# The Median Method 

Emin Karayel

March 11, 2024


#### Abstract

The median method is an amplification result for randomized approximation algorithms described in [1]. Given an algorithm whose result is in a desired interval with a probability larger than $\frac{1}{2}$, it is possible to improve the success probability, by running the algorithm multiple times independently and using the median. In contrast to using the mean, the amplification of the success probability grows exponentially with the number of independent runs.

This entry contains a formalization of the underlying theorem: Given a sequence of $n$ independent random variables, which are in a desired interval with a probability $\frac{1}{2}+\alpha$. Then their median will be in the desired interval with a probability of $1-\exp \left(-2 \alpha^{2} n\right)$. In particular, the success probability approaches 1 exponentially with the number of variables.

In addition to that, this entry also contains a proof that orderstatistics of Borel-measurable random variables are themselves measurable and that generalized intervals in linearly ordered Borel-spaces are measurable.


## Contents

1 Intervals are Borel measurable 1
2 Order statistics are Borel measurable 4
3 The Median Method 8
4 Some additional results about the median 19

## 1 Intervals are Borel measurable

theory Median<br>imports<br>HOL-Probability.Probability HOL-Library.Multiset<br>Universal-Hash-Families.Universal-Hash-Families-More-Independent-Families

## begin

This section contains a proof that intervals are Borel measurable, where an interval is defined as a convex subset of linearly ordered space, more precisely, a set is an interval, if for each triple of points $x<y<z$ : If $x$ and $z$ are in the set so is $y$. This includes ordinary intervals like $\{a . . b\},\{a<. .<b\}$ but also for example $\{x::$ rat. $x * x<$ (2::rat) $\}$ which cannot be expressed in the standard notation.

In the HOL-Analysis.Borel-Space there are proofs for the measurability of each specific type of interval, but those unfortunately do not help if we want to express the result about the median bound for arbitrary types of intervals.

```
definition interval :: ('a :: linorder) set }=>\mathrm{ bool where
    interval }I=(\forallxyz.x\inI\longrightarrowz\inI\longrightarrowx\leqy\longrightarrowy\leqz\longrightarrowy\inI
definition up-ray :: ('a :: linorder) set }=>\mathrm{ bool where
    up-ray I = (\forallxy.x 趶 \longrightarrowx\leqy\longrightarrowy\inI)
```

lemma up-ray-borel:
assumes up-ray (I :: (('a :: linorder-topology) set))
shows $I \in$ borel
proof (cases closed I)
case True
then show ?thesis using borel-closed by blast
next
case False
hence $b: \neg$ closed I by blast
have open I
proof (rule Topological-Spaces.openI)
fix $x$
assume $c: x \in I$
show $\exists T$. open $T \wedge x \in T \wedge T \subseteq I$
proof (cases $\exists y . y<x \wedge y \in I$ )
case True
then obtain $y$ where $a: y<x \wedge y \in I$ by blast
have open $\{y<.$.$\} by simp$
moreover have $x \in\{y<.$.$\} using a$ by $\operatorname{simp}$
moreover have $\{y<..\} \subseteq I$
using a assms(1) by (auto simp: up-ray-def)
ultimately show ?thesis by blast
next
case False
hence $I \subseteq\{x .$.$\} using linorder-not-less by auto$
moreover have $\{x ..\} \subseteq I$
using $c$ assms(1) unfolding up-ray-def by blast
ultimately have $I=\{x .$.
by (rule order-antisym)
moreover have closed $\{x .$.$\} by simp$
ultimately have False using $b$ by auto then show? thesis by simp
qed
qed
then show? ?thesis by simp
qed
definition down-ray $::$ ('a :: linorder) set $\Rightarrow$ bool where
down-ray $I=(\forall x y . y \in I \longrightarrow x \leq y \longrightarrow x \in I)$
lemma down-ray-borel:
assumes down-ray (I :: (('a :: linorder-topology) set))
shows $I \in$ borel
proof -
have up-ray ( $-I$ ) using assms
by (simp add: up-ray-def down-ray-def, blast)
hence $(-I) \in$ borel using up-ray-borel by blast
thus $I \in$ borel
by (metis borel-comp double-complement)
qed

