The Balog–Szemerédi–Gowers Theorem

Angeliki Koutsoukou-Argyraki, Mantas Bakšys, and Chelsea Edmonds
University of Cambridge
{ak2110, mb2412, cle47}@cam.ac.uk

September 13, 2023

Abstract

We formalise the Balog–Szemerédi–Gowers Theorem, a profound result in additive combinatorics which played a central role in Gowers's proof deriving the first effective bounds for Szemerédi's Theorem [2]. The proof is of great mathematical interest given that it involves an interplay between different mathematical areas, namely applications of graph theory and probability theory to additive combinatorics involving algebraic objects. This interplay is what made the process of the formalisation, for which we had to develop formalisations of new background material in the aforementioned areas, more rich and technically challenging. We demonstrate how locales, Isabelle's module system, can be employed to handle such interplays. To treat the graph-theoretic aspects of the proof, we make use of a new, more general undirected graph theory library developed recently by Chelsea Edmonds, which is both flexible and extensible [1]. For the formalisation we followed a proof presented in the 2022 lecture notes by Timothy Gowers "Introduction to Additive Combinatorics" for Part III of the Mathematical Tripos taught at the University of Cambridge [3]. In addition to the main theorem, which, following our source, is formulated for difference sets, we also give an alternative version for sumsets which required a formalisation of an auxiliary triangle inequality following a proof by Yufei Zhao from his book "Graph Theory and Additive Combinatorics" [4]. We moreover formalise a few additional results in additive combinatorics that are not used in the proof of the main theorem. This is the first formalisation of the Balog–Szemerédi–Gowers Theorem in any proof assistant to our knowledge.

Contents

Acknowledgements

Angeliki Koutsoukou-Argyraki is funded by the ERC Advanced Grant ALEXANDRIA (Project GA 742178) funded by the European Research Council and led by Lawrence C. Paulson (University of Cambridge, Department of Computer Science and Technology). Mantas Bakšys received funding for his internship supervised by Koutsoukou-Argyraki by the Cambridge Mathematics Placements (CMP) Programme and by the ALEXANDRIA Project. Chelsea Edmonds is jointly funded by the Cambridge Trust (Cambridge Australia Scholarship) and a Cambridge Department of Computer Science and Technology Premium Research Studentship.

1 Miscellaneous technical lemmas

theory *Miscellaneous-Lemmas*
imports
HOL−Library.Indicator-Function
HOL−Analysis.Convex
begin

lemma *set-pairs-filter-subset*: $A \subseteq B \Longrightarrow \{p . p \in A \times A \wedge P\ p\} \subseteq \{p. p \in B \times B \wedge P\ p\}$
by (*intro subsetI*) *blast*

lemma *card-set-ss-indicator*:
assumes $A \subseteq B$
assumes *finite* B
shows *card* $A = (\sum p \in B.\ indicator\ A\ p)$
proof −
obtain C **where** *ceq*: $C = B - A$ **by** *blast*
then have *beq*: $B = A \cup C$ **using** *assms* **by** *blast*
have *bint*: $A \cap C = \{\}$ **using** *ceq* **by** *blast*
have *finite*: *finite* A **using** *assms finite-subset* **by** *auto*
have *zero*: $\bigwedge p.\ p \in C \Longrightarrow indicator\ (A)\ p = 0$
by (*simp add*: *ceq*)
then have *card* $A = (\sum p \in A .\ indicator\ (A)\ p)$
by *simp*
also have ... $= (\sum p \in A .\ indicator\ A\ p) + (\sum p \in C .\ indicator\ A\ p)$
using *zero* **by** (*metis add-cancel-left-right sum.neutral*)
finally show *card* $A = (\sum p \in B.\ indicator\ A\ p)$ **using** *beq bint assms*
by (*metis add.commute ceq sum.subset-diff*)
qed

lemma *card-cartesian-prod-square*: *finite* $X \Longrightarrow card\ (X \times X) = (card\ X)\hat{}2$
using *card-cartesian-product* **by** (*simp add*: *power2-eq-square*)

lemma (**in** *ordered-ab-group-add*) *diff-strict1-mono*:
assumes $a > a'$ $b \le b'$
shows $a - b > a' - b'$
using *diff-strict-mono assms*
by (*metis local.diff-strict-right-mono local.dual-order.not-eq-order-implies-strict*)

lemma *card-cartesian-product-6*: *card* $(A \times A \times A \times A \times A \times A) = (card\ A)\ \hat{}\ 6$
proof−
have *card* $(A \times A \times A \times A \times A \times A) =$
$card\ A * card\ A * card\ A * card\ A * card\ A * card\ A$
using *card-cartesian-product mult.commute* **by** *metis*
then show *?thesis* **by** *algebra*
qed

lemma *card-cartesian-product3*: *card* $(X \times Y \times Z) =$ *card* $X *$ *card* $Y *$ *card* Z
using *card-cartesian-product* **by** (*metis mult.commute mult.left-commute*)

lemma *card-le-image-div*:
fixes A:: *'a set* **and** B:: *'b set* **and** f:: *'a* $\Rightarrow$ *'b set* **and** r:: *real*
assumes *finite B* **and** *pairwise* (λ *s t. disjnt* (*f s*) (*f t*)) *A* **and** $\forall\ d \in A$. (*card* (*f d*)) $\geq r$
and $\forall\ d \in A$. $f\ d \subseteq B$ **and** $r > 0$
shows *card* $A \leq$ *card* B / r

proof (*cases finite A*)
assume *hA*: *finite A*
have *hpair-disj*: *pairwise disjnt* (*f* ' *A*) **using** *assms* **by** (*metis pairwiseD pairwise-imageI*)
have $r *$ *card* $A = (\sum\ d \in A.\ r)$ **by** *simp*
also have ... $\leq (\sum\ d \in A.$ *card* (*f d*)) **using** *assms sum-mono* **by** *fastforce*
also have ... = *sum card* (*f* ' *A*) **using** *assms hA* **by** (*simp add*: *sum-card-image*)
also have ... = *card* ($\bigcup\ d \in A$. *f d*) **using** *assms hA hpair-disj*
by (*metis Sup-upper card-Union-disjoint finite-UN-I rev-finite-subset*)
also have ... $\leq$ *card B* **using** *assms card-mono*
by (*metis UN-subset-iff of-nat-le-iff*)
finally have $r *$ *card* $A \leq$ *card B* **by** *linarith*
thus *?thesis* **using** *divide-le-eq assms* **by** (*simp add*: *mult-imp-le-div-pos mult-of-nat-commute*)
next
assume ¬ *finite A*
thus *?thesis* **using** *assms* **by** *auto*
qed

lemma *list-middle-eq*:
length xs = *length ys* $\Longrightarrow$ *hd xs* = *hd ys* $\Longrightarrow$ *last xs* = *last ys*
$\Longrightarrow$ *butlast* (*tl xs*) = *butlast* (*tl ys*) $\Longrightarrow$ *xs* = *ys*
apply (*induct xs ys rule*: *list-induct2*, *simp*)
by (*metis append-butlast-last-id butlast.simps*(*1*) *butlast.simps*(*2*) *butlast-tl hd-Cons-tl*
impossible-Cons le-refl list.sel(*3*) *neq-Nil-conv*)

lemma *list2-middle-singleton*:
assumes *length xs* = *3*
shows *butlast* (*tl xs*) = [*xs* ! *1*]
proof (*simp add*: *list-eq-iff-nth-eq assms*)
have *l*: *length* (*butlast* (*tl xs*)) = *1* **using** *length-butlast length-tl assms* **by** *simp*
then have *butlast* (*tl xs*) ! *0* = (*tl xs*) ! *0* **using** *nth-butlast*[*of 0 tl xs*] **by** *simp*
then show *butlast* (*tl xs*) ! *0* = *xs* ! *Suc 0* **using** *nth-tl*[*of 0 xs*] *l* **by** *simp*
qed

lemma *le-powr-half-mult*:

fixes *x y z*:: *real*
assumes $x \hat{} 2 \leq y * z$ **and** $0 \leq y$ **and** $0 \leq z$
shows $x \leq y$ *powr*$(1/2) * z$ *powr* $(1/2)$
using *assms power2-eq-square*
by (*metis dual-order.trans linorder-linear powr-ge-pzero powr-half-sqrt powr-mult real-le-rsqrt*
real-sqrt-le-0-iff)

lemma *Cauchy-Schwarz-ineq-sum2*:
fixes *f g*:: *'a* ⇒ *real* **and** *A*:: *'a set*
shows $(\sum d \in A.\ f\ d * g\ d) \leq$
$(\sum d \in A.\ (f\ d)\hat{}2)$ *powr* $(1/2) * (\sum d \in A.\ (g\ d)\hat{}2)$ *powr* $(1/2)$
using *Convex.Cauchy-Schwarz-ineq-sum*[*of f g A*] *le-powr-half-mult sum-nonneg zero-le-power2*
by (*metis* (*mono-tags*, *lifting*))

end

2 Background material for the graph-theoretic aspects of the main proof

This section includes a number of lemmas on project specific definitions for graph theory, building on the general undirected graph theory library [1]

theory *Graph-Theory-Preliminaries*
imports
Miscellaneous-Lemmas
Undirected-Graph-Theory.Bipartite-Graphs
Undirected-Graph-Theory.Connectivity
Random-Graph-Subgraph-Threshold.Ugraph-Misc
begin

2.1 On graphs with loops

context *ulgraph*

begin

definition *degree-normalized*:: *'a* ⇒ *'a set* ⇒ *real* **where**
degree-normalized v S ≡ *card* (*neighbors-ss v S*) / (*card S*)

lemma *degree-normalized-le-1*: *degree-normalized x S* ≤ *1*

proof(*cases finite S*)
assume *hA*: *finite S*
then have *card* (*neighbors-ss x S*) ≤ *card S* **using** *neighbors-ss-def card-mono hA*
by *fastforce*
then show *?thesis* **using** *degree-normalized-def divide-le-eq-1*

by (*metis antisym-conv3 of-nat-le-iff of-nat-less-0-iff*)
next
case *False*
then show *?thesis* **using** *degree-normalized-def* **by** *auto*
qed

end

2.2 On bipartite graphs

context *bipartite-graph*
begin

definition *codegree*:: $'a \Rightarrow 'a \Rightarrow nat$ **where**
codegree v u $\equiv$ *card* $\{x \in V$. *vert-adj v x* $\wedge$ *vert-adj u x*$\}$

lemma *codegree-neighbors*: *codegree v u* = *card* (*neighborhood v* $\cap$ *neighborhood u*)
unfolding *codegree-def neighborhood-def*

proof –
have $\{x \in V$. *vert-adj v x* $\wedge$ *vert-adj u x*$\}$ = $\{va \in V$. *vert-adj v va*$\}$ $\cap$ $\{v \in V$. *vert-adj u v*$\}$
by *blast*
thus *card* $\{x \in V$. *vert-adj v x* $\wedge$ *vert-adj u x*$\}$ = *card* ($\{va \in V$. *vert-adj v va*$\}$ $\cap$ $\{v \in V$. *vert-adj u v*$\}$)
by *auto*
qed

lemma *codegree-sym*: *codegree v u* = *codegree u v*
by (*simp add*: *Int-commute codegree-neighbors*)

definition *codegree-normalized*:: $'a \Rightarrow 'a \Rightarrow 'a$ *set* $\Rightarrow$ *real* **where**
codegree-normalized v u S $\equiv$ *codegree v u* / *card S*

lemma *codegree-normalized-altX*:
assumes $x \in X$ **and** $x' \in X$
shows *codegree-normalized* x x' Y = *card* (*neighbors-ss* x Y $\cap$ *neighbors-ss* x' Y) / *card* Y

proof –
have ((*neighbors-ss* x Y) $\cap$ (*neighbors-ss* x' Y)) = *neighborhood* x $\cap$ *neighborhood* x'
using *neighbors-ss-eq-neighborhoodX assms* **by** *auto*
then show *?thesis* **unfolding** *codegree-normalized-def*
using *codegree-def codegree-neighbors* **by** *presburger*
qed

lemma *codegree-normalized-altY*:

assumes $y \in Y$ **and** $y' \in Y$
shows *codegree-normalized* $y\ y'\ X = card\ (neighbors\text{-}ss\ y\ X \cap neighbors\text{-}ss\ y'\ X)$ $/$ $card\ X$

proof –
have *neighbors-ss* $y\ X \cap$ *neighbors-ss* $y'\ X =$ *neighborhood* $y \cap$ *neighborhood* y'
using *neighbors-ss-eq-neighborhoodY assms* **by** *auto*
then show *?thesis* **unfolding** *codegree-normalized-def*
using *codegree-def codegree-neighbors* **by** *presburger*
qed

lemma *codegree-normalized-sym*: *codegree-normalized* $u\ v\ S =$ *codegree-normalized* $v\ u\ S$
unfolding *codegree-normalized-def* **using** *codegree-sym* **by** *simp*

definition *bad-pair*:: $'a \Rightarrow 'a \Rightarrow 'a\ set \Rightarrow real \Rightarrow bool$ **where**
bad-pair $v\ u\ S\ c \equiv$ *codegree-normalized* $v\ u\ S < c$

lemma *bad-pair-sym*:
assumes *bad-pair* $v\ u\ S\ c$ **shows** *bad-pair* $u\ v\ S\ c$
using *assms bad-pair-def codegree-normalized-def*
by (*simp add*: *codegree-normalized-sym*)

definition *bad-pair-set*:: $'a\ set \Rightarrow 'a\ set \Rightarrow real \Rightarrow ('a \times 'a)\ set$ **where**
bad-pair-set $S\ T\ c \equiv \{(u, v) \in S \times S.$ *bad-pair* $u\ v\ T\ c\}$

lemma *bad-pair-set-ss*: *bad-pair-set* $S\ T\ c \subseteq S \times S$
by (*auto simp add*: *bad-pair-set-def*)

lemma *bad-pair-set-filter-alt*:
bad-pair-set $S\ T\ c =$ *Set.filter* (λ p . *bad-pair* (*fst* p) (*snd* p) $T\ c$) $(S \times S)$
using *bad-pair-set-def* **by** *auto*

lemma *bad-pair-set-finite*:
assumes *finite* S
shows *finite* (*bad-pair-set* $S\ T\ c$)
proof –
have *finite* $(S \times S)$ **using** *finite-cartesian-product assms* **by** *blast*
thus *?thesis* **using** *bad-pair-set-filter-alt finite-filter* **by** *auto*
qed

lemma *codegree-is-path-length-two*:
codegree $x\ x' = card\ \{p\ .$ *connecting-path* $x\ x'\ p \wedge$ *walk-length* $p = 2\}$
unfolding *codegree-def*
proof–
define f:: $'a\ list \Rightarrow 'a$ **where** $f = (\lambda\ p.\ p!1)$
have *f-inj*: *inj-on* f $\{p\ .$ *connecting-path* $x\ x'\ p \wedge$ *walk-length* $p = 2\}$
unfolding *f-def*
proof (*intro inj-onI*, *simp del*: *One-nat-def*)

fix *a b* **assume** *ha*: *connecting-path x x′ a* ∧ *walk-length a* = *2* **and**
hb: *connecting-path x x′ b* ∧ *walk-length b* = *2* **and** *1*: *a*!*1* = *b*!*1*
then have *len*: *length a* = *3 length b* = *3* **using** *walk-length-conv* **by** *auto*
show *a* = *b* **using** *list2-middle-singleton 1 len list-middle-eq ha hb connecting-path-def len* **by** *metis*
qed
have *f-image*: *f* ' {*p* . *connecting-path x x′ p* ∧ *walk-length p* = *2*} =
{*xa* ∈ *V*. *vert-adj x xa* ∧ *vert-adj x′ xa*}
proof(*intro subset-antisym*)
show *f* ' {*p*. *connecting-path x x′ p* ∧ *walk-length p* = *2*}
⊆ {*xa* ∈ *V*. *vert-adj x xa* ∧ *vert-adj x′ xa*}
proof (*intro subsetI*)
fix *a* **assume** *a* ∈ *f* ' {*p*. *connecting-path x x′ p* ∧ *walk-length p* = *2*}
then obtain *p* **where** *ha*: *p*!*1* = *a* **and** *hp*: *connecting-path x x′ p* **and** *hpl*: *length p* = *3*
using *f-def walk-length-conv* **by** *auto*
have *p* ! *0* = *x* **using** *hd-conv-nth*[*of p*] *hpl hp connecting-path-def* **by** *fastforce*

then have *va1*: *vert-adj x a* **using** *is-walk-index*[*of 0 p*] *hp connecting-path-def is-gen-path-def*
vert-adj-def ha hpl **by** *auto*
have *p* ! *2* = *x′* **using** *last-conv-nth*[*of p*] *hpl hp connecting-path-def* **by** *fastforce*
then have *vert-adj a x′* **using** *is-walk-index*[*of 1 p*] *hp connecting-path-def is-gen-path-def*
vert-adj-def ha hpl **by** (*metis One-nat-def le0 lessI numeral-3-eq-3 one-add-one*)
then show *a* ∈ {*a* ∈ *V*. *vert-adj x a* ∧ *vert-adj x′ a*}
using *va1 vert-adj-sym* **by** (*simp add*: *vert-adj-imp-inV*)
qed
show {*xa* ∈ *V*. *vert-adj x xa* ∧ *vert-adj x′ xa*}
⊆ *f* ' {*p*. *connecting-path x x′ p* ∧ *walk-length p* = *2*}
proof (*intro subsetI*)
fix *a* **assume** *ha*: *a* ∈ {*xa* ∈ *V*. *vert-adj x xa* ∧ *vert-adj x′ xa*}
then have *a* ∈ *V* **and** *x* ∈ *V* **and** *x′* ∈ *V* **and** *vert-adj x a* **and** *vert-adj x′ a*
using *vert-adj-imp-inV* **by** *auto*
then have *is-gen-path* [*x*, *a*, *x′*]
using *is-walk-def vert-adj-def vert-adj-sym ha singleton-not-edge is-gen-path-def* **by** *auto*
then have *connecting-path x x′* [*x*, *a*, *x′*]
unfolding *connecting-path-def vert-adj-def hd-conv-nth last-conv-nth* **by** *simp*
moreover have *walk-length* [*x*, *a*, *x′*] = *2* **using** *walk-length-conv* **by** *simp*
ultimately show *a* ∈ *f* ' {*p*. *connecting-path x x′ p* ∧ *walk-length p* = *2*} **using** *f-def* **by** *force*
qed
qed
then show *card* {*xa* ∈ *V*. *vert-adj x xa* ∧ *vert-adj x′ xa*} =
card {*p*. *connecting-path x x′ p* ∧ *walk-length p* = *2*}

using *f-inj card-image* **by** *fastforce*
qed

lemma *codegree-bipartite-eq*:
$\forall\ x \in X.\ \forall\ x' \in X.$ *codegree x x'* = *card* $\{y \in Y.$ *vert-adj x y* $\wedge$ *vert-adj x' y*$\}$
unfolding *codegree-def* **using** *vert-adj-imp-inV X-vert-adj-Y*
by (*metis* (*no-types*, *lifting*) *Collect-cong*)

lemma (**in** *fin-bipartite-graph*) *bipartite-deg-square-eq*:
$\forall\ y \in Y.$ ($\sum\ x' \in X.\ \sum\ x \in X.$ *indicator* $\{z.$ *vert-adj x z* $\wedge$ *vert-adj x' z*$\}$ *y*) = (*degree y*)^*2*
proof
have *hX*: *finite X* **by** (*simp add*: *partitions-finite*(*1*))
fix *y* **assume** *hy*: $y \in Y$
have *1*: $\forall\ x' \in X.\ \forall\ x \in X.$ *indicator* $\{z.$ *vert-adj x z* $\wedge$ *vert-adj x' z*$\}$ *y* =
indicator ($\{z.$ *vert-adj x' z*$\}$ $\cap$ $\{z.$ *vert-adj x z*$\}$) *y*
by (*metis* (*mono-tags*, *lifting*) *Int-Collect indicator-simps*(*1*) *indicator-simps*(*2*) *mem-Collect-eq*)
have *2*: $\forall\ x' \in X.\ \forall\ x \in X.$ (*indicator* ($\{z.$ *vert-adj x' z*$\}$ $\cap$ $\{z.$ *vert-adj x z*$\}$) *y*:: *nat*) =
indicator $\{z.$ *vert-adj x' z*$\}$ *y* $*$ *indicator* $\{z.$ *vert-adj x z*$\}$ *y* **using** *indicator-inter-arith*
by *auto*
have ($\sum\ x' \in X.\ \sum\ x \in X.$ (*indicator* $\{z.$ *vert-adj x z* $\wedge$ *vert-adj x' z*$\}$ *y*:: *nat*)) =
($\sum\ x' \in X.\ \sum\ x \in X.$ *indicator* ($\{z.$ *vert-adj x' z*$\}$ $\cap$ $\{z.$ *vert-adj x z*$\}$) *y*)
using *1 sum.cong* **by** (*metis* (*no-types*, *lifting*))
also have ... = ($\sum\ x' \in X.\ \sum\ x \in X.$ *indicator* $\{z.$ *vert-adj x' z*$\}$ *y* $*$
indicator $\{z.$ *vert-adj x z*$\}$ *y*) **using** *2 sum.cong* **by** *auto*
also have ... = *sum* (λ *x*. *indicator* $\{z.$ *vert-adj x z*$\}$ *y*) *X* $*$ *sum* (λ *x*. *indicator* $\{z.$ *vert-adj x z*$\}$ *y*) *X*
using *sum-product*[*of* (λ *x*. (*indicator* $\{z.$ *vert-adj x z*$\}$ *y*:: *nat*)) *X*
(λ *x*. *indicator* $\{z.$ *vert-adj x z*$\}$ *y*) *X*] **by** *auto*
finally have *3*: ($\sum\ x' \in X.\ \sum\ x \in X.$ (*indicator* $\{z.$ *vert-adj x z* $\wedge$ *vert-adj x' z*$\}$ *y*:: *nat*)) =
(*sum* (λ *x*. *indicator* $\{z.$ *vert-adj x z*$\}$ *y*) *X*) ^ *2* **using** *power2-eq-square*
by (*metis* (*no-types*, *lifting*))
have $\forall\ x \in X.$ *indicator* $\{z.$ *vert-adj x z*$\}$ *y* = *indicator* $\{x.$ *vert-adj x y*$\}$ *x*
by (*simp add*: *indicator-def*)
from *this* **have** (*sum* (λ *x*. *indicator* $\{z.$ *vert-adj x z*$\}$ *y*) *X*) = *sum* (λ *x*. *indicator* $\{x.$ *vert-adj x y*$\}$ *x*) *X*
using *sum.cong* **by** *fastforce*
also have ... = *card* ($\{x \in X.$ *vert-adj x y*$\}$) **using** *sum-indicator-eq-card hX*
by (*metis Collect-conj-eq Collect-mem-eq*)
finally show ($\sum x' \in X.\ \sum x \in X.$ *indicator* $\{z.$ *vert-adj x z* $\wedge$ *vert-adj x' z*$\}$ *y*) = (*degree y*)^*2*
using *3 hy degree-neighbors-ssY neighbors-ss-def vert-adj-sym* **by** *presburger*
qed

lemma (**in** *fin-bipartite-graph*) *codegree-degree*:
($\sum$ $x' \in X$. $\sum$ $x \in X$. (*codegree* x x')) = ($\sum$ $y \in Y$. (*degree* y)^*2*)

proof−
have *hX*: *finite X* **and** *hY*: *finite Y*
by (*simp-all add*: *partitions-finite*)
have $\forall$ $x' \in X$. $\forall$ $x \in X$. {$z \in V$. *vert-adj* x z $\wedge$ *vert-adj* x' z} = $Y \cap$ {z. *vert-adj* x z $\wedge$ *vert-adj* x' z}
using *XY-union X-vert-adj-Y* **by** *fastforce*
from *this* **have** ($\sum$ $x' \in X$. $\sum$ $x \in X$. (*codegree* x x')) = ($\sum$ $x' \in X$. $\sum$ $x \in X$. *card* ($Y \cap$ {z. *vert-adj* x z $\wedge$ *vert-adj* x' z}))
using *codegree-def sum.cong* **by** *auto*
also have ... = ($\sum$ $x' \in X$. $\sum$ $x \in X$. $\sum$ $y \in Y$. *indicator* {z. *vert-adj* x z $\wedge$ *vert-adj* x' z} y)
using *sum-indicator-eq-card hY* **by** *fastforce*
also have ... = ($\sum$ $x' \in X$. $\sum$ $y \in Y$. ($\sum$ $x \in X$. *indicator* {z. *vert-adj* x z $\wedge$ *vert-adj* x' z} y))
using *sum.swap* **by** (*metis* (*no-types*))
also have ... = ($\sum$ $y \in Y$. $\sum$ $x' \in X$. ($\sum$ $x \in X$. *indicator* {z. *vert-adj* x z $\wedge$ *vert-adj* x' z} y))
using *sum.swap* **by** *fastforce*
also have ... = ($\sum$ $y \in Y$. (*degree* y)^*2*) **using** *bipartite-deg-square-eq sum.cong* **by** *force*
finally show *?thesis* **by** *simp*
qed

lemma (**in** *fin-bipartite-graph*) *sum-degree-normalized-X-density*:
($\sum$ $x \in X$. *degree-normalized* x Y) / *card* X = *edge-density* X Y
by (*smt* (*z3*) *card-all-edges-betw-neighbor card-edges-between-set degree-normalized-def divide-divide-eq-left′ density-simp of-nat-mult of-nat-sum partitions-finite*(*1*) *partitions-finite*(*2*) *sum.cong sum-left-div-distrib*)

lemma (**in** *fin-bipartite-graph*) *sum-degree-normalized-Y-density*:
($\sum$ $y \in Y$. *degree-normalized* y X) / *card* Y = *edge-density* X Y
using *bipartite-sym fin-bipartite-graph.sum-degree-normalized-X-density fin-bipartite-graph-def*

fin-graph-system-axioms edge-density-commute **by** *fastforce*

end
end

3 Auxiliary probability space results

theory *Prob-Space-Lemmas*
imports
Random-Graph-Subgraph-Threshold.Prob-Lemmas
begin

context *prob-space*

begin

lemma *expectation-uniform-count*:
  assumes M = *uniform-count-measure* X **and** *finite* X
  shows *expectation* $f = (\sum x \in X.\ f\ x)$ / *card* X

proof−
  have *expectation* $f = (\sum x \in X.\ (1\ /\ (card\ X)) * f\ x)$
   using *assms uniform-count-measure-def bot-nat-0.extremum of-nat-0 of-nat-le-iff real-scaleR-def*
    lebesgue-integral-point-measure-finite[*of* - (λ *x*. *1* / *card X*) *f*]
    scaleR-sum-right sum-distrib-left zero-le-divide-1-iff **by** *metis*
  then show *?thesis* **using** *sum-left-div-distrib* **by** *fastforce*
qed

A lemma to obtain a value for x where the inequality is satisfied

lemma *expectation-obtains-ge*:
  fixes f :: $'a \Rightarrow$ *real*
  assumes M = *uniform-count-measure* X **and** *finite* X
  assumes *expectation* $f \geq c$
  obtains x **where** $x \in X$ **and** $f\ x \geq c$

proof −
  have *ne*: $X \neq \{\}$
    using *assms*(*1*) *subprob-not-empty* **by** *auto*
  then have *ne0*: *card* $X > 0$
    by (*simp add*: *assms*(*2*) *card-gt-0-iff*)
  have $\exists\ x \in X\ .\ f\ x \geq c$
  proof (*rule ccontr*)
    assume $\neg\ (\exists x{\in}X.\ c \leq f\ x)$
    then have $\forall x{\in}X.\ c > f\ x$ **by** *auto*
    then have $(\sum x \in X.\ f\ x) < (\sum x \in X.\ c)$
      by (*meson assms*(*2*) *ne sum-strict-mono*)
    then have *lt*: $(\sum x \in X.\ f\ x) < (card\ X) * c$ **by** *simp*
   have *expectation* $f = (\sum x \in X.\ f\ x)$ / *card* X **using** *expectation-uniform-count assms* **by** *auto*
    then have $(\sum x \in X.\ f\ x) \geq (card\ X) * c$ **using** *ne0 assms*
      by (*simp add*: *le-divide-eq mult-of-nat-commute*)
    then show *False* **using** *lt* **by** *auto*
  qed
  then show *?thesis* **using** *that* **by** *auto*
qed

The following is the variation on the Cauchy-Schwarz inequality presented in Gowers's notes before Lemma 2.13 [3].

lemma *cauchy-schwarz-ineq-var*:
  fixes X :: $'a \Rightarrow$ *real*
  assumes *integrable* M ($\lambda x.\ (X\ x)$^*2*) **and** $X \in$ *borel-measurable* M

shows *expectation* (λ *x*. (*X x*)^*2*) ≥ (*expectation* (λ *x* . (*X x*)))^*2*

proof −
have *expectation* (λ *x*. (*X x*)^*2*) − (*expectation* (λ *x* . (*X x*)))^*2* = *expectation* (λ*x*. (*X x* − *expectation X*)^*2*)
using *variance-expectation assms*(*1*) *assms*(*2*) **by** *presburger*
then have *expectation* (λ *x*. (*X x*)^*2*) − (*expectation* (λ *x* . (*X x*)))^*2* ≥ *0* **by** *simp*
thus *?thesis* **by** *simp*
qed

lemma *integrable-uniform-count-measure-finite*:
fixes *g* :: ′*a* ⇒ ′b::{*banach*, *second-countable-topology*}
shows *finite A* ⟹ *integrable* (*uniform-count-measure A*) *g*
unfolding *uniform-count-measure-def* **by** (*simp add*: *integrable-point-measure-finite*)

lemma *cauchy-schwarz-ineq-var-uniform*:
fixes *X* :: ′*a* ⇒ *real*
assumes *M* = *uniform-count-measure S*
assumes *finite S*
shows *expectation* (λ *x*. (*X x*)^*2*) ≥ (*expectation* (λ *x* . (*X x*)))^*2*

proof −
have *borel*: *X* ∈ *borel-measurable M* **using** *assms* **by** (*simp*)
have *integrable M X* **using** *assms* **by** (*simp add*: *integrable-uniform-count-measure-finite*)
then have *integrable M* (λ *x*. (*X x*)^*2*) **using** *assms* **by** (*simp add*: *integrable-uniform-count-measure-finite*)
thus *?thesis* **using** *cauchy-schwarz-ineq-var borel* **by** *simp*
qed

An equation for expectation over a discrete random variables distribution:

lemma *expectation-finite-uniform-space*:
assumes *M* = *uniform-count-measure S* **and** *finite S*
fixes *X* :: ′*a* ⇒ *real*
shows *expectation X* = (∑ *y* ∈ *X* ‘ *S* . *prob* {*x* ∈ *S* . *X x* = *y*} ∗ *y*)

proof −
have *Bochner-Integration.simple-bochner-integrable M X*
proof (*safe intro*!: *Bochner-Integration.simple-bochner-integrable.intros*)
show *simple-function M X* **unfolding** *simple-function-def* **using** *assms*
by (*auto simp add*: *space-uniform-count-measure*)
show *emeasure M* {*y* ∈ *space M*. *X y* ≠ *0*} = ∞ ⟹ *False*
using *emeasure-subprob-space-less-top* **by** (*auto*)
qed
then have *expectation X* = *Bochner-Integration.simple-bochner-integral M X*
using *simple-bochner-integrable-eq-integral* **by** *fastforce*
thus *?thesis* **using** *Bochner-Integration.simple-bochner-integral-def space-uniform-count-measure*
by (*metis* (*no-types*, *lifting*) *Collect-cong assms*(*1*) *real-scaleR-def sum.cong*)

qed

lemma *expectation-finite-uniform-indicator*:
assumes M = *uniform-count-measure* S **and** *finite* S
shows *expectation* (λ *x*. *indicator* (T *x*) *y*) = *prob* $\{x \in S$. *indicator* (T *x*) y = *1*$\}$ (**is** *expectation ?X* = -)

proof −
have *ss*: *?X* ' $S \subseteq \{0, 1\}$
by (*intro subsetI*, *auto simp add*: *indicator-eq-1-iff*)
have *diff*: $\bigwedge y'. y' \in (\{0, 1\} - ?X ` S) \Longrightarrow prob \{x \in S . ?X x = y'\} = 0$
by (*metis* (*mono-tags*, *lifting*) *DiffD2 empty-Collect-eq image-eqI measure-empty*)
have *expectation ?X* = ($\sum$ $y \in$ *?X* ' S . *prob* $\{x \in S$. *?X* $x = y\} * y$)
using *expectation-finite-uniform-space assms* **by** *auto*
also have ... = ($\sum$ $y \in$ *?X* ' S . *prob* $\{x \in S$. *?X* $x = y\} * y$) +
($\sum$ $y \in (\{0, 1\} -$ *?X* ' S) . *prob* $\{x \in S$. *?X* $x = y\} * y$)
using *diff* **by** *auto*
also have ... = ($\sum$ $y \in \{0, 1\}$. *prob* $\{x \in S$. *?X* $x = y\} * y$)
using *sum.subset-diff*[*of ?X* ' S $\{0, 1\}$ λ *y*. *prob* $\{x \in S$. *?X* $x = y\} * y$] *ss* **by** *fastforce*
also have ... = *prob* $\{x \in S$. *?X* $x = 0\} * 0$ + *prob* $\{x \in S$. *?X* $x = 1\} * 1$ **by** *auto*
finally have *expectation ?X* = *prob* $\{x \in S$. *?X* $x = 1\}*1$ **by** *auto*
thus *?thesis* **by** (*smt* (*verit*) *Collect-cong indicator-eq-1-iff*)
qed

end
end

4 A triangle inequality for sumsets

theory *Sumset-Triangle-Inequality*
imports
Pluennecke-Ruzsa-Inequality.Pluennecke-Ruzsa-Inequality
begin

context *additive-abelian-group*

begin

We show a useful triangle inequality for sumsets that does *not* follow from the Ruzsa triangle inequality. The proof follows the exposition in Zhao's book [4].

