*)+
then obtain $i j k$ **where** $i: x = \text{prod-encode}(0, i)$ **and** $j: y = \text{prod-encode}(1, j)$

and $k: z = \text{prod-encode}(2, k)$
by (*metis One-nat-def Suc-1 enc-iff prod-decode-aux.cases prod-decode-inverse*)
obtain $a1$ **where** $a1 \in A$ **and** $a1: (int j - int i) \bmod int M = int a1$
using $xy \langle x \in ?X \rangle i j$ **by** (*auto simp add: XY-def Edges-def doubleton-eq-iff diff-def*)

```

obtain  $a3$  where  $a3 \in A$  and  $a3: (int\ k - int\ j) \bmod\ int\ M = int\ a3$ 
using  $yz \langle x \in ?X \rangle j\ k$  by (auto simp add: YZ-def Edges-def doubleton-eq-iff
diff-def)
obtain  $a2$  where  $a2 \in A$  and  $a2: (int\ k - int\ i) \bmod\ int\ M = int\ (a2 * 2)$ 
mod int M
using  $xz \langle x \in ?X \rangle i\ k$  apply (auto simp add: XZ-def Edges-def double-
ton-eq-iff)
by (metis diff2-by2 diff-def int-plus mult-2-right)
obtain  $a1 < N$   $a2 < N$   $a3 < N$ 
using  $A \langle a1 \in A \rangle \langle a2 \in A \rangle \langle a3 \in A \rangle$  by blast
then obtain  $a1 + a3 < M$   $a2 * 2 < M$ 
by (simp add: M-def)
then have  $int\ (a2 * 2) = int\ (a2 * 2) \bmod\ M$ 
by force
also have  $\dots = int\ (a1 + a3) \bmod\ int\ M$ 
using  $a1\ a2\ a3$  by (smt (verit, del-insts) int-plus mod-add-eq)
also have  $\dots = int\ (a1 + a3)$ 
using  $\langle a1 + a3 < M \rangle$  by force
finally have  $a2 * 2 = a1 + a3$ 
by presburger
then obtain equal:  $a3 - a2 = a2 - a1$   $a2 - a3 = a1 - a2$ 
by (metis Nat.diff-cancel diff-cancel2 mult-2-right)
with  $\langle a1 \in A \rangle \langle a2 \in A \rangle \langle a3 \in A \rangle$  have progression3  $a1\ (a2 - a1) \subseteq A$ 
apply (clarsimp simp: progression3-def)
by (metis diff-is-0-eq' le-add-diff-inverse nle-le)
with non equal have  $a2 = a1$ 
unfolding progression3-def
by (metis \langle a2 \in A \rangle \langle a3 \in A \rangle add.right-neutral diff-is-0-eq insert-subset
le-add-diff-inverse not-gr-zero)
then have  $a3 = a2$ 
using  $\langle a2 * 2 = a1 + a3 \rangle$  by force
have k-minus-j:  $(int\ k - int\ j) \bmod\ int\ M = int\ a1$ 
by (simp add: \langle a2 = a1 \rangle \langle a3 = a2 \rangle a3)
have i-to-j:  $j \bmod\ M = (i + a1) \bmod\ M$ 
by (metis a1 add-diff-cancel-left' add-diff-eq mod-add-right-eq nat-int of-nat-add
of-nat-mod)
have j-to-k:  $k \bmod\ M = (j + a1) \bmod\ M$ 
by (metis \langle a2 = a1 \rangle \langle a3 = a2 \rangle a3 add-diff-cancel-left' add-diff-eq mod-add-right-eq
nat-int of-nat-add of-nat-mod)
have  $i < M$ 
using  $\langle x \in ?X \rangle i$  by simp
then show ?thesis
using  $i\ j\ k\ x\ y\ z \langle a1 \in A \rangle$ 
by (metis \langle y \in ?Y \rangle \langle z \in ?Z \rangle enc-iff i-to-j j-to-k mod-add-left-eq mod-less)
qed