Main result of this section:
lemma interval-borel:
assumes interval ( $I$ :: (('a :: linorder-topology) set))
shows $I \in$ borel
proof (cases $I=\{ \}$ )
case True
then show? ?hesis by simp
next
case False
then obtain $x$ where $a: x \in I$ by blast
have $\bigwedge y z . y \in I \cup\{x ..\} \Longrightarrow y \leq z \Longrightarrow z \in I \cup\{x .$.
by (metis assms a interval-def IntE UnE Un-Int-eq(1) Un-Int-eq(2) atLeast-iff
nle-le order.trans)
hence up-ray $(I \cup\{x .\}$.
using up-ray-def by blast
hence $b: I \cup\{x ..\} \in$ borel
using up-ray-borel by blast
have $\bigwedge y z . y \in I \cup\{. . x\} \Longrightarrow z \leq y \Longrightarrow z \in I \cup\{. . x\}$
by (metis assms a interval-def UnE UnI1 UnI2 atMost-iff dual-order.trans
linorder-le-cases)
hence down-ray $(I \cup\{. . x\})$
using down-ray-def by blast
hence $c: I \cup\{. . x\} \in$ borel
using down-ray-borel by blast
have $I=(I \cup\{x .\}.) \cap(I \cup\{. . x\})$
using a by fastforce

```
    then show ?thesis using b c
    by (metis sets.Int)
qed
```


## 2 Order statistics are Borel measurable

This section contains a proof that order statistics of Borel measurable random variables are themselves Borel measurable.
The proof relies on the existence of branch-free comparison-sort algorithms. Given a sequence length these algorithms perform compare-swap operations on predefined pairs of positions. In particular the result of a comparison does not affect future operations. An example for a branch-free comparison sort algorithm is shell-sort and also bubble-sort without the early exit.
The advantage of using such a comparison-sort algorithm is that it can be lifted to work on random variables, where the result of a comparison-swap operation on two random variables $X$ and $Y$ can be represented as the expressions $\lambda \omega$. min $(X \omega)(Y \omega)$ and $\lambda \omega$. max $(X \omega)(Y \omega)$.
Because taking the point-wise minimum (resp. maximum) of two random variables is still Borel measurable, and because the entire sorting operation can be represented using such compare-swap operations, we can show that all order statistics are Borel measuable.

```
fun sort-primitive where
    sort-primitive i \(j f k=(\) if \(k=i\) then \(\min (f i)(f j)\) else (if \(k=j\) then \(\max (f i)\)
( \(f\) j) else \(f k\) )
fun sort-map where
    sort-map \(f n=\) fold id \([\) sort-primitive \(j\) i. \(i<-[0 . .<n], j<-[0 . .<i]] f\)
lemma sort-map-ind:
    sort-map \(f(\) Suc \(n)=\) fold id \([\) sort-primitive \(j n . j<-[0 . .<n]](\) sort-map \(f n)\)
    by \(\operatorname{simp}\)
lemma sort-map-strict-mono:
    fixes \(f::\) nat \(\Rightarrow\) ' \(b::\) linorder
    shows \(j<n \Longrightarrow i<j \Longrightarrow\) sort-map \(f n i \leq\) sort-map \(f n j\)
proof (induction \(n\) arbitrary: \(i j\) )
    case 0
    then show ?case by simp
next
    case (Suc n)
    define \(g\) where \(g=(\lambda k\). fold id [sort-primitive \(j n . j<-[0 . .<k]\) ] (sort-map \(f\)
n))
    define \(k\) where \(k=n\)
    have \(a:(\forall i j . j<n \longrightarrow i<j \longrightarrow g k i \leq g k j) \wedge(\forall l . l<k \longrightarrow g k l \leq g k n)\)
    proof (induction \(k\) )
```

```
    case 0
    then show ?case using Suc by (simp add:g-def del:sort-map.simps)
    next
    case (Suc k)
    have g(Suc k)= sort-primitive kn (gk)
        by (simp add:g-def)
    then show ?case using Suc
        apply (cases gkk\leqgkn)
        apply (simp add:min-def max-def)
        using less-antisym apply blast
        apply (cases gkn\leqgkk)
        apply (simp add:min-def max-def)
        apply (metis less-antisym max.coboundedI2 max.orderE)
        by simp
qed
```