The following auxiliary lemma corresponds to Lemma 7.3.4 in Zhao's book [4].

lemma *triangle-ineq-sumsets-aux*:
fixes X B Y :: *'a set*

assumes *hX*: *finite X* **and** *hB*: *finite B* **and** *hXG*: $X \subseteq G$ **and** *hBG*: $B \subseteq G$ **and**
hXne: $X \neq \{\}$ **and** *hYX*: $\bigwedge Y.\ Y \subseteq X \Longrightarrow Y \neq \{\} \Longrightarrow$ *card* (*sumset Y B*) / *card Y* $\geq$
card (*sumset X B*) / *card X* **and** *hC*: *finite C* **and** *hCne*: $C \neq \{\}$ **and** *hCG*: $C \subseteq G$
shows *card* (*sumset X* (*sumset C B*)) / *card* (*sumset X C*) $\leq$ *card* (*sumset X B*) / *card X*
using *hC hCne hCG* **proof** (*induct*)
case *empty*
then show *?case* **by** *blast*
next
case *hcase*: (*insert c C*)
have *hc* : $c \in G$ **using** *hcase* **by** *auto*
show *card* (*sumset X B*) / *card X* $\geq$
card (*sumset X* (*sumset* (*insert c C*) *B*)) / *card* (*sumset X* (*insert c C*))
proof(*cases* $C = \{\}$)
case *True*
then have *card* (*sumset X* (*insert c C*)) = *card X* **using** *hc hXG hX*
by (*simp add*: *card-sumset-singleton-eq le-iff-inf*)
moreover have *card* (*sumset X* (*sumset* (*insert c C*) *B*)) $\leq$ *card* (*sumset X B*) **using** *hX hB hBG hXG*
hc **by** (*metis True card-sumset-le finite-sumset sumset-assoc sumset-commute*)
ultimately show *?thesis* **by** (*simp add*: *divide-right-mono*)
next
case *hCne*: *False*
have *hCG* : $C \subseteq G$ **using** *hcase* **by** *auto*
have *hstep*: *card* (*sumset X* (*sumset* $\{c\}$ *B*) $-$ *sumset X* (*sumset C B*)) $\leq$
card (*sumset X B*) $*$ *card* (*sumset X* $\{c\}$ $-$ *sumset X C*) / *card X*
proof$-$
let *?Y* = $\{x \in X.$ *sumset* $\{x\}$ (*sumset* $\{c\}$ *B*) $\subseteq$ *sumset X* (*sumset C B*)$\}$
have *hYX* : *?Y* $\subseteq$ *X* **and** *hY*: *finite ?Y* **using** *finite-subset hX* **by** *auto*
have *hsub1*: *sumset X* (*sumset* $\{c\}$ *B*) $-$ *sumset X* (*sumset C B*) $\subseteq$
sumset (*sumset X* $\{c\}$ $-$ *sumset X C*) *B*
by (*metis Diff-subset-conv Un-Diff-cancel sumset-assoc sumset-subset-Un1 sup.cobounded2*)
have *hcard1* : *card* (*sumset X B*) = *card* (*sumset X* (*sumset* $\{c\}$ *B*))
by (*metis card-sumset-singleton-eq finite-sumset hB hX hc sumset-Int-carrier sumset-assoc*
sumset-commute)
have *hcard2* : *card* (*sumset ?Y B*) = *card* (*sumset* (*sumset ?Y* $\{c\}$) *B*)
using *card-sumset-singleton-eq finite-sumset hB hY hc sumset-Int-carrier sumset-assoc*
sumset-commute **by** (*smt* (*verit*, *ccfv-threshold*))
have *sumset* (*sumset ?Y* $\{c\}$) *B* $\subseteq$ *sumset X* (*sumset C B*)
proof
fix *d* **assume** $d \in$ *sumset* (*sumset ?Y* $\{c\}$) *B*
then obtain *a b* **where** *ha*: $a \in$ *?Y* **and** $b \in B$ **and** *hd*: $d = a \oplus c \oplus b$
by (*smt* (*verit*) *empty-iff insert-iff sumset.cases*)

then have $a \oplus c \oplus b \in$ *sumset* $\{a\}$ (*sumset* $\{c\}$ B)
by (*smt* (*verit*) *associative composition-closed hBG hXG hc insertCI mem-Collect-eq subsetD*
sumset.sumsetI)
then show $d \in$ *sumset* X (*sumset* C B) **using** *ha hd* **by** *blast*
qed
then have *hdisj* : *disjnt* ((*sumset* X (*sumset* $\{c\}$ B)) $-$ (*sumset* X (*sumset* C B)))
(*sumset* (*sumset* *?Y* $\{c\}$) B)
by (*auto simp add* : *disjnt-iff*)
have *hsub2* : *sumset* (*sumset* X $\{c\}$ $-$ *sumset* X C) $B \cup$ *sumset* (*sumset* *?Y* $\{c\}$) $B \subseteq$
sumset (*sumset* X $\{c\}$) B **by** (*simp add*: *sumset-mono*)
then have *ineq1*: *card* (*sumset* X (*sumset* $\{c\}$ B) $-$ *sumset* X (*sumset* C B)) $+$ *card* (*sumset* *?Y* B) $\leq$
card (*sumset* X B)
proof−
have *card* (*sumset* X (*sumset* $\{c\}$ B) $-$ *sumset* X (*sumset* C B)) $+$ *card* (*sumset* *?Y* B) $=$
card ((*sumset* X (*sumset* $\{c\}$ B) $-$ *sumset* X (*sumset* C B)) $\cup$ *sumset* (*sumset* *?Y* $\{c\}$) B)
using *hdisj hcard2 card-Un-disjnt finite-sumset hX hYX finite-subset hB*
by (*metis* (*no-types*, *lifting*) *finite.emptyI finite.insertI finite-Diff*)
also have ... $\leq$ *card* (*sumset* X (*sumset* $\{c\}$ B)) **using** *card-mono finite-sumset hX hB*
by (*metis* (*no-types*, *lifting*) *Diff-subset Un-subset-iff finite.emptyI finite.insertI*
hsub2 sumset-assoc)
finally show *card* (*sumset* X (*sumset* $\{c\}$ B) $-$ *sumset* X (*sumset* C B)) $+$ *card* (*sumset* *?Y* B) $\leq$
card (*sumset* X B) **using** *hcard1* **by** *auto*
qed
have *ineq2*: *card* (*sumset* X $\{c\}$ $-$ *sumset* X C) $\geq$ *card* X $-$ *card* *?Y*
proof−
let *?Z* $= \{x \in X.$ *sumset* $\{x\}$ $\{c\} \subseteq$ *sumset* X $C\}$
have *hZY*: *?Z* $\subseteq$ *?Y*
by (*smt* (*verit*, *del-insts*) *Collect-mono-iff subset-refl sumset-assoc sumset-mono*)
have *hinj*: *inj-on* (λ x. $x \oplus c$) (X $-$ *?Z*)
proof (*intro inj-onI*)
fix x y **assume** $x \in X$ $-$ *?Z* **and** $y \in X$ $-$ *?Z* **and** h: $x \oplus c = y \oplus c$
then have $x \in G$ **and** $y \in G$ **using** *hXG* **by** *auto*
then show $x = y$ **using** *h hc* **by** *simp*
qed
have *himage*: (λ x. $x \oplus c$) ‘ (X $-$ *?Z*) $=$ *sumset* X $\{c\}$ $-$ *sumset* X C
proof
show (λx. $x \oplus c$) ‘ (X $-$ *?Z*) $\subseteq$ *sumset* X $\{c\}$ $-$ *sumset* X C
proof(*intro image-subsetI*)
fix x **assume** *hx*: $x \in X$ $-$ *?Z*

then have *hxG* : $x \in G$ **using** *hXG* **by** *auto*
then have *hxc1*: $x \oplus c \in$ *sumset X* $\{c\}$ **using** *hXG hc hx* **by** *auto*
have $x \oplus c \in$ *sumset* $\{x\}$ $\{c\}$ **using** *hxG hc* **by** *auto*
then have *hxc2*: $x \oplus c \notin$ *sumset X C* **using** *hx hXG hc hCG*
using *DiffD2 sumset.simps sumset.sumsetI* **by** *auto*
then show $x \oplus c \in$ *sumset X* $\{c\}$ $-$ *sumset X C* **using** *hxc1 hxc2* **by** *simp*
qed
show *sumset X* $\{c\}$ $-$ *sumset X C* $\subseteq$ $(\lambda x.\ x \oplus c)$ ‘ $(X - ?Z)$
proof
fix *d* **assume** $d \in$ *sumset X* $\{c\}$ $-$ *sumset X C*
then obtain *x* **where** *hd*: $d = x \oplus c$ **and** *hxc*: $x \oplus c \notin$ *sumset X C* **and** *hx*: $x \in X$
using *sumset.cases* **by** *force*
then show $d \in (\lambda x.\ x \oplus c)$ ‘ $(X - ?Z)$ **using** *hd hxc hx hXG hc* **by** *auto*

qed
qed
have *hcard3*: *card* $(X - ?Z)$ = *card* (*sumset X* $\{c\}$ $-$ *sumset X C*)
using *card-image hinj himage* **by** *fastforce*
have *card X* = *card ?Z* + *card* $(X - ?Z)$
by (*simp add*: *card-Diff-subset card-mono hX*)
also have ... $\leq$ *card ?Y* + *card* (*sumset X* $\{c\}$ $-$ *sumset X C*)
using *hcard3 card-mono hZY hY* **by** *auto*
finally show *?thesis* **by** *simp*
qed
have *ineq3*: *card* (*sumset X B*) $-$ *card* (*sumset ?Y B*) $\leq$
card (*sumset X B*) $*$ (*card X* $-$ *card ?Y*) / *card X*
proof(*cases ?Y* = $\{\}$)
case *True*
then show *?thesis* **using** *card-eq-0-iff*
by (*smt* (*verit*) *hX hXne minus-nat.diff-0 nonzero-mult-div-cancel-right of-nat-eq-0-iff*
of-nat-mult sumset-empty(*2*))
next
case *hYne*: *False*
have *card* (*sumset ?Y B*) $\geq$ *card* (*sumset X B*) / *card X* $*$ *card ?Y* **using** *assms*(*6*)[*OF hYX hYne*]
hX hYne hX hYX finite-subset divide-le-eq
by (*smt* (*z3*) *card-gt-0-iff hY mult-imp-div-pos-less of-nat-0-less-iff*)
moreover have *card* (*sumset X B*) / *card X* $*$ *card ?Y* = (*card* (*sumset X B*) $*$ *card ?Y*) / *card X*
by *auto*
moreover have *card* (*sumset X B*) $*$ *card ?Y* / *card X* $*$ *card X* = *card* (*sumset X B*) $*$ *card ?Y*
using *hX* **by** (*simp add*: *field-simps*)
ultimately have *card* (*sumset ?Y B*) $*$ *card X* $\geq$ *card* (*sumset X B*) $*$ *card ?Y*
using *hX hXne of-nat-0-less-iff le-divide-eq*

by (*smt* (*z3*) *card-sumset-0-iff hBG hXG mult-cancel1 mult-cancel2 of-nat-le-0-iff*

of-nat-le-iff of-nat-mult)
then have *real* ((*card* (*sumset X B*) − *card* (*sumset ?Y B*)) ∗ *card X*) ≤
card (*sumset X B*) ∗ (*card X* − *card ?Y*)
by (*simp add*: *diff-mult-distrib diff-mult-distrib2*)
thus *?thesis* **using** *le-divide-eq card-eq-0-iff hX hXne*
by (*smt* (*z3*) *of-nat-le-0-iff of-nat-mult*)
qed
show *?thesis*
proof−
have *real* (*card* ((*sumset X* (*sumset* {*c*} *B*)) − (*sumset X* (*sumset C B*))))
≤
card (*sumset X B*) − *card* (*sumset ?Y B*) **using** *ineq1* **by** *auto*
also have ... ≤ *card* (*sumset X B*) ∗ (*card X* − *card ?Y*) / *card X* **using**
ineq3 **by** *auto*
also have ... ≤ *card* (*sumset X B*) ∗ *card* (*sumset X* {*c*} − *sumset X C*) /
card X **using** *ineq2*
divide-le-cancel of-nat-less-0-iff of-nat-mono **by** (*smt* (*verit*, *del-insts*)
mult-le-mono2)
finally show *?thesis* **by** *simp*
qed
qed
have *hinsert*: *real* (*card* (*sumset X* (*sumset* (*insert c C*) *B*))) / *real* (*card* (*sumset*
X (*insert c C*))) =
(*card* (*sumset X* (*sumset* {*c*} *B*) − *sumset X* (*sumset C B*)) + *card* (*sumset X*
(*sumset C B*))) /
(*card* (*sumset X* {*c*} − *sumset X C*) + *card* (*sumset X C*))
using *sumset-insert2 card-Un-disjoint finite-sumset hX hB hC Diff-disjoint*
Int-commute
Un-commute finite.emptyI finite.insertI finite-Diff hcase.hyps(*1*)
by (*smt* (*verit*) *Un-Diff-cancel2 insert-is-Un sumset-commute sumset-subset-Un2*)
have *hsplit*: *real* (*card* (*sumset X B*)) ∗ (*card* (*sumset X* {*c*} − *sumset X C*) +
card (*sumset X C*)) / *card X* =
real (*card* (*sumset X B*)) ∗ *card* (*sumset X* {*c*} − *sumset X C*) / *card X* +
card (*sumset X B*) ∗ *card* (*sumset X C*) / *card X*
by (*smt* (*verit*, *ccfv-threshold*) *add-divide-distrib add-mult-distrib2 of-nat-add*
of-nat-mult)
have *hind*: *card* (*sumset X B*) ∗ *card* (*sumset X C*) / *card X* ≥ *card* (*sumset X*
(*sumset C B*))
using *hcase*(*3*)[*OF hCne hCG*] *hXne hCne hcase*(*1*) *hX finite-sumset card-gt-0-iff*
add-mult-distrib2 of-nat-mult card-eq-0-iff card-sumset-0-iff hCG
hXG of-nat-0-less-iff **by** (*metis* (*no-types*, *opaque-lifting*) *divide-le-eq times-divide-eq-left*)
have *real* (*card* (*sumset X B*)) ∗ (*card* (*sumset X* {*c*} − *sumset X C*) + *card*
(*sumset X C*)) / *card X* ≥
(*card* (*sumset X* (*sumset* {*c*} *B*) − *sumset X* (*sumset C B*)) + *card* (*sumset X*
(*sumset C B*)))
using *hsplit hind hstep* **by** *simp*
then have *card* (*sumset X B*) / *card X* ≥

(*card* (*sumset* *X* (*sumset* {*c*} *B*) − *sumset* *X* (*sumset* *C* *B*)) + *card* (*sumset* *X* (*sumset* *C* *B*))) /
(*card* (*sumset* *X* {*c*} − *sumset* *X* *C*) + *card* (*sumset* *X* *C*)) **using** *card-sumset-0-iff hCG hXG*
card-eq-0-iff hC hX hXne hCne divide-self le-divide-eq
by (*smt* (*z3*) *add-is-0 hcase.hyps*(*1*) *of-nat-le-0-iff times-divide-eq-left*)
thus *real* (*card* (*sumset* *X* (*sumset* (*insert* *c* *C*) *B*))) / *real* (*card* (*sumset* *X* (*insert* *c* *C*))) ≤
real (*card* (*sumset* *X* *B*)) / *real* (*card* *X*) **using** *hinsert* **by** *auto*
qed
qed

The following inequality is the result corresponding to Corollary 7.3.6 in Zhao's book [4].

lemma *triangle-ineq-sumsets*:
assumes *hA*: *finite A* **and** *hB*: *finite B* **and** *hC*: *finite C* **and**
hAG : *A* ⊆ *G* **and** *hBG*: *B* ⊆ *G* **and** *hCG*: *C* ⊆ *G*
shows *card A* ∗ *card* (*sumset B C*) ≤ *card* (*sumset A B*) ∗ *card* (*sumset A C*)

proof(*cases A* = {})
case *True*
then show *?thesis* **by** *simp*
next
case *hAne*: *False*
show *card A* ∗ *card* (*sumset B C*) ≤ *card* (*sumset A B*) ∗ *card* (*sumset A C*)
proof(*cases B* = {})
case *True*
then show *?thesis* **by** *simp*
next
case *hBne*: *False*
define *KS* **where** *KS* ≡ (λ*X*. *card* (*sumset X C*) / *real* (*card X*)) ' (*Pow A* − {{}})
define *K* **where** *K* ≡ *Min KS*
define *X* **where** *X* ≡ @*X*. *X* ∈ *Pow A* − {{}} ∧ *K* = *card* (*sumset X C*) / *real* (*card X*)
obtain *KS*: *finite KS KS* ≠ {}
using *KS-def hA hAne* **by** *blast*
then have *K* ∈ *KS*
using *K-def Min-in* **by** *blast*
then have ∃ *X*. *X* ∈ *Pow A* − {{}} ∧ *K* = *card* (*sumset X C*) / *real* (*card X*)
using *KS-def* **by** *blast*
then obtain *X* ∈ *Pow A* − {{}} **and** *Keq*: *K* = *card* (*sumset X C*) / *real* (*card X*)
by (*metis* (*mono-tags*, *lifting*) *X-def someI-ex*)
then have *hX*: *X* ⊆ *A X* ≠ {}
by *auto*
have *hXmin* : ⋀ *Y*. *Y* ⊆ *A* ⟹ *Y* ≠ {} ⟹
card (*sumset X C*) / *card X* ≤ *card* (*sumset Y C*) / *card Y*
using *K-def KS-def Keq Min-le KS*(*1*) **by** *auto*

```
then have hXAineq: card (sumset X C) / card X ≤ card (sumset A C) / card
A
  by (metis hAne subset-refl)
 have haux: real (card (sumset X (sumset B C))) / real (card (sumset X B))
  ≤ real (card (sumset X C)) / real (card X) using triangle-ineq-sumsets-aux[of
X C B]
   hXmin hX hA hAG finite-subset hB hC hBne hBG hC hCG subset-trans by
metis
 have hXAsumset : real (card (sumset X B)) ≤ card (sumset A B)
  using hX(1) card-mono hA finite-sumset hB order-refl sumset-mono
  by (metis of-nat-le-iff)
 have card (sumset B C) ≤ card (sumset X (sumset B C)) using assms hX
  finite-sumset hAG card-le-sumset
  by (metis bot.extremum-uniqueI dual-order.trans infinite-super subsetD subsetI

    sumset-subset-carrier)
 also have ... ≤ (card (sumset X C) / card X) * card (sumset X B) using haux
divide-le-eq
   card-sumset-0-iff hBne hX hB hA finite-subset card-0-eq by (smt (verit) hCG
mult-eq-0-iff
   of-nat-0-eq-iff of-nat-0-le-iff sumset-assoc sumset-subset-carrier)
 also have ... ≤ (card (sumset A C) / card A) * card (sumset A B)
    using hXAineq hXAsumset by (meson divide-nonneg-nonneg mult-mono′
of-nat-0-le-iff)
 finally have card (sumset B C) ≤ (card (sumset A C) * card (sumset A B))
/ card A by simp
 then have card (sumset B C) * card A ≤ card (sumset A C) * card (sumset
A B)
  using le-divide-eq hAne hA card-gt-0-iff by (smt (verit, ccfv-threshold)
  card-0-eq of-nat-le-0-iff of-nat-le-iff of-nat-mult)
 thus card A * card (sumset B C) ≤ card (sumset A B) * card (sumset A C)
  by (simp add: mult.commute)
 qed
qed

end
end
```

5 Background material in additive combinatorics

This section outlines some background definitions and basic lemmas in additive combinatorics based on the notes by Gowers [3].

```
theory Additive-Combinatorics-Preliminaries
 imports
  Pluennecke-Ruzsa-Inequality.Pluennecke-Ruzsa-Inequality
begin
```

5.1 Additive quadruples and additive energy

context *additive-abelian-group*

begin

definition *additive-quadruple*:: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$ **where**
additive-quadruple $a\ b\ c\ d\ \equiv a \in G \wedge b \in G \wedge c \in G \wedge d \in G \wedge a \oplus b = c \oplus d$

lemma *additive-quadruple-aux*:
assumes *additive-quadruple* $a\ b\ c\ d$
shows $d = a \oplus b \ominus c$
by (*metis additive-quadruple-def assms associative commutative inverse-closed invertible*
invertible-right-inverse2)

lemma *additive-quadruple-diff*:
assumes *additive-quadruple* $a\ b\ c\ d$
shows $a \ominus c = d \ominus b$
by (*smt* (*verit*, *del-insts*) *additive-quadruple-def assms associative commutative*
composition-closed inverse-closed invertible invertible-inverse-inverse invertible-right-inverse2)

definition *additive-quadruple-set*:: $'a\ set \Rightarrow ('a \times 'a \times 'a \times 'a)\ set$ **where**
additive-quadruple-set $A \equiv \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$
additive-quadruple $a\ b\ c\ d\}$

lemma *additive-quadruple-set-sub*:
additive-quadruple-set $A \subseteq \{(a, b, c, d) \mid a\ b\ c\ d.\ d = a \oplus b \ominus c \wedge a \in A \wedge b \in A \wedge$
$c \in A \wedge d \in A\}$ **using** *additive-quadruple-set-def additive-quadruple-def additive-quadruple-aux*
by *auto*

definition *additive-energy*:: $'a\ set \Rightarrow real$ **where**
additive-energy $A \equiv$ *card* (*additive-quadruple-set* A) / (*card* A)^3

lemma *card-ineq-aux-quadruples*:
assumes *finite* A
shows *card* (*additive-quadruple-set* A) $\leq$ (*card* A)^3

proof−
define f:: $'a \times 'a \times 'a \times 'a \Rightarrow 'a \times 'a \times 'a$ **where** $f = (\lambda\ (a, b, c, d)\ .\ (a, b, c))$
have *hinj*: *inj-on* f $\{(a, b, c, d) \mid a\ b\ c\ d.\ d = a \oplus b \ominus c \wedge a \in A \wedge b \in A \wedge c \in A \wedge d \in A\}$
unfolding *inj-on-def f-def* **by** *auto*
moreover have *himage*: f ‘ $\{(a, b, c, d) \mid a\ b\ c\ d.\ d = a \oplus b \ominus c \wedge a \in A \wedge b \in A \wedge c \in A \wedge d \in A\} \subseteq A \times A \times A$

unfolding *f-def* **by** *auto*
ultimately have *card* (*additive-quadruple-set* A) $\leq$ *card* ($\{(a, b, c, d) \mid a\ b\ c\ d.\ d = a \oplus b \ominus c \wedge a \in A \wedge b \in A \wedge c \in A \wedge d \in A\}$)
using *card-mono inj-on-finite*[*of f*] *assms additive-quadruple-set-sub finite-SigmaI*
by (*metis* (*no-types*, *lifting*))
also have ... $\leq$ *card* ($A \times A \times A$) **using** *himage hinj assms card-inj-on-le finite-SigmaI*
by (*metis* (*no-types*, *lifting*))
finally show *?thesis* **by** (*simp add*: *card-cartesian-product power3-eq-cube*)
qed

lemma *additive-energy-upper-bound*: *additive-energy* $A \leq 1$

proof (*cases finite* A)
assume *hA*: *finite* A
show *?thesis* **unfolding** *additive-energy-def* **using** *card-ineq-aux-quadruples hA*

card-cartesian-product power3-eq-cube **by** (*simp add*: *divide-le-eq*)
next
assume *infinite* A
thus *?thesis* **unfolding** *additive-energy-def* **by** *simp*
qed

5.2 On sums

definition *f-sum*:: $'a \Rightarrow 'a\ set \Rightarrow nat$ **where**
f-sum $d\ A \equiv$ *card* $\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \oplus b = d\}$

lemma *pairwise-disjnt-sum-1*:
pairwise ($\lambda s\ t.$ *disjnt* (($\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \oplus b = d)\}$) s)
(($\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \oplus b = d)\}$) t)) (*sumset* A A)
unfolding *disjnt-def* **by** (*intro pairwiseI*) (*auto*)

lemma *pairwise-disjnt-sum-2*:
pairwise disjnt (($\lambda\ d.\ \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \oplus b = d\}$) ‘ (*sumset* A A))
unfolding *disjnt-def* **by** (*intro pairwiseI*) (*auto*)

lemma *sum-Union-span*:
assumes $A \subseteq G$
shows $\bigcup$ (($\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \oplus b = d)\}$) ‘ (*sumset* A A)) $= A \times A$

proof
show $\bigcup$ (($\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \oplus b = d)\}$) ‘ (*sumset* A A)) $\subseteq A \times A$ **by** *blast*
next
show $A \times A \subseteq \bigcup$ (($\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \oplus b = d)\}$) ‘ (*sumset*

A A))
  proof (*intro subsetI*)
    fix *x* **assume** *hxA*: $x \in A \times A$
    then obtain *y z* **where** *hxyz*: $x = (y, z)$ **and** *hy*: $y \in A$ **and** *hz*:$z \in A$ **by** *blast*
    show $x \in (\bigcup d\in(sumset\ A\ A).\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \oplus b = d\})$
      using *hy hz assms hxA hxyz* **by** *auto*
  qed
qed

lemma *f-sum-le-card*:
  assumes *finite A* **and** $A \subseteq G$
  shows *f-sum d A* $\leq$ *card A*

proof−
  define *f*:: $('a \times 'a) \Rightarrow 'a$ **where** $f \equiv (\lambda\ (a, b).\ a)$
  have *inj-on f* $\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \oplus b = d\}$
  unfolding *f-def* **proof** (*intro inj-onI*)
  fix *x y* **assume** $x \in \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \oplus b = d\}$ **and**
    $y \in \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \oplus b = d\}$ **and**
    hcase: (*case x of* $(a, b) \Rightarrow a$) = (*case y of* $(a, b) \Rightarrow a$)
  then obtain *x1 x2 y1 y2* **where** *hx*: $x = (x1, x2)$ **and** *hy*:$y = (y1, y2)$ **and** *h1*: $x1 \oplus x2 = d$ **and**
    h2: $y1 \oplus y2 = d$ **and** *hx1*: $x1 \in A$ **and** *hx2*: $x2 \in A$ **and** *hy1*: $y1 \in A$ **and** *hy2*: $y2 \in A$ **by** *blast*
  have *hxsub*: $x2 = d \ominus x1$
    using *h1 hx1 hx2 assms* **by** (*metis additive-abelian-group.inverse-closed composition-closed*
      additive-abelian-group-axioms commutative invertible invertible-left-inverse2 subsetD)
  have *hysub*: $y2 = d \ominus y1$
   using *h2 hy1 hy2 assms* **by** (*metis inverse-closed commutative composition-closed hy1 hy2*
      invertible invertible-left-inverse2 subset-iff)
  show $x = y$ **using** *hx hy hxsub hysub hcase* **by** *auto*
  qed
  moreover have *f* ‘ $\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \oplus b = d\} \subseteq A$ **using** *f-def* **by** *auto*
  ultimately show *?thesis* **using** *card-mono assms f-sum-def card-image*[*of f*]
    by (*metis* (*mono-tags*, *lifting*))
qed

lemma *f-sum-card*:
  assumes $A \subseteq G$ **and** *hA*: *finite A*
  shows $(\sum\ d \in (sumset\ A\ A).\ (f\text{-}sum\ d\ A)) = (card\ A)\hat{\ }2$

proof−
  have *fin*: $\forall\ X \in ((\lambda\ d.\ \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \oplus b = d)\ \})$ ‘ $(sumset\ A\ A)).$ *finite X*
  proof

fix X **assume** hX: $X \in (\lambda d.\ \{(a,\ b) \mid a\ b.\ a \in A \land b \in A \land a \oplus b = d\})$ ‘ (*sumset A A*)
then obtain d **where** hXd: $X = \{(a,\ b) \mid a\ b.\ a \in A \land b \in A \land a \oplus b = d\}$ **by** *blast*
show *finite X* **using** *hA hXd finite-subset finite-cartesian-product*
by (*smt* (*verit, best*) *mem-Collect-eq mem-Sigma-iff rev-finite-subset subrelI*)
qed
have ($\sum d \in$(*sumset A A*). *f-sum d A*) = *card* ($A \times A$)
unfolding *f-sum-def*
using *sum-card-image*[*of sumset A A* (λ *d* .$\{(a,\ b) \mid a\ b.\ a \in A \land b \in A \land (a \oplus b = d)\}$)]
pairwise-disjnt-sum-1 hA finite-sumset card-Union-disjoint[*of* (($\lambda d.\ \{(a,\ b) \mid a\ b.\ a \in A \land b \in A \land a \oplus b = d\}$) ‘ *sumset A A*)]
fin pairwise-disjnt-sum-2 hA finite-sumset sum-Union-span assms **by** *auto*
thus *?thesis* **using** *card-cartesian-product power2-eq-square* **by** *metis*
qed

lemma *f-sum-card-eq*:
assumes $A \subseteq G$
shows $\forall\ x \in$ *sumset A A*. (*f-sum x A*)^*2* =
card $\{(a,\ b,\ c,\ d) \mid a\ b\ c\ d.\ a \in A \land b \in A \land c \in A \land d \in A \land$
additive-quadruple a b c d $\land\ a \oplus b = x \land c \oplus d = x\}$

proof
fix x **assume** $x \in$ *sumset A A*
define C **where** hC: $C = \{(a,\ b,\ c,\ d) \mid a\ b\ c\ d.\ a \in A \land b \in A \land c \in A \land d \in A \land$
additive-quadruple a b c d $\land\ a \oplus b = x \land c \oplus d = x\}$
define f:: $'a \times 'a \times 'a \times 'a \Rightarrow ('a \times 'a) \times ('a \times 'a)$ **where** $f = (\lambda\ (a,\ b,\ c,\ d).\ ((a,\ b),\ (c,\ d)))$
have *hfinj*: *inj-on f C* **unfolding** *f-def* **by** (*intro inj-onI*) (*auto*)
have f ‘ $C = \{((a,b),\ (c,d)) \mid a\ b\ c\ d.\ a \in A \land b \in A \land c \in A \land d \in A \land a \oplus b = x \land c \oplus d = x\}$
proof
show f ‘ $C \subseteq \{((a,\ b),\ (c,\ d)) \mid a\ b\ c\ d.\ a \in A \land b \in A \land c \in A \land d \in A \land a \oplus b = x \land c \oplus d = x\}$
unfolding *f-def hC* **by** *auto*
next
show $\{((a,\ b),\ c,\ d) \mid a\ b\ c\ d.\ a \in A \land b \in A \land c \in A \land d \in A \land a \oplus b = x \land c \oplus d = x\} \subseteq f$ ‘ C
proof
fix z **assume** $z \in \{((a,\ b),\ (c,\ d)) \mid a\ b\ c\ d.\ a \in A \land b \in A \land c \in A \land d \in A \land a \oplus b = x \land c \oplus d = x\}$
then obtain $a\ b\ c\ d$ **where** hz: $z = ((a,\ b),\ (c,\ d))$ **and** ha: $a \in A$ **and** hb: $b \in A$ **and** hc: $c \in A$ **and** hd: $d \in A$
and hab: $a \oplus b = x$ **and** hcd: $c \oplus d = x$ **by** *blast*
then have *habcd*: $(a,\ b,\ c,\ d) \in C$ **using** *additive-quadruple-def assms hC* **by** *auto*
show $z \in f$ ‘ C **using** *hz f-def habcd* **by** *force*