```

Every edge of the graph G lies in exactly one triangle.

have *unique-triangles* G

```

unfolding unique-triangles-def
proof (intro strip)
  fix  $e$ 
  assume  $e \in \text{uedges } G$ 
  then consider  $e \in XY \mid e \in YZ \mid e \in XZ$ 
    using  $G\text{-def}$  by fastforce
  then show  $\exists! T. \exists x y z. T = \{x, y, z\} \wedge \text{triangle-in-graph } x y z G \wedge e \subseteq T$ 
  proof cases
    case 1
      then obtain  $i j a$  where  $eeq: e = \{\text{prod-encode}(0,i), \text{prod-encode}(1,j)\}$ 
        and  $i < M$  and  $j < M$ 
        and  $df: \text{diff } j i = \text{int } a$  and  $a \in A$ 
        by (auto simp: XY-def Edges-def part-of-def)
      let  $?x = \text{prod-encode } (0, i)$ 
      let  $?y = \text{prod-encode } (1, j)$ 
      let  $?z = \text{prod-encode } (2, (j+a) \bmod M)$ 
      have  $yeq: j = (i+a) \bmod M$ 
        using diff-invert using  $\langle a \in A \rangle df \langle j < M \rangle$  by blast
      with  $\langle a \in A \rangle \langle j < M \rangle$  have  $\{?y, ?z\} \in YZ$ 
        by (fastforce simp: YZ-def Edges-def image-iff diff-invert)
      moreover have  $\{?x, ?z\} \in XZ$ 
        using  $\langle a \in A \rangle$  by (fastforce simp: XZ-def Edges-def yeq diff2-invert  $\langle i < M \rangle$ )
      ultimately have  $T: \text{triangle-in-graph } ?x ?y ?z G$ 
        using  $\langle e \in \text{uedges } G \rangle$  by (force simp add: G-def eeq triangle-in-graph-def)
      show ?thesis
      proof (intro ex1I)
        show  $\exists x y z. \{?x, ?y, ?z\} = \{x, y, z\} \wedge \text{triangle-in-graph } x y z G \wedge e \subseteq \{?x, ?y, ?z\}$ 
          using  $T eeq$  by blast
        fix  $T$ 
        assume  $\exists p q r. T = \{p, q, r\} \wedge \text{triangle-in-graph } p q r G \wedge e \subseteq T$ 
        then obtain  $p q r$  where  $Teq: T = \{p, q, r\}$ 
          and  $tri: \text{triangle-in-graph } p q r G$  and  $e \subseteq T$ 
          by blast
        with uniq
          obtain  $i' a' x y z$  where  $i' < M a' \in A$ 
            and  $x: x \in \{p, q, r\}$  and  $y: y \in \{p, q, r\}$  and  $z: z \in \{p, q, r\}$ 
            and  $xeq: x = \text{prod-encode}(0, i')$ 
            and  $yeq: y = \text{prod-encode}(1, (i'+a') \bmod M)$ 
            and  $zeq: z = \text{prod-encode}(2, (i'+a'+a') \bmod M)$ 
            by metis
          then have  $\text{sets-eq}: \{x, y, z\} = \{p, q, r\}$  by auto
          with  $Teq \langle e \subseteq T \rangle$  have  $esub': e \subseteq \{x, y, z\}$  by blast
          have  $a' < M$ 
            using  $A \langle N < M \rangle \langle a' \in A \rangle$  by auto
          obtain  $?x \in e ?y \in e$  using  $eeq$  by force
          then have  $x = ?x$ 
            using  $esub' eeq yeq zeq$  by simp
          then have  $y = ?y$ 