hence $\bigwedge i j . j<S u c n \Longrightarrow i<j \Longrightarrow g n i \leq g n j$
apply (simp add:k-def) using less-antisym by blast
moreover have sort-map $f$ (Suc $n)=g n$
by (simp add:sort-map-ind g-def del:sort-map.simps)
ultimately show ?case
using Suc by (simp del:sort-map.simps)
qed
lemma sort-map-mono:
fixes $f::$ nat $\Rightarrow$ ' $b::$ linorder
shows $j<n \Longrightarrow i \leq j \Longrightarrow$ sort-map $f n i \leq$ sort-map $f n j$
by (metis sort-map-strict-mono eq-iff le-imp-less-or-eq)
lemma sort-map-perm:
fixes $f::$ nat $\Rightarrow$ ' $b$ :: linorder
shows $\operatorname{image}-m s e t(\operatorname{sort}-\operatorname{map} f n)(\operatorname{mset}[0 . .<n])=\operatorname{image-mset} f(\operatorname{mset}[0 . .<n])$
proof -
define is-swap where is-swap $=\left(\lambda\left(t s::\left(\left(n a t \Rightarrow{ }^{\prime} b\right) \Rightarrow n a t \Rightarrow{ }^{\prime} b\right)\right) . \exists i<n . \exists j\right.$
$<n . t s=$ sort-primitive $i j$ )
define $t::\left(\left(n a t \Rightarrow{ }^{\prime} b\right) \Rightarrow\right.$ nat $\left.\Rightarrow{ }^{\prime} b\right)$ list
where $t=[$ sort-primitive $j$ i. $i<-[0 . .<n], j<-[0 . .<i]]$
have $a: \bigwedge x f$. is-swap $x \Longrightarrow$ image-mset $(x f)($ mset-set $\{0 . .<n\})=$ image-mset
$f($ mset-set $\{0 . .<n\})$
proof -
fix $x$
fix $f::$ nat $\Rightarrow$ ' $b$ :: linorder
assume is-swap $x$
then obtain $i j$ where $x$-def: $x=$ sort-primitive $i j$ and $i$-bound: $i<n$ and
j-bound: $; n$
using is-swap-def by blast
define inv where inv $=$ mset-set $\{k . k<n \wedge k \neq i \wedge k \neq j\}$
have $b:\{0 . .<n\}=\{k . k<n \wedge k \neq i \wedge k \neq j\} \cup\{i, j\}$
apply (rule order-antisym, rule subsetI, simp, blast, rule subsetI, simp) using $i$-bound $j$-bound by meson
have $c: \bigwedge k . k \in \#$ inv $\Longrightarrow(x f) k=f k$ by ( simp add: $x$-def inv-def)
have image-mset ( $x f$ ) inv $=$ image-mset $f$ inv apply (rule multiset-eqI)
using $c$ multiset.map-cong0 by force
moreover have image-mset $(x f)($ mset-set $\{i, j\})=$ image-mset $f$ (mset-set $\{i, j\})$
apply (cases $i=j$ )
by (simp add:x-def max-def min-def)+
moreover have mset-set $\{0 . .<n\}=$ inv + mset-set $\{i, j\}$
by (simp only:inv-def b, rule mset-set-Union, simp, simp, simp)
ultimately show image-mset $(x f)($ mset-set $\{0 . .<n\})=\operatorname{image}-m s e t f($ mset-set $\{0 . .<n\}$ )
by $\operatorname{simp}$
qed
have $(\forall x \in$ set $t$. is-swap $x) \Longrightarrow$ image-mset (fold id $t$ f) $($ mset $[0 . .<n])=$ image-mset $f$ (mset $[0 . .<n])$
by (induction $t$ arbitrary:f, simp, simp add:a)
moreover have $\bigwedge x . x \in$ set $t \Longrightarrow$ is-swap $x$
apply (simp add:t-def is-swap-def)
by (meson atLeastLessThan-iff imageE less-imp-le less-le-trans)
ultimately have image-mset (fold id $t f)($ mset $[0 . .<n])=$ image-mset $f($ mset $[0 . .<n])$ by blast
then show ?thesis by (simp add:t-def)
qed
lemma list-eq-iff:
assumes mset $x s=$ mset ys
assumes sorted xs
assumes sorted ys
shows $x s=y s$
using assms properties-for-sort by blast
lemma sort-map-eq-sort:
fixes $f::$ nat $\Rightarrow$ (' $b::$ linorder $)$
shows map $($ sort-map $f n)[0 . .<n]=\operatorname{sort}(\operatorname{map} f[0 . .<n])($ is $? A=? B)$
proof -
have mset $? A=$ mset $? B$
using sort-map-perm [where $f=f$ and $n=n]$
by (simp del:sort-map.simps)
moreover have sorted ? $B$
by $\operatorname{simp}$
moreover have sorted?A
apply (subst sorted-wrt-iff-nth-less)
apply (simp del:sort-map.simps)
by (metis sort-map-mono nat-less-le)