qed
qed
moreover have $\{((a, b), c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge a \oplus b = x \wedge c \oplus d = x\} =$
$\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \oplus b = x\} \times \{(c, d) \mid c\ d.\ c \in A \wedge d \in A \wedge c \oplus d = x\}$ **by** *blast*
ultimately have *card C* = *card* $(\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \oplus b = x\})$ ^ *2*
using *hfinj card-image*[*of f*] *card-cartesian-product* **by** (*metis* (*no-types*, *lifting*) *Sigma-cong power2-eq-square*)
thus (*f-sum x A*)^*2* = *card* $(\{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$
additive-quadruple a b c d $\wedge\ a \oplus b = x \wedge c \oplus d = x\})$ **using** *hC f-sum-def* **by** *auto*
qed

lemma *pairwise-disjoint-sum*:
pairwise ($\lambda s\ t.$ *disjnt* (($\lambda\ x.\ \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$
additive-quadruple a b c d $\wedge\ a \oplus b = x \wedge c \oplus d = x\}$) *s*)
(($\lambda\ x.\ \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$
additive-quadruple a b c d $\wedge\ a \oplus b = x \wedge c \oplus d = x\}$) *t*)) (*sumset A A*)
unfolding *disjnt-def* **by** (*intro pairwiseI*) (*auto*)

lemma *pairwise-disjnt-quadruple-sum*:
pairwise disjnt (($\lambda\ x.\ \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$ *additive-quadruple a b c d* $\wedge\ a \oplus b = x \wedge c \oplus d = x\}$) ' (*sumset A A*))
unfolding *disjnt-def* **by** (*intro pairwiseI*) (*auto*)

lemma *quadruple-sum-Union-eq*:
$\bigcup$ (($\lambda\ x.\ \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$
additive-quadruple a b c d $\wedge\ a \oplus b = x \wedge c \oplus d = x\}$) ' (*sumset A A*)) = *additive-quadruple-set A*

proof
show ($\bigcup x \in$*sumset A A*.
$\{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$ *additive-quadruple a b c d* $\wedge$
$a \oplus b = x \wedge c \oplus d = x\}) \subseteq$ *additive-quadruple-set A*
unfolding *additive-quadruple-set-def* **by** (*intro Union-least*) (*auto*)
next
show *additive-quadruple-set A* $\subseteq$ ($\bigcup x \in$*sumset A A*.
$\{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$
additive-quadruple a b c d $\wedge\ a \oplus b = x \wedge c \oplus d = x\})$
unfolding *additive-quadruple-set-def additive-quadruple-def* **by** (*intro subsetI*) (*auto*)
qed

lemma *f-sum-card-quadruple-set*:
  assumes *hAG*: *A* ⊆ *G* **and** *hA*: *finite A*
  shows (∑ *d* ∈ (*sumset A A*). (*f-sum d A*)^*2*) = *card* (*additive-quadruple-set A*)

proof−
  have *fin*: ∀ *X* ∈ ((λ *x*. {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧
   additive-quadruple a b c d ∧ *a* ⊕ *b* = *x* ∧ *c* ⊕ *d* = *x*}) ‘ (*sumset A A*)). *finite X*
  proof
    fix *X* **assume** *X* ∈ (λ*x*. {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧
      additive-quadruple a b c d ∧ *a* ⊕ *b* = *x* ∧ *c* ⊕ *d* = *x*}) ‘ *sumset A A*
    then obtain *x* **where** *hX*: *X* = {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧
      additive-quadruple a b c d ∧ *a* ⊕ *b* = *x* ∧ *c* ⊕ *d* = *x*}
      by *blast*
    show *finite X* **using** *hA hX finite-subset finite-cartesian-product*
      by (*smt* (*verit*, *best*) *mem-Collect-eq mem-Sigma-iff rev-finite-subset subrelI*)
  qed
  have (∑ *d*∈*sumset A A*. (*f-sum d A*)^2) =
    card (⋃ ((λ *x*. {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧
    additive-quadruple a b c d ∧ *a* ⊕ *b* = *x* ∧ *c* ⊕ *d* = *x*}) ‘ (*sumset A A*)))
   using *f-sum-card-eq hAG sum-card-image*[*of sumset A A* (λ*x*. {(*a*, *b*, *c*, *d*) |*a b c d*.
      a ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *additive-quadruple a b c d* ∧ *a* ⊕ *b* = *x* ∧ *c* ⊕ *d* = *x*})]
      pairwise-disjoint-sum card-Union-disjoint[*of* (λ *x*. {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧
      c ∈ *A* ∧ *d* ∈ *A* ∧ *additive-quadruple a b c d* ∧ *a* ⊕ *b* = *x* ∧ *c* ⊕ *d* = *x*}) ‘ (*sumset A A*)]
      fin pairwise-disjnt-quadruple-sum hA finite-sumset **by** *auto*
  then show *?thesis* **using** *quadruple-sum-Union-eq* **by** *auto*
qed

lemma *f-sum-card-quadruple-set-additive-energy*: **assumes** *A* ⊆ *G* **and** *finite A*
  shows (∑ *d* ∈ *sumset A A*. (*f-sum d A*)^*2*) = *additive-energy A* ∗ (*card A*)^*3*
  using *assms f-sum-card-quadruple-set additive-energy-def* **by** *force*

definition *popular-sum*:: ′*a* ⇒ *real* ⇒ ′*a set* ⇒ *bool* **where**
  popular-sum d ϑ *A* ≡ *f-sum d A* ≥ ϑ ∗ *of-real* (*card A*)

definition *popular-sum-set*:: *real* ⇒ ′*a set* ⇒ ′*a set* **where**
  popular-sum-set ϑ *A* ≡ {*d* ∈ *sumset A A*. *popular-sum d* ϑ *A*}

5.3 On differences

The following material is directly analogous to the material given previously on sums. All definitions and lemmas are the corresponding ones for differences. E.g. *f-diff* corresponds to *f-sum*.

definition *f-diff*:: $'a \Rightarrow 'a$ *set* $\Rightarrow$ *nat* **where**
f-diff d A $\equiv$ *card* $\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\}$

lemma *pairwise-disjnt-diff-1*:
pairwise ($\lambda s\ t.$ *disjnt* $((\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \ominus b = d)\})\ s)$
$((\lambda\ d.\ \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \ominus b = d)\})\ t))$ (*differenceset A A*)
using *disjnt-def* **by** (*intro pairwiseI*) (*auto*)

lemma *pairwise-disjnt-diff-2*:
pairwise disjnt $((\lambda\ d.\ \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\})$ ‘ (*differenceset A A*))
unfolding *disjnt-def* **by** (*intro pairwiseI*) (*auto*)

lemma *diff-Union-span*:
assumes $A \subseteq G$
shows $\bigcup\ ((\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \ominus b = d)\})$ ‘ (*differenceset A A*)) $= A \times A$

proof
show $\bigcup\ ((\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \ominus b = d)\})$ ‘ (*differenceset A A*)) $\subseteq A \times A$
by *blast*
next
show $A \times A \subseteq \bigcup\ ((\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \ominus b = d)\})$ ‘ (*differenceset A A*))
proof (*intro subsetI*)
fix *x* **assume** *hxA*: $x \in A \times A$
then obtain *y z* **where** *hxyz*: $x = (y, z)$ **and** *hy*: $y \in A$ **and** *hz*:$z \in A$ **by** *blast*
show $x \in (\bigcup d \in (\textit{differenceset } A\ A).\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\})$
using *hy hz assms hxA hxyz* **by** *auto*
qed
qed

lemma *f-diff-le-card*:
assumes *finite A* **and** $A \subseteq G$
shows *f-diff d A* $\leq$ *card A*

proof−
define *f*:: $('a \times 'a) \Rightarrow 'a$ **where** $f \equiv (\lambda\ (a, b).\ a)$
have *inj-on f* $\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\}$
unfolding *f-def* **proof** (*intro inj-onI*)
fix *x y* **assume** $x \in \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\}$ **and**
$y \in \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\}$ **and**
hcase: (*case x of* $(a, b) \Rightarrow a$) = (*case y of* $(a, b) \Rightarrow a$)
then obtain *x1 x2 y1 y2* **where** *hx*: $x = (x1, x2)$ **and** *hy*:$y = (y1, y2)$ **and** *h1*: $x1 \ominus x2 = d$ **and**
h2: $y1 \ominus y2 = d$ **and** *hx1*: $x1 \in A$ **and** *hx2*: $x2 \in A$ **and** *hy1*: $y1 \in A$ **and** *hy2*: $y2 \in A$ **by** *blast*

have *hxsub*: $x2 = x1 \ominus d$
using *h1 assms associative commutative composition-closed hx1 hx2*
by (*smt* (*verit*, *best*) *inverse-closed invertible invertible-left-inverse2 subset-iff*)
have *hysub*: $y2 = y1 \ominus d$
using *h2 assms associative commutative composition-closed hy1 hy2*
by (*smt* (*verit*, *best*) *inverse-closed invertible invertible-left-inverse2 subset-iff*)
show $x = y$ **using** *hx hy hxsub hysub hcase* **by** *auto*
qed
moreover have f ' $\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\} \subseteq A$ **using** *f-def* **by** *auto*
ultimately show *?thesis* **using** *card-mono assms f-diff-def card-image*[*of f*]
by (*metis* (*mono-tags*, *lifting*))
qed

lemma *f-diff-card*:
assumes $A \subseteq G$ **and** *hA*: *finite A*
shows $(\sum d \in (differenceset\ A\ A).\ f\text{-}diff\ d\ A) = (card\ A)\hat{\ }2$

proof−
have *fin*: $\forall X \in ((\lambda\ d.\ \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \ominus b = d)\})$ ' $(differenceset\ A\ A))$.
finite X
proof
fix *X* **assume** *hX*: $X \in (\lambda d.\ \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\})$ ' $(differenceset\ A\ A)$
then obtain *d* **where** *hXd*: $X = \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\}$ **and**
$d \in (differenceset\ A\ A)$ **by** *blast*
have *hXA*: $X \subseteq A \times A$ **using** *hXd* **by** *blast*
show *finite X* **using** *hXA hA finite-subset* **by** *blast*
qed
have $(\sum d\in(differenceset\ A\ A).\ f\text{-}diff\ d\ A) = card\ (A \times A)$
unfolding *f-diff-def* **using** *sum-card-image*[*of differenceset A A*
$(\lambda\ d\ .\{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge (a \ominus b = d)\})$] *pairwise-disjnt-diff-1*
card-Union-disjoint[*of* $((\lambda d.\ \{(a, b) \mid a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\})$ ' *differenceset A A*)]
fin pairwise-disjnt-diff-2 diff-Union-span assms hA finite-minusset finite-sumset **by** *auto*
thus *?thesis* **using** *card-cartesian-product power2-eq-square* **by** *metis*
qed

lemma *f-diff-card-eq*:
assumes $A \subseteq G$
shows $\forall\ x \in differenceset\ A\ A.\ (f\text{-}diff\ x\ A)\hat{\ }2 =$
$card\ \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$
$additive\text{-}quadruple\ a\ b\ c\ d \wedge a \ominus c = x \wedge d \ominus b = x\}$

proof
fix *x* **assume** $x \in differenceset\ A\ A$

define *C* **where** *hC*: *C*= {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *additive-quadruple a b c d* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x* }
define *f*:: *'a* × *'a* × *'a* × *'a* ⇒ (*'a* × *'a*) × (*'a* × *'a*) **where** *f* = (λ (*a*, *b*, *c*, *d*). ((*a*, *c*), (*d*, *b*)))
have *hfinj*: *inj-on f C* **using** *f-def* **by** (*intro inj-onI*) (*auto*)
have *f* ' *C* = {((*a*,*c*), (*d*,*b*)) | *a b c d*.
a ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}
proof
show *f* ' *C* ⊆ {((*a*, *c*), (*d*, *b*)) |*a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}
unfolding *f-def hC* **by** *auto*
next
show {((*a*, *c*), *d*, *b*) |*a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*} ⊆ *f* ' *C*
proof
fix *z* **assume** *z* ∈ {((*a*, *c*), (*d*, *b*)) |*a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}
then obtain *a b c d* **where** *hz*: *z* = ((*a*, *c*), (*d*, *b*)) **and** *ha*: *a* ∈ *A* **and** *hb*: *b* ∈ *A* **and** *hc*: *c* ∈ *A* **and** *hd*: *d* ∈ *A*
and *hab*: *a* ⊖ *c* = *x* **and** *hcd*: *d* ⊖ *b* = *x* **by** *blast*
have *additive-quadruple a b c d*
using *assms* **by** (*metis* (*no-types*, *lifting*) *ha hb hc hd additive-quadruple-def associative*
commutative composition-closed hab hcd inverse-closed invertible invertible-right-inverse2 subset-eq)
then have *habcd*: (*a*, *b*, *c*, *d*) ∈ *C* **using** *hab hcd hC ha hb hc hd* **by** *blast*
show *z* ∈ *f* ' *C* **using** *hz f-def habcd image-iff* **by** *fastforce*
qed
qed
moreover have {((*a*, *c*), (*d*, *b*))|*a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*} =
{(*a*, *c*) | *a c*. *a* ∈ *A* ∧ *c* ∈ *A* ∧ *a* ⊖ *c* = *x*} × {(*d*, *b*) | *d b*. *d* ∈ *A* ∧ *b* ∈ *A* ∧ *d* ⊖ *b* = *x*} **by** *blast*
moreover have *card C* = *card* (*f* ' *C*) **using** *hfinj card-image*[*of f*] **by** *auto*
ultimately have *card C* = *card* ({(*a*, *c*) | *a c*. *a* ∈ *A* ∧ *c* ∈ *A* ∧ *a* ⊖ *c* = *x*}) ^ *2*
using *hfinj card-image*[*of f*] *card-cartesian-product Sigma-cong power2-eq-square* **by** (*smt* (*verit*, *best*))
thus (*f-diff x A*)^2 = *card* {(*a*, *b*, *c*, *d*) |*a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *additive-quadruple a b c d* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}
using *f-diff-def hC* **by** *simp*
qed

lemma *pairwise-disjoint-diff*:
pairwise (λ*s t*. *disjnt* ((λ *x*. {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *additive-quadruple a b c d* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}) *s*)
((λ *x*. {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *additive-quadruple a b c d* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}) *t*)) (*differenceset A A*)
unfolding *disjnt-def* **by** (*intro pairwiseI*) (*auto*)

lemma *pairwise-disjnt-quadruple-diff*:
pairwise disjnt ((λ *x*. {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *additive-quadruple a b c d* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}) ' (*differenceset A A*))
unfolding *disjnt-def* **by** (*intro pairwiseI*) (*auto*)

lemma *quadruple-diff-Union-eq*:
⋃ ((λ *x*. {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *additive-quadruple a b c d* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}) ' (*differenceset A A*)) =

additive-quadruple-set A

proof
show (⋃*x*∈*differenceset A A*. {(*a*, *b*, *c*, *d*) |*a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧
additive-quadruple a b c d ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}) ⊆ *additive-quadruple-set A*
unfolding *additive-quadruple-set-def* **by** (*intro Union-least*)(*auto*)
next
show *additive-quadruple-set A* ⊆ (⋃*x*∈*differenceset A A*.
{(*a*, *b*, *c*, *d*) |*a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧ *additive-quadruple a b c d* ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*})
proof (*intro subsetI*)
fix *x* **assume***x* ∈ *additive-quadruple-set A*
then obtain *x1 x2 x3 x4* **where** *hx*: *x* = (*x1*, *x2*, *x3*, *x4*) **and** *hx1*: *x1* ∈ *A* **and** *hx2*: *x2* ∈ *A* **and** *hx3*: *x3* ∈ *A*
and *hx4*: *x4* ∈ *A* **and** *hxadd*: *additive-quadruple x1 x2 x3 x4*
using *additive-quadruple-set-def* **by** *auto*
have *hxmem*: (*x1*, *x2*, *x3*, *x4*) ∈ {(*a*, *b*, *c*, *d*) |*a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧
additive-quadruple a b c d ∧ *a* ⊖ *c* = *x1* ⊖ *x3* ∧ *d* ⊖ *b* = *x1* ⊖ *x3*}
using *additive-quadruple-diff hx1 hx2 hx3 hx4 hxadd* **by** *auto*
show *x* ∈ (⋃*x*∈*differenceset A A*. {(*a*, *b*, *c*, *d*) |*a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧
additive-quadruple a b c d ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*})
using *hx hxmem hx1 hx3 additive-quadruple-def hxadd* **by** *auto*
qed
qed

lemma *f-diff-card-quadruple-set*:
assumes *hAG*: *A* ⊆ *G* **and** *hA*: *finite A*
shows (∑ *d* ∈ (*differenceset A A*). (*f-diff d A*)^2) = *card* (*additive-quadruple-set A*)

proof−
have *fin*: ∀ *X* ∈ ((λ *x*. {(*a*, *b*, *c*, *d*) | *a b c d*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *c* ∈ *A* ∧ *d* ∈ *A* ∧
additive-quadruple a b c d ∧ *a* ⊖ *c* = *x* ∧ *d* ⊖ *b* = *x*}) ' (*differenceset A A*)). *finite X*

proof
fix X **assume** $X \in (\lambda x. \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$
additive-quadruple $a\ b\ c\ d \wedge a \ominus c = x \wedge d \ominus b = x\})$ ‘ *differenceset* $A\ A$
then obtain x **where** hX: $X = \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$
additive-quadruple $a\ b\ c\ d \wedge a \ominus c = x \wedge d \ominus b = x\}$ **and** $x \in$ *differenceset* $A\ A$ **by** *blast*
show *finite X* **using** *hX hA finite-subset finite-cartesian-product*
by (*smt* (*verit*, *best*) *mem-Collect-eq mem-Sigma-iff rev-finite-subset subrelI*)
qed
have $(\sum d \in$*differenceset* $A\ A.$ (*f-diff* $d\ A)^2) =$ *card* $(\bigcup\ ((\lambda\ x. \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge$
$b \in A \wedge c \in A \wedge d \in A \wedge$ *additive-quadruple* $a\ b\ c\ d \wedge a \ominus c = x \wedge d \ominus b = x\})$ ‘ (*differenceset* $A\ A$)))
using *f-diff-card-eq hAG sum-card-image*[*of differenceset* $A\ A$ $(\lambda x. \{(a, b, c, d) \mid a\ b\ c\ d.$
$a \in A \wedge b \in A \wedge c \in A \wedge d \in A \wedge$ *additive-quadruple* $a\ b\ c\ d \wedge a \ominus c = x \wedge d \ominus b = x\})$]
pairwise-disjoint-diff card-Union-disjoint[*of* $(\lambda\ x. \{(a, b, c, d) \mid a\ b\ c\ d.\ a \in A \wedge$
$b \in A \wedge c \in A \wedge d \in A \wedge$ *additive-quadruple* $a\ b\ c\ d \wedge a \ominus c = x \wedge d \ominus b = x\})$ ‘
(*differenceset* $A\ A$)] *fin pairwise-disjnt-quadruple-diff hA finite-minusset finite-sumset* **by** *auto*
thus *?thesis* **using** *quadruple-diff-Union-eq* **by** *auto*
qed

lemma *f-diff-card-quadruple-set-additive-energy*: **assumes** $A \subseteq G$ **and** *finite A*
shows $(\sum\ d \in$ *differenceset* $A\ A.$ (*f-diff* $d\ A)^{*2*}) =$ *additive-energy* $A *$ (*card* A)^*3*
using *assms f-diff-card-quadruple-set additive-energy-def* **by** *force*

definition *popular-diff*:: $'a \Rightarrow real \Rightarrow 'a\ set \Rightarrow bool$ **where**
popular-diff $d\ \vartheta\ A \equiv$ *f-diff* $d\ A \geq \vartheta *$ *of-real* (*card* A)

definition *popular-diff-set*:: $real \Rightarrow 'a\ set \Rightarrow 'a\ set$ **where**
popular-diff-set $\vartheta\ A \equiv \{d \in$ *differenceset* $A\ A.$ *popular-diff* $d\ \vartheta\ A\}$

end
end

6 Results on lower bounds on additive energy

theory *Additive-Energy-Lower-Bounds*
imports
Additive-Combinatorics-Preliminaries
Miscellaneous-Lemmas

begin

context *additive-abelian-group*

begin

The following corresponds to Proposition 2.11 in Gowers's notes [3].

proposition *additive-energy-lower-bound-sumset*: **fixes** *C*::*real*
assumes *finite A* **and** $A \subseteq G$ **and** $(card\ (sumset\ A\ A)) \leq C * card\ A$ **and** $card\ A \neq 0$
shows $additive\text{-}energy\ A \geq 1/C$

proof−
have $(card\ A)\hat{}2 = (\sum x \in sumset\ A\ A.\ real\ (f\text{-}sum\ x\ A))$
using *assms f-sum-card* **by** (*metis of-nat-sum*)
also have ... $\leq (card((sumset\ A\ A))\ powr(1/2)) * (\sum x \in sumset\ A\ A\ .\ (f\text{-}sum\ x\ A)\hat{}2)\ powr(1/2)$
using *Cauchy-Schwarz-ineq-sum2*[*of* λ *d. 1* λ *d. f-sum d A*] **by** *auto*
also have ...$\leq ((C * (card\ A))\ powr(1/2)) * ((\sum x \in sumset\ A\ A\ .\ (f\text{-}sum\ x\ A)\hat{}2))\ powr(1/2)$
by (*metis mult.commute mult-left-mono assms*(*3*) *of-nat-0-le-iff powr-ge-pzero powr-mono2*
zero-le-divide-1-iff zero-le-numeral)
finally have $((card\ A)\hat{}2)\hat{}2 \leq (((C * (card\ A))\ powr(1/2)) * ((\sum x \in sumset\ A\ A\ .\ (f\text{-}sum\ x\ A)\hat{}2))\ powr(1/2))\hat{}2$
by (*metis of-nat-0-le-iff of-nat-power-eq-of-nat-cancel-iff power-mono*)
then have $(card\ A)\hat{}4 \leq (((C * (card\ A)) * ((\sum x \in sumset\ A\ A.\ (f\text{-}sum\ x\ A)\hat{}2)))\ powr\ (1/2))\hat{}2$
by (*smt* (*verit*) *assms of-nat-0-le-iff powr-mult*
mult.left-commute power2-eq-square power3-eq-cube power4-eq-xxxx power-commutes)
then have $(card\ A)\hat{}4 \leq ((C * (card\ A)) * ((\sum x \in sumset\ A\ A.\ (f\text{-}sum\ x\ A)\hat{}2)))$
using *assms powr-half-sqrt of-nat-0 of-nat-le-0-iff power-mult-distrib*
real-sqrt-pow2 **by** (*smt* (*verit, best*) *powr-mult*)
moreover have $additive\text{-}energy\ A = (\sum x \in sumset\ A\ A.\ (f\text{-}sum\ x\ A)\hat{}2)/(card\ A)\hat{}3$
using *additive-energy-def f-sum-card-quadruple-set assms* **by** *simp*
moreover then have $additive\text{-}energy\ A * (card\ A)\hat{}3 = (\sum x \in sumset\ A\ A.\ (f\text{-}sum\ x\ A)\hat{}2)$
using *assms* **by** *simp*
ultimately have $(additive\text{-}energy\ A) \geq ((card\ A)\hat{}4)/(C * (card\ A)\hat{}4)$
using *additive-energy-upper-bound*
additive-abelian-group-axioms assms divide-le-eq divide-le-eq-1-pos mult.left-commute

mult-left-mono of-nat-0-eq-iff of-nat-0-le-iff power-eq-0-iff power3-eq-cube power4-eq-xxxx
linorder-not-less mult.assoc mult-zero-left of-nat-0-less-iff of-nat-mult
order-trans-rules(*23*) *times-divide-eq-right* **by** (*smt* (*verit*) *card-sumset-0-iff*
div-by-1 mult-cancel-left1 nonzero-mult-div-cancel-left nonzero-mult-divide-mult-cancel-right

nonzero-mult-divide-mult-cancel-right2 of-nat-1 of-nat-le-0-iff times-divide-eq-left)
then show *?thesis* **by** (*simp add*: *assms*)
qed

An analogous version of Proposition 2.11 where the assumption is on a difference set is given below. The proof is identical to the proof of *additive-energy-lower-bound-sumset* above (with the obvious modifications).

proposition *additive-energy-lower-bound-differenceset*: **fixes** *C*::*real*
assumes *finite A* **and** $A \subseteq G$ **and** (*card* (*differenceset A A*)) $\leq C *$ *card A* **and** *card A* $\neq$ *0*
shows *additive-energy A* $\geq$ *1/C*

proof−
have (*card A*)^*2* = ($\sum$ $x \in$ *differenceset A A*. *real* (*f-diff x A*))
using *assms f-diff-card* **by** (*metis of-nat-sum*)
also have ... $\leq$ (*card*((*differenceset A A*)) *powr* (*1/2*)) $*$ ($\sum$ $x \in$ *differenceset A A* . (*f-diff x A*)^*2*) *powr*(*1/2*)
using *Cauchy-Schwarz-ineq-sum2*[*of* λ *d*. *1* λ *d*. *f-diff d A*] **by** *auto*
also have ...$\leq$ ((*C* $*$ (*card A*))*powr* (*1/2*)) $*$ (($\sum$ $x \in$ *differenceset A A* . (*f-diff x A*)^*2*)) *powr*(*1/2*)
by (*metis mult.commute mult-left-mono assms*(*3*) *of-nat-0-le-iff powr-ge-pzero powr-mono2*
zero-le-divide-1-iff zero-le-numeral)
finally have ((*card A*)^*2*)^*2* $\leq$ (((*C* $*$ (*card A*))*powr* (*1/2*)) $*$ (($\sum$ $x \in$ *differenceset A A* . (*f-diff x A*)^*2*)) *powr*(*1/2*))^*2*
by (*metis of-nat-0-le-iff of-nat-power-eq-of-nat-cancel-iff power-mono*)
then have (*card A*)^*4* $\leq$ (((*C* $*$ (*card A*)) $*$ (($\sum$ $x \in$ *differenceset A A*. (*f-diff x A*)^*2*))) *powr* (*1/2*))^*2*
by (*smt* (*verit*) *assms of-nat-0-le-iff powr-mult*
mult.left-commute power2-eq-square power3-eq-cube power4-eq-xxxx power-commutes)
then have (*card A*)^*4* $\leq$ ((*C* $*$ (*card A*)) $*$ (($\sum$ $x \in$ *differenceset A A* . (*f-diff x A*)^*2*)))
using *assms powr-half-sqrt of-nat-0 of-nat-le-0-iff power-mult-distrib*
real-sqrt-pow2 **by** (*smt* (*verit*, *best*) *powr-mult*)
moreover have *additive-energy A* = ($\sum$ $x \in$ *differenceset A A*. (*f-diff x A*)^*2*)/(*card A*)^*3*
using *additive-energy-def f-diff-card-quadruple-set assms* **by** *simp*
moreover then have *additive-energy A* $*$ (*card A*)^*3* = ($\sum$ $x \in$ *differenceset A A*. (*f-diff x A*)^*2*)
using *assms* **by** *simp*
ultimately have (*additive-energy A*) $\geq$ ((*card A*)^*4*)/ (*C* $*$ (*card A*)^*4*)
using *additive-energy-upper-bound*
additive-abelian-group-axioms assms divide-le-eq divide-le-eq-1-pos mult.left-commute

mult-left-mono of-nat-0-eq-iff of-nat-0-le-iff power-eq-0-iff power3-eq-cube power4-eq-xxxx
linorder-not-less mult.assoc mult-zero-left of-nat-0-less-iff of-nat-mult
order-trans-rules(*23*) *times-divide-eq-right* **by** (*smt* (*verit*) *card-sumset-0-iff*

div-by-1 mult-cancel-left1 nonzero-mult-div-cancel-left nonzero-mult-divide-mult-cancel-right

nonzero-mult-divide-mult-cancel-right2 of-nat-1 of-nat-le-0-iff times-divide-eq-left)
then show *?thesis* **by** (*simp add*: *assms*)
qed

end
end

7 Towards the proof of the Balog–Szemerédi–Gowers Theorem

theory *Balog-Szemeredi-Gowers-Main-Proof*
imports
Prob-Space-Lemmas
Graph-Theory-Preliminaries
Sumset-Triangle-Inequality
Additive-Combinatorics-Preliminaries
begin

context *additive-abelian-group*

begin

After having introduced all the necessary preliminaries in the imported files, we are now ready to follow the chain of the arguments for the main proof as in Gowers's notes [3].