```

```

    using esub' eqq zeq by simp
  obtain eq':  $i' = i (i'+a') \bmod M = j$ 
    using  $\langle x = ?x \rangle$  xeq using  $\langle y = ?y \rangle$  yeq by auto
  then have diff  $(i'+a') i' = \text{int } a'$ 
    by (simp add: diff-def  $\langle a' < M \rangle$ )
  then have  $a' = a$ 
    by (metis eq' df diff-def mod-diff-left-eq nat-int zmod-int)
  then have  $z = ?z$ 
    by (metis  $\langle y = ?y \rangle$  mod-add-left-eq prod-encode-eq snd-conv yeq zeq)
  then show  $T = \{?x, ?y, ?z\}$ 
    using Teq  $\langle x = ?x \rangle \langle y = ?y \rangle$  sets-eq by presburger
qed
next
case 2
then obtain  $j k a$  where eqq:  $e = \{\text{prod-encode}(1,j), \text{prod-encode}(2,k)\}$ 
  and  $j < M k < M$ 
  and df:  $\text{diff } k j = \text{int } a$  and  $a \in A$ 
  by (auto simp: YZ-def Edges-def part-of-def numeral-2-eq-2)
let  $?x = \text{prod-encode } (0, (M+j-a) \bmod M)$ 
let  $?y = \text{prod-encode } (1, j)$ 
let  $?z = \text{prod-encode } (2, k)$ 
have zeq:  $k = (j+a) \bmod M$ 
  using diff-invert using  $\langle a \in A \rangle$  df  $\langle k < M \rangle$  by blast
with  $\langle a \in A \rangle \langle k < M \rangle$  have  $\{?x, ?z\} \in XZ$ 
  unfolding XZ-def Edges-def image-iff
  apply (clarsimp simp: mod-add-left-eq doubleton-eq-iff conj-disj-distribR
ex-disj-distrib)
  apply (smt (verit, ccfv-threshold) A  $\langle N < M \rangle$  diff2-invert le-add-diff-inverse2
lessThan-iff
linorder-not-less mod-add-left-eq mod-add-self1 not-add-less1
order.strict-trans subsetD)
done
moreover
have  $a < N$  using A  $\langle a \in A \rangle$  by blast
with  $\langle N < M \rangle$  have  $((M + j - a) \bmod M + a) \bmod M = j \bmod M$ 
  by (simp add: mod-add-left-eq)
then have  $\{?x, ?y\} \in XY$ 
  using  $\langle a \in A \rangle \langle j < M \rangle$  unfolding XY-def Edges-def
  by (force simp add: zeq image-iff diff-invert doubleton-eq-iff ex-disj-distrib)
ultimately have  $T: \text{triangle-in-graph } ?x ?y ?z G$ 
  using  $\langle e \in \text{uedges } G \rangle$  by (auto simp: G-def eqq triangle-in-graph-def)
show ?thesis
proof (intro ex1I)
  show  $\exists x y z. \{?x, ?y, ?z\} = \{x, y, z\} \wedge \text{triangle-in-graph } x y z G \wedge e \subseteq \{?x, ?y, ?z\}$ 
    using T eqq by blast
fix T
assume  $\exists p q r. T = \{p, q, r\} \wedge \text{triangle-in-graph } p q r G \wedge e \subseteq T$ 
then obtain  $p q r$  where Teq:  $T = \{p, q, r\}$  and tri:  $\text{triangle-in-graph } p$ 