```
    ultimately show ?A = ?B
    using list-eq-iff by blast
qed
lemma order-statistics-measurable-aux:
    fixes }X:: nat => ' a = ('b :: {linorder-topology, second-countable-topology})
    assumes n\geq1
    assumes j<n
    assumes }\i.i<n\LongrightarrowXi\in measurable M borel
    shows (\lambdax. (sort-map (\lambdai. X i x) n) j) \in measurable M borel
proof -
    have n-ge-0:n > 0 using assms by simp
    define is-swap where is-swap = (\lambda(ts :: ((nat m'b) => nat 缶'b)).\existsi<n.\existsj
< n.ts = sort-primitive i j)
    define t:: ((nat => 'b) => nat => 'b) list
    where t=[sort-primitive j i. i<- [0..<n], j<- [0..<i]]
    define meas-ptw :: (nat => ' a # 'b) => bool
    where meas-ptw = (\lambdaf. (\forallk.k<n\longrightarrowfk\in borel-measurable M)}
    have ind-step:
    \x(g:: nat 殒 a m 'b). meas-ptw g\Longrightarrow is-swap x \Longrightarrow meas-ptw ( }\lambdak\omega.x(\lambdai
gi \omega)k)
    proof -
    fix }x
    assume meas-ptw g
    hence a:\k. k<n\Longrightarrowgk\in borel-measurable M by (simp add:meas-ptw-def)
    assume is-swap x
    then obtain ij where x-def:x=sort-primitive i j and i-le:i<n and j-le:j<
n
        by (simp add:is-swap-def, blast)
    have }\k.k<n\Longrightarrow(\lambda\omega.x(\lambdai.g i\omega)k)\in\mathrm{ borel-measurable M
    proof -
        fix }
        assume }k<
        thus (\lambda\omega.x (\lambdai.gi\omega)k)\in borel-measurable M
            apply (simp add:x-def)
            apply (cases k=i, simp)
            using a i-le j-le borel-measurable-min apply blast
            apply (cases k=j, simp)
            using a i-le j-le borel-measurable-max apply blast
            using a by simp
    qed
    thus meas-ptw (\lambdak \omega. x (\lambdai.g i \omega)k)
        by (simp add:meas-ptw-def)
    qed
    have }(\forallx\in\mathrm{ set t. is-swap }x)\Longrightarrow\mathrm{ meas-ptw ( }\lambdak\omega.(fold id t (\lambdak.X k \omega)) k
    proof (induction t rule:rev-induct)
```

```
        case Nil
        then show ?case using assms by (simp add:meas-ptw-def)
    next
        case (snoc x xs)
        have a:meas-ptw (\lambdak\omega. fold (\lambdaa. a) xs (\lambdak. X k \omega) k) using snoc by simp
        have b:is-swap x using snoc by simp
        show ?case using ind-step[OF a b] by simp
    qed
    moreover have }\x.x\in\mathrm{ set }t\Longrightarrow\mathrm{ is-swap x
        apply (simp add:t-def is-swap-def)
        by (meson atLeastLessThan-iff imageE less-imp-le less-le-trans)
    ultimately show ?thesis using assms
    by (simp add:t-def[symmetric] meas-ptw-def)
qed
Main results of this section:
lemma order-statistics-measurable:
    fixes X :: nat = ' }a=>\mathrm{ ('b :: {linorder-topology, second-countable-topology})
    assumes n\geq1
    assumes j<n
    assumes \i. i<n\LongrightarrowXi\in measurable M borel
    shows (\lambdax.(sort (map (\lambdai. X i x) [0..<n]))!j)\in measurable M borel
    apply (subst sort-map-eq-sort[symmetric])
    using assms by (simp add:order-statistics-measurable-aux del:sort-map.simps)
definition median :: nat }=>(nat=>('a :: linorder )) 㷋'夕'' a where
    median nf = sort (map f [0..<n])! (n div 2)
lemma median-measurable:
    fixes }X:: nat = ' a m ('b :: {linorder-topology, second-countable-topology})
    assumes n\geq1
    assumes }\i.i<n\LongrightarrowXi\in measurable M bore
    shows (\lambdax. median n (\lambdai. X i x)) \in measurable M borel
    apply (simp add:median-def)
    apply (rule order-statistics-measurable[OF assms(1) - assms(2)])
    using assms(1) by force+
```