The following lemma corresponds to Lemma 2.13 in Gowers's notes [3].

lemma (**in** *fin-bipartite-graph*) *proportion-bad-pairs-subset-bipartite*:
fixes *c*::*real*
assumes $c > 0$
obtains X' **where** $X' \subseteq X$ **and** *card* $X' \geq$ *density* $*$ *card* X / *sqrt 2* **and**
card (*bad-pair-set* X' Y c) / (*card* X')^2 ≤ 2 ∗ *c* / *density*^2
proof (*cases density* = *0*)
case *True*
then show *?thesis* **using** *that*[*of* {}] *bad-pair-set-def* **by** *auto*
next
case *False*
then have *dgt0*: *density* > *0* **using** *density-simp* **by** *auto*
let *?M* = *uniform-count-measure Y*
interpret *P*: *prob-space ?M*
by (*simp add*: *Y-not-empty partitions-finite prob-space-uniform-count-measure*)
have *sp*: *space ?M* = *Y*
by (*simp add*: *space-uniform-count-measure*)

have *avg-degree*: *P.expectation* (λ *y* . *card* (*neighborhood y*)) = *density* ∗ (*card* *X*)
proof −

have *density* = ($\sum y \in Y$. *degree y*)/(*card X* ∗ *card Y*)
using *edge-size-degree-sumY density-simp* **by** *simp*
then have *d*: *density* ∗ (*card X*) = ($\sum y \in Y$. *degree y*)/(*card Y*)
using *card-edges-between-set edge-size-degree-sumY partitions-finite*(*1*) *partitions-finite*(*2*) **by** *auto*
have *P.expectation* (λ *y* . *card* (*neighborhood y*)) = *P.expectation* (λ *y* . *degree y*)
using *alt-deg-neighborhood* **by** *simp*
also have ... =($\sum y \in Y$. *degree y*)/(*card Y*) **using** *P.expectation-uniform-count*
by (*simp add*: *partitions-finite*(*2*))
finally show *?thesis* **using** *d* **by** *simp*
qed

then have *card-exp-gt*: *P.expectation* (λ *y*. (*card* (*neighborhood y*))^*2*) ≥ *density*^*2* ∗ (*card X*)^*2*
proof −
have *P.expectation* (λ *y*. (*card* (*neighborhood y*))^*2*) ≥ (*P.expectation* (λ *y* . *card* (*neighborhood y*)))^*2*
using *P.cauchy-schwarz-ineq-var-uniform partitions-finite*(*2*) **by** *auto*
thus *?thesis* **using** *avg-degree*
by (*metis of-nat-power power-mult-distrib*)
qed

define *B* **where** *B* ≡ *bad-pair-set X Y c*
define *B′* **where** *B′* ≡ λ *y*. *bad-pair-set* (*neighborhood y*) *Y c*
have *finB*: *finite B* **using** *bad-pair-set-finite partitions-finite B-def* **by** *auto*
have ⋀ *x*. *x* ∈ *X* ⟹ *x* ∈ *V* **using** *partitions-ss*(*1*) **by** *auto*
have *card B* ≤ (*card* (*X* × *X*)) **using** *B-def bad-pair-set-ss partitions-finite card-mono finite-cartesian-product-iff* **by** *metis*
then have *card-B*: *card B* ≤ (*card X*)^*2*
by (*metis card-cartesian-prod-square partitions-finite*(*1*))

have ⋀ *x x′* . (*x*, *x′*) ∈ *B* ⟹ *P.prob* {*y* ∈ *Y* . {*x*, *x′*} ⊆ *neighborhood y*} < *c*
proof −
fix *x x′* **assume** *assm*: (*x*, *x′*) ∈ *B*
then have *x* ∈ *X x′* ∈ *X* **unfolding** *B-def bad-pair-set-def bad-pair-def* **by** *auto*
then have *card-eq*: *card* {*v* ∈ *V* . *vert-adj v x* ∧ *vert-adj v x′*} = *card* {*y* ∈ *Y*. *vert-adj y x* ∧ *vert-adj y x′*}
by (*metis* (*no-types*, *lifting*) *X-vert-adj-Y vert-adj-edge-iff2 vert-adj-imp-inV*)

have *ltc*: *card* {*v* ∈ *V* . *vert-adj v x* ∧ *vert-adj v x′*}/(*card Y*) < *c*
using *assm* **by** (*auto simp add*: *B-def bad-pair-set-def bad-pair-def codegree-normalized-def codegree-def vert-adj-sym*)
have {*y* ∈ *Y* . {*x*, *x′*} ⊆ *neighborhood y*} = {*y* ∈ *Y* . *vert-adj y x* ∧ *vert-adj y x′*}
using *bad-pair-set-def bad-pair-def neighborhood-def vert-adj-imp-inV vert-adj-imp-inV* **by** *auto*
then have *P.prob* {*y* ∈ *Y* . {*x*, *x′*} ⊆ *neighborhood y*} = *card* {*y* ∈ *Y*. *vert-adj*

y x ∧ *vert-adj y x*′}/ *card Y*
using *measure-uniform-count-measure partitions-finite*(*2*) **by** *fastforce*
thus *P.prob* {*y* ∈ *Y* . {*x*, *x*′} ⊆ *neighborhood y*} < *c* **using** *card-eq ltc* **by** *simp*
qed
then have ⋀ *x x*′ . (*x*, *x*′) ∈ *B* ⟹ *P.prob* {*y* ∈ *Y* . (*x*, *x*′) ∈ *B*′ *y*} < *c*
by (*simp add*: *B-def B*′*-def bad-pair-set-def*)
then have *prob*: ⋀ *p* .*p* ∈ *B* ⟹ *P.prob* {*y* ∈ *Y* . *indicator* (*B*′ *y*) *p* = *1*} ≤ *c*
unfolding *indicator-def* **by** *fastforce*

have *dsimp*: (*density*^*2* − (*density*^*2*/(*2* ∗ *c*)) ∗ *c*) ∗ (*card X*)^*2* =(*density*^*2*/*2*) ∗ (*card X*)^*2*
using *assms* **by** (*simp add*: *algebra-simps*)
then have *gt0*: (*density*^*2*/*2*) ∗ (*card X*)^*2* > *0*
using *dgt0* **by** (*metis density-simp division-ring-divide-zero half-gt-zero linorder-neqE-linordered-idom of-nat-less-0-iff of-nat-mult power2-eq-square zero-less-mult-iff*)
have *Cgt0*: (*density*^*2*/(*2* ∗ *c*)) > *0* **using** *dgt0 assms* **by** *auto*
have ⋀ *y* . *y* ∈ *Y* ⟹ *card* (*B*′ *y*) = (∑ *p* ∈ *B*. *indicator* (*B*′ *y*) *p*)
proof −
fix *y* **assume** *y* ∈ *Y*
then have *neighborhood y* ⊆ *X* **by** (*simp add*: *neighborhood-subset-oppY*)
then have *ss*: *B*′ *y* ⊆ *B* **unfolding** *B-def B*′*-def bad-pair-set-def*
using *set-pairs-filter-subset* **by** *blast*
then show *card* (*B*′ *y*) = (∑ *p* ∈ *B*. *indicator* (*B*′ *y*) *p*)
using *card-set-ss-indicator*[*of B*′ *y B*] *finB* **by** *auto*
qed
then have *P.expectation* (λ *y*. *card* (*B*′ *y*)) = *P.expectation* (λ *y*. (∑ *p* ∈ *B*. *indicator* (*B*′ *y*) *p*))
by (*metis* (*mono-tags*, *lifting*) *P.prob-space-axioms of-nat-sum partitions-finite*(*2*)

prob-space.expectation-uniform-count real-of-nat-indicator sum.cong)
also have ... = (∑ *p* ∈ *B* . *P.expectation* (λ *y*. *indicator* (*B*′ *y*) *p*))
by (*rule Bochner-Integration.integral-sum*[*of B ?M* λ *p y* . *indicator* (*B*′ *y*) *p*])
(*auto simp add*: *P.integrable-uniform-count-measure-finite partitions-finite*(*2*))
finally have *P.expectation* (λ *y*. *card* (*B*′ *y*)) = (∑ *p* ∈ *B* . *P.prob* {*y* ∈ *Y*. *indicator* (*B*′ *y*) *p* = *1*})
using *P.expectation-finite-uniform-indicator*[*of Y B*′] **using** *partitions-finite*(*2*)
by (*smt* (*verit*, *best*) *sum.cong*)
then have *P.expectation* (λ *y*. *card* (*B*′ *y*)) ≤ (∑ *p* ∈ *B* . *c*)
using *prob sum-mono*[*of B* λ *p*. *P.prob* {*y* ∈ *Y*. *indicator* (*B*′ *y*) *p* = *1*} λ *p*. *c*]
by (*simp add*: *indicator-eq-1-iff*)
then have *lt1*: *P.expectation* (λ *y*. *card* (*B*′ *y*)) ≤ *c* ∗ (*card B*) **using** *finB*
by (*simp add*: *mult-of-nat-commute*)

have *c* ∗ (*card B*) ≤ *c* ∗ (*card X*)^*2* **using** *assms card-B* **by** *auto*
then have *P.expectation* (λ *y*. *card* (*B*′ *y*)) ≤ *c* ∗ (*card X*)^*2*
using *lt1* **by** *linarith*
then have ⋀ *C*. *C*> *0* ⟹ *C* ∗ *P.expectation* (λ *y*. *card* (*B*′ *y*)) ≤ *C* ∗ *c* ∗ (*card X*)^*2*

by *auto*
then have ⋀ C . C> 0 ⟹(P.expectation (λ y. (card (neighborhood y))^2) − C ∗ (P.expectation (λ y. card (B′ y)))
≥ (density^2 ∗(card X)^2) − (C∗c∗(card X)^2))
using *card-exp-gt diff-strict1-mono* **by** (*smt* (*verit*))
then have ⋀ C .C> 0 ⟹ (P.expectation (λ y. (card (neighborhood y))^2) − C ∗ (P.expectation (λ y. card (B′ y)))
≥ (density^2 − C ∗ c) ∗ (card X)^2)
by (*simp add*: *field-simps*)

then have (P.expectation (λ y. (card (neighborhood y))^2) − (density^2/(2 ∗ c)) ∗ (P.expectation (λ y. card (B′ y)))
≥ (density^2 − (density^2/(2 ∗ c)) ∗ c) ∗ (card X)^2)
using *Cgt0 assms* **by** *blast*
then have P.expectation (λ y. (card (neighborhood y))^2) − (density^2/(2 ∗ c)) ∗ (P.expectation (λ y. card (B′ y)))
≥ (density^2/2) ∗ (card X)^2 **using** *dsimp* **by** *linarith*
then have P.expectation (λ y. (card (neighborhood y))^2) − (P.expectation (λ y. (density^2/(2 ∗ c)) ∗ card (B′ y)))
≥ (density^2/2) ∗ (card X)^2 **by** *auto*
then have P.expectation (λ y. (card (neighborhood y))^2 − ((density^2/(2∗c)) ∗ card (B′ y)))
≥ (density^2/2) ∗ (card X)^2
using *Bochner-Integration.integral-diff*[*of ?M* (λ y. (card (neighborhood y))^2) (λ y. (density^2/(2 ∗ c)) ∗ card (B′ y))]
P.integrable-uniform-count-measure-finite partitions-finite(2) **by** *fastforce*

then obtain y **where** *yin*: y ∈ Y **and** *ineq*: (card (neighborhood y))^2 − ((density^2/(2 ∗ c)) ∗ card (B′ y)) ≥ (density^2/2) ∗ (card X)^2
using *P.expectation-obtains-ge partitions-finite*(2) **by** *blast*

let ?X′ = neighborhood y
have *ss*: ?X′ ⊆ X
using *yin* **by** (*simp add*: *neighborhood-subset-oppY*)
have local.density^2 / (2 ∗ c) ∗ real (card (B′ y)) ≥ 0
using *assms density-simp* **by** *simp*
then have *d1*: (card ?X′)^2 ≥ (density^2/2) ∗ (card X)^2
using *ineq* **by** *linarith*
then have (card ?X′) ≥ sqrt(((density) ∗ (card X))^2/2)
by (*simp add*: *field-simps real-le-lsqrt*)
then have *den*: ((card ?X′) ≥ (density ∗ (card X)/(sqrt 2)))
by (*smt* (*verit*, *del-insts*) *divide-nonneg-nonneg divide-nonpos-nonneg real-sqrt-divide*

real-sqrt-ge-0-iff real-sqrt-unique zero-le-power2)
have *xgt0*: (card ?X′) > 0 **using** *dgt0 gt0*
using *d1 gr0I* **by** *force*
then have (card ?X′)^2 ≥ (density^2/(2 ∗ c)) ∗ card (B′ y)
using *gt0 ineq* **by** *simp*
then have (card ?X′)^2/(density^2/(2 ∗ c)) ≥ card (B′ y)

using *Cgt0* **by** (*metis mult.commute pos-le-divide-eq*)
then have $((2 * c)/(density\hat{}2)) \geq card\ (B'\ y)/(card\ ?X')\hat{}2$
using *pos-le-divide-eq xgt0* **by** (*simp add*: *field-simps*)
thus *?thesis* **using** *that*[*of ?X'*] *den ss B'-def* **by** *auto*
qed

The following technical probability lemma corresponds to Lemma 2.14 in Gowers's notes [3].

lemma (**in** *prob-space*) *expectation-condition-card-1*:
fixes $X::'a\ set$ **and** $f::'a \Rightarrow real$ **and** $\delta::real$
assumes *finite X* **and** $\forall\ x \in X.\ f\ x \leq 1$ **and** $M = uniform\text{-}count\text{-}measure\ X$
and $expectation\ f \geq \delta$
shows $card\ \{x \in X.\ (f\ x \geq \delta\ /\ 2)\} \geq \delta * card\ X\ /\ 2$
proof (*cases* $\delta \geq 0$)
assume $h\delta$: $\delta \geq 0$
have *ineq1*: $real\ (card\ (X - \{x \in X.\ \delta \leq f\ x * 2\})) * \delta \leq real\ (card\ X) * \delta$
using *card-mono assms Diff-subset hδ mult-le-cancel-right nat-le-linear of-nat-le-iff*

by (*smt* (*verit*, *best*))
have *ineq2*: $\forall\ x \in X - \{x.\ x \in X \wedge (f\ x \geq \delta/2)\}.\ f\ x \leq \delta\ /\ 2$ **by** *auto*
have $expectation\ f * card\ X = (\sum x \in X.\ f\ x)$
using *assms*(*1*) *expectation-uniform-count assms*(*3*) **by** *force*
also have ... $= (\sum\ x \in \{x.\ x \in X \wedge (f\ x \geq \delta/2)\}.\ f\ x)$
$+(\sum\ x \in X - \{x.\ x \in X \wedge (f\ x \geq \delta/2)\}.\ f\ x)$
using *assms*
by (*metis* (*no-types*, *lifting*) *add.commute mem-Collect-eq subsetI sum.subset-diff*)
also have ... $\leq (\sum\ x \in\ \{x.\ x \in X \wedge (f\ x \geq \delta/2)\}.\ 1) +$
$(\sum\ x \in X - \{x.\ x \in X \wedge (f\ x \geq \delta/2)\}\ .\ \delta\ /\ 2)$
using *assms sum-mono ineq2* **by** (*smt* (*verit*, *ccfv-SIG*) *mem-Collect-eq*)
also have ... $\leq card\ (\{x.\ x \in X \wedge (f\ x \geq \delta/2)\}) + (card\ X) * \delta\ /\ 2$
using *ineq1* **by** *auto*
finally have $\delta * card\ X \leq card\ \ \{x.\ x \in X \wedge (f\ x \geq \delta/2)\} + (\delta/2)*(card\ X)$
using *ineq1 mult-of-nat-commute assms*(*4*) *mult-right-mono le-trans*
by (*smt* (*verit*, *del-insts*) *of-nat-0-le-iff times-divide-eq-left*)
then show *?thesis* **by** *auto*
next
assume $\neg\ \delta \geq 0$
thus *?thesis* **by** (*smt* (*verit*, *del-insts*) *divide-nonpos-nonneg mult-nonpos-nonneg of-nat-0-le-iff*)
qed

The following technical probability lemma corresponds to Lemma 2.15 in Gowers's notes.

lemma (**in** *prob-space*) *expectation-condition-card-2*:
fixes $X::'a\ set$ **and** $\beta::real$ **and** $\alpha::real$ **and** $f::\ 'a \Rightarrow real$
assumes *finite X* **and** $\bigwedge\ x.\ x \in X \Longrightarrow f\ x \leq 1$ **and** $\beta > 0$ **and** $\alpha > 0$
and $expectation\ f \geq 1 - \alpha$ **and** $M = uniform\text{-}count\text{-}measure\ X$
shows $card\ \{x \in X.\ f\ x \geq 1 - \beta\} \geq (1-\ \alpha\ /\ \beta) * card\ X$

proof −
have *hcard*: *card* $\{x \in X.\ 1 - \beta \leq f\ x\} \leq$ *card X* **using** *card-mono assms*(*1*) **by** *fastforce*
have $h\beta$: $\forall\ x \in X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\}.\ f\ x \leq 1 - \beta$ **by** *auto*
have *expectation f* $*$ *card X* $= (\sum\ x \in X.\ f\ x)$
using *assms*(*1*) *expectation-uniform-count assms*(*6*) **by** *force*
then have $(1 - \alpha) *$ *card X* $\leq (\sum\ x \in X.\ f\ x)$ **using** *assms*
by (*metis mult.commute sum-bounded-below sum-constant*)
also have ... $= (\sum\ x \in \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\}.\ f\ x) + (\sum\ x \in X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\}.\ f\ x)$ **using** *assms*
by (*metis* (*no-types*, *lifting*) *add.commute mem-Collect-eq subsetI sum.subset-diff*)
also have ... $\leq (\sum\ x \in \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\}.\ 1) + (\sum\ x \in X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\}.\ (1 - \beta))$
using *assms* $h\beta$ *sum-mono* **by** (*smt* (*verit*, *ccfv-SIG*) *mem-Collect-eq*)
also have ... $=$ *card* $\{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\} + (1 - \beta) *$ *card* $(X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\})$
by *auto*
also have ... $= ($*card* $\{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\} +$ *card* $(X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\})) - \beta *$ *card* $(X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\})$
using *left-diff-distrib*
by (*smt* (*verit*, *ccfv-threshold*) *mult.commute mult.right-neutral of-nat-1 of-nat-add of-nat-mult*)
also have *heq*: ... $=$ *card X* $- \beta *$ *card* $(X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\})$
using *assms*(*1*) *card-Diff-subset*[*of* $\{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\}$ *X*] *hcard* **by** *auto*
finally have $(1 - \alpha) *$ *card X* $\leq$ *card X* $- \beta *$ *card* $(X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\})$ **by** *blast*
then have $- (1 - \alpha) *$ *card X* $+$ *card X* $\geq \beta *$ *card* $(X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\})$ **by** *linarith*
then have $-$ *card X* $+ \alpha *$ *card X* $+$ *card X* $\geq \beta *$ *card*$(X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\})$
using *add.assoc add.commute*
add.right-neutral add-0 add-diff-cancel-right′ add-diff-eq add-uminus-conv-diff diff-add-cancel
distrib-right minus-diff-eq mult.commute mult-1 of-int-minus of-int-of-nat-eq uminus-add-conv-diff
cancel-comm-monoid-add-class.diff-cancel **by** (*smt* (*verit*, *del-insts*) *mult-cancel-right2*)
then have $\alpha *$ *card X* $\geq \beta *$ *card*$(X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\})$ **by** *auto*
then have $\alpha *$ *card X* $/ \beta \geq$ *card*$(X - \{x.\ x \in X \wedge (f\ x \geq 1 - \beta)\})$ **using** *assms*
by (*smt* (*verit*, *ccfv-SIG*) *mult.commute pos-divide-less-eq*)
then show *?thesis* **by** (*smt* (*verit*) *heq combine-common-factor left-diff-distrib′ mult-of-nat-commute*
nat-mult-1-right of-nat-1 of-nat-add of-nat-mult times-divide-eq-left scale-minus-left)
qed

The following lemma corresponds to Lemma 2.16 in Gowers's notes [3]. For the proof, we will apply Lemma 2.13 (*proportion-bad-pairs-subset-bipartite*, the technical probability Lemmas 2.14 (*expectation-condition-card-1*) and

2.15 (*expectation-condition-card-2*) as well as background material on graphs with loops and bipartite graphs that was previously presented.

lemma (**in** *fin-bipartite-graph*) *walks-of-length-3-subsets-bipartite*:
obtains X' **and** Y' **where** $X' \subseteq X$ **and** $Y' \subseteq Y$ **and**
card $X' \geq$ (*edge-density* X Y)^*2* $*$ *card* X / *16* **and**
card $Y' \geq$ *edge-density* X Y $*$ *card* Y / *4* **and**
$\forall\ x \in X'. \forall\ y \in Y'.$ *card* {*p. connecting-walk x y p* $\wedge$ *walk-length p* = *3*} $\geq$
(*edge-density* X Y)^*6* $*$ *card* X $*$ *card* Y / *2*^*13*

proof (*cases edge-density X Y* > *0*)
let *?δ* = *edge-density X Y*
assume *hδ*: *?δ* > *0*
interpret *P1*: *prob-space uniform-count-measure X*
by (*simp add*: *X-not-empty partitions-finite*(*1*) *prob-space-uniform-count-measure*)
have *hP1exp*: *P1.expectation* (λ *x. degree-normalized x Y*) ≥ *?δ*
using *P1.expectation-uniform-count partitions-finite sum-degree-normalized-X-density*
by *auto*
let *?X1* = {*x* ∈ *X*. (*degree-normalized x Y* ≥ *?δ*/*2*)}
have *hX1X*: *?X1* ⊆ *X* **and** *hX1card*: *card ?X1* ≥ *?δ* ∗ (*card X*)/*2*
and *hX1degree*: ∀ *x* ∈ *?X1*. *degree-normalized x Y* ≥ *?δ* /*2* **using**
P1.expectation-condition-card-1 partitions-finite degree-normalized-le-1 hP1exp
by *auto*
have *hX1cardpos*: *card ?X1* > *0* **using** *hX1card hδ X-not-empty*
by (*smt* (*verit*, *del-insts*) *divide-divide-eq-right divide-le-0-iff density-simp gr0I less-eq-real-def mult-is-0 not-numeral-le-zero of-nat-le-0-iff of-nat-less-0-iff*)
interpret *H*: *fin-bipartite-graph* (*?X1* ∪ *Y*) {*e* ∈ *E*. *e* ⊆ (*?X1* ∪ *Y*)} *?X1 Y*
proof (*unfold-locales*, *simp add*: *partitions-finite*)
have *disjoint* {*?X1*, *Y*} **using** *hX1X partition-on-def partition*
by (*metis* (*no-types*, *lifting*) *disjnt-subset2 disjnt-sym ne pairwise-insert singletonD*)
moreover have {} ∉ {*?X1*, *Y*} **using** *hX1cardpos Y-not-empty*
by (*metis* (*no-types*, *lifting*) *card.empty insert-iff neq0-conv singleton-iff*)
ultimately show *partition-on* (*?X1* ∪ *Y*) {*?X1*, *Y*} **using** *partition-on-def*
by *auto*
next
show *?X1* ≠ *Y* **using** *ne partition* **by** (*metis Int-absorb1 Y-not-empty hX1X part-intersect-empty*)
next
show ⋀*e*. *e* ∈ {*e* ∈ *E*. *e* ⊆ *?X1* ∪ *Y*} ⟹ *e* ∈ *all-bi-edges* {*x* ∈ *X*. *edge-density X Y* / *2* ≤
degree-normalized x Y} *Y*
using *Un-iff Y-verts-not-adj edge-betw-indiv in-mono insert-subset mem-Collect-eq*
subset-refl that vert-adj-def all-bi-edges-def[*of ?X1 Y*] *in-mk-uedge-img-iff*
by (*smt* (*verit*, *ccfv-threshold*) *all-edges-betw-I all-edges-between-bi-subset*)
next
show *finite* (*?X1* ∪ *Y*) **using** *hX1X* **by** (*simp add*: *partitions-finite*(*1*) *partitions-finite*(*2*))
qed

have *neighborhood-unchanged*: ∀ *x* ∈ *?X1*. *neighbors-ss x Y* = *H.neighbors-ss x Y*
using *neighbors-ss-def H.neighbors-ss-def vert-adj-def H.vert-adj-def* **by** *auto*
then have *degree-unchanged*: ∀ *x* ∈ *?X1*. *degree x* = *H.degree x*
using *H.degree-neighbors-ssX degree-neighbors-ssX* **by** *auto*
have *hHdensity*: (*H.edge-density ?X1 Y*) ≥ *?δ* / *2*
proof−
have *?δ* / *2* = (∑ *x* ∈ *?X1*. (*?δ* / *2*)) / *card ?X1* **using** *hX1cardpos* **by** *auto*
also have ... ≤ (∑ *x* ∈ *?X1*. *degree-normalized x Y*) / *card ?X1*
using *sum-mono hX1degree hX1cardpos divide-le-cancel*
by (*smt* (*z3*) *H.X-not-empty H.partitions-finite*(*1*)
calculation divide-pos-pos hδ sum-pos zero-less-divide-iff)
also have ... = (*H.edge-density ?X1 Y*)
using *H.degree-normalized-def degree-normalized-def degree-unchanged sum.cong*
H.degree-neighbors-ssX degree-neighbors-ssX H.sum-degree-normalized-X-density
by *auto*
finally show *?thesis* **by** *simp*
qed
have *hδ3pos*: *?δ^3* / *128* > *0* **using** *hδ* **by** *auto*
then obtain *X2* **where** *hX2subX1*: *X2* ⊆ *?X1* **and** *hX2card*: *card X2* ≥ (*H.edge-density ?X1 Y*) ∗
(*card ?X1*)/ (*sqrt 2*) **and** *hX2badtemp*: (*card* (*H.bad-pair-set X2 Y* (*?δ^3* / *128*))) / *real* ((*card X2*)^*2*)
≤ *2* ∗ (*?δ^3* / *128*) / (*H.edge-density ?X1 Y*)^*2* **using** *H.proportion-bad-pairs-subset-bipartite*
by *blast*
have (*H.edge-density ?X1 Y*) ∗ (*card ?X1*)/ (*sqrt 2*) > *0* **using** *hHdensity hX1cardpos hδ hX2card*
by *auto*
then have *hX2cardpos*: *card X2* > *0* **using** *hX2card* **by** *auto*
then have *hX2finite*: *finite X2* **using** *card-ge-0-finite* **by** *auto*
have *hX2bad*: (*card* (*H.bad-pair-set X2 Y* (*?δ^3* / *128*))) ≤ (*?δ* / *16*) ∗ (*card X2*)^*2*
proof−
have *hpos*: *real* ((*card X2*)^*2*) > *0* **using** *hX2cardpos* **by** *auto*
have *trivial*: (*3*:: *nat*) − *2* = *1* **by** *simp*
then have *hδpow*: *?δ* ^ *3* / *?δ* ^ *2* = *?δ^1* **using** *power-diff hδ*
by (*metis div-greater-zero-iff less-numeral-extra*(*3*) *numeral-Bit1-div-2 zero-less-numeral*)

have *card* (*H.bad-pair-set X2 Y* (*?δ^3* / *128*)) ≤ (*2* ∗ (*?δ^3* / *128*) / (*H.edge-density ?X1 Y*)^*2*) ∗
(*card X2*)^*2* **using** *hX2badtemp hX2cardpos* **by** (*simp add*: *field-simps*)
also have ... ≤ (*2* ∗ (*?δ^3* / *128*) / (*?δ* / *2*)^*2*) ∗ (*card X2*)^*2*
using *hδ hHdensity divide-left-mono frac-le hpos* **by** (*smt* (*verit*) *divide-pos-pos*

edge-density-ge0 le-divide-eq power-mono zero-le-divide-iff zero-less-power)
also have ... = (*?δ^3* / *?δ^2*) ∗ (*1*/*16*) ∗ (*card X2*)^*2* **by** (*simp add*: *field-simps*)
also have ... = (*?δ* / *16*) ∗ (*card X2*) ^ *2* **using** *hδpow* **by** *auto*
finally show *?thesis* **by** *simp*

qed
let *?E-loops* = *mk-edge* ‘ {(*x*, *x*′) | *x x*′. *x* ∈ *X2* ∧ *x*′ ∈ *X2* ∧
(*H.codegree-normalized x x*′ *Y*) ≥ *?δ* ^ *3* / *128*}
interpret Γ: *ulgraph X2 ?E-loops*
proof(*unfold-locales*)
show ⋀*e*. *e* ∈ *?E-loops* ⟹ *e* ⊆ *X2* **by** *auto*
next
have ⋀*a b*. *a* ∈ *X2* ⟹ *b* ∈ *X2* ⟹ *0* < *card* {*a*, *b*}
by (*meson card-0-eq finite.emptyI finite-insert insert-not-empty neq0-conv*)
moreover have ⋀ *a b*. *a* ∈ *X2* ⟹ *b* ∈ *X2* ⟹ *card* {*a*, *b*} ≤ *2*
by (*metis card-2-iff dual-order.refl insert-absorb2 is-singletonI*
is-singleton-altdef one-le-numeral)
ultimately show ⋀*e*. *e* ∈ *?E-loops* ⟹ *0* < *card e* ∧ *card e* ≤ *2* **by** *auto*
qed
have *h*Γ*-edges*: ⋀ *a b*. *a* ∈ *X2* ⟹ *b* ∈ *X2* ⟹
{*a*, *b*} ∈ *?E-loops* ⟷ *H.codegree-normalized a b Y* ≥ *?δ*^*3*/*128*
proof
fix *a b* **assume** {*a*, *b*} ∈ *?E-loops*
then show *H.codegree-normalized a b Y* ≥ *?δ*^*3*/*128*
using *in-mk-uedge-img-iff*[*of a b* {(*x*, *x*′) | *x x*′. *x* ∈ *X2* ∧ *x*′ ∈ *X2* ∧
(*H.codegree-normalized x x*′ *Y*) ≥ *?δ* ^ *3* / *128*}] *doubleton-eq-iff H.codegree-normalized-sym*

by *auto*
next
fix *a b* **assume** *a* ∈ *X2* **and** *b* ∈ *X2* **and** *H.codegree-normalized a b Y* ≥
?δ^*3*/*128*
then show {*a*, *b*} ∈ *?E-loops* **using** *in-mk-uedge-img-iff*[*of a b* {(*x*, *x*′) | *x*
x′. *x* ∈ *X2* ∧
x′ ∈ *X2* ∧ (*H.codegree-normalized x x*′ *Y*) ≥ *?δ* ^ *3* / *128*}] *H.codegree-normalized-sym*
by *auto*
qed
interpret *P2*: *prob-space uniform-count-measure X2*
using *hX2finite hX2cardpos prob-space-uniform-count-measure* **by** *fastforce*
have *hP2exp*: *P2.expectation* (λ *x*. Γ.*degree-normalized x X2*) ≥ *1* − *?δ* / *16*
proof−
have *h*Γ*all*: Γ.*all-edges-between X2 X2* = (*X2* × *X2*) − *H.bad-pair-set X2 Y*
(*?δ*^*3* / *128*)
proof
show Γ.*all-edges-between X2 X2* ⊆ *X2* × *X2* − *H.bad-pair-set X2 Y* (*?δ* ^
3 / *128*)
proof
fix *x* **assume** *x* ∈ Γ.*all-edges-between X2 X2*
then obtain *a b* **where** *a* ∈ *X2* **and** *b* ∈ *X2* **and** *x* = (*a*, *b*) **and**
H.codegree-normalized a b Y ≥ *?δ*^*3* / *128*
using Γ.*all-edges-between-def in-mk-uedge-img-iff h*Γ*-edges*
by (*smt* (*verit*, *del-insts*) Γ.*all-edges-betw-D3* Γ.*wellformed-alt-snd edge-density-commute*

mk-edge.cases mk-edge.simps that)
then show *x* ∈ *X2* × *X2* − *H.bad-pair-set X2 Y* (*?δ* ^ *3* / *128*)

```
          using H.bad-pair-set-def H.bad-pair-def by auto
        qed
      next
        show X2 × X2 − H.bad-pair-set X2 Y (?δ ^ 3 / 128) ⊆ Γ.all-edges-between
X2 X2
          using H.bad-pair-set-def H.bad-pair-def Γ.all-edges-between-def hΓ-edges by
auto
      qed
      then have hΓall-le: card (Γ.all-edges-between X2 X2) ≥ (1 − ?δ / 16) ∗ (card
X2 ∗ card X2)
      proof−
          have hbadsub: H.bad-pair-set X2 Y (?δ ^3 / 128) ⊆ X2 × X2 using
H.bad-pair-set-def by auto
        have (1 − ?δ / 16) ∗ (card X2 ∗ card X2) = card (X2 × X2) − ?δ / 16 ∗
(card X2) ^2
          using card-cartesian-product power2-eq-square
          by (metis Rings.ring-distribs(4) more-arith-simps(6) mult-of-nat-commute)
        also have ... ≤ card (X2 × X2) − card (H.bad-pair-set X2 Y (?δ ^ 3 /
128))
          using hX2bad by auto
        also have ... = card (X2 × X2 − H.bad-pair-set X2 Y (?δ ^ 3 / 128)) using
card-Diff-subset
          finite-cartesian-product[of X2 X2] hX2finite hbadsub
          by (metis (mono-tags, lifting) finite-subset)
        finally show card(Γ.all-edges-between X2 X2) ≥ (1 − ?δ/16) ∗ (card X2 ∗
card X2)
          using hΓall by simp
      qed
      have 1 − ?δ / 16 = ((1 − ?δ / 16) ∗ (card X2 ∗ card X2)) / (card X2 ∗ card
X2)
        using hX2cardpos by auto
      also have ... ≤ card (Γ.all-edges-between X2 X2) / (card X2 ∗ card X2)
        using hΓall-le hX2cardpos divide-le-cancel of-nat-less-0-iff by fast
      also have ... = (∑ x ∈ X2. real (card (Γ.neighbors-ss x X2))) / card X2 /
card X2
        using Γ.card-all-edges-betw-neighbor[of X2 X2] hX2finite by (auto simp add:
field-simps)
      also have ... = (∑ x ∈ X2. Γ.degree-normalized x X2) / card X2
        unfolding Γ.degree-normalized-def
          using sum-divide-distrib[of λ x. real (card (Γ.neighbors-ss x X2)) X2 card
X2] by auto
      also have ... = P2.expectation (λ x. Γ.degree-normalized x X2)
        using P2.expectation-uniform-count hX2finite by auto
      finally show ?thesis by simp
    qed
    let ?X′ = {x ∈ X2. Γ.degree-normalized x X2 ≥ 1 − ?δ / 8}
    have hX′subX2: ?X′ ⊆ X2 by blast
    have hX′cardX2: card ?X′ ≥ card X2 / 2 using hX2finite divide-self hδ
      P2.expectation-condition-card-2 [of X2 (λ x. Γ.degree-normalized x X2) ?δ /8
```