```

$q r G$ and $e \subseteq T$
by *blast*
with *uniq*
obtain $i' a' x y z$ **where** $i' < M$ $a' \in A$
and $x: x \in \{p, q, r\}$ **and** $y: y \in \{p, q, r\}$ **and** $z: z \in \{p, q, r\}$
and $x_{eq}: x = \text{prod-encode}(0, i')$
and $y_{eq}: y = \text{prod-encode}(1, (i'+a') \bmod M)$
and $z_{eq}: z = \text{prod-encode}(2, (i'+a'+a') \bmod M)$
by *metis*
then have $\text{sets-eq}: \{x, y, z\} = \{p, q, r\}$ **by** *auto*
with $T_{eq} \langle e \subseteq T \rangle$ **have** $e_{\text{sub}'}: e \subseteq \{x, y, z\}$ **by** *blast*
have $a' < M$
using $A \langle N < M \rangle \langle a' \in A \rangle$ **by** *auto*
obtain $?y \in e$ $?z \in e$
using e_{eq} **by** *force*
then have $y = ?y$
using $e_{\text{sub}'}$ e_{eq} x_{eq} z_{eq} **by** *simp*
then have $z = ?z$
using $e_{\text{sub}'}$ e_{eq} x_{eq} **by** *simp*
obtain $eq': (i'+a') \bmod M = j$ $(i'+a'+a') \bmod M = k$
using $\langle y = ?y \rangle y_{eq}$ **using** $\langle z = ?z \rangle z_{eq}$ **by** *auto*
then have $\text{diff} (i'+a'+a') (i'+a') = \text{int } a'$
by (*simp add: diff-def* $\langle a' < M \rangle$)
then have $a' = a$
by (*metis M-mod-bound* $\langle a' \in A \rangle$ *df diff-invert eq' mod-add-eq mod-if*
of-nat-eq-iff)
have $(M + ((i'+a') \bmod M) - a') \bmod M = (M + (i' + a') - a') \bmod M$
by (*metis Nat.add-diff-assoc2* $\langle a' < M \rangle$ *less-imp-le-nat mod-add-right-eq*)
with $\langle i' < M \rangle$ **have** $(M + ((i'+a') \bmod M) - a') \bmod M = i'$
by *force*
with $\langle a' = a \rangle eq'$ **have** $(M + j - a) \bmod M = i'$
by *force*
with x_{eq} **have** $x = ?x$ **by** *blast*
then show $T = \{?x, ?y, ?z\}$
using $T_{eq} \langle z = ?z \rangle \langle y = ?y \rangle \text{sets-eq}$ **by** *presburger*
qed
next
case 3
then obtain $i k a$ **where** $e_{eq}: e = \{\text{prod-encode}(0, i), \text{prod-encode}(2, k)\}$
and $i < M$ **and** $k < M$
and $df: \text{diff2 } k i = \text{int } a$ **and** $a \in A$
by (*auto simp: XZ-def Edges-def part-of-def eval-nat-numeral*)
let $?x = \text{prod-encode}(0, i)$
let $?y = \text{prod-encode}(1, (i+a) \bmod M)$
let $?z = \text{prod-encode}(2, k)$
have $keq: k = (i+a+a) \bmod M$
using diff2-invert [*OF* $\langle a \in A \rangle$, *of* i] *df* $\langle k < M \rangle$ **using** *inj-on-diff2* [*of* i]
by (*simp add: inj-on-def Ball-def mod-add-left-eq*)
with $\langle a \in A \rangle$ **have** $\{?x, ?y\} \in XY$

```

    using ⟨a ∈ A⟩ ⟨i < M⟩ ⟨k < M⟩ apply (auto simp: XY-def Edges-def)
    by (metis M-mod-bound diff-invert enc-iff from-enc imageI)
  moreover have {?y, ?z} ∈ YZ
    apply (auto simp: YZ-def Edges-def image-iff eval-nat-numeral)
    by (metis M-mod-bound ⟨a ∈ A⟩ diff-invert enc-iff from-enc mod-add-left-eq
keq)
  ultimately have T: triangle-in-graph ?x ?y ?z G
    using ⟨e ∈ uedges G⟩ by (force simp add: G-def eeq triangle-in-graph-def)
  show ?thesis
  proof (intro exI)
    show ∃ x y z. {?x, ?y, ?z} = {x, y, z} ∧ triangle-in-graph x y z G ∧ e ⊆
    {?x, ?y, ?z}
      using T eeq by blast
    fix T
    assume ∃ p q r. T = {p, q, r} ∧ triangle-in-graph p q r G ∧ e ⊆ T
    then obtain p q r where Teq: T = {p, q, r} and tri: triangle-in-graph p
q r G and e ⊆ T
      by blast
    with uniq obtain i' a' x y z where i' < M a' ∈ A
      and x: x ∈ {p, q, r} and y: y ∈ {p, q, r} and z: z ∈ {p, q, r}
      and xeq: x = prod-encode(0, i')
      and yeq: y = prod-encode(1, (i' + a') mod M)
      and zeq: z = prod-encode(2, (i' + a' + a') mod M)
      by metis
    then have sets-eq: {x, y, z} = {p, q, r} by auto
    with Teq ⟨e ⊆ T⟩ have esub': e ⊆ {x, y, z} by blast
    have a' < M
      using A ⟨N < M⟩ ⟨a' ∈ A⟩ by auto
    obtain ?x ∈ e ?z ∈ e using eeq by force
    then have x = ?x
      using esub' eeq yeq zeq by simp
    then have z = ?z
      using esub' eeq yeq by simp
    obtain eq': i' = i (i' + a' + a') mod M = k
      using ⟨x = ?x⟩ xeq using ⟨z = ?z⟩ zeq by auto
    then have diff (i' + a') i' = int a'
      by (simp add: diff-def ⟨a' < M⟩)
    then have a' = a
      by (metis ⟨a' ∈ A⟩ add commute df diff2-invert eq' mod-add-right-eq
nat-int)
    then have y = ?y
      by (metis ⟨x = ?x⟩ prod-encode-eq snd-conv yeq xeq)
    then show T = {?x, ?y, ?z}
      using Teq ⟨x = ?x⟩ ⟨z = ?z⟩ sets-eq by presburger
  qed
qed
qed
have *: card (uedges G) ≤ ε/12 * (card (uverts G))2
  using L ⟨L < card (uverts G)⟩ ⟨unique-triangles G⟩ ⟨uwellformed G⟩ by blast