## 3 The Median Method

This section contains the proof for the probability that the median of independent random variables will be in an interval with high probability if the individual variables are in the same interval with probability larger than $\frac{1}{2}$. The proof starts with the elementary observation that the median of a seqeuence with $n$ elements is in an interval $I$ if at least half of them are in $I$. This works because after sorting the sequence the elements that will be in the interval must necessarily form a consecutive subsequence, if its length is larger than $\frac{n}{2}$ the median must be in it.

The remainder follows the proof in $[1, \S 2.1]$ using the Hoeffding inequality to estimate the probability that at least half of the sequence elements will be in the interval $I$.

```
lemma interval-rule:
    assumes interval I
    assumes a\leqx x\leqb
    assumes a\inI
    assumes b\inI
    shows }x\in
    using assms(1) apply (simp add:interval-def)
    using assms by blast
lemma sorted-int:
    assumes interval I
    assumes sorted xs
    assumes k< length xs i\leqjj\leqk
    assumes xs !i\inIxs!k\inI
    shows xs ! j f I
    apply (rule interval-rule[where }a=xs!i\mathrm{ and }b=xs!k]
    using assms by (simp add: sorted-nth-mono)+
lemma mid-in-interval:
    assumes 2*length (filter ( }\lambdax.x\inI) xs)> length xs
    assumes interval I
    assumes sorted xs
    shows xs ! (length xs div 2) }\in
proof -
    have length (filter (\lambdax. x \inI) xs) > 0 using assms(1) by linarith
    then obtain v}\mathrm{ where v-1:v<length xs and v-2:xs!v 
    by (metis filter-False in-set-conv-nth length-greater-0-conv)
    define J where }J={k.k<length xs ^xs!k\inI
    have card-J-min: 2*card J > length xs
    using assms(1) by (simp add:J-def length-filter-conv-card)
    consider
    (a) xs!(length xs div 2) }\inI
    (b) xs!(length xs div 2) }\not\inI\wedgev>(length xs div 2) |
    (c) xs ! (length xs div 2) }\not\inI\wedgev< (length xs div 2)
    by (metis linorder-cases v-2)
thus ?thesis
proof (cases)
    case a
    then show ?thesis by simp
    next
    case b
    have p:\k. k\leq length xs div 2 \Longrightarrowxs ! k\not\inI
        using b v-2 sorted-int[OF assms(2) assms(3) v-1, where j=length xs div 2]
```

```
apply simp by blast
    have card J\leqcard {Suc (length xs div 2)..<length xs}
        apply (rule card-mono, simp)
        apply (rule subsetI, simp add:J-def not-less-eq-eq[symmetric])
        using p by metis
    hence card J \leq length xs - (Suc (length xs div 2))
        using card-atLeastLessThan by metis
    hence length xs \leq2*( length xs - (Suc (length xs div 2)))
        using card-J-min by linarith
    hence False
        apply (simp add:nat-distrib)
        apply (subst (asm) le-diff-conv2) using b v-1 apply linarith