?δ / 16]
hP2exp Γ.*degree-normalized-le-1* **by** *auto*
interpret *P3*: *prob-space uniform-count-measure Y*
by (*simp add*: *Y-not-empty partitions-finite*(*2*) *prob-space-uniform-count-measure*)
have *hP3exp*: *P3.expectation* (λ *y*. *H.degree-normalized y X2*) ≥ *?δ / 2*
proof−
have *hHdegree-normalized*: ⋀ *x*. *x* ∈ *X2* ⟹ (*?δ / 2*) ≤ *H.degree-normalized x Y*
using *hX1degree degree-normalized-def H.degree-normalized-def neighborhood-unchanged*
hX2subX1 subsetD **by** (*metis* (*no-types*, *lifting*))
have *?δ / 2* = (∑ *x* ∈ *X2*. (*?δ / 2*)) / *card X2* **using** *hX2cardpos* **by** *auto*
also have ... ≤ (∑ *x* ∈ *X2*. *real* (*card* (*H.neighbors-ss x Y*))) / *card Y* / *card X2*
using *hHdegree-normalized sum-mono divide-le-cancel hX2cardpos of-nat-0-le-iff*
H.degree-normalized-def sum.cong sum-divide-distrib **by** (*smt* (*verit*, *best*))
also have ... = (*card* (*H.all-edges-between Y X2*)) / *card X2* / *card Y*
using *H.card-all-edges-between-commute H.card-all-edges-betw-neighbor hX2finite*
H.partitions-finite(*2*) **by** *auto*
also have ... = (∑ *y* ∈ *Y*. *real* (*card*(*H.neighbors-ss y X2*))) / *card X2* / *card Y* **using**
H.card-all-edges-betw-neighbor hX2finite H.partitions-finite(*2*) **by** *auto*
also have ... = (∑ *y* ∈ *Y*. *H.degree-normalized y X2*) / *card Y* **using** *H.degree-normalized-def*
sum.cong sum-divide-distrib **by** (*smt* (*verit*, *best*))
also have ... = *P3.expectation* (λ *y*. *H.degree-normalized y X2*)
using *P3.expectation-uniform-count H.partitions-finite*(*2*) **by** *auto*
finally show *?thesis* **by** *linarith*
qed
let *?Y′* = {*x* ∈ *Y*. *H.degree-normalized x X2* ≥ *?δ / 4*}
have *hY′subY*: *?Y′* ⊆ *Y* **by** *blast*
then have *hY′card*: *card ?Y′* ≥ *?δ* ∗ *card Y / 4*
using *P3.expectation-condition-card-1*[*of Y* (λ *y*. *H.degree-normalized y X2*) *?δ / 2*] *H.partitions-finite*(*2*)
hP3exp H.degree-normalized-le-1 **by** *auto*

have *hX2adjcard*: ⋀ *x y*. *x* ∈ *?X′* ⟹ *y* ∈ *?Y′* ⟹
card {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*} ≥ *?δ / 8* ∗ *card X2*
proof−
fix *x y* **assume** *hx*: *x* ∈ *?X′* **and** *hy*: *y* ∈ *?Y′*
have *hinter*: {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*} =
{*x′* ∈ *X2*. Γ.*vert-adj x x′*} ∩ {*x′* ∈ *X2*. *vert-adj y x′*} **by** *auto*
have *huncardX2*: *card* ({*x′* ∈ *X2*. Γ.*vert-adj x x′*} ∪ {*x′* ∈ *X2*. *vert-adj y x′*}) ≤ *card X2*
using *card-mono hX2finite* **by** *fastforce*
have *fin1*: *finite* {*x′* ∈ *X2*. Γ.*vert-adj x x′*} **and** *fin2*: *finite* {*x′* ∈ *X2*. *vert-adj y x′*}
using *hX2finite* **by** *auto*

have $\{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x'\} = \Gamma.neighbors\text{-}ss\ x\ X2$
using Γ.*vert-adj-def vert-adj-def* Γ.*neighbors-ss-def hX2subX1* Γ.*neighbors-ss-def*
by *auto*
then have *hcard1*: $card\ \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x'\} \geq (1 - ?\delta/8) * card\ X2$
using *hx*
Γ.*degree-normalized-def divide-le-eq hX2cardpos* **by** (*simp add*: *hX2card le-divide-eq*)
have $\{x' \in X2.\ vert\text{-}adj\ y\ x'\} = H.neighbors\text{-}ss\ y\ X2$ **using** *H.vert-adj-def vert-adj-def*
H.neighbors-ss-def hy hY'subY hX2subX1 H.neighbors-ss-def **by** *auto*
then have *hcard2*: $card\ \{x' \in X2.\ vert\text{-}adj\ y\ x'\} \geq (?\delta\ /\ 4) * card\ X2$
using *hy H.degree-normalized-def divide-le-eq hX2cardpos* **by** (*simp add*: *hX2card le-divide-eq*)
have $?\delta\ /\ 8 * card\ X2 = (1 - ?\delta\ /\ 8) * card\ X2 + ?\delta/4 * card\ X2 - card\ X2$
by (*simp add*: *algebra-simps*)
also have ... $\leq card\ \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x'\} + card\ \{x' \in X2.\ vert\text{-}adj\ y\ x'\} -$
$card\ (\{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x'\} \cup \{x' \in X2.\ vert\text{-}adj\ y\ x'\})$
using *huncardX2 hcard1 hcard2* **by** *linarith*
also have ... $= card\ \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\}$
using *card-Un-Int fin1 fin2 hinter* **by** *fastforce*
finally show $card\ \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\} \geq ?\delta\ /\ 8 * card\ X2$
by *linarith*
qed
have *hYpos*: $real\ (card\ Y) > 0$ **using** *Y-not-empty partitions-finite*(*2*) **by** *auto*
have $\bigwedge x\ y.\ x \in ?X' \Longrightarrow y \in ?Y' \Longrightarrow card\ \{p.\ connecting\text{-}walk\ x\ y\ p \wedge walk\text{-}length\ p = 3\} \geq$
$(?\delta\ \hat{}\ 3\ /\ 128 * (card\ Y)) * ((?\delta\ /\ 8) * (card\ X2))$
proof−
fix *x y* **assume** *hx*: $x \in ?X'$ **and** *hy*: $y \in ?Y'$
then have *hxV*: $x \in V$ **and** *hyV*: $y \in V$ **using** *hY'subY hX'subX2 hX2subX1 hX1X partitions-ss*(*1*)
partitions-ss(*2*) *subsetD* **by** *auto*
define *f*:: *'a* ⇒ *'a list set* **where** $f \equiv (\lambda\ a.\ ((\lambda\ z.\ z\ @\ [y])\ `\ \{p.\ connecting\text{-}path\ x\ a\ p \wedge walk\text{-}length\ p = 2\}))$
have *h-norm*: $\bigwedge\ a.\ a \in \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\} \Longrightarrow codegree\text{-}normalized\ x\ a\ Y \geq ?\delta\hat{}3\ /\ 128$
using Γ.*vert-adj-def codegree-normalized-sym hx hX'subX2 subsetD*
codegree-normalized-altX H.codegree-normalized-altX neighborhood-unchanged

*h*Γ*-edges hX2subX1 hX1X* **by** (*smt* (*verit*, *del-insts*) *mem-Collect-eq*)
have *inj-concat*: *inj* (λ *z*. *z* @ [*y*]) **using** *inj-def* **by** *blast*
then have *card-f-eq*: $\bigwedge\ a.\ card\ (f\ a) = card\ \{p.\ connecting\text{-}path\ x\ a\ p \wedge walk\text{-}length\ p = 2\}$
using *f-def card-image inj-eq* **by** (*smt* (*verit*) *inj-onI*)
then have *card-f-ge*: $\bigwedge\ a.\ a \in \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\} \Longrightarrow$
$card\ (f\ a) \geq ?\delta\hat{}3\ /\ 128 * card\ Y$

using *codegree-is-path-length-two codegree-normalized-def hYpos f-def h-norm*
by (*simp add*: *field-simps*)
have *f-disjoint*: *pairwise* (λ *s t*. *disjnt* (*f s*) (*f t*)) $\{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\}$
proof (*intro pairwiseI*)
fix *s t* **assume** $s \in \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\}$ **and**
$t \in \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\}$ **and** $s \neq t$
moreover have $\bigwedge$ *a l*. $l \in f\ a \Longrightarrow l\ !\ 2 = a$
proof−
fix *a l* **assume** $l \in f\ a$
then obtain *z* **where** *hz*: $z \in \{p.\ connecting\text{-}path\ x\ a\ p \wedge walk\text{-}length\ p = 2\}$ **and** *hlz*: $l = z\ @\ [y]$
using *f-def* **by** *blast*
then have $z\ !\ 2 = a$ **using** *walk-length-conv connecting-path-def last-conv-nth*
by (*metis* (*mono-tags*, *lifting*) *diff-add-inverse length-tl list.sel*(*2*) *mem-Collect-eq*

nat-1-add-1 one-eq-numeral-iff rel-simps(*18*))
then show $l\ !\ 2 = a$ **using** *hlz nth-append hz walk-length-conv less-diff-conv mem-Collect-eq*
by (*metis* (*mono-tags*, *lifting*) *nat-1-add-1 one-less-numeral-iff rel-simps*(*9*))
qed
ultimately show *disjnt* (*f s*) (*f t*) **by** (*metis disjnt-iff*)
qed
have *hwalk-subset*: $\{p.\ connecting\text{-}walk\ x\ k\ p \wedge walk\text{-}length\ p = n\} \subseteq \{p.\ set\ p \subseteq V \wedge length\ p = n + 1\}$ **for** *n k*
using *connecting-walk-def is-walk-def walk-length-conv* **by** *auto*
have *finite-walk*: *finite* $\{p.\ connecting\text{-}walk\ x\ k\ p \wedge walk\text{-}length\ p = n\}$ **for** *n k*
using *finV finite-lists-length-eq finite-subset hwalk-subset*[*of k n*] *rev-finite-subset*
by *blast*
have *f-finite*: $\bigwedge$ *A*. $A \in (f\ `\ \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\}) \Longrightarrow$ *finite A*
proof−
fix *A* **assume** $A \in (f\ `\ \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\})$
then obtain *a* **where** $a \in \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\}$ **and** *hA*: $A = f\ a$ **by** *blast*
have $\{p.\ connecting\text{-}path\ x\ a\ p \wedge walk\text{-}length\ p = 2\} \subseteq \{p.\ connecting\text{-}walk\ x\ a\ p \wedge walk\text{-}length\ p = 2\}$
using *connecting-path-walk* **by** *blast*
then have *finite* $\{p.\ connecting\text{-}path\ x\ a\ p \wedge walk\text{-}length\ p = 2\}$
using *finite-walk finite-subset connecting-path-walk* **by** *blast*
then show *finite A* **using** *f-def hA* **by** *auto*
qed
moreover have *f-image-sub*:
$(\bigcup\ x \in \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\}.\ f\ x) \subseteq \{p.\ connecting\text{-}walk\ x\ y\ p \wedge walk\text{-}length\ p = 3\}$
proof(*intro Union-least*)
fix *X* **assume** $X \in f\ `\ \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\}$
then obtain *a* **where** *ha*: $a \in \{x' \in X2.\ \Gamma.vert\text{-}adj\ x\ x' \wedge vert\text{-}adj\ y\ x'\}$ **and** *haX*: $f\ a = X$ **by** *blast*

have ⋀*z*. *connecting-path x a z* ⟹ *walk-length z* = *2* ⟹ *connecting-walk x y* (*z* @ [*y*])
proof−
fix *z* **assume** *hpath*: *connecting-path x a z* **and** *hlen*: *walk-length z* = *2*
then obtain *y′* **where** *z* ! *1* = *y′* **by** *blast*
then have *hz*: *z* = [*x*, *y′*, *a*] **using** *list2-middle-singleton walk-length-conv connecting-path-def hpath hlen add-diff-cancel-left′ append-butlast-last-id butlast.simps connecting-path-walk connecting-walk-def diff-diff-add diff-le-self*

is-walk-not-empty is-walk-tl last-ConsL last-tl list.expand list.sel list.simps(*3*)

nat-1-add-1 le-imp-diff-is-add
by (*metis* (*no-types*, *lifting*) *arith-simps*(*45*) *arithmetic-simps*(*2*) *numerals*(*1*))
moreover have *hwalk*: *connecting-walk x a z* **using** *connecting-path-walk hpath* **by** *blast*
then show *connecting-walk x y* (*z* @ [*y*]) **using** *connecting-walk-def is-walk-def*

connecting-path-def is-gen-path-def is-walk-def ha hz hwalk hyV vert-adj-sym vert-adj-def **by** *auto*
qed
then show *X* ⊆ {*p*. *connecting-walk x y p* ∧ *walk-length p* = *3*}
using *haX f-def walk-length-conv* **by** *auto*
qed
ultimately have *hUn-le*: *card* (⋃ *x* ∈ {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*}. *f x*) ≤ *card* {*p*. *connecting-walk x y p* ∧ *walk-length p* = *3*}
using *card-mono finite-walk*[*of y 3*] **by** *blast*
have *disjoint* (*f* ‘ {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*}) **using** *f-disjoint pairwise-def*
pairwise-image[*of disjnt f* {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*}] **by** (*metis* (*mono-tags*, *lifting*))
then have *card* (⋃ (*f* ‘ {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*})) = *sum card* (*f* ‘ {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*})
using *card-Union-disjoint*[*of f* ‘ {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*}] *f-finite* **by** *blast*
also have ... = (∑ *a* ∈ {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*}. *card* (*f a*))
using *sum-card-image*[*OF - f-disjoint*] *hX2finite finite-subset* **by** *fastforce*
also have ... ≥ *card* {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*} ∗ *?δ*ˆ*3* / *128* ∗ *card Y*
using *sum-mono*[*of* {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*} (λ *a*. *?δ*ˆ*3* / *128* ∗ *card Y*) (λ *a*. *card* (*f a*))]
card-f-ge **by** *auto*
finally have *card* {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*} ∗ *?δ*ˆ*3* / *128* ∗ *card Y* ≤ *card* {*p*. *connecting-walk x y p* ∧ *walk-length p* = *3*}
using *hUn-le* **by** *linarith*
then have (*?δ* ˆ *3* / *128* ∗ (*card Y*)) ∗ *card* {*x′* ∈ *X2*. Γ.*vert-adj x x′* ∧ *vert-adj y x′*} ≤ *card* {*p*. *connecting-walk x y p* ∧ *walk-length p* = *3*}
by *argo*
then show (*?δ* ˆ *3* / *128* ∗ (*card Y*)) ∗ ((*?δ* / *8*) ∗ (*card X2*)) ≤ *card* {*p*.

```
connecting-walk x y p ∧ walk-length p = 3}
      using hX2adjcard[OF hx hy] hYpos mult-left-mono hδ3pos mult-pos-pos
      by (smt (verit, del-insts))
  qed
  moreover have hX2cardX: card X2 ≥ (?δ ^2 / 8) *(card X)
  proof−
    have card X2 ≥ (H.edge-density ?X1 Y / sqrt 2) ∗ (card ?X1)
      using hX2card by (simp add: algebra-simps)
    moreover have (H.edge-density ?X1 Y / sqrt 2) ∗ (card ?X1) ≥ (?δ / (2 ∗
sqrt 2)) ∗ card ?X1
      using hHdensity hX1cardpos by (simp add: field-simps)
    moreover have (?δ / (2 ∗ sqrt 2)) ∗ card ?X1 ≥ (?δ / 4) ∗ card ?X1
      using sqrt2-less-2 hX1cardpos hδ by (simp add: field-simps)
    moreover have (?δ / 4) ∗ card ?X1 ≥ (?δ / 4) ∗ (?δ / 2 ∗ card X)
      using hX1card hδ by (simp add: field-simps)
    moreover have (?δ / 4) ∗ (?δ / 2 ∗ card X) = (?δ ^2 / 8) ∗(card X) using
power2-eq-square
       by (metis (no-types, opaque-lifting) Groups.mult-ac(2) Groups.mult-ac(3)
divide-divide-eq-left
      num-double numeral-times-numeral times-divide-eq-left times-divide-eq-right)
    ultimately show ?thesis by linarith
  qed
  moreover have (?δ ^ 3 / 128 ∗ (card Y)) ∗ ((?δ / 8) ∗ (card X2)) ≥
    ?δ ^ 6 ∗ card X ∗ card Y / 2 ^ 13
  proof−
    have hinter: (?δ / 8) ∗ (?δ ^2 / 8 ∗ card X) ≤ (?δ / 8) ∗ (card X2)
      using hX2cardX hδ by (simp add: algebra-simps)
    have ?δ ^ 6 ∗ real (card X ∗ card Y) / 2 ^ 13 =
      ?δ ^ 3 ∗ ?δ ∗ ?δ ^ 2 ∗ real (card X ∗ card Y) / (128 ∗ 8 ∗ 8) by algebra
    also have ... = (?δ ^ 3 / 128 ∗ (card Y)) ∗ ((?δ / 8) ∗ (?δ ^2 / 8 ∗ card X))
by auto
    also have ... ≤ (?δ ^ 3 / 128 ∗ (card Y)) ∗ ((?δ / 8) ∗ (card X2))
      using hinter hYpos hδ power3-eq-cube
    by (smt (verit) ‹0 < edge-density X Y ^ 3 / 128› mult-left-mono zero-compare-simps(6))
    finally show ?thesis by simp
  qed
  moreover have hX′card: card ?X′ ≥ ?δ ^ 2 ∗ card X / 16 using hX′cardX2
hX2cardX by auto
  moreover have hX′subX: ?X′ ⊆ X using hX′subX2 hX2subX1 hX1X by auto
  ultimately show ?thesis using hY′card hX′card hY′subY hX′subX that
    by (smt (verit, best))
next
  assume ¬ 0 < edge-density X Y
  then have edge-density X Y = 0 by (smt (verit, ccfv-threshold) edge-density-ge0)
  then show ?thesis using that by auto
qed
```

The following lemma corresponds to Lemma 2.17 in Gowers's notes [3].

Note that here we have set(*additive-energy A = 2 ∗ c* (instead of (*ad-*

ditive-energy A = c as in the notes) and we are accordingly considering c-popular differences (instead of c/2-popular differences as in the notes) so that we will still have (*ϑ = additive-energy A / 2*.

lemma *popular-differences-card*: **fixes** *A::'a set* **and** *c::real*
  assumes *finite A* **and** *A ⊆ G* **and** *additive-energy A = 2 ∗ c*
  shows *card* (*popular-diff-set c A*) ≥ *c ∗ card A*

proof(*cases card A ≠ 0*)
  assume *hA*: *card A ≠ 0*
  have *hc*: *c ≥ 0* **using** *assms additive-energy-def of-nat-0-le-iff*
    by (*smt* (*verit*, *best*) *assms*(*3*) *divide-nonneg-nonneg of-nat-0-le-iff*)
  have (*2 ∗ c*) ∗ (*card A*)^*3* = (∑ *d* ∈ (*differenceset A A*). (*f-diff d A*)^*2*)
    using *assms f-diff-card-quadruple-set-additive-energy* **by** *auto*
  also have ...= ((∑ *d* ∈ (*popular-diff-set c A*). (*f-diff d A*)^*2*))
    +((∑ *d* ∈ ((*differenceset A A*) − (*popular-diff-set c A*)). (*f-diff d A*)^*2*))
   using *popular-diff-set-def assms finite-minusset finite-sumset* **by** (*metis* (*no-types*, *lifting*)
      add.commute mem-Collect-eq subsetI sum.subset-diff)
  also have ... ≤ ((*card* (*popular-diff-set c A*)) ∗ (*card A*)^*2*)
    + *c ∗ card A ∗* ((∑ *d* ∈ (*differenceset A A* − (*popular-diff-set c A*)) . (*f-diff d A*)))
  proof−
    have ∀ *d* ∈ ((*differenceset A A*) − (*popular-diff-set c A*)) . (*f-diff d A*)^*2* ≤
      (*c ∗* (*card A*)) ∗ (*f-diff d A*)
    proof
     fix *d* **assume** *hd1*: *d* ∈ *differenceset A A* − *popular-diff-set c A*
     have *hnonneg*: *f-diff d A* ≥ *0* **by** *auto*
     have ¬ *popular-diff d c A* **using** *hd1 popular-diff-set-def* **by** *blast*
     from *this* **have** *f-diff d A* ≤ *c ∗ card A* **using** *popular-diff-def* **by** *auto*
     thus *real* ((*f-diff d A*)^2) ≤ *c ∗ real* (*card A*) ∗ *real* (*f-diff d A*)
     using *power2-eq-square hnonneg mult-right-mono of-nat-0 of-nat-le-iff of-nat-mult* **by** *metis*
   qed
   moreover have ∀ *d* ∈ (*differenceset A A*) . *f-diff d A* ≤ (*card A*)^*2*
     using *f-diff-def finite-minusset finite-sumset assms*
     by (*metis f-diff-le-card le-antisym nat-le-linear power2-nat-le-imp-le*)
   ultimately have ((∑ *d* ∈ ((*differenceset A A*) − *popular-diff-set c A*) . (*f-diff d A*)^*2*)) ≤
     ((∑ *d* ∈ ((*differenceset A A*) − *popular-diff-set c A*). (*c ∗ card A*) ∗ (*f-diff d A*)))
      using *assms finite-minusset finite-sumset sum-distrib-left sum-mono* **by** *fastforce*
  then have ((∑ *d* ∈ ((*differenceset A A*) − *popular-diff-set c A*) . (*f-diff d A*)^*2*)) ≤
     (*c ∗ card A*) ∗ ((∑ *d* ∈ ((*differenceset A A*) − *popular-diff-set c A*) . (*f-diff d A*)))
     by (*metis* (*no-types*) *of-nat-sum sum-distrib-left*)
   moreover have (∑ *d* ∈ *popular-diff-set c A*. (*f-diff d A*)^*2*) ≤

($\sum$ *d* ∈ *popular-diff-set c A*. (*card A*)^*2*) **using** *f-diff-le-card assms sum-mono assms popular-diff-set-def*
by (*metis* (*no-types*, *lifting*) *power2-nat-le-eq-le*)
moreover then have *ddd*: ($\sum$ *d* ∈ *popular-diff-set c A* . (*f-diff d A*)^*2*) ≤ (*card* (*popular-diff-set c A*)) ∗ (*card A*)^*2*
using *sum-distrib-right* **by** *simp*
ultimately show *?thesis* **by** *linarith*
qed
also have ... ≤ ((*card* (*popular-diff-set c A*)) ∗ (*card A*)^*2*) + (*c* ∗ *card A*) ∗ (*card A*)^*2*
proof−
have ($\sum$ *d* ∈ (*differenceset A A* − *popular-diff-set c A*) . (*f-diff d A*)) ≤ ($\sum$ *d* ∈ *differenceset A A* . (*f-diff d A*)) **using** *DiffD1 subsetI assms sum-mono2*

finite-minusset finite-sumset zero-le **by** *metis*
then have (*c* ∗ *card A*) ∗ (($\sum$ *d* ∈ (*differenceset A A* − *popular-diff-set c A*). (*f-diff d A*)))
≤ (*c* ∗ *card A*) ∗ (*card A*)^*2*
using *f-diff-card hc le0 mult-left-mono of-nat-0 of-nat-mono zero-le-mult-iff assms* **by** *metis*
then show *?thesis* **by** *linarith*
qed
finally have (*2* ∗ *c*) ∗ (*card A*)^*3* ≤ ((*card* (*popular-diff-set c A*)) ∗ (*card A*)^*2*) +
(*c* ∗ *card A*) ∗ (*card A*)^*2* **by** *linarith*
then have (*card* (*popular-diff-set c A*)) ≥
((*2* ∗ *c*) ∗ (*card A*)^*3* − (*c* ∗ *card A*) ∗ (*card A*)^*2*)/((*card A*)^*2*)
using *hA* **by** (*simp add*: *field-simps*)
moreover have ((*2* ∗ *c*)∗ (*card A*)^*3* − (*c* ∗ *card A*) ∗ (*card A*)^*2*)/((*card A*)^*2*) = *2* ∗ *c* ∗ *card A* − *c* ∗ *card A*
using *hA* **by** (*simp add*: *power2-eq-square power3-eq-cube*)
ultimately show *?thesis* **by** *linarith*
next
assume ¬ *card A* ≠ *0*
thus *?thesis* **by** *auto*
qed

The following lemma corresponds to Lemma 2.18 in Gowers's notes [3]. It includes the key argument of the main proof and its proof applies Lemmas 2.16 (*walks-of-length-3-subsets-bipartite*) and 2.17 (*popular-differences-card*). In the proof we will use an appropriately defined bipartite graph as an intermediate/auxiliary construct so as to apply lemma *walks-of-length-3-subsets-bipartite*. As each vertex set of the bipartite graph is constructed to be a copy of a finite subset of an Abelian group, we need flexibility regarding types, which is what prompted the introduction and use of the new graph theory library [1] (that does not impose any type restrictions e.g. by representing vertices as natural numbers).

lemma *obtains-subsets-differenceset-card-bound*:

fixes *A*::*'a set* **and** *c*::*real*
assumes *finite A* **and** *c>0* **and** $A \neq \{\}$ **and** $A \subseteq G$ **and** *additive-energy* $A = 2 * c$
obtains *B* **and** *A'* **where** $B \subseteq A$ **and** $B \neq \{\}$ **and** *card* $B \geq c$^*4* $*$ *card A / 16*
and $A' \subseteq A$ **and** $A' \neq \{\}$ **and** *card* $A' \geq c$^*2* $*$ *card A / 4*
and *card* (*differenceset* A' *B*) $\leq$ *2*^*13* $*$ *card A / c*^*15*

proof−
let *?X* = *A* × {*0*:: *nat*}
let *?Y* = *A* × {*1*:: *nat*}
let *?E* = *mk-edge* ' {(*x*, *y*)| *x y*. *x* ∈ *?X* ∧ *y* ∈ *?Y* ∧ (*popular-diff* (*fst y* ⊖ *fst x*) *c A*)}
interpret *H*: *fin-bipartite-graph ?X* ∪ *?Y ?E ?X ?Y*
proof (*unfold-locales*, *auto simp add*: *partition-on-def assms*(*3*) *assms*(*1*) *disjoint-def*)
show {} = *A* × {*0*} ⟹ *False* **using** *assms*(*3*) **by** *auto*
next
show {} = *A* × {*Suc 0*} ⟹ *False* **using** *assms*(*3*) **by** *auto*
next
show *A* × {*0*} = *A* × {*Suc 0*} ⟹ *False* **using** *assms*(*3*) **by** *fastforce*
next
fix *x y* **assume** *x* ∈ *A* **and** *y* ∈ *A* **and** *popular-diff* (*y* ⊖ *x*) *c A*
thus {(*x*, *0*), (*y*, *Suc 0*)} ∈ *all-bi-edges* (*A* × {*0*}) (*A* × {*Suc 0*})
using *all-bi-edges-def*[*of A* × {*0*} *A* × {*Suc 0*}]
by (*simp add*: *in-mk-edge-img*)
qed
have *edges1*: ∀ *a* ∈ *A*. ∀ *b* ∈ *A*. ({(*a*, *0*), (*b*, *1*)} ∈ *?E* ⟷ *popular-diff* (*b* ⊖ *a*) *c A*)
by (*auto simp add*: *in-mk-uedge-img-iff*)
have *hXA*: *card A* = *card ?X* **by** (*simp add*: *card-cartesian-product*)
have *hYA*: *card A* = *card ?Y* **by** (*simp add*: *card-cartesian-product*)
have *hA*: *card A* ≠ *0* **using** *assms card-0-eq* **by** *blast*
have *edge-density*: *H*.*edge-density ?X ?Y* ≥ *c*^*2*
proof−
define *f*:: *'a* ⇒ (*'a* × *nat*) *edge set* **where** *f* ≡ (λ *x*. {{(*a*, *0*), (*b*, *1*)} | *a b*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *b* ⊖ *a* = *x*})
have *f-disj*: *pairwise* (λ *s t*. *disjnt* (*f s*) (*f t*)) (*popular-diff-set c A*)
proof (*intro pairwiseI*)
fix *x y* **assume** *hx*: *x* ∈ *popular-diff-set c A* **and** *hy*: *y* ∈ *popular-diff-set c A*
and *hxy*: *x* ≠ *y*
show *disjnt* (*f x*) (*f y*)
proof−
have ∀ *a*. ¬ (*a* ∈ *f x* ∧ *a* ∈ *f y*)
proof (*intro allI notI*)
fix *a* **assume** *a* ∈ *f x* ∧ *a* ∈ *f y*
then obtain *z w* **where** *hazw*: *a* = {(*z*, *0*), (*w*, *1*)} **and** *hx*: {(*z*,*0*), (*w*, *1*)} ∈ *f x*
and *hy*: {(*z*, *0*), (*w*, *1*)} ∈ *f y* **using** *f-def* **by** *blast*
have *w* ⊖ *z* = *x* **using** *f-def hx* **by** (*simp add*: *doubleton-eq-iff*)

moreover have $w \ominus z = y$ **using** *f-def hy* **by** (*simp add*: *doubleton-eq-iff*)
ultimately show *False* **using** *hxy* **by** *auto*
qed
thus *?thesis* **using** *disjnt-iff* **by** *auto*
qed
qed
have *f-sub-edges*: $\forall\ d \in$ *popular-diff-set* $c\ A.\ (f\ d) \subseteq$ *?E*
using *popular-diff-set-def f-def edges1* **by** *auto*
have *f-union-sub*: $(\bigcup\ d \in$ *popular-diff-set* $c\ A.\ (f\ d)) \subseteq$ *?E* **using** *popular-diff-set-def*
f-def edges1 **by** *auto*
have *f-disj2*: *disjoint* (*f* ‘ (*popular-diff-set c A*)) **using** *f-disj*
pairwise-image[*of disjnt f popular-diff-set c A*] **by** (*simp add*: *pairwise-def*)
have *f-finite*: $\bigwedge B.\ B \in f$ ‘ *popular-diff-set* $c\ A \Longrightarrow$ *finite* B
using *finite-subset f-sub-edges H.fin-edges* **by** *auto*
have *card-eq-f-diff*: $\forall\ d \in$ *popular-diff-set* $c\ A.$ *card* $(f\ d) =$ *f-diff* $d\ A$
proof
fix d **assume** $d \in$ *popular-diff-set* $c\ A$
define g:: $('a \times 'a) \Rightarrow ('a \times nat)$ *edge* **where** $g = (\lambda\ (a,\ b).\ \{(b,\ 0),\ (a,\ 1)\})$
have *g-inj*: *inj-on* $g\ \{(a,\ b)\ |\ a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\}$
proof (*intro inj-onI*)
fix $x\ y$ **assume** $x \in \{(a,\ b)\ |a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\}$ **and**
$y \in \{(a,\ b)\ |a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\}$ **and** *hg*: $g\ x = g\ y$
then obtain *a1 a2 b1 b2* **where** *hx*: $x = (a1,\ a2)$ **and** *hy*: $y = (b1,\ b2)$
by *blast*
thus $x = y$ **using** *g-def hg hx hy* **by** (*simp add*: *doubleton-eq-iff*)
qed
have *g-image*: g ‘ $\{(a,\ b)\ |a\ b.\ a \in A \wedge b \in A \wedge a \ominus b = d\} = f\ d$ **using**
f-def g-def **by** *auto*
show *card* $(f\ d) =$ *f-diff* $d\ A$ **using** *card-image g-inj g-image f-diff-def* **by**
fastforce
qed
have c ^ *2* $*$ (*card A*) ^ *2* $= c *$ (*card A*) $*$ ($c *$ (*card A*)) **using** *power2-eq-square*
by (*metis of-nat-power power-mult-distrib*)
also have ... $\leq$ (*card* (*popular-diff-set c A*)) $*$ ($c *$ (*card A*))
using *assms popular-differences-card hA* **by** *force*
also have ... $\leq (\sum\ d \in$ *popular-diff-set* $c\ A.$ *f-diff* $d\ A)$ **using** *sum-mono*
popular-diff-set-def
popular-diff-def **by** (*smt* (*verit*, *ccfv-SIG*) *mem-Collect-eq of-nat-sum of-real-of-nat-eq*

sum-constant)
also have ... $= (\sum\ d \in$ *popular-diff-set* $c\ A.$ *card* $(f\ d))$
using *card-eq-f-diff sum.cong* **by** *auto*
also have ... $=$ *sum card* (*f* ‘ (*popular-diff-set c A*))
using *f-disj sum-card-image*[*of popular-diff-set c A f*] *popular-diff-set-def*
finite-minusset finite-sumset assms(*1*) *finite-subset* **by** *auto*
also have ... $=$ *card* $(\bigcup\ d \in$ *popular-diff-set* $c\ A.\ (f\ d))$
using *card-Union-disjoint*[*of f* ‘ (*popular-diff-set c A*)] *f-disj2 f-finite* **by** *auto*
also have ... $\leq$ *card ?E* **using** *card-mono f-union-sub H.fin-edges* **by** *auto*

finally have *c* ^ *2* ∗ (*card A*) ^ *2* ≤ *card ?E* **by** *linarith*
then have *c* ^ *2* ∗ (*card A*) ^ *2* ≤ *card* (*H.all-edges-between ?X ?Y*)
using *H.card-edges-between-set* **by** *auto*
moreover have *H.edge-density ?X ?Y* = *card* (*H.all-edges-between ?X ?Y*) / (*card A*) ^ *2*
using *H.edge-density-def power2-eq-square hXA hYA*
by (*smt* (*verit*, *best*))
ultimately have (*c* ^ *2* ∗ (*card A*) ^ *2*) / (*card A*) ^ *2* ≤ *H.edge-density ?X ?Y* **using** *hA*
divide-le-cancel **by** (*smt* (*verit*, *del-insts*) *H.edge-density-ge0* ‹c^2 ∗ *real* ((*card A*)^2) =
c ∗ *real* (*card A*) ∗ (*c* ∗ *real* (*card A*))› *divide-divide-eq-right zero-le-divide-iff*)
thus *?thesis* **using** *hA assms*(*2*) **by** *auto*
qed
obtain *X′* **and** *Y′* **where** *X′sub*: *X′* ⊆ *?X* **and** *Y′sub*: *Y′* ⊆ *?Y* **and**
hX′: *card X′* ≥ (*H.edge-density ?X ?Y*)^*2* ∗ (*card ?X*)/*16* **and**
hY′: *card Y′* ≥ (*H.edge-density ?X ?Y*) ∗ (*card ?Y*)/*4* **and**
hwalks: ∀ *x* ∈ *X′*. ∀ *y* ∈ *Y′*. *card* ({*p*. *H.connecting-walk x y p* ∧ *H.walk-length p* = *3*})
≥ (*H.edge-density ?X ?Y*)^*6* ∗ *card ?X* ∗ *card ?Y* / *2*^*13*
using *H.walks-of-length-3-subsets-bipartite* ‹*c*>*0*› **by** *auto*
have ((*c*^*2*)^*2*) ∗ (*card A*) ≤ (*H.edge-density ?X ?Y*)^*2* ∗ (*card A*)
using *edge-density assms*(*2*) *hA power-mono zero-le-power2 mult-le-cancel-right*
by (*smt* (*verit*) *of-nat-less-of-nat-power-cancel-iff of-nat-zero-less-power-iff*
power2-less-eq-zero-iff power-0-left)
then have *cardX′*: *card X′* ≥ (*c*^*4*) ∗ (*card A*)/*16* **using** *hX′ divide-le-cancel hXA* **by** *fastforce*
have *c*^*2* ∗ (*card A*) / *4* ≤ (*H.edge-density ?X ?Y*) ∗ *card ?Y* / *4* **using** *hYA hA edge-density*
mult-le-cancel-right **by** *simp*
then have *cardY′*: *card Y′* ≥ *c*^*2* ∗ (*card A*)/*4* **using** *hY′* **by** *linarith*
have (*H.edge-density ?X ?Y*)^*6* ∗ (*card ?X* ∗ *card ?Y*)/ *2*^*13* ≥ (*c*^*2*)^*6* ∗ ((*card A*)^*2*) / *2*^*13* **using**
hXA hYA power2-eq-square edge-density divide-le-cancel mult-le-cancel-right hA
by (*smt* (*verit*, *ccfv-SIG*) *of-nat-power power2-less-0 power-less-imp-less-base zero-le-power*)
then have *card-walks*: ∀ *x* ∈ *X′*. ∀ *y* ∈ *Y′*.
card ({*p*. *H.connecting-walk x y p* ∧ *H.walk-length p* = *3*}) ≥ (*c*^*12*) ∗ ((*card A*)^*2*) / *2*^*13*
using *hwalks* **by** *fastforce*

let *?B* = (λ (*a*, *b*). *a*) ‘ *X′*
let *?C* = (λ (*a*, *b*). *a*) ‘ *Y′*
have *hBA*: *?B* ⊆ *A* **and** *hCA*: *?C* ⊆ *A* **using** *Y′sub X′sub* **by** *auto*
have *inj-on-X′*: *inj-on* (λ (*a*, *b*). *a*) *X′* **using** *X′sub* **by** (*intro inj-onI*) (*auto*)
have *inj-on-Y′*: *inj-on* (λ (*a*, *b*). *a*) *Y′* **using** *Y′sub* **by** (*intro inj-onI*) (*auto*)
have *hBX′*: *card ?B* = *card X′* **and** *hCY′*: *card ?C* = *card Y′*
using *card-image inj-on-X′ inj-on-Y′* **by** *auto*
then have *cardB*: *card ?B* ≥ (*c*^*4*) ∗ (*card A*)/*16* **and** *cardC*: *card ?C* ≥ *c*^*2* ∗