```


have *diff-cancel*: $\exists j < M. \text{diff } j \ i = \text{int } a$ **if** $a \in A$ **for** $i \ a$
using *M-mod-bound diff-invert that by blast*
have *diff2-cancel*: $\exists j < M. \text{diff2 } j \ i = \text{int } a$ **if** $a \in A$ **for** $i \ a$
using *M-mod-bound diff2-invert that by blast*

have *card-Edges*: $\text{card } (\text{Edges } (\text{part-of } \xi) (\text{part-of } \zeta) \text{ df}) = M * \text{card } A$ (**is** *card*
?E = -)
if $\xi \neq \zeta$ **and** *df-cancel*: $\forall a \in A. \forall i < M. \exists j < M. \text{df } j \ i = \text{int } a$
and *df-inj*: $\forall a. \text{inj-on } (\lambda x. \text{df } x \ a) \ \{.. < M\}$ **for** $\xi \ \zeta \ \text{df}$

proof –
define *R* **where** $R \equiv \lambda \xi \ Y \ \text{df } u \ p. \exists x \ y \ i \ a. u = \{x, y\} \wedge p = (i, a) \wedge x =$
prod-encode (ξ, i)
 $\wedge y \in Y \wedge a \in A \wedge \text{df}(\text{from-part } y) (\text{from-part}$
 $x) = \text{int } a$
have *R-uniq*: $\llbracket R \ \xi \ (\text{part-of } \zeta) \ \text{df } u \ p; R \ \xi \ (\text{part-of } \zeta) \ \text{df } u \ p'; \xi \neq \zeta \rrbracket \implies p' = p$
for $u \ p \ p' \ \xi \ \zeta \ \text{df}$
by (*auto simp add: R-def doubleton-eq-iff*)
define *f* **where** $f \equiv \lambda \xi \ Y \ \text{df } u. @p. R \ \xi \ Y \ \text{df } u \ p$
have *f-if-R*: $f \ \xi \ (\text{part-of } \zeta) \ \text{df } u = p$ **if** $R \ \xi \ (\text{part-of } \zeta) \ \text{df } u \ p \ \xi \neq \zeta$ **for** $u \ p \ \xi \ \zeta \ \text{df}$
using *R-uniq f-def that by blast*
have *bij-betw* $(f \ \xi \ (\text{part-of } \zeta) \ \text{df}) \ ?E \ (\{.. < M\} \times A)$
unfolding *bij-betw-def inj-on-def*
proof (*intro conjI strip*)
fix $u \ u'$
assume $u \in ?E$ **and** $u' \in ?E$
and *eq*: $f \ \xi \ (\text{part-of } \zeta) \ \text{df } u = f \ \xi \ (\text{part-of } \zeta) \ \text{df } u'$
obtain $x \ y \ a$ **where** $u: u = \{x, y\} \ x \in \text{part-of } \xi \ y \in \text{part-of } \zeta \ a \in A$
and *df*: $\text{df} (\text{from-part } y) (\text{from-part } x) = \text{int } a$
using $\langle u \in ?E \rangle$ **by** (*force simp add: Edges-def image-iff*)
then obtain i **where** $i: x = \text{prod-encode } (\xi, i)$
using *part-of-def by blast*
with $u \ \text{df}$ *R-def f-if-R* **that** **have** $fu: f \ \xi \ (\text{part-of } \zeta) \ \text{df } u = (i, a)$
by *blast*
obtain $x' \ y' \ a'$ **where** $u': u' = \{x', y'\} \ x' \in \text{part-of } \xi \ y' \in \text{part-of } \zeta \ a' \in A$
and *df'*: $\text{df} (\text{from-part } y') (\text{from-part } x') = \text{int } a'$
using $\langle u' \in ?E \rangle$ **by** (*force simp add: Edges-def image-iff*)
then obtain i' **where** $i': x' = \text{prod-encode } (\xi, i')$
using *part-of-def by blast*
with $u' \ \text{df}'$ *R-def f-if-R* **that** **have** $fu': f \ \xi \ (\text{part-of } \zeta) \ \text{df } u' = (i', a')$
by *blast*
have $i' = i \ a' = a$
using $fu \ fu'$ *eq* **by** *auto*
with $i \ i'$ **have** $x = x'$
by *meson*
moreover **have** $\text{from-part } y = \text{from-part } y'$