        by simp
    then show ?thesis by simp
    next
    case c
    have p:\k. k\geq length xs div 2 \Longrightarrow k length xs \Longrightarrow xs ! k\not\inI
        using c v-1 v-2 sorted-int[OF assms(2) assms(3), where i=v and j=length
xs div 2] apply simp by blast
    have card J \leq card {0..<(length xs div 2)}
        apply (rule card-mono, simp)
        apply (rule subsetI, simp add:J-def not-less-eq-eq[symmetric])
        using p linorder-le-less-linear by blast
    hence card J\leq(length xs div 2)
        using card-atLeastLessThan by simp
    then show ?thesis using card-J-min by linarith
    qed
qed
lemma median-est:
    assumes interval I
    assumes 2*card {k.k<n^fk\inI}>n
    shows median nf}\in
proof -
    have a:{k.k<n\wedgefk\inI}={i.i<n\wedge map f[0..<n]!i\inI}
        apply (rule order-antisym, rule subsetI, simp)
        by (rule subsetI, simp, metis add-0 diff-zero nth-map-upt)
    show ?thesis
        apply (simp add:median-def)
        apply (rule mid-in-interval[where I=I and xs=sort (map f [0..<n]), simpli-
fied])
        using assms a apply (simp add:filter-sort comp-def length-filter-conv-card)
    by (simp add:assms)
qed
lemma prod-pmf-bernoulli-mono:
    assumes finite I
    assumes \i.i\inI\Longrightarrow0\leqfi\wedgefi\leqgi^gi\leq1
```

```
    assumes \(\bigwedge x y . x \in A \Longrightarrow(\forall i \in I . x i \leq y i) \Longrightarrow y \in A\)
    shows measure (Pi-pmf Id (bernoulli-pmf \(\circ f\) )) \(A \leq\) measure (Pi-pmf I d
(bernoulli-pmf \(\circ g\) )) A
    \((\) is \(? L \leq ? R)\)
proof -
    define \(q\) where \(q i=p m f\)-of-list \([(0:: n a t, f i),(1, g i-f i),(2,1-g i)]\) for \(i\)
    have wf:pmf-of-list-wf \([(0:: n a t, f i),(1, g i-f i),(2,1-g i)]\) if \(i \in I\) for \(i\)
    using assms(2)[OF that] by (intro pmf-of-list-wfI) auto
    have 0: bernoulli-pmf \((f i)=m a p-p m f(\lambda x . x=0)(q i)(\) is \(? L 1=? R 1)\)
    if \(i \in I\) for \(i\)
proof -
    have \(0 \leq f i f i \leq 1\) using \(\operatorname{assms}\) (2) \([\) OF that \(]\) by auto
    hence \(p m f\) ? \(L 1 x=p m f\) ? \(R 1 x\) for \(x\)
        unfolding \(q\)-def pmf-map measure-pmf-of-list[OF wf[OF that]]
        by (cases x;simp-all add:vimage-def)
    thus ?thesis
        by (intro pmf-eqI) auto
    qed
    have 1: bernoulli-pmf \((g i)=\operatorname{map-pmf}(\lambda x . x \in\{0,1\})(q i)(\) is ? \(L 1=? R 1)\)
    if \(i \in I\) for \(i\)
proof -
    have \(0 \leq g i g i \leq 1\) using \(\operatorname{assms}(2)[O F\) that \(]\) by auto
    hence \(p m f\) ? L1 \(x=p m f\) ? \(R 1 x\) for \(x\)
        unfolding \(q\)-def pmf-map measure-pmf-of-list[OF wf[OF that]]
        by (cases \(x\);simp-all add:vimage-def)
    thus ?thesis
        by (intro pmf-eqI) auto
    qed
```