(*card A*)/*4*
  using *cardX′ cardY′* **by** *auto*
 have *card-ineq1*: $\bigwedge$ *x y*. *x* ∈ *?B* ⟹ *y* ∈ *?C* ⟹ *card* ({(*z*, *w*) | *z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧
 popular-diff (*z* ⊖ *x*) *c A* ∧ *popular-diff* (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*}) ≥
 (*c*^*12*) ∗ ((*card A*)^*2*) / *2*^*13*
 proof−
  fix *x y* **assume** *hx*: *x* ∈ *?B* **and** *hy*: *y* ∈ *?C*
  have *hxA*: *x* ∈ *A* **and** *hyA*: *y* ∈ *A* **using** *hx hy hBA hCA* **by** *auto*
  define *f*:: ′*a* × ′*a* ⇒ (′*a* × *nat*) *list*
   where *f* ≡ (λ (*z*, *w*). [(*x*, *0*), (*z*, *1*), (*w*, *0*), (*y*, *1*)])
  have *f-inj-on*: *inj-on f* {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
   popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} **using** *f-def* **by** (*intro inj-onI*) (*auto*)
  have *f-image*: *f* ‘ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
   popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} =
   {*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length p* = *3*}
  proof
   show *f* ‘ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
    popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} ⊆
    {*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length p* = *3*}
   proof
    fix *p* **assume** *hp*: *p* ∈ *f* ‘ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
     popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*}
    then obtain *z w* **where** *hz*: *z* ∈ *A* **and** *hw*: *w* ∈ *A* **and** *hzx*: *popular-diff* (*z* ⊖ *x*) *c A* **and**
     hzw: *popular-diff* (*z* ⊖ *w*) *c A* **and** *hyw*: *popular-diff* (*y* ⊖ *w*) *c A* **and**
     hp: *p* = [(*x*, *0*), (*z*, *1*), (*w*, *0*), (*y*, *1*)] **using** *f-def hp* **by** *fast*
    then have *hcon*: *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p*
     unfolding *H*.*connecting-walk-def H*.*is-walk-def*
     using *hxA hyA H*.*vert-adj-def H*.*vert-adj-sym edges1* **by** *simp*
    thus *p* ∈ {*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length p* = *3*}
     using *hp H*.*walk-length-conv* **by** *auto*
    qed
   next
    show {*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length p* = *3*} ⊆ *f* ‘ {(*z*, *w*) |*z w*.
     z ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧ *popular-diff* (*z* ⊖ *w*) *c A* ∧
     popular-diff (*y* ⊖ *w*) *c A*}
    proof(*intro subsetI*)
    fix *p* **assume** *hp*: *p* ∈ {*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length p* = *3*}
     then have *len*: *length p* = *4* **using** *H*.*walk-length-conv* **by** *auto*
     have *hpsub*: *set p* ⊆ *A* × {*0*} ∪ *A* × {*1*} **using** *hp H*.*connecting-walk-def H*.*is-walk-def*
      by *auto*

then have *fst-sub*: *fst* ‘ *set* $p \subseteq A$ **by** *auto*
have *h1A*: *fst* $(p!1) \in A$ **and** *h2A*: *fst* $(p!2) \in A$ **using** *fst-sub len* **by** *auto*
have *hpnum*: $p = [p!0, p!1, p!2, p!3]$
proof (*auto simp add*: *list-eq-iff-nth-eq len*)
fix k **assume** $k < (4:: nat)$
then have $k = 0 \vee k = 1 \vee k = 2 \vee k = 3$ **by** *auto*
thus $p ! k = [p ! 0, p ! Suc\ 0, p ! 2, p ! 3] ! k$ **by** *fastforce*
qed
then have *set* (*H.walk-edges p*) $= \{\{p!0, p!1\}, \{p!1, p!2\}, \{p!2, p!3\}\}$
using
comp-sgraph.walk-edges.simps(*2*) *comp-sgraph.walk-edges.simps*(*3*)
by (*metis empty-set list.simps*(*15*))
then have *h1*: $\{p!0, p!1\} \in ?E$ **and** *h2*: $\{p!2, p!1\} \in ?E$ **and** *h3*: $\{p!2, p!3\} \in ?E$
using *hp H.connecting-walk-def H.is-walk-def len* **by** *auto*
have *hxp*: $p!0 = (x, 0)$ **using** *hp len hd-conv-nth H.connecting-walk-def H.is-walk-def*
by *fastforce*
have *hyp*: $p!3 = (y,1)$ **using** *hp len last-conv-nth H.connecting-walk-def H.is-walk-def*
by *fastforce*
have *h1p*: $p!1 = (fst\ (p!1), 1)$
proof−
have $p!1 \in A \times \{0\} \cup A \times \{1\}$ **using** *hpnum hpsub*
by (*metis* (*no-types*, *lifting*) *insertCI list.simps*(*15*) *subsetD*)
then have *hsplit*: *snd* $(p!1) = 0 \vee$ *snd* $(p!1) = 1$ **by** *auto*
then have *snd* $(p!1) = 1$
proof(*cases snd* $(p!1) = 0$)
case *True*
then have *1*: $\{(x, 0), (fst\ (p!1), 0)\} \in ?E$ **using** *h1 hxp doubleton-eq-iff*
by (*smt* (*verit*, *del-insts*) *surjective-pairing*)
have *hY*: $(fst\ (p!1), 0) \notin ?Y$ **and** *hX*: $(x, 0) \in ?X$ **using** *hxA* **by** *auto*
then have *2*: $\{(x, 0), (fst\ (p!1), 0)\} \notin ?E$ **using** *H.X-vert-adj-Y H.vert-adj-def* **by** *meson*
then show *?thesis* **using** *1 2* **by** *blast*
next
case *False*
then show *?thesis* **using** *hsplit* **by** *auto*
qed
thus $(p ! 1) = (fst\ (p ! 1), 1)$
by (*metis* (*full-types*) *split-pairs*)
qed
have *h2p*: $p!2 = (fst\ (p!2), 0)$
proof−
have $p!2 \in A \times \{0\} \cup A \times \{1\}$ **using** *hpnum hpsub*
by (*metis* (*no-types*, *lifting*) *insertCI list.simps*(*15*) *subsetD*)
then have *hsplit*: *snd* $(p!2) = 0 \vee$ *snd* $(p!2) = 1$ **by** *auto*
then have *snd* $(p!2) = 0$

proof(*cases snd* (*p*!*2*) = *1*)
case *True*
then have *1*: {(*fst* (*p*!*2*), *1*), (*y*, *1*)} ∈ *?E* **using** *h3 hyp doubleton-eq-iff*
by (*smt* (*verit*, *del-insts*) *surjective-pairing*)
have *hY*: (*y*, *1*) ∉ *?X* **and** *hX*: (*fst* (*p*!*2*), *1*) ∈ *?Y* **using** *hyA h2A* **by** *auto*
then have *2*: {(*fst* (*p*!*2*), *1*), (*y*, *1*)} ∉ *?E* **using** *H.Y-vert-adj-X H.vert-adj-def*
by *meson*
then show *?thesis* **using** *1 2* **by** *blast*
next
case *False*
then show *?thesis* **using** *hsplit* **by** *auto*
qed
thus (*p* ! *2*) = (*fst* (*p* ! *2*), *0*)
by (*metis* (*full-types*) *split-pairs*)
qed
have *hpop1*: *popular-diff* ((*fst* (*p*!*1*)) ⊖ *x*) *c A* **using** *edges1 h1 hxp h1p hxA h1A*
by (*smt* (*z3*))
have *hpop2*: *popular-diff*((*fst* (*p*!*1*)) ⊖ (*fst* (*p*!*2*))) *c A* **using** *edges1 h2 h1p h2p h1A h2A*
by (*smt* (*z3*))
have *hpop3*: *popular-diff* (*y* ⊖ (*fst* (*p*!*2*))) *c A* **using** *edges1 h3 h2p hyp hyA h2A*
by (*smt* (*z3*))
thus *p* ∈ *f* ‘ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} **using** *f-def hpnum hxp h1p h2p hyp*
h1A h2A hpop1 hpop2 hpop3 **by** *force*
qed
qed
have *hx1*: (*x*, *0*) ∈ *X′* **and** *hy2*: (*y*, *1*) ∈ *Y′* **using** *hx X′sub hy Y′sub* **by** *auto*
have *card* {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} =
card {*p*. *H.connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H.walk-length p* = *3*}
using *card-image f-inj-on f-image* **by** *fastforce*
thus *card* {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} ≥ *c* ^ *12* ∗ ((*card A*)^*2*) / *2* ^ *13*
using *hx1 hy2 card-walks* **by** *auto*
qed
have *card-ineq2*: ⋀ *x y z w*. *x* ∈ *?B* ⟹ *y* ∈ *?C* ⟹ (*z*, *w*) ∈ {(*z*, *w*) | *z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧
popular-diff (*z* ⊖ *x*) *c A* ∧ *popular-diff* (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} ⟹
card {(*p*, *q*, *r*, *s*, *t*, *u*)| *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧

$p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus w\} \geq c\hat{\ }3 * card\ A\hat{\ }3$
proof (*auto*)
fix *x x2 y y2 z w* **assume** $(x, x2) \in X'$ **and** $(y, y2) \in Y'$ **and** $z \in A$ **and** $w \in A$ **and**
1: *popular-diff* $(z \ominus x)$ *c A* **and** *2*: *popular-diff* $(z \ominus w)$ *c A* **and**
3: *popular-diff* $(y \ominus w)$ *c A*
define $f{::}\ 'a \times 'a \times 'a \times 'a \times 'a \times 'a \Rightarrow ('a \times 'a) \times ('a \times 'a) \times ('a \times 'a)$ **where**
$f \equiv (\lambda\ (p, q, r, s, t, u).\ ((p, q), (r, s), (t, u)))$

have *f-inj*: *inj-on f* $\{(p, q, r, s, t, u)|\ p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge$
$t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus w\}$ **using** *f-def*
by (*intro inj-onI*) (*auto*)
have *f-image*: f ‘ $\{(p, q, r, s, t, u)|\ p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge$
$t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus w\} =$
$\{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus x\} \times$
$\{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus w\} \times$
$\{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = y \ominus w\}$ **using** *f-def* **by** *force*
have *card* $\{(p, q, r, s, t, u)|\ p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge$
$t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus w\} = card$ $(\{(p, q).\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus x\} \times$
$\{(p, q).\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus w\} \times \{(p, q).\ p \in A \wedge q \in A \wedge p \ominus q = y \ominus w\})$
using *card-image f-inj f-image* **by** *fastforce*
moreover have *card* $(\{(p, q).\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus x\} \times$
$\{(p, q).\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus w\} \times \{(p, q).\ p \in A \wedge q \in A \wedge p \ominus q = y \ominus w\}) =$
card $\{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus x\}\ *$
card $\{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus w\}\ *$
card $\{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = y \ominus w\}$
using *card-cartesian-product3* **by** *auto*
moreover have $c * card\ A \leq card\ \{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus x\}$
using *1 popular-diff-def f-diff-def* **by** *auto*
moreover then have $(c * card\ A) * (c * card\ A) \leq card\ \{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus x\}$
$*\ card\ \{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus w\}$
using *2 popular-diff-def f-diff-def mult-mono assms*(*2*) *mult-nonneg-nonneg of-nat-0-le-iff of-nat-mult* **by** *fastforce*
moreover then have $(c * card\ A) * (c * card\ A) * (c * card\ A) \leq card\ \{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus x\}$
$*\ card\ \{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = z \ominus w\}\ *$
$card\ \{(p, q)\ |\ p\ q.\ p \in A \wedge q \in A \wedge p \ominus q = y \ominus w\}$
using *3 popular-diff-def f-diff-def mult-mono assms*(*2*) *mult-nonneg-nonneg of-nat-0-le-iff*
of-nat-mult **by** *fastforce*

moreover have $c \,\hat{}\; 3 * card\ A \,\hat{}\; 3 = (c * card\ A) * ((c * card\ A) * (c * card\ A))$
by (*simp add*: *power3-eq-cube algebra-simps*)
ultimately show $c \,\hat{}\; 3 * real\ (card\ A) \,\hat{}\; 3 \le$
$(card\ \{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge$
$p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus w\})$ **by** *auto*
qed
have *card-ineq3*: $\bigwedge x\ y.\ \ x \in ?B \Longrightarrow y \in ?C \Longrightarrow card\ (\bigcup\ (z, w) \in \{(z, w) \mid z\ w.\ z \in A \wedge w \in A \wedge$
$popular\text{-}diff\ (z \ominus x)\ c\ A \wedge popular\text{-}diff\ (z \ominus w)\ c\ A \wedge popular\text{-}diff\ (y \ominus w)\ c\ A\}.$
$\{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge$
$t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus w\}) \ge$
$c \,\hat{}\; 15 * ((card\ A)\hat{}5)\ /\ 2 \,\hat{}\; 13$
proof−
fix $x\ y$ **assume** *hx*: $x \in ?B$ **and** *hy*: $y \in ?C$
have *hxG*: $x \in G$ **and** *hyG*: $y \in G$ **using** *hx hy hBA hCA assms*(*4*) **by** *auto*
let $?f = (\lambda\ (z, w).\ \{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge$
$r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus w\})$
have *h-pairwise-disjnt*: $pairwise\ (\lambda\ a\ b.\ disjnt\ (?f\ a)\ (?f\ b))$
$\{(z, w) \mid z\ w.\ z \in A \wedge w \in A \wedge popular\text{-}diff\ (z \ominus x)\ c\ A \wedge popular\text{-}diff\ (z \ominus w)\ c\ A \wedge$
$popular\text{-}diff\ (y \ominus w)\ c\ A\}$
proof (*intro pairwiseI*)
fix $a\ b$ **assume** $a \in \{(z, w) \mid z\ w.\ z \in A \wedge w \in A \wedge popular\text{-}diff\ (z \ominus x)\ c\ A \wedge$
$popular\text{-}diff\ (z \ominus w)\ c\ A \wedge popular\text{-}diff\ (y \ominus w)\ c\ A\}$ $b \in \{(z, w) \mid z\ w.\ z \in A \wedge w \in A \wedge$
$popular\text{-}diff\ (z \ominus x)\ c\ A \wedge popular\text{-}diff\ (z \ominus w)\ c\ A \wedge popular\text{-}diff\ (y \ominus w)\ c\ A\}$ **and**
$a \neq b$
then obtain *a1 a2 b1 b2* **where** *ha*: $a = (a1, a2)$ **and** *hb*: $b = (b1, b2)$ **and** *ha1*: $a1 \in G$ **and**
ha2: $a2 \in G$ **and** *hb1*: $b1 \in G$ **and** *hb2*: $b2 \in G$ **and** *hne*: $(a1, a2) \neq (b1, b2)$
using *assms*(*4*) **by** *blast*
have $(\forall x.\ \neg\ (x \in (?f\ a) \wedge x \in (?f\ b)))$
proof(*intro allI notI*)
fix d **assume** $d \in (?f\ a) \wedge d \in (?f\ b)$
then obtain $p\ q\ r\ s\ t\ u$ **where** $d = (p, q, r, s, t, u)$ **and** *hpq1*: $p \ominus q = a1 \ominus x$ **and**
htu1: $t \ominus u = y \ominus a2$ **and** *hpq2*: $p \ominus q = b1 \ominus x$ **and** *htu2*: $t \ominus u = y \ominus b2$
using *ha hb* **by** *auto*
then have $y \ominus a2 = y \ominus b2$ **using** *htu1 htu2* **by** *auto*
then have *2*: $a2 = b2$ **using** *ha2 hb2 hyG*
by (*metis additive-abelian-group.inverse-closed additive-abelian-group-axioms*

invertible invertible-inverse-inverse invertible-left-cancel)
have *1*: *a1* = *b1* **using** *hpq1 hpq2 ha1 hb1 hxG* **by** *simp*
show *False* **using** *1 2 hne* **by** *auto*
qed
thus *disjnt* (*?f a*) (*?f b*) **using** *disjnt-iff*[*of* (*?f a*) (*?f b*)] **by** *auto*
qed
have *hfinite-walks*: ⋀ *B*. *B* ∈ ((λ (*z*, *w*). {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧
r ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊖ *q* = *z* ⊖ *x* ∧ *r* ⊖ *s* = *z* ⊖ *w* ∧ *t* ⊖ *u* = *y* ⊖ *w*}) ‘
{(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧ *popular-diff* (*z* ⊖ *w*) *c A* ∧
popular-diff (*y* ⊖ *w*) *c A*}) ⟹ *finite B*
proof−
fix *B* **assume** *B* ∈ ((λ (*z*, *w*). {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧
r ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊖ *q* = *z* ⊖ *x* ∧ *r* ⊖ *s* = *z* ⊖ *w* ∧ *t* ⊖ *u* = *y* ⊖ *w*}) ‘
{(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧ *popular-diff* (*z* ⊖ *w*) *c A* ∧
popular-diff (*y* ⊖ *w*) *c A*})
then have *B* ⊆ *A* × *A* × *A* × *A* × *A* × *A* **by** *auto*
thus *finite B* **using** *assms*(*1*)
by (*auto simp add*: *finite-subset*)
qed
have *hdisj*: *disjoint* ((λ (*z*, *w*). {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧
r ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊖ *q* = *z* ⊖ *x* ∧ *r* ⊖ *s* = *z* ⊖ *w* ∧ *t* ⊖ *u* = *y* ⊖ *w*}) ‘
{(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧ *popular-diff* (*z* ⊖ *w*) *c A* ∧
popular-diff (*y* ⊖ *w*) *c A*}) **using** *h-pairwise-disjnt pairwise-image*[*of disjnt* (λ (*z*, *w*). {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧
r ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊖ *q* = *z* ⊖ *x* ∧ *r* ⊖ *s* = *z* ⊖ *w* ∧ *t* ⊖ *u* = *y* ⊖ *w*})
{(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧ *popular-diff* (*z* ⊖ *w*) *c A* ∧
popular-diff (*y* ⊖ *w*) *c A*}] **by** (*simp add*: *pairwise-def*)
have {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} ⊆ *A* × *A* **by** *auto*
then have *hwalks-finite*: *finite* {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} **using** *finite-subset assms*(*1*)
by *fastforce*
have *f-ineq*: ∀ *a* ∈ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*}. *c* ^ *3* ∗ (*card A*) ^ *3* ≤
card (*?f a*) **using** *card-ineq2 hx hy* **by** *auto*
have *c* ^ *15* ∗ ((*card A*)^*5*) / *2* ^ *13* = (*c* ^ *12* ∗ (*card A*) ^ *2* / *2* ^ *13*) ∗ (*c*

^ *3* ∗ *card A* ^ *3*)
by (*simp add*: *algebra-simps*)
also have ... ≤ *card* {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} ∗ (*c* ^ *3* ∗ (*card A*) ^ *3*)
using *card-ineq1*[*of x y*] *hx hy mult-le-cancel-right hA* **by** (*smt* (*verit*, *best*) *assms*(*2*)
mult-pos-pos of-nat-0-less-iff of-nat-le-0-iff zero-less-power)
also have ... = (∑ *a* ∈ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*}. (*c* ^ *3* ∗ (*card A*) ^ *3*)) **by** *auto*
also have ... ≤ (∑*a*∈{(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*}. *card* (*?f a*))
using *sum-mono f-ineq* **by** (*smt* (*verit*, *del-insts*) *of-nat-sum*)
also have ... = *sum card* (*?f* ‘ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*})
using *sum-card-image*[*of* {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*} *?f*] *h-pairwise-disjnt hwalks-finite* **by** *auto*
also have ... = *card* (⋃ (*z*, *w*)∈{(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*}. {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*.
p ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊖ *q* = *z* ⊖ *x* ∧ *r* ⊖ *s* = *z* ⊖ *w* ∧
t ⊖ *u* = *y* ⊖ *w*}) **using** *card-Union-disjoint hfinite-walks hdisj* **by** (*metis* (*no-types*, *lifting*))
finally show *c* ^ *15* ∗ *real* (*card A* ^ *5*) / *2* ^ *13* ≤ *real* (*card* (⋃(*z*, *w*)∈{(*z*, *w*) |*z w*.
z ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧ *popular-diff* (*z* ⊖ *w*) *c A* ∧
popular-diff (*y* ⊖ *w*) *c A*}. {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧
s ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊖ *q* = *z* ⊖ *x* ∧ *r* ⊖ *s* = *z* ⊖ *w* ∧ *t* ⊖ *u* = *y* ⊖ *w*})) **by** *simp*
qed
have *hdsub*: ∀ *d* ∈ *differenceset ?C ?B*. ∃ *y* ∈ *?C*. ∃ *x* ∈ *?B*.
(⋃ (*z*, *w*) ∈ {(*z*, *w*) | *z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*}.
{(*p*, *q*, *r*, *s*, *t*, *u*)| *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧
t ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊖ *q* = *z* ⊖ *x* ∧ *r* ⊖ *s* = *z* ⊖ *w* ∧ *t* ⊖ *u* = *y* ⊖ *w*})
⊆ {(*p*, *q*, *r*, *s*, *t*, *u*) | *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧
s ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *d* = *p* ⊖ *q* ⊖ *r* ⊕ *s* ⊕ *t* ⊖ *u*}
proof
fix *d* **assume** *d* ∈ *differenceset ?C ?B*

then obtain *y x* **where** *hy*: $y \in$ *?C* **and** *hx*: $x \in$ *?B* **and** *hxy*: $d = y \ominus x$
using *sumset-def minusset-def hBA hCA assms(4) subset-trans*
by (*smt* (*verit*, *best*) *minusset.simps sumset.cases*)
have *hxG*: $x \in G$ **and** *hyG*: $y \in G$ **using** *hx hy hBA hCA assms(4)* **by** *auto*
have ($\bigcup$(z, w)$\in${(z, w) |z w. $z \in A \wedge w \in A \wedge$ *popular-diff* ($z \ominus x$) c $A \wedge$
popular-diff ($z \ominus w$) c $A \wedge$ *popular-diff* ($y \ominus w$) c A}. {(p, q, r, s, t, u) |p q
r s t u.
$p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s$
$= z \ominus w \wedge t \ominus u = y \ominus w$})
$\subseteq$ {(p, q, r, s, t, u) |p q r s t u. $p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge$
$u \in A \wedge$
$d = p \ominus q \ominus r \oplus s \oplus t \ominus u$}
proof (*rule Union-least*)
fix X **assume** $X \in$ (λ(z, w). {(p, q, r, s, t, u) |p q r s t u.
$p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s$
$= z \ominus w \wedge$
$t \ominus u = y \ominus w$}) ' {($z$, w) |z w. $z \in A \wedge w \in A \wedge$ *popular-diff* ($z \ominus x$) c $A \wedge$
popular-diff ($z \ominus w$) c $A \wedge$ *popular-diff* ($y \ominus w$) c A}
then obtain *z w* **where** *hX*: $X =$ {(p, q, r, s, t, u) |p q r s t u. $p \in A \wedge q$
$\in A \wedge r \in A \wedge$
$s \in A \wedge t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus$
w}
and *hz*: $z \in A$ **and** *hw*: $w \in A$ **by** *auto*
have *hzG*: $z \in G$ **and** *hwG*: $w \in G$ **using** *hz hw assms(4)* **by** *auto*
have {(p, q, r, s, t, u) |p q r s t u. $p \in A \wedge q \in A \wedge r \in A \wedge$
$s \in A \wedge t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus$
w} $\subseteq$
{(p, q, r, s, t, u) |p q r s t u. $p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge u$
$\in A \wedge$
$d = p \ominus q \ominus r \oplus s \oplus t \ominus u$}
proof
fix e **assume** $e \in$ {(p, q, r, s, t, u) |p q r s t u. $p \in A \wedge q \in A \wedge r \in A \wedge$
$s \in A \wedge$
$t \in A \wedge u \in A \wedge p \ominus q = z \ominus x \wedge r \ominus s = z \ominus w \wedge t \ominus u = y \ominus w$}
then obtain *p q r s t u* **where** $p \ominus q = z \ominus x$ **and** $r \ominus s = z \ominus w$ **and** t
$\ominus u = y \ominus w$
and *hp*: $p \in A$ **and** *hq*: $q \in A$ **and** *hr*: $r \in A$ **and** *hs*: $s \in A$ **and** *ht*: $t \in$
A
and *hu*: $u \in A$ **and** *he*: $e = (p, q, r, s, t, u)$ **by** *blast*
then have $p \ominus q \ominus r \oplus s \oplus t \ominus u = (z \ominus x) \ominus (z \ominus w) \oplus (y \ominus w)$
by (*smt* (*z3*) *additive-abelian-group.inverse-closed additive-abelian-group-axioms*
assms(4)
associative commutative-monoid.commutative commutative-monoid-axioms
composition-closed
invertible invertible-inverse-inverse monoid.inverse-composition-commute
monoid-axioms subsetD)
also have ... $= (w \ominus x) \oplus (y \ominus w)$ **using** *hxG hyG hzG hwG associative*
commutative
inverse-composition-commute invertible-right-inverse2 **by** *auto*

also have ... = *y* ⊖ *x* **using** *hxG hwG hyG associative commutative*
by (*simp add*: *monoid.invertible-right-inverse2 monoid-axioms*)
finally have *p* ⊖ *q* ⊖ *r* ⊕ *s* ⊕ *t* ⊖ *u* = *d* **using** *hxy* **by** *simp*
thus *e* ∈ {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧
u ∈ *A* ∧ *d* = *p* ⊖ *q* ⊖ *r* ⊕ *s* ⊕ *t* ⊖ *u*} **using** *he hp hq hr hs ht hu* **by** *auto*
qed
thus *X* ⊆ {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*.
p ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *d* = *p* ⊖ *q* ⊖ *r* ⊕ *s* ⊕ *t* ⊖ *u*}
using *hX* **by** *auto*
qed
thus ∃ *y*∈(λ(*a*, *b*). *a*) ' *Y'*. ∃ *x*∈(λ(*a*, *b*). *a*) ' *X'*. (⋃(*z*, *w*)∈{(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧
popular-diff (*z* ⊖ *x*) *c A* ∧ *popular-diff* (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*}.
{(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧
p ⊖ *q* = *z* ⊖ *x* ∧ *r* ⊖ *s* = *z* ⊖ *w* ∧ *t* ⊖ *u* = *y* ⊖ *w*}) ⊆ {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*.
p ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *d* = *p* ⊖ *q* ⊖ *r* ⊕ *s* ⊕ *t* ⊖ *u*}
using *hx hy* **by** *meson*
qed
have *pos*: *0* < *c* ^ *15* ∗ *real* (*card A* ^ *5*) / *2* ^ *13* **using** *hA* ‹*c*>*0*› **by** *auto*
have (*5*:: *nat*) ≤ *6* **by** *auto*
then have (*card A* ^ *6* / *card A* ^ *5*) = (*card A*) ^(*6* − *5*)
using *hA power-diff* **by** (*metis of-nat-eq-0-iff of-nat-power*)
then have *cardApow*: (*card A* ^ *6* / *card A* ^ *5*) = *card A* **using** *power-one-right* **by** *simp*
moreover have ∀ *d* ∈ *differenceset ?C ?B*. *card* {(*p*, *q*, *r*, *s*, *t*, *u*) | *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧
r ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ (*d* = *p* ⊖ *q* ⊖ *r* ⊕ *s* ⊕ *t* ⊖ *u*)} ≥ *c* ^ *15* ∗ (*card A*) ^ *5* / *2* ^*13*
proof
fix *d* **assume** *d* ∈ *differenceset ?C ?B*
then obtain *x y* **where** *hy*: *y* ∈ *?C* **and** *hx*: *x* ∈ *?B* **and** *hsub*:
(⋃ (*z*, *w*) ∈ {(*z*, *w*) | *z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-diff* (*z* ⊖ *x*) *c A* ∧
popular-diff (*z* ⊖ *w*) *c A* ∧ *popular-diff* (*y* ⊖ *w*) *c A*}.
{(*p*, *q*, *r*, *s*, *t*, *u*)| *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧
t ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊖ *q* = *z* ⊖ *x* ∧ *r* ⊖ *s* = *z* ⊖ *w* ∧ *t* ⊖ *u* = *y* ⊖ *w*})
⊆ {(*p*, *q*, *r*, *s*, *t*, *u*) | *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧
s ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *d* = *p* ⊖ *q* ⊖ *r* ⊕ *s* ⊕ *t* ⊖ *u*} **using** *hdsub* **by** *meson*
have {(*p*, *q*, *r*, *s*, *t*, *u*) | *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧
s ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *d* = *p* ⊖ *q* ⊖ *r* ⊕ *s* ⊕ *t* ⊖ *u*} ⊆ *A* × *A* × *A* × *A* × *A* × *A* **by** *auto*
then have *fin*: *finite* {(*p*, *q*, *r*, *s*, *t*, *u*) | *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧
s ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *d* = *p* ⊖ *q* ⊖ *r* ⊕ *s* ⊕ *t* ⊖ *u*}
using *finite-subset assms*(*1*) *finite-cartesian-product* **by** *fastforce*

have *c ^ 15 ∗ (card A) ^ 5 / 2 ^13 ≤ card (⋃ (z, w) ∈ {(z, w) | z w. z ∈ A ∧ w ∈ A ∧ popular-diff (z ⊖ x) c A ∧*
popular-diff (z ⊖ w) c A ∧ popular-diff (y ⊖ w) c A}.
{(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧ r ∈ A ∧ s ∈ A ∧
t ∈ A ∧ u ∈ A ∧ p ⊖ q = z ⊖ x ∧ r ⊖ s = z ⊖ w ∧ t ⊖ u = y ⊖ w})
using *card-ineq3 hx hy* **by** *auto*
also have ... *≤ card {(p, q, r, s, t, u) | p q r s t u. p ∈ A ∧ q ∈ A ∧ r ∈ A ∧*
s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊖ q ⊖ r ⊕ s ⊕ t ⊖ u}
using *hsub card-mono fin* **by** *auto*
finally show *c ^ 15 ∗ (card A) ^ 5 / 2 ^13 ≤ card {(p, q, r, s, t, u) | p q r s t u. p ∈ A ∧ q ∈ A ∧ r ∈ A ∧*
s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊖ q ⊖ r ⊕ s ⊕ t ⊖ u} **by** *linarith*
qed
moreover have *pairwise (λ s t. disjnt ((λ d. {(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧*
r ∈ A ∧ s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ (d = p ⊖ q ⊖ r ⊕ s ⊕ t ⊖ u)}) s)
((λ d. {(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧
r ∈ A ∧ s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ (d = p ⊖ q ⊖ r ⊕ s ⊕ t ⊖ u)}) t))
(differenceset ?C ?B)
unfolding *disjnt-def* **by** *(intro pairwiseI) (auto)*
moreover have *∀ d ∈ differenceset ?C ?B. ((λ d. {(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧*
r ∈ A ∧ s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ (d = p ⊖ q ⊖ r ⊕ s ⊕ t ⊖ u)}) d) ⊆ A × A × A × A × A × A
by *blast*
ultimately have *card (differenceset ?C ?B) ≤ ((card A) ^ 6) / (c^15 ∗ (card A)^5 /2^13)*
using *assms(1) hA finite-cartesian-product card-cartesian-product-6[of A]*
pos card-le-image-div[of A × A × A × A × A × A (λ d. {(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧
r ∈ A ∧ s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ (d = p ⊖ q ⊖ r ⊕ s ⊕ t ⊖ u)}) differenceset ?C ?B
(c^15 ∗ (card A)^5 /2^13)] **by** *auto*
also have ... *= (card A ^ 6 / card A ^ 5) / (c^15 / 2^13)*
using *hA assms(3) field-simps* **by** *simp*
also have ... *= (card A) / (c ^ 15 / 2 ^ 13)*
using *cardApow* **by** *metis*
finally have *final: card (differenceset ?C ?B) ≤ 2 ^ 13 ∗ (1 / c ^ 15) ∗ real (card A)*
by *argo*
have *0 < c ^ 4 ∗ real (card A) / 16* **and** *0 < c ^ 2 ∗ real (card A) / 4* **using** *assms(2) hA* **by** *auto*
then have *?B ≠ {}* **and** *?C ≠ {}* **using** *cardB cardC* **by** *auto*
then show *?thesis* **using** *hCA hBA cardC cardB final that* **by** *auto*
qed