using $\text{df } \text{df}' \ \langle x = x' \rangle \ \langle a' = a \rangle \ \text{df-inj } u'(3) \ u(3)$
by (*clarsimp simp add: inj-on-def*) (*metis part-of-M lessThan-iff*)

```

ultimately show  $u = u'$ 
using  $u u'$  by (metis enc-iff from-part-def prod.collapse prod-decode-inverse)
next
have  $f \xi$  (part-of  $\zeta$ ) df '  $?E \subseteq \{..<M\} \times A$ 
proof (clarsimp simp: Edges-def)
  fix  $i a x y b$ 
  assume  $x \in \text{part-of } \xi$   $y \in \text{part-of } \zeta$  df (from-part  $y$ ) (from-part  $x$ ) = int  $b$ 
   $b \in A$  and feq:  $(i, a) = f \xi$  (part-of  $\zeta$ ) df  $\{x, y\}$ 
  then have  $R \xi$  (part-of  $\zeta$ ) df  $\{x, y\}$  (from-part  $x, b$ )
  by (auto simp: R-def doubleton-eq-iff part-of-def)
  then have (from-part  $x, b$ ) =  $(i, a)$ 
  by (simp add: f-if-R feq from-part-def that)
  then show  $i < M \wedge a \in A$ 
  using  $\langle x \in \text{part-of } \xi \rangle \langle b \in A \rangle$  part-of-M by fastforce
qed
moreover have  $\{..<M\} \times A \subseteq f \xi$  (part-of  $\zeta$ ) df '  $?E$ 
proofclarsimp
  fix  $i a$  assume  $a \in A$  and  $i < M$ 
  then obtain  $j$  where  $j < M$  and  $j$ : df  $j i = \text{int } a$ 
  using df-cancel by metis
  then have  $fj$ :  $f \xi$  (part-of  $\zeta$ ) df  $\{\text{prod-encode } (\xi, i), \text{prod-encode } (\zeta, j)\}$ 
=  $(i, a)$ 
  by (metis R-def  $\langle a \in A \rangle$  enc-iff f-if-R from-enc  $\langle \xi \neq \zeta \rangle$ )
  then have  $\{\text{prod-encode } (\xi, i), \text{prod-encode } (\zeta, j \bmod M)\} \in \text{Edges}$  (part-of
 $\xi$ ) (part-of  $\zeta$ ) df
  apply (clarsimp simp: Edges-def doubleton-eq-iff)
  by (metis  $\langle a \in A \rangle \langle i < M \rangle \langle j < M \rangle$  enc-iff from-enc image-eqI  $j \bmod$ -if)
  then show  $(i, a) \in f \xi$  (part-of  $\zeta$ ) df '  $\text{Edges}$  (part-of  $\xi$ ) (part-of  $\zeta$ ) df
  using  $\langle j < M \rangle$   $fj$  image-iff by fastforce
qed
ultimately show  $f \xi$  (part-of  $\zeta$ ) df '  $?E = \{..<M\} \times A$  by blast
qed
then show ?thesis
by (simp add: bij-betw-same-card card-cartesian-product)
qed
have [simp]:  $\text{disjnt } XY \ YZ$   $\text{disjnt } XY \ XZ$   $\text{disjnt } YZ \ XZ$ 
using  $\text{disjnt-part-of unfolding } XY\text{-def } YZ\text{-def } XZ\text{-def Edges-def disjnt-def}$ 
by (clarsimp simp add: disjoint-iff doubleton-eq-iff, meson disjnt-iff n-not-Suc-n
nat.discI)+
have [simp]:  $\text{card } XY = M * \text{card } A$   $\text{card } YZ = M * \text{card } A$ 
by (simp-all add:  $XY\text{-def } YZ\text{-def card-Edges diff-cancel inj-on-diff}$ )
have [simp]:  $\text{card } XZ = M * \text{card } A$ 
by (simp-all add:  $XZ\text{-def card-Edges diff2-cancel inj-on-diff2}$ )
have  $\text{card } (\text{uedges } G) = 3 * M * \text{card } A$ 
by (simp add:  $G\text{-def card-Un-disjnt}$ )
then have  $\text{card } A \leq \varepsilon * (\text{real } M / 4)$ 
using  $* \langle 0 < M \rangle$  by (simp add:  $\text{card}G$  power2-eq-square)
also have  $\dots < \varepsilon * N$ 
using  $\langle N > 0 \rangle$  by (simp add:  $M\text{-def assms}$ )