    have 2: \((\lambda k . x k=0) \in A \Longrightarrow(\lambda k . x k=0 \vee x k=\) Suc 0\() \in A\) for \(x\)
    by (erule assms(3)) auto
    have ? \(L=\) measure \((\) Pi-pmf Id \((\lambda i . \operatorname{map-pmf}(\lambda x . x=0)(q i))) A\)
        unfolding comp-def by (simp add:0 cong: Pi-pmf-cong)
    also have \(\ldots=\) measure (map-pmf \(((\circ)(\lambda x . x=0))(\) Pi-pmf I (if \(d\) then 0 else
    2) q)) $A$
by (intro arg-cong2[where $f=$ measure-pmf.prob] Pi-pmf-map[OF $\operatorname{assms}(1)])$
auto
also have $\ldots=$ measure (Pi-pmf I (if d then 0 else 2) $q$ ) $\{x .(\lambda k . x k=0) \in A\}$
by (simp add:comp-def vimage-def)
also have $\ldots \leq$ measure (Pi-pmf I (if $d$ then 0 else 2) $q)\{x$. ( $\lambda k . x k \in\{0,1\}$ )
$\in A\}$
using 2 by (intro measure-pmf.finite-measure-mono subsetI) auto
also have $\ldots=$ measure (map-pmf $((\circ)(\lambda x . x \in\{0,1\}))(P i-p m f I$ (if $d$ then 0
else 2) q)) $A$
by (simp add:vimage-def comp-def)
```
    also have ... = measure (Pi-pmf I d (\lambdai.map-pmf (\lambdax. x \in{0,1}) (qi)))A
    by (intro arg-cong2[where f=measure-pmf.prob] Pi-pmf-map[OF assms(1),
symmetric]) auto
    also have ... = ?R
        unfolding comp-def by (simp add:1 cong: Pi-pmf-cong)
    finally show ?thesis by simp
qed
lemma discrete-measure-eqI:
    assumes sets M = count-space UNIV
    assumes sets N = count-space UNIV
    assumes countable \Omega
```



```
\not=\infty
    assumes AE x in M. x\in\Omega
    assumes AE x in N. x 琽
    shows M=N
proof -
    define E where E = insert {} (( }\lambdax.{x})'\Omega
    have 0: Int-stable E unfolding E-def by (intro Int-stableI) auto
    have 1: countable E using assms(3) unfolding E-def by simp
    have E\subseteq Pow \Omega unfolding E-def by auto
    have emeasure MA= emeasure NA if A-range: A E E for }
    using that assms(4) unfolding E-def by auto
    moreover have sets M = sets N using assms(1,2) by simp
    moreover have }\Omega\in\mathrm{ sets M using assms(1) by simp
    moreover have }E\not={}\mathrm{ unfolding E-def by simp
    moreover have }\bigcupE=\Omega\mathrm{ unfolding E-def by simp
    moreover have emeasure Ma\not=\infty if a\inE for a
        using that assms(4) unfolding E-def by auto
    moreover have sets (restrict-space M \Omega)=Pow \Omega
        using assms(1) by (simp add:sets-restrict-space range-inter)
    moreover have sets (restrict-space N \Omega)=Pow \Omega
    using assms(2) by (simp add:sets-restrict-space range-inter)
    moreover have sigma-sets \Omega E=Pow \Omega
        unfolding E-def by (intro sigma-sets-singletons-and-empty assms(3))
    ultimately show ?thesis
    by (intro measure-eqI-restrict-generator[OF 0 - - - - assms(5,6) 1]) auto
qed
```