We now show the main theorem, which is a direct application of lemma *obtains-subsets-differenceset-card-bound* and the Ruzsa triangle inequality. (The main theorem corresponds to Corollary 2.19 in Gowers's notes [3].)

theorem *Balog-Szemeredi-Gowers*: **fixes** *A*::′*a set* **and** *c*::*real*
assumes *afin*: *finite A* **and** *A* ≠ {} **and** *c>0* **and** *additive-energy A* = *2* ∗ *c* **and** *ass*: *A* ⊆ *G*
obtains *A′* **where** *A′* ⊆ *A* **and** *card A′* ≥ *c^2* ∗ *card A* / *4* **and**
card (*differenceset A′ A′*) ≤ *2^30* ∗ *card A* / *c^34*
proof−
obtain *B* **and** *A′* **where** *bss*: *B* ⊆ *A* **and** *bne*: *B* ≠ {} **and** *bge*: *card B* ≥ (*c^4*) ∗ (*card A*)/*16*
and *a2ss*: *A′* ⊆ *A* **and** *a2ge*: *card A′* ≥ (*c^2*) ∗ (*card* (*A*))/*4*
and *hcardle*: *card* (*differenceset A′ B*) ≤ *2^13* ∗ *card A* / *c^15*
using *assms obtains-subsets-differenceset-card-bound* **by** *metis*
have *Bg0*: (*card B* :: *real*) > *0* **using** *bne afin bss infinite-super* **by** *fastforce*
have (*card B*) ∗ *card* (*differenceset A′ A′*) ≤
card (*differenceset A′ B*) ∗ *card* (*differenceset A′ B*)
using *afin a2ss bss infinite-super ass Ruzsa-triangle-ineq1 card-minusset′ differenceset-commute*
sumset-subset-carrier subset-trans sumset-commute **by** (*smt* (*verit*, *best*))
then have *card B* ∗ *card* (*differenceset A′ A′*) ≤ (*card* (*differenceset A′ B*))*^2*
using *bss bss power2-eq-square* **by** *metis*
then have (*card* (*differenceset A′ A′*)) ≤ (*card* (*differenceset A′ B*))*^2*/*card B*
using *Bg0 nonzero-mult-div-cancel-left*[*of card B card*(*differenceset A′ A′*)]
divide-right-mono **by** (*smt* (*verit*) *of-nat-0 of-nat-mono real-of-nat-div4*)
moreover have (*card* (*differenceset A′ B*))*^2* ≤ ((*2^13*) ∗ (*1*/*c^15*)∗(*card A*))*^2*
using *hcardle* **by** *simp*
ultimately have (*card* (*differenceset A′ A′*)) ≤ ((*2^13*) ∗ (*1*/*c^15*)∗(*card A*))*^2*/(*card B*)
using *pos-le-divide-eq*[*OF Bg0*] **by** *simp*
moreover have (*c^4*) ∗ (*card A*)/*16* >*0*
using *assms card-0-eq* **by** *fastforce*
moreover have ((*2^13*) ∗ (*1*/*c^15*) ∗ (*card A*))*^2*/(*card B*) =
((*2^13*)∗ (*1*/*c^15*)∗(*card A*))*^2* ∗ (*1*/(*card B*)) **by** *simp*
moreover have ((*2^13*)∗ (*1*/*c^15*) ∗ (*card A*))*^2* ∗ (*1*/(*card B*)) ≤
((*2^13*) ∗ (*1*/*c^15*) ∗ (*card A*))*^2*/ ((*c^4*) ∗ (*card A*)/*16*)
using *bge calculation*(*2*, *3*) *frac-le less-eq-real-def zero-le-power2* **by** *metis*
ultimately have (*card* (*differenceset A′ A′*)) ≤ ((*2^13*) ∗ (*1*/*c^15*) ∗ (*card A*))*^2*/ ((*c^4*) ∗ (*card A*)/*16*)
by *linarith*
then have (*card* (*differenceset A′ A′*)) ≤ (*2^30*) ∗ (*card A*)/(*c^34*)
using *card-0-eq assms* **by** (*simp add*: *power2-eq-square*)
then show *?thesis* **using** *a2ss a2ge that* **by** *blast*
qed

The following is an analogous version of the Balog–Szemerédi–Gowers Theorem for a sumset instead of a difference set. The proof is similar to that of the original version, again using *obtains-subsets-differenceset-card-bound*, however, instead of the Ruzsa triangle inequality we will use the alternative triangle inequality for sumsets *triangle-ineq-sumsets*.

theorem *Balog-Szemeredi-Gowers-sumset*: **fixes** *A*::′*a set* **and** *c*::*real*

assumes *afin*: *finite A* **and** $A \neq \{\}$ **and** *c>0* **and** *additive-energy A* = *2* ∗ *c* **and** *ass*: $A \subseteq G$
obtains A' **where** $A' \subseteq A$ **and** *card* $A' \geq c\hat{}2 * card\ A\ /\ 4$ **and**
card (*sumset* A' A') $\leq 2\hat{}30 * card\ A\ /\ c\hat{}34$

```
proof−
  obtain B and A' where bss: B ⊆ A and bne: B ≠ {} and bge: card B ≥ (c^4)
∗ (card A)/16
    and a2ss: A' ⊆ A  and a2ne: A' ≠ {} and a2ge: card A' ≥ (c^2) ∗ (card
(A))/4
    and hcardle: card (differenceset A' B) ≤ 2^13 ∗ card A / c^15
    using assms obtains-subsets-differenceset-card-bound by metis
  have finA': finite A' and finB: finite B using afin a2ss bss using infinite-super
by auto
  have bg0: (card B :: real) > 0 using bne afin bss infinite-super by fastforce
  have card (minusset B) ∗ card (sumset A' A') ≤
    card (sumset (minusset B) A') ∗ card (sumset (minusset B) A')
    using finA' finB ass a2ss bss triangle-ineq-sumsets
    finite-minusset minusset-subset-carrier subset-trans by metis
  then have card B ∗ card (sumset A' A') ≤ (card (differenceset A' B))^2
    using card-minusset bss ass power2-eq-square
    by (metis card-minusset' subset-trans sumset-commute)
  then have (card (sumset A' A')) ≤ (card (differenceset A' B))^2/card B
    using bg0 nonzero-mult-div-cancel-left[of card B card(sumset A' A')]
      divide-right-mono by (smt (verit) of-nat-0 of-nat-mono real-of-nat-div4)
  moreover have (card (differenceset A' B))^2  ≤ ((2^13) ∗ (1/c^15) ∗ (card
A))^2
    using hcardle  by simp
  ultimately have (card (sumset A' A')) ≤ ((2^13) ∗ (1/c^15) ∗ (card A))^2/(card
B)
    using pos-le-divide-eq[OF bg0] by simp
  moreover have (c^4) ∗ (card A)/16 >0
    using assms card-0-eq by fastforce
  moreover have ((2^13) ∗ (1/c^15) ∗ (card A))^2/(card B) =
    ((2^13)∗ (1/c^15) ∗ (card A))^2 ∗ (1/(card B)) by simp
  moreover have ((2^13) ∗ (1/c^15)∗(card A))^2 ∗ (1/(card B)) ≤
     ((2^13) ∗ (1/c^15) ∗ (card A))^2/ ((c^4) ∗ (card A)/16) using bge frac-le
less-eq-real-def
    zero-le-power2 calculation(2, 3) by metis
  ultimately have (card (sumset A' A')) ≤ ((2^13) ∗ (1/c^15)∗(card A))^2/
((c^4) ∗ (card A)/16)
    by linarith
  then have (card (sumset A' A')) ≤ (2^30) ∗ (card A)/(c^34)
    using card-0-eq assms by (simp add: power2-eq-square)
  then show ?thesis using a2ss a2ne a2ge that by blast
qed

end
end
```

8 Supplementary results related to intermediate lemmas used in the proof of the Balog–Szemerédi–Gowers Theorem

theory *Balog-Szemeredi-Gowers-Supplementary*
  imports
    Balog-Szemeredi-Gowers-Main-Proof
begin

context *additive-abelian-group*

begin

Even though it is not applied anywhere in this development, for the sake of completeness we give the following analogous version of Lemma 2.17 (*popular-differences-card*) but for popular sums instead of popular differences. The proof is identical to that of Lemma 2.17, with the obvious modifications.

```
lemma popular-sums-card:
  fixes A::'a set and c::real
  assumes finite A and additive-energy A = 2 ∗ c and A ⊆ G
  shows card (popular-sum-set c A) ≥ c ∗ card A

proof(cases card A ≠ 0)
  assume hA: card A ≠ 0
  have hc: c ≥ 0 using assms additive-energy-def of-nat-0-le-iff
    by (smt (verit, best) assms(3) divide-nonneg-nonneg of-nat-0-le-iff)
  have (2 ∗ c) ∗ (card A)^3 = (∑ d ∈ (sumset A A). (f-sum d A)^2)
    using assms f-sum-card-quadruple-set-additive-energy by auto
  also have ...= ((∑ d ∈ (popular-sum-set c A). (f-sum d A)^2))
   +((∑  d ∈ ((sumset A A) − (popular-sum-set c A)). (f-sum d A)^2))
    using popular-sum-set-def assms finite-sumset by (metis (no-types, lifting)
      add.commute mem-Collect-eq subsetI sum.subset-diff)
  also have ... ≤ ((card (popular-sum-set c A)) ∗ (card A)^2)
   + c ∗ card A ∗ ((∑ d ∈ (sumset A A − (popular-sum-set c A)) . (f-sum d A)))
  proof−
    have ∀  d ∈ ((sumset A A) − (popular-sum-set c A)) . (f-sum d A)^2 ≤
      (c ∗ (card A)) ∗ (f-sum d A)
    proof
     fix d assume hd1: d ∈ sumset A A − popular-sum-set c A
     have hnonneg: f-sum d A ≥ 0 by auto
     have ¬ popular-sum d c A using hd1 popular-sum-set-def by blast
     from this have f-sum d A ≤ c ∗ card A using popular-sum-def by auto
     thus real ((f-sum d A)²) ≤ c ∗ real (card A) ∗ real (f-sum d A)
     using power2-eq-square hnonneg mult-right-mono of-nat-0 of-nat-le-iff of-nat-mult
by metis
   qed
   moreover have ∀  d ∈ (sumset A A) . f-sum d A ≤ (card A)^2
     using f-sum-def finite-sumset assms
```

by (*metis f-sum-le-card le-antisym nat-le-linear power2-nat-le-imp-le*)
ultimately have (($\sum$ *d* ∈ ((*sumset A A*) − *popular-sum-set c A*) . (*f-sum d A*)^*2*)) ≤
((∑ *d* ∈ ((*sumset A A*) − *popular-sum-set c A*). (*c* ∗ *card A*) ∗ (*f-sum d A*)))
using *assms finite-sumset sum-distrib-left sum-mono* **by** *fastforce*
then have ((∑ *d* ∈ ((*sumset A A*) − *popular-sum-set c A*) . (*f-sum d A*)^*2*)) ≤
(*c* ∗ *card A*) ∗ ((∑ *d* ∈ ((*sumset A A*) − *popular-sum-set c A*) . (*f-sum d A*)))
by (*metis* (*no-types*) *of-nat-sum sum-distrib-left*)
moreover have (∑ *d* ∈ *popular-sum-set c A*. (*f-sum d A*)^*2*) ≤
(∑ *d* ∈ *popular-sum-set c A*. (*card A*)^*2*) **using** *f-sum-le-card assms sum-mono assms popular-sum-set-def*
by (*metis* (*no-types*, *lifting*) *power2-nat-le-eq-le*)
moreover then have (∑ *d* ∈ *popular-sum-set c A* . (*f-sum d A*)^*2*) ≤
(*card* (*popular-sum-set c A*)) ∗ (*card A*)^*2*
using *sum-distrib-right* **by** *simp*
ultimately show *?thesis* **by** *linarith*
qed
also have ... ≤ ((*card* (*popular-sum-set c A*)) ∗ (*card A*)^*2*) + (*c* ∗ *card A*) ∗ (*card A*)^*2*
proof−
have (∑ *d* ∈ (*sumset A A* − *popular-sum-set c A*) . (*f-sum d A*)) ≤
(∑ *d* ∈ *sumset A A* . (*f-sum d A*)) **using** *DiffD1 subsetI assms sum-mono2 finite-sumset zero-le* **by** *metis*
then have (*c* ∗ *card A*) ∗ ((∑ *d* ∈ (*sumset A A* − *popular-sum-set c A*). (*f-sum d A*)))
≤ (*c* ∗ *card A*) ∗ (*card A*)^*2*
using *f-sum-card hc le0 mult-left-mono of-nat-0 of-nat-mono zero-le-mult-iff assms* **by** *metis*
then show *?thesis* **by** *linarith*
qed
finally have (*2* ∗ *c*) ∗ (*card A*)^*3* ≤ ((*card* (*popular-sum-set c A*)) ∗ (*card A*)^*2*) +
(*c* ∗ *card A*) ∗ (*card A*)^*2* **by** *linarith*
then have (*card* (*popular-sum-set c A*)) ≥
((*2* ∗ *c*)∗ (*card A*)^*3* − (*c* ∗ *card A*) ∗ (*card A*)^*2*)/((*card A*)^*2*)
using *hA* **by** (*simp add*: *field-simps*)
moreover have ((*2* ∗ *c*)∗ (*card A*)^*3* − (*c* ∗ *card A*) ∗ (*card A*)^*2*)/((*card A*)^*2*) = *2* ∗ *c* ∗ *card A* − *c* ∗ *card A*
using *hA* **by** (*simp add*: *power2-eq-square power3-eq-cube*)
ultimately show *?thesis* **by** *linarith*
next
assume ¬ *card A* ≠ *0*
thus *?thesis* **by** *auto*
qed

The following is an analogous version of lemma *obtains-subsets-differenceset-card-bound* (2.18 in Gowers's notes [3]) but for a sumset instead of a difference set. It is not used anywhere in this development but we provide it for the sake of completeness. The proof is identical to that of lemma *obtains-subsets-differenceset-card-bound* with *f-diff* changed to *f-sum*, *popular-diff* changed to *popular-sum*, ⊕interchanged with ⊖, and instead of lemma *popular-differences-card* we apply its analogous version for popular sums, that is lemma *popular-sums-card*.

lemma *obtains-subsets-sumset-card-bound*: **fixes** *A*::*'a set* **and** *c*::*real*
assumes *finite A* **and** *c>0* **and** *A* ≠ {} **and** *A* ⊆ *G* **and** *additive-energy A* = *2* ∗ *c*
obtains *B* **and** *A'* **where** *B* ⊆ *A* **and** *B* ≠ {} **and** *card B* ≥ *c^4* ∗ *card A* / *16*
and *A'* ⊆ *A* **and** *A'* ≠ {} **and** *card A'* ≥ *c^2* ∗ *card A* / *4*
and *card* (*sumset A' B*) ≤ *2^13* ∗ *card A* / *c^15*

proof−
let *?X* = *A* × {*0*:: *nat*}
let *?Y* = *A* × {*1*:: *nat*}
let *?E* = *mk-edge* ‘ {(*x*, *y*)| *x y*. *x* ∈ *?X* ∧ *y* ∈ *?Y* ∧ (*popular-sum* (*fst y* ⊕ *fst x*) *c A*)}
interpret *H*: *fin-bipartite-graph ?X* ∪ *?Y ?E ?X ?Y*
proof (*unfold-locales*, *auto simp add*: *partition-on-def assms*(*3*) *assms*(*1*) *disjoint-def*)
show {} = *A* × {*0*} ⟹ *False* **using** *assms*(*3*) **by** *auto*
next
show {} = *A* × {*Suc 0*} ⟹ *False* **using** *assms*(*3*) **by** *auto*
next
show *A* × {*0*} = *A* × {*Suc 0*} ⟹ *False* **using** *assms*(*3*) **by** *fastforce*
next
fix *x y* **assume** *x* ∈ *A* **and** *y* ∈ *A* **and** *popular-sum* (*y* ⊕ *x*) *c A*
thus {(*x*, *0*), (*y*, *Suc 0*)} ∈ *all-bi-edges* (*A* × {*0*}) (*A* × {*Suc 0*})
using *all-bi-edges-def*[*of A* × {*0*} *A* × {*Suc 0*}]
by (*simp add*: *in-mk-edge-img*)
qed
have *edges1*: ∀ *a* ∈ *A*. ∀ *b* ∈ *A*. ({(*a*, *0*), (*b*, *1*)} ∈ *?E* ⟷ *popular-sum* (*b* ⊕ *a*) *c A*)
by (*auto simp add*: *in-mk-uedge-img-iff*)
have *hXA*: *card A* = *card ?X* **by** (*simp add*: *card-cartesian-product*)
have *hYA*: *card A* = *card ?Y* **by** (*simp add*: *card-cartesian-product*)
have *hA*: *card A* ≠ *0* **using** *assms card-0-eq* **by** *blast*
have *edge-density*: *H.edge-density ?X ?Y* ≥ *c^2*
proof−
define *f*:: *'a* ⇒ (*'a* × *nat*) *edge set* **where** *f* ≡ (λ *x*. {{(*a*, *0*), (*b*, *1*)} | *a b*. *a* ∈ *A* ∧ *b* ∈ *A* ∧ *b* ⊕ *a* = *x*})
have *f-disj*: *pairwise* (λ *s t*. *disjnt* (*f s*) (*f t*)) (*popular-sum-set c A*)
proof (*intro pairwiseI*)
fix *x y* **assume** *hx*: *x* ∈ *popular-sum-set c A* **and** *hy*: *y* ∈ *popular-sum-set c A*
and *hxy*: *x* ≠ *y*
show *disjnt* (*f x*) (*f y*)
proof−
have ∀ *a*. ¬ (*a* ∈ *f x* ∧ *a* ∈ *f y*)
proof (*intro allI notI*)

```
          fix a assume a ∈ f x ∧ a ∈ f y
          then obtain z w where hazw: a = {(z, 0), (w, 1)} and hx: {(z,0), (w,
1)} ∈ f x
            and hy: {(z, 0), (w, 1)} ∈ f y using f-def by blast
          have w ⊕ z = x using f-def hx by (simp add: doubleton-eq-iff)
         moreover have w ⊕ z = y using f-def hy by (simp add: doubleton-eq-iff)
          ultimately show False using hxy by auto
        qed
        thus ?thesis using disjnt-iff by auto
      qed
    qed
    have f-sub-edges: ∀ d ∈ popular-sum-set c A. (f d) ⊆ ?E
      using popular-sum-set-def f-def edges1 by auto
     have f-union-sub: (⋃ d ∈ popular-sum-set c A. (f d)) ⊆ ?E using popu-
lar-sum-set-def
      f-def edges1 by auto
    have f-disj2: disjoint (f ‘ (popular-sum-set c A)) using f-disj
      pairwise-image[of disjnt f popular-sum-set c A] by (simp add: pairwise-def)
    have f-finite: ⋀B. B ∈ f ‘ popular-sum-set c A ⟹ finite B
      using finite-subset f-sub-edges H.fin-edges by auto
    have card-eq-f-diff: ∀ d ∈ popular-sum-set c A. card (f d) = f-sum d A
    proof
      fix d assume d ∈ popular-sum-set c A
     define g:: ('a × 'a) ⇒ ('a × nat) edge where g = (λ (a, b). {(b, 0), (a, 1)})
      have g-inj: inj-on g {(a, b) | a b. a ∈ A ∧ b ∈ A ∧ a ⊕ b = d}
      proof (intro inj-onI)
        fix x y assume x ∈ {(a, b) |a b. a ∈ A ∧ b ∈ A ∧ a ⊕ b = d} and
          y ∈ {(a, b) |a b. a ∈ A ∧ b ∈ A ∧ a ⊕ b = d} and hg: g x = g y
        then obtain a1 a2 b1 b2  where hx: x = (a1, a2) and hy: y = (b1, b2)
by blast
        thus x = y using g-def hg hx hy by (simp add: doubleton-eq-iff)
      qed
       have g-image: g ‘ {(a, b) |a b. a ∈ A ∧ b ∈ A ∧ a ⊕ b = d} = f d using
f-def g-def by auto
       show card (f d) = f-sum d A using card-image g-inj g-image f-sum-def by
fastforce
    qed
   have c ^2 ∗ (card A) ^2 = c ∗ (card A) ∗ (c ∗ (card A)) using power2-eq-square
      by (metis of-nat-power power-mult-distrib)
    also have ... ≤ (card (popular-sum-set c A)) ∗ (c ∗ (card A))
      using assms  popular-sums-card hA by force
     also have ... ≤ (∑ d ∈ popular-sum-set c A. f-sum d A) using sum-mono
popular-sum-set-def
     popular-sum-def by (smt (verit, ccfv-SIG) mem-Collect-eq of-nat-sum of-real-of-nat-eq

      sum-constant)
    also have ... = (∑ d ∈ popular-sum-set c A. card (f d))
      using card-eq-f-diff sum.cong by auto
    also have ... = sum card (f ‘ (popular-sum-set c A))
```

using *f-disj sum-card-image*[*of popular-sum-set c A f*] *popular-sum-set-def finite-sumset assms*(*1*) *finite-subset* **by** *auto*
also have ... = *card* ($\bigcup$ *d* ∈ *popular-sum-set c A*. (*f d*))
using *card-Union-disjoint*[*of f* ‘ (*popular-sum-set c A*)] *f-disj2 f-finite* **by** *auto*
also have ... ≤ *card ?E* **using** *card-mono f-union-sub H.fin-edges* **by** *auto*
finally have *c* ^ *2* ∗ (*card A*) ^ *2* ≤ *card ?E* **by** *linarith*
then have *c* ^ *2* ∗ (*card A*) ^ *2* ≤ *card* (*H.all-edges-between ?X ?Y*)
using *H.card-edges-between-set* **by** *auto*
moreover have *H.edge-density ?X ?Y* = *card* (*H.all-edges-between ?X ?Y*) / (*card A*) ^ *2*
using *H.edge-density-def power2-eq-square hXA hYA*
by (*smt* (*verit*, *best*))
ultimately have (*c* ^ *2* ∗ (*card A*) ^ *2*) / (*card A*) ^ *2* ≤ *H.edge-density ?X ?Y* **using** *hA*
divide-le-cancel **by** (*smt* (*verit*, *del-insts*) *H.edge-density-ge0* ‹c^2 ∗ *real* ((*card A*)^2) =
c ∗ *real* (*card A*) ∗ (*c* ∗ *real* (*card A*))› *divide-divide-eq-right zero-le-divide-iff*)
thus *?thesis* **using** *hA assms*(*2*) **by** *auto*
qed
obtain *X′* **and** *Y′* **where** *X′sub*: *X′* ⊆ *?X* **and** *Y′sub*: *Y′* ⊆ *?Y* **and**
hX′: *card X′* ≥ (*H.edge-density ?X ?Y*)^*2* ∗ (*card ?X*)/*16* **and**
hY′: *card Y′* ≥ (*H.edge-density ?X ?Y*) ∗ (*card ?Y*)/*4* **and**
hwalks: ∀ *x* ∈ *X′*. ∀ *y* ∈ *Y′*. *card* ({*p*. *H.connecting-walk x y p* ∧ *H.walk-length p* = *3*}) ≥
(*H.edge-density ?X ?Y*)^*6* ∗ *card ?X* ∗ *card ?Y* / *2*^*13*
using *H.walks-of-length-3-subsets-bipartite* ‹*c*>*0*› **by** *auto*
have ((*c*^*2*)^*2*) ∗ (*card A*) ≤ (*H.edge-density ?X ?Y*)^*2* ∗ (*card A*)
using *edge-density assms*(*2*) *hA power-mono zero-le-power2 mult-le-cancel-right*
by (*smt* (*verit*) *of-nat-less-of-nat-power-cancel-iff of-nat-zero-less-power-iff power2-less-eq-zero-iff power-0-left*)
then have *cardX′*: *card X′* ≥ (*c*^*4*) ∗ (*card A*)/*16* **using** *hX′ divide-le-cancel hXA* **by** *fastforce*
have *c*^*2* ∗ (*card A*) / *4* ≤ (*H.edge-density ?X ?Y*) ∗ *card ?Y* / *4* **using** *hYA hA edge-density*
mult-le-cancel-right **by** *simp*
then have *cardY′*: *card Y′* ≥ *c*^*2* ∗ (*card A*)/*4* **using** *hY′* **by** *linarith*
have (*H.edge-density ?X ?Y*)^*6* ∗ (*card ?X* ∗ *card ?Y*)/ *2*^*13* ≥ (*c*^*2*)^*6* ∗ ((*card A*)^*2*) / *2*^*13* **using**
hXA hYA power2-eq-square edge-density divide-le-cancel mult-le-cancel-right hA
by (*smt* (*verit*, *ccfv-SIG*) *of-nat-power power2-less-0 power-less-imp-less-base zero-le-power*)
then have *card-walks*: ∀ *x* ∈ *X′*. ∀ *y* ∈ *Y′*.
card ({*p*. *H.connecting-walk x y p* ∧ *H.walk-length p* = *3*}) ≥ (*c*^*12*) ∗ ((*card A*)^*2*) / *2*^*13*
using *hwalks* **by** *fastforce*

let *?B* = (λ (*a*, *b*). *a*) ‘ *X′*
let *?C* = (λ (*a*, *b*). *a*) ‘ *Y′*
have *hBA*: *?B* ⊆ *A* **and** *hCA*: *?C* ⊆ *A* **using** *Y′sub X′sub* **by** *auto*

have *inj-on-X'*: *inj-on* (λ (*a*, *b*). *a*) *X'* **using** *X'sub* **by** (*intro inj-onI*) (*auto*)
have *inj-on-Y'*: *inj-on* (λ (*a*, *b*). *a*) *Y'* **using** *Y'sub* **by** (*intro inj-onI*) (*auto*)
have *hBX'*: *card ?B* = *card X'* **and** *hCY'*: *card ?C* = *card Y'*
using *card-image inj-on-X' inj-on-Y'* **by** *auto*
then have *cardB*: *card ?B* ≥ (*c*^*4*) ∗ (*card A*)/*16* **and** *cardC*: *card ?C* ≥ *c*^*2* ∗ (*card A*)/*4*
using *cardX' cardY'* **by** *auto*
have *card-ineq1*: ⋀ *x y*. *x* ∈ *?B* ⟹ *y* ∈ *?C* ⟹ *card* ({(*z*, *w*) | *z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧
popular-sum (*z* ⊕ *x*) *c A* ∧ *popular-sum* (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*}) ≥
(*c*^*12*) ∗ ((*card A*)^*2*) / *2*^*13*
proof−
fix *x y* **assume** *hx*: *x* ∈ *?B* **and** *hy*: *y* ∈ *?C*
have *hxA*: *x* ∈ *A* **and** *hyA*: *y* ∈ *A* **using** *hx hy hBA hCA* **by** *auto*
define *f*:: *'a* × *'a* ⇒ (*'a* × *nat*) *list*
where *f* ≡ (λ (*z*, *w*). [(*x*, *0*), (*z*, *1*), (*w*, *0*), (*y*, *1*)])
have *f-inj-on*: *inj-on f* {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*} **using** *f-def* **by** (*intro inj-onI*) (*auto*)
have *f-image*: *f* ' {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*} =
{*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length p* = *3*}
proof
show *f* ' {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*} ⊆
{*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length p* = *3*}
proof
fix *p* **assume** *hp*: *p* ∈ *f* ' {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*}
then obtain *z w* **where** *hz*: *z* ∈ *A* **and** *hw*: *w* ∈ *A* **and** *hzx*: *popular-sum* (*z* ⊕ *x*) *c A* **and**
hzw: *popular-sum* (*z* ⊕ *w*) *c A* **and** *hyw*: *popular-sum* (*y* ⊕ *w*) *c A* **and**
hp: *p* = [(*x*, *0*), (*z*, *1*), (*w*, *0*), (*y*, *1*)] **using** *f-def hp* **by** *fast*
then have *hcon*: *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p*
unfolding *H*.*connecting-walk-def H*.*is-walk-def*
using *hxA hyA H*.*vert-adj-def H*.*vert-adj-sym edges1* **by** *simp*
thus *p* ∈ {*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length p* = *3*}
using *hp H*.*walk-length-conv* **by** *auto*
qed
next
show {*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length p* = *3*} ⊆ *f* ' {(*z*, *w*) |*z w*.
z ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧ *popular-sum* (*z* ⊕ *w*) *c A* ∧
popular-sum (*y* ⊕ *w*) *c A*}
proof(*intro subsetI*)
fix *p* **assume** *hp*: *p* ∈ {*p*. *H*.*connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H*.*walk-length*