```

finally show $\text{card } A < \varepsilon * N$.
qed
qed

We finally present the main statement formulated using the upper asymptotic density condition.

theorem *RothArithmeticProgressions:*

assumes *upper-asymptotic-density* $A > 0$

shows $\exists k d. d > 0 \wedge \text{progression3 } k d \subseteq A$

proof (*rule ccontr*)

assume *non*: $\nexists k d. 0 < d \wedge \text{progression3 } k d \subseteq A$

obtain M **where** $X: \forall N \geq M. \forall A' \subseteq \{..<N\}. (\nexists k d. d > 0 \wedge \text{progression3 } k d \subseteq A')$

$\longrightarrow \text{card } A' < \text{upper-asymptotic-density } A / 2 * \text{real } N$

by (*metis half-gt-zero RothArithmeticProgressions-aux assms*)

then have $\forall N \geq M. \text{card } (A \cap \{..<N\}) < \text{upper-asymptotic-density } A / 2 * N$

by (*meson order-trans inf-le1 inf-le2 non*)

then have $\text{upper-asymptotic-density } A \leq \text{upper-asymptotic-density } A / 2$

by (*force simp add: eventually-sequentially less-eq-real-def intro!: upper-asymptotic-densityI*)

with *assms* **show** *False* **by** *linarith*

qed

end