p = 3}
then have len: length p = 4 using H.walk-length-conv by auto
have hpsub: set p ⊆ A × {0} ∪ A × {1} using hp H.connecting-walk-def H.is-walk-def
by auto
then have fst-sub: fst ‘ set p ⊆ A by auto
have h1A: fst (p!1) ∈ A and h2A: fst (p!2) ∈ A using fst-sub len by auto
have hpnum: p = [p!0, p!1, p!2, p!3]
proof (auto simp add: list-eq-iff-nth-eq len)
fix k assume k < (4:: nat)
then have k = 0 ∨ k = 1 ∨ k = 2 ∨ k = 3 by auto
thus p ! k = [p ! 0, p ! Suc 0, p ! 2, p ! 3] ! k by fastforce
qed
then have set (H.walk-edges p) = {{p!0, p!1} , {p!1, p!2}, {p!2, p!3}} using
comp-sgraph.walk-edges.simps(2) comp-sgraph.walk-edges.simps(3)
by (metis empty-set list.simps(15))
then have h1: {p!0, p!1} ∈ ?E and h2: {p!2, p!1} ∈ ?E and h3: {p!2, p!3} ∈ ?E
using hp H.connecting-walk-def H.is-walk-def len by auto
have hxp: p!0 = (x, 0) using hp len hd-conv-nth H.connecting-walk-def H.is-walk-def
by fastforce
have hyp: p!3 = (y, 1) using hp len last-conv-nth H.connecting-walk-def H.is-walk-def
by fastforce
have h1p: p!1 = (fst (p!1), 1)
proof−
have p!1 ∈ A × {0} ∪ A × {1} using hpnum hpsub
by (metis (no-types, lifting) insertCI list.simps(15) subsetD)
then have hsplit: snd (p!1) = 0 ∨ snd (p!1) = 1 by auto
then have snd (p!1) = 1
proof(cases snd (p!1) = 0)
case True
then have 1: {(x, 0), (fst (p!1), 0)} ∈ ?E using h1 hxp doubleton-eq-iff
by (smt (verit, del-insts) surjective-pairing)
have hY: (fst (p!1), 0) ∉ ?Y and hX: (x, 0) ∈ ?X using hxA by auto
then have 2: {(x, 0), (fst (p!1), 0)} ∉ ?E using H.X-vert-adj-Y H.vert-adj-def by meson
then show ?thesis using 1 2 by blast
next
case False
then show ?thesis using hsplit by auto
qed
thus (p ! 1) = (fst (p ! 1), 1)
by (metis (full-types) split-pairs)
qed
have h2p: p!2 = (fst (p!2), 0)

proof−
have *p*!*2* ∈ *A* × {*0*} ∪ *A* × {*1*} **using** *hpnum hpsub*
by (*metis* (*no-types*, *lifting*) *insertCI list.simps*(*15*) *subsetD*)
then have *hsplit*: *snd* (*p*!*2*) = *0* ∨ *snd* (*p*!*2*) = *1* **by** *auto*
then have *snd* (*p*!*2*) = *0*
proof(*cases snd* (*p*!*2*) = *1*)
case *True*
then have *1*: {(*fst* (*p*!*2*), *1*), (*y*, *1*)} ∈ *?E* **using** *h3 hyp doubleton-eq-iff*
by (*smt* (*verit*, *del-insts*) *surjective-pairing*)
have *hY*: (*y*, *1*) ∉ *?X* **and** *hX*: (*fst* (*p*!*2*), *1*) ∈ *?Y* **using** *hyA h2A*
by *auto*
then have *2*: {(*fst* (*p*!*2*), *1*), (*y*, *1*)} ∉ *?E* **using** *H.Y-vert-adj-X*
H.vert-adj-def
by *meson*
then show *?thesis* **using** *1 2* **by** *blast*
next
case *False*
then show *?thesis* **using** *hsplit* **by** *auto*
qed
thus (*p* ! *2*) = (*fst* (*p* ! *2*), *0*)
by (*metis* (*full-types*) *split-pairs*)
qed
have *hpop1*: *popular-sum* ((*fst* (*p*!*1*)) ⊕ *x*) *c A* **using** *edges1 h1 hxp h1p*
hxA h1A
by (*smt* (*z3*))
have *hpop2*: *popular-sum*((*fst* (*p*!*1*)) ⊕ (*fst* (*p*!*2*))) *c A* **using** *edges1 h2*
h1p h2p h1A h2A
by (*smt* (*z3*))
have *hpop3*: *popular-sum* (*y* ⊕ (*fst* (*p*!*2*))) *c A* **using** *edges1 h3 h2p hyp*
hyA h2A
by (*smt* (*z3*))
thus *p* ∈ *f* ‘ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*} **using** *f-def hpnum*
hxp h1p h2p hyp
h1A h2A hpop1 hpop2 hpop3 **by** *force*
qed
qed
have *hx1*: (*x*, *0*) ∈ *X′* **and** *hy2*: (*y*, *1*) ∈ *Y′* **using** *hx X′sub hy Y′sub* **by**
auto
have *card* {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*} =
card {*p*. *H.connecting-walk* (*x*, *0*) (*y*, *1*) *p* ∧ *H.walk-length p* = *3*}
using *card-image f-inj-on f-image* **by** *fastforce*
thus *card* {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*} ≥ *c* ^ *12* ∗ ((*card*
A)^*2*) / *2* ^ *13*
using *hx1 hy2 card-walks* **by** *auto*
qed
have ⋀ *x x2 y y2 z w* . (*x*, *x2*) ∈ *X′*⟹ (*y*, *y2*) ∈ *Y′*⟹ *z* ∈ *A* ⟹ *w* ∈ *A*

$\Longrightarrow$ *popular-sum* $(z \oplus x)$ c $A \Longrightarrow$ *popular-sum* $(z \oplus w)$ c $A \Longrightarrow$ *popular-sum* $(y \oplus w)$ c $A \Longrightarrow$
c ^ $3 *$ *real* (*card* A) ^ $3 \leq$
(*card* $\{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge$
$p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\})$
proof −
fix x $x2$ y $y2$ z w **assume** $(x, x2) \in X'$ **and** $(y, y2) \in Y'$ **and** $z \in A$ **and** $w \in A$ **and**
1: *popular-sum* $(z \oplus x)$ c A **and** *2*: *popular-sum* $(z \oplus w)$ c A **and**
3: *popular-sum* $(y \oplus w)$ c A
define f:: $'a \times 'a \times 'a \times 'a \times 'a \times 'a \Rightarrow ('a \times 'a) \times ('a \times 'a) \times ('a \times 'a)$ **where**
$f \equiv (\lambda\ (p, q, r, s, t, u).\ ((p, q), (r, s), (t, u)))$

have *f-inj*: *inj-on* f $\{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge$
$t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\}$ **using** *f-def*
by (*intro inj-onI*) (*auto*)
have *f-image*: f ‘ $\{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge$
$t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\} =$
$\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus x\} \times$
$\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus w\} \times$
$\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = y \oplus w\}$ **using** *f-def* **by** *force*

have *4*: *card* $\{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge$
$t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\} =$
card $(\{(p, q).\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus x\} \times$
$\{(p, q).\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus w\} \times \{(p, q).\ p \in A \wedge q \in A \wedge p \oplus q = y \oplus w\})$
using *card-image f-inj f-image* **by** *fastforce*
moreover have *5*: *card* $(\{(p, q).\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus x\} \times$
$\{(p, q).\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus w\} \times \{(p, q).\ p \in A \wedge q \in A \wedge p \oplus q = y \oplus w\}) =$
card $\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus x\} *$
card $\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus w\} *$
card $\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = y \oplus w\}$
using *card-cartesian-product3* **by** *auto*
have $c *$ *card* $A \leq$ *card* $\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus x\}$
using *1 popular-sum-def f-sum-def* **by** *auto*
then have $(c *$ *card* $A) * (c *$ *card* $A) \leq$ *card* $\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus x\} *$
card $\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus w\}$
using *2 popular-sum-def f-sum-def mult-mono assms*(*2*) *mult-nonneg-nonneg of-nat-0-le-iff of-nat-mult* **by** *fastforce*
then have *6*: $(c *$ *card* $A) * (c *$ *card* $A) * (c *$ *card* $A) \leq$ *card* $\{(p, q) \mid p\ q.\ p \in A \wedge q \in A \wedge p \oplus q = z \oplus x\} *$

card {(*p*, *q*) | *p q*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *p* ⊕ *q* = *z* ⊕ *w*} ∗
card {(*p*, *q*) | *p q*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *p* ⊕ *q* = *y* ⊕ *w*}
using *3 popular-sum-def f-sum-def mult-mono assms*(*2*) *mult-nonneg-nonneg of-nat-0-le-iff*
of-nat-mult **by** *fastforce*
have *7*: *c* ^ *3* ∗ *card A* ^ *3* = (*c* ∗ *card A*) ∗ ((*c* ∗ *card A*) ∗ (*c* ∗ *card A*))
by (*simp add*: *power3-eq-cube algebra-simps*)
show *c* ^ *3* ∗ *real* (*card A*) ^ *3* ≤
(*card* {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧
p ⊕ *q* = *z* ⊕ *x* ∧ *r* ⊕ *s* = *z* ⊕ *w* ∧ *t* ⊕ *u* = *y* ⊕ *w*}) **using** *4 5 6 7* **by** *auto*
qed
then have *card-ineq2*: ⋀ *x y z w*. *x* ∈ *?B* ⟹ *y* ∈ *?C* ⟹ (*z*, *w*) ∈ {(*z*, *w*) | *z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧
popular-sum (*z* ⊕ *x*) *c A* ∧ *popular-sum* (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*} ⟹
card {(*p*, *q*, *r*, *s*, *t*, *u*)| *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧
p ⊕ *q* = *z* ⊕ *x* ∧ *r* ⊕ *s* = *z* ⊕ *w* ∧ *t* ⊕ *u* = *y* ⊕ *w*} ≥ *c*^*3* ∗ *card A*^*3*
by *auto*
have *card-ineq3*: ⋀ *x y*. *x* ∈ *?B* ⟹ *y* ∈ *?C* ⟹ *card* (⋃ (*z*, *w*) ∈ {(*z*, *w*) | *z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧
popular-sum (*z* ⊕ *x*) *c A* ∧ *popular-sum* (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*}.
{(*p*, *q*, *r*, *s*, *t*, *u*)| *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧
t ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊕ *q* = *z* ⊕ *x* ∧ *r* ⊕ *s* = *z* ⊕ *w* ∧ *t* ⊕ *u* = *y* ⊕ *w*}) ≥
c ^ *15* ∗ ((*card A*)^*5*) / *2* ^ *13*
proof−
fix *x y* **assume** *hx*: *x* ∈ *?B* **and** *hy*: *y* ∈ *?C*
have *hxG*: *x* ∈ *G* **and** *hyG*: *y* ∈ *G* **using** *hx hy hBA hCA assms*(*4*) **by** *auto*
let *?f* = (λ (*z*, *w*). {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧
r ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊕ *q* = *z* ⊕ *x* ∧ *r* ⊕ *s* = *z* ⊕ *w* ∧ *t* ⊕ *u* = *y* ⊕ *w*})
have *hpair-disj*: *pairwise* (λ *a b*. *disjnt* (*?f a*) (*?f b*))
{(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧ *popular-sum* (*z* ⊕ *w*) *c A* ∧
popular-sum (*y* ⊕ *w*) *c A*}
proof (*intro pairwiseI*)
fix *a b* **assume** *a* ∈ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*} *b* ∈ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧
popular-sum (*z* ⊕ *x*) *c A* ∧ *popular-sum* (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*} **and**
a ≠ *b*
then obtain *a1 a2 b1 b2* **where** *ha*: *a* = (*a1*, *a2*) **and** *hb*: *b* = (*b1*, *b2*) **and** *ha1*: *a1* ∈ *G* **and**
ha2: *a2* ∈ *G* **and** *hb1*: *b1* ∈ *G* **and** *hb2*: *b2* ∈ *G* **and** *hne*: (*a1*, *a2*) ≠ (*b1*, *b2*)

using *assms*(*4*) **by** *blast*
have $(\forall x. \neg (x \in (?f\ a) \wedge x \in (?f\ b)))$
proof(*intro allI notI*)
fix *d* **assume** $d \in (?f\ a) \wedge d \in (?f\ b)$
then obtain *p q r s t u* **where** $d = (p, q, r, s, t, u)$ **and** *hpq1*: $p \oplus q = a1 \oplus x$ **and**
htu1: $t \oplus u = y \oplus a2$ **and** *hpq2*: $p \oplus q = b1 \oplus x$ **and** *htu2*: $t \oplus u = y \oplus b2$
using *ha hb* **by** *auto*
then have $y \oplus a2 = y \oplus b2$ **using** *htu1 htu2* **by** *auto*
then have *2*: $a2 = b2$ **using** *ha2 hb2 hyG* **by** (*metis invertible invertible-left-cancel*)
have *1*: $a1 = b1$ **using** *hpq1 hpq2 ha1 hb1 hxG* **by** *simp*
show *False* **using** *1 2 hne* **by** *auto*
qed
thus *disjnt* (*?f a*) (*?f b*) **using** *disjnt-iff*[*of* (*?f a*) (*?f b*)] **by** *auto*
qed
have *hfinite-walks*: $\bigwedge B.\ B \in ((\lambda (z, w). \{(p, q, r, s, t, u) |p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge$
$r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\})$ ‘
$\{(z, w) |z\ w.\ z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)\ c\ A \wedge$ *popular-sum* $(z \oplus w)\ c\ A \wedge$
popular-sum $(y \oplus w)\ c\ A\}) \Longrightarrow$ *finite B*
proof−
fix *B* **assume** $B \in ((\lambda (z, w). \{(p, q, r, s, t, u) |p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge$
$r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\})$ ‘
$\{(z, w) |z\ w.\ z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)\ c\ A \wedge$ *popular-sum* $(z \oplus w)\ c\ A \wedge$
popular-sum $(y \oplus w)\ c\ A\})$
then have $B \subseteq A \times A \times A \times A \times A \times A$ **by** *auto*
thus *finite B* **using** *assms*(*1*)
by (*auto simp add*: *finite-subset*)
qed
have *hdisj*: *disjoint* $((\lambda (z, w). \{(p, q, r, s, t, u) |p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge$
$r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\})$ ‘
$\{(z, w) |z\ w.\ z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)\ c\ A \wedge$ *popular-sum* $(z \oplus w)\ c\ A \wedge$
popular-sum $(y \oplus w)\ c\ A\})$ **using** *hpair-disj pairwise-image*[*of disjnt* $(\lambda (z, w). \{(p, q, r, s, t, u) |p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge$
$r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\})$
$\{(z, w) |z\ w.\ z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)\ c\ A \wedge$ *popular-sum* $(z \oplus w)\ c\ A \wedge$
popular-sum $(y \oplus w)\ c\ A\}]$ **by** (*simp add*: *pairwise-def*)
have $\{(z, w) |z\ w.\ z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)\ c\ A \wedge$

popular-sum $(z \oplus w)$ *c A* $\wedge$ *popular-sum* $(y \oplus w)$ *c A*$\} \subseteq A \times A$ **by** *auto*
then have *hwalks-finite*: *finite* $\{(z, w)$ |*z w*. $z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)$ *c A* $\wedge$
popular-sum $(z \oplus w)$ *c A* $\wedge$ *popular-sum* $(y \oplus w)$ *c A*$\}$ **using** *finite-subset assms*(*1*)
by *fastforce*
have *f-ineq*: $\forall$ $a \in \{(z, w)$ |*z w*. $z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)$ *c A* $\wedge$
popular-sum $(z \oplus w)$ *c A* $\wedge$ *popular-sum* $(y \oplus w)$ *c A*$\}$. *c* ^ *3* $*$ (*card A*) ^ *3* $\leq$
card (*?f a*) **using** *card-ineq2 hx hy* **by** *auto*
have *c* ^ *15* $*$ ((*card A*)^*5*) / *2* ^ *13* = (*c* ^ *12* $*$ (*card A*) ^ *2* / *2* ^ *13*) $*$ (*c* ^ *3* $*$ *card A* ^ *3*)
by (*simp add*: *algebra-simps*)
also have ... $\leq$ *card* $\{(z, w)$ |*z w*. $z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)$ *c A* $\wedge$
popular-sum $(z \oplus w)$ *c A* $\wedge$ *popular-sum* $(y \oplus w)$ *c A*$\}$ $*$ (*c* ^ *3* $*$ (*card A*) ^ *3*)
using *card-ineq1*[*of x y*] *hx hy mult-le-cancel-right hA* **by** (*smt* (*verit*, *best*) *assms*(*2*)
mult-pos-pos of-nat-0-less-iff of-nat-le-0-iff zero-less-power)
also have ... = ($\sum$ $a \in \{(z, w)$ |*z w*. $z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)$ *c A* $\wedge$
popular-sum $(z \oplus w)$ *c A* $\wedge$ *popular-sum* $(y \oplus w)$ *c A*$\}$. (*c* ^ *3* $*$ (*card A*) ^ *3*)) **by** *auto*
also have ... $\leq$ ($\sum$ $a \in \{(z, w)$ |*z w*. $z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)$ *c A* $\wedge$
popular-sum $(z \oplus w)$ *c A* $\wedge$ *popular-sum* $(y \oplus w)$ *c A*$\}$. *card* (*?f a*))
using *sum-mono f-ineq* **by** (*smt* (*verit*, *del-insts*) *of-nat-sum*)
also have ... = *sum card* (*?f* ' $\{(z, w)$ |*z w*. $z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)$ *c A* $\wedge$
popular-sum $(z \oplus w)$ *c A* $\wedge$ *popular-sum* $(y \oplus w)$ *c A*$\}$)
using *sum-card-image*[*of* $\{(z, w)$ |*z w*. $z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)$ *c A* $\wedge$
popular-sum $(z \oplus w)$ *c A* $\wedge$ *popular-sum* $(y \oplus w)$ *c A*$\}$ *?f*] *hpair-disj hwalks-finite* **by** *auto*
also have ... = *card* ($\bigcup$ $(z, w) \in \{(z, w)$ |*z w*. $z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)$ *c A* $\wedge$
popular-sum $(z \oplus w)$ *c A* $\wedge$ *popular-sum* $(y \oplus w)$ *c A*$\}$. $\{(p, q, r, s, t, u)$ |*p q r s t u*.
$p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge$
$t \oplus u = y \oplus w\}$) **using** *card-Union-disjoint hdisj hfinite-walks* **by** (*metis* (*no-types*, *lifting*))
finally show *c* ^ *15* $*$ *real* (*card A* ^ *5*) / *2* ^ *13* $\leq$ *real* (*card* ($\bigcup$ $(z, w) \in \{(z, w)$ |*z w*.
$z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)$ *c A* $\wedge$ *popular-sum* $(z \oplus w)$ *c A* $\wedge$
popular-sum $(y \oplus w)$ *c A*$\}$. $\{(p, q, r, s, t, u)$ |*p q r s t u*. $p \in A \wedge q \in A \wedge r \in A \wedge$
$s \in A \wedge t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus$

$w\}$)) **by** *simp*
qed
have *pos*: $0 < c \,\hat{}\, 15 * real\ (card\ A \,\hat{}\, 5)\ /\ 2 \,\hat{}\, 13$ **using** *hA* ‹*c*>*0*› **by** *auto*
have (*5*:: *nat*) ≤ *6* **by** *auto*
then have (*card A* ^ *6* / *card A* ^ *5*) = (*card A*) ^(*6* − *5*)
using *hA power-diff* **by** (*metis of-nat-eq-0-iff of-nat-power*)
then have *cardApow*: (*card A* ^ *6* / *card A* ^ *5*) = *card A* **using** *power-one-right* **by** *simp*
have *hdsub*: ∀ *d* ∈ *sumset ?C ?B*. ∃ *y* ∈ *?C*. ∃ *x* ∈ *?B*.
(⋃ (*z*, *w*) ∈ {(*z*, *w*) | *z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*}.
{(*p*, *q*, *r*, *s*, *t*, *u*)| *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧
t ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊕ *q* = *z* ⊕ *x* ∧ *r* ⊕ *s* = *z* ⊕ *w* ∧ *t* ⊕ *u* = *y* ⊕ *w*})
⊆ {(*p*, *q*, *r*, *s*, *t*, *u*) | *p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧
s ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *d* = *p* ⊕ *q* ⊖ (*r* ⊕ *s*) ⊕ *t* ⊕ *u*}
proof
fix *d* **assume** *d* ∈ *sumset ?C ?B*
then obtain *y x* **where** *hy*: *y* ∈ *?C* **and** *hx*: *x* ∈ *?B* **and** *hxy*: *d* = *y* ⊕ *x*
using *sumset-def minusset-def hBA hCA assms*(*4*) *subset-trans*
by (*smt* (*verit*, *best*) *minusset.simps sumset.cases*)
have *hxG*: *x* ∈ *G* **and** *hyG*: *y* ∈ *G* **using** *hx hy hBA hCA assms*(*4*) **by** *auto*
have (⋃(*z*, *w*)∈{(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*}. {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*.
p ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊕ *q* = *z* ⊕ *x* ∧ *r* ⊕ *s* = *z* ⊕ *w* ∧ *t* ⊕ *u* = *y* ⊕ *w*})
⊆ {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧
d = *p* ⊕ *q* ⊖ (*r* ⊕ *s*) ⊕ *t* ⊕ *u*}
proof (*rule Union-least*)
fix *X* **assume** *X* ∈ (λ(*z*, *w*). {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*.
p ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊕ *q* = *z* ⊕ *x* ∧ *r* ⊕ *s* = *z* ⊕ *w* ∧
t ⊕ *u* = *y* ⊕ *w*}) ‘ {(*z*, *w*) |*z w*. *z* ∈ *A* ∧ *w* ∈ *A* ∧ *popular-sum* (*z* ⊕ *x*) *c A* ∧
popular-sum (*z* ⊕ *w*) *c A* ∧ *popular-sum* (*y* ⊕ *w*) *c A*}
then obtain *z w* **where** *hX*: *X* = {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧
s ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊕ *q* = *z* ⊕ *x* ∧ *r* ⊕ *s* = *z* ⊕ *w* ∧ *t* ⊕ *u* = *y* ⊕ *w*}
and *hz*: *z* ∈ *A* **and** *hw*: *w* ∈ *A* **by** *auto*
have *hzG*: *z* ∈ *G* **and** *hwG*: *w* ∈ *G* **using** *hz hw assms*(*4*) **by** *auto*
have {(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧
s ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧ *p* ⊕ *q* = *z* ⊕ *x* ∧ *r* ⊕ *s* = *z* ⊕ *w* ∧ *t* ⊕ *u* = *y* ⊕ *w*} ⊆
{(*p*, *q*, *r*, *s*, *t*, *u*) |*p q r s t u*. *p* ∈ *A* ∧ *q* ∈ *A* ∧ *r* ∈ *A* ∧ *s* ∈ *A* ∧ *t* ∈ *A* ∧ *u* ∈ *A* ∧
d = *p* ⊕ *q* ⊖ (*r* ⊕ *s*) ⊕ *t* ⊕ *u*}
proof

fix e **assume** $e \in \{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge$
$t \in A \wedge u \in A \wedge p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\}$
then obtain $p\ q\ r\ s\ t\ u$ **where** $p \oplus q = z \oplus x$ **and** $r \oplus s = z \oplus w$ **and** $t \oplus u = y \oplus w$
and *hp*: $p \in A$ **and** *hq*: $q \in A$ **and** *hr*: $r \in A$ **and** *hs*: $s \in A$ **and** *ht*: $t \in A$
and *hu*: $u \in A$ **and** *he*: $e = (p, q, r, s, t, u)$ **by** *blast*
then have $p \oplus q \ominus (r \oplus s) \oplus t \oplus u = z \oplus x \ominus (z \oplus w) \oplus y \oplus w$
by (*smt* (*verit*, *ccfv-threshold*) *assms*(*4*) *associative composition-closed hwG hxG hyG hzG*
inverse-closed subset-eq)
also have ... $= y \oplus x$ **using** *hxG hyG hzG hwG*
by (*smt* (*verit*) *associative commutative composition-closed inverse-closed invertible*
invertible-right-inverse2)
finally have $d = p \oplus q \ominus (r \oplus s) \oplus t \oplus u$ **using** *hxy* **by** *simp*
thus $e \in \{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge$
$u \in A \wedge d = p \oplus q \ominus (r \oplus s) \oplus t \oplus u\}$ **using** *he hp hq hr hs ht hu* **by** *auto*
qed
thus $X \subseteq \{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.$
$p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge d = p \oplus q \ominus (r \oplus s) \oplus t \oplus u\}$
using *hX* **by** *auto*
qed
thus $\exists y{\in}(\lambda(a, b).\ a)$ ‘ Y'. $\exists x{\in}(\lambda(a, b).\ a)$ ‘ X'. $(\bigcup(z, w){\in}\{(z, w) \mid z\ w.\ z \in A \wedge w \in A \wedge$
popular-sum $(z \oplus x)\ c\ A \wedge$ *popular-sum* $(z \oplus w)\ c\ A \wedge$ *popular-sum* $(y \oplus w)\ c\ A\}$.
$\{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge$
$p \oplus q = z \oplus x \wedge r \oplus s = z \oplus w \wedge t \oplus u = y \oplus w\}) \subseteq \{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.$
$p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge d = p \oplus q \ominus (r \oplus s) \oplus t \oplus u\}$
using *hx hy* **by** *meson*
qed
moreover have $\forall\ d \in$ *sumset ?C ?B*. *card* $\{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge$
$r \in A \wedge s \in A \wedge t \in A \wedge u \in A \wedge d = p \oplus q \ominus (r \oplus s) \oplus t \oplus u\} \geq c\ \hat{}\ 15 * (card\ A)\ \hat{}\ 5\ /\ 2\ \hat{}13$
proof
fix d **assume** $d \in$ *sumset* $((\lambda(a, b).\ a)$ ‘ $Y')$ $((\lambda(a, b).\ a)$ ‘ $X')$
then obtain $x\ y$ **where** *hy*: $y \in$ *?C* **and** *hx*: $x \in$ *?B* **and** *hsub*:
$(\bigcup\ (z, w) \in \{(z, w) \mid z\ w.\ z \in A \wedge w \in A \wedge$ *popular-sum* $(z \oplus x)\ c\ A \wedge$
popular-sum $(z \oplus w)\ c\ A \wedge$ *popular-sum* $(y \oplus w)\ c\ A\}$.
$\{(p, q, r, s, t, u) \mid p\ q\ r\ s\ t\ u.\ p \in A \wedge q \in A \wedge r \in A \wedge s \in A \wedge$

t ∈ A ∧ u ∈ A ∧ p ⊕ q = z ⊕ x ∧ r ⊕ s = z ⊕ w ∧ t ⊕ u = y ⊕ w})
⊆ {(p, q, r, s, t, u) | p q r s t u. p ∈ A ∧ q ∈ A ∧ r ∈ A ∧
s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊕ q ⊖ (r ⊕ s) ⊕ t ⊕ u} **using** *hdsub* **by** *meson*
have {(p, q, r, s, t, u) | p q r s t u. p ∈ A ∧ q ∈ A ∧ r ∈ A ∧
s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊕ q ⊖ (r ⊕ s) ⊕ t ⊕ u} ⊆ A × A × A × A × A × A **by** *auto*
then have *fin*: *finite* {(p, q, r, s, t, u) | p q r s t u. p ∈ A ∧ q ∈ A ∧ r ∈ A ∧
s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊕ q ⊖ (r ⊕ s) ⊕ t ⊕ u}
using *finite-subset assms*(*1*) *finite-cartesian-product* **by** *fastforce*
have c ^ 15 ∗ (card A) ^ 5 / 2 ^13 ≤ card (⋃ (z, w) ∈ {(z, w) | z w. z ∈ A ∧ w ∈ A ∧ popular-sum (z ⊕ x) c A ∧
popular-sum (z ⊕ w) c A ∧ popular-sum (y ⊕ w) c A}.
{(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧ r ∈ A ∧ s ∈ A ∧
t ∈ A ∧ u ∈ A ∧ p ⊕ q = z ⊕ x ∧ r ⊕ s = z ⊕ w ∧ t ⊕ u = y ⊕ w})
using *card-ineq3 hx hy* **by** *auto*
also have ... ≤ card {(p, q, r, s, t, u) | p q r s t u. p ∈ A ∧ q ∈ A ∧ r ∈ A ∧
s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊕ q ⊖ (r ⊕ s) ⊕ t ⊕ u}
using *hsub card-mono fin* **by** *auto*
finally show c ^ 15 ∗ (card A) ^ 5 / 2 ^13 ≤ card {(p, q, r, s, t, u) | p q r s t u. p ∈ A ∧ q ∈ A ∧ r ∈ A ∧
s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊕ q ⊖ (r ⊕ s) ⊕ t ⊕ u} **by** *linarith*
qed
moreover have *pairwise* (λ s t. *disjnt* ((λ d. {(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧
r ∈ A ∧ s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊕ q ⊖ (r ⊕ s) ⊕ t ⊕ u}) s)
((λ d. {(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧
r ∈ A ∧ s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊕ q ⊖ (r ⊕ s) ⊕ t ⊕ u}) t)) (*sumset ?C ?B*)
unfolding *disjnt-def* **by** (*intro pairwiseI*) (*auto*)
moreover have ∀ d ∈ *sumset ?C ?B*. ((λ d. {(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧
r ∈ A ∧ s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ (d = p ⊕ q ⊕ r ⊕ s ⊕ t ⊕ u)}) d) ⊆ A × A × A × A × A × A
by *blast*
ultimately have card (*sumset ?C ?B*) ≤ ((card A) ^ 6) / (c^15 ∗(card A)^5 /2^13)
using *assms*(*1*) *hA finite-cartesian-product card-cartesian-product-6*[*of A*]
pos card-le-image-div[*of* A × A × A × A × A × A (λ d. {(p, q, r, s, t, u)| p q r s t u. p ∈ A ∧ q ∈ A ∧
r ∈ A ∧ s ∈ A ∧ t ∈ A ∧ u ∈ A ∧ d = p ⊕ q ⊖ (r ⊕ s) ⊕ t ⊕ u}) *sumset ?C ?B*
(c^15 ∗ (card A)^5 /2^13)] **by** *auto*
also have ... = (card A ^ 6 / card A ^ 5) / (c^15 / 2^13)
using *hA assms*(*3*) *field-simps* **by** *simp*
also have ... = (card A) / (c ^ 15 / 2 ^ 13)
using *cardApow* **by** *metis*
finally have *final*: card (*sumset ?C ?B*) ≤ 2 ^ 13 ∗ (1 / c ^ 15) ∗ *real* (card A)
by *argo*

have $0 < c \,\hat{}\, 4 * real\ (card\ A)\ /\ 16$ **and** $0 < c \,\hat{}\, 2 * real\ (card\ A)\ /\ 4$ **using** *assms*(*2*) *hA* **by** *auto*
then have $?B \neq \{\}$ **and** $?C \neq \{\}$ **using** *cardB cardC* **by** *auto*
then show *?thesis* **using** *hCA hBA cardC cardB final that* **by** *auto*
qed

end
end

References

[1] C. Edmonds. Undirected graph theory. *Archive of Formal Proofs*, September 2022. https://isa-afp.org/entries/Undirected_Graph_Theory.html, Formal proof development.

[2] W. T. Gowers. A new proof of Szemerédi's theorem. *Geometric & Functional Analysis GAFA*, 11(3):465–588, 2001.

[3] W. T. Gowers. Introduction to additive combinatorics, 2022. Lecture notes for Part III of the Mathematical Tripos taught at the University of Cambridge, available at https://drive.google.com/file/d/1ut0mUqSyPMweoxoDTfhXverEONyFgcuO/view.

[4] Y. Zhao. Graph theory and additive combinatorics. Online at https://yufeizhao.com/gtacbook/, 2022. book draft.