Main results of this section:
The next theorem establishes a bound for the probability of the median of independent random variables using the binomial distribution. In a follow-up step, we will establish tail bounds for the binomial distribution and corresponding median bounds.
This two-step strategy was suggested by Yong Kiam Tan. In a previ-
ous version, I only had verified the exponential tail bound (see theorem median_bound below).
theorem (in prob-space) median-bound-raw:
fixes $I::(' b::\{$ linorder-topology, second-countable-topology\}) set
assumes $n>0 p \geq 0$
assumes interval I
assumes indep-vars ( $\lambda$-. borel) $X\{0 . .<n\}$
assumes $\bigwedge i . i<n \Longrightarrow \mathcal{P}(\omega$ in $M . X i \omega \in I) \geq p$
shows $\mathcal{P}(\omega$ in $M$. median $n(\lambda i . X i \omega) \in I) \geq 1$ - measure (binomial-pmf $n p$ )
\{..n div 2\}
(is ? $L \geq$ ? $R$ )
proof -
let ?pi $=\operatorname{Pi}$-pmf $\{. .<n\}$ undefined
define $q$ where $q i=\mathcal{P}(\omega$ in $M . X i \omega \in I)$ for $i$
have $n$-ge-1: $n \geq 1$ using $\operatorname{assms}(1)$ by simp
have 0: $\{k . k<n \wedge(k<n \longrightarrow X k \omega \in I)\}=\{k . k<n \wedge X k \omega \in I\}$ for $\omega$ by auto
have countable $\left(\{. .<n\} \rightarrow_{E}(U N I V::\right.$ bool set $\left.)\right)$
by (intro countable-PiE) auto
hence countable-ext: countable (extensional $\{. .<n\}::($ nat $\Rightarrow$ bool) set)
unfolding PiE-def by auto
have $m 0: I \in$ sets borel
using interval-borel[OF assms(3)] by simp
have $m 1$ : random-variable borel $(\lambda x . X k x)$ if $k \in\{. .<n\}$ for $k$
using assms(4) that unfolding indep-vars-def by auto
have m2: $(\lambda x . x \in I) \in$ borel $\rightarrow_{M}($ measure-pmf $(($ bernoulli-pmf $\circ q) k))$
for $k$ using $m 0$ by measurable
hence m3: random-variable (measure-pmf ((bernoulli-pmf $\circ q) k))(\lambda x . X k x \in$ I)
if $k \in\{. .<n\}$ for $k$
by (intro measurable-compose[OF m1] that)
hence m4: random-variable (PiM $\{. .<n\}$ (bernoulli-pmf $\circ q)$ ) $(\lambda \omega . \lambda k \in\{. .<n\}$.
$X k \omega \in I)$
by (intro measurable-restrict) auto
moreover have $A \in$ sets $\left(P i_{M}\{. .<n\}(\lambda x\right.$. measure-pmf (bernoulli-pmf $\left.\left.(q x))\right)\right)$
if $A \subseteq$ extensional $\{. .<n\}$ for $A$
proof -
have $A=(\bigcup a \in A .\{a\})$ by auto
also have $\ldots=(\bigcup a \in A . \operatorname{PiE}\{. .<n\}(\lambda k .\{a k\}))$
using that by (intro arg-cong[where $f=$ Union $]$ image-cong refl PiE-singleton[symmetric]) auto
also have $\ldots \in \operatorname{sets}\left(P i_{M}\{. .<n\}(\lambda x\right.$. measure-pmf (bernoulli-pmf $\left.\left.(q x))\right)\right)$
using that countable-ext countable-subset
by (intro sets.countable-Union countable-image image-subsetI sets-PiM-I-finite) auto
finally show?thesis by simp
qed
hence $m 5: i d \in(\operatorname{PiM}\{. .<n\}($ bernoulli-pmf $\circ q)) \rightarrow_{M}($ count-space UNIV)
by (intro measurableI) (simp-all add:vimage-def space-PiM PiE-def)
ultimately have random-variable (count-space UNIV) $(i d \circ(\lambda \omega . \lambda k \in\{. .<n\} . X$ $k \omega \in I)$ )
by (rule measurable-comp)
hence m6: random-variable (count-space UNIV) $(\lambda \omega . \lambda k \in\{. .<n\} . X k \omega \in I)$ by $\operatorname{simp}$
have indep: indep-vars (bernoulli-pmf $\circ q$ ) $(\lambda i x . X i x \in I)\{0 . .<n\}$
by (intro indep-vars-compose2[OF assms(4)] m2)
have measure $M\{x \in$ space $M .(X k x \in I)=\omega\}=$ measure (bernoulli-pmf $(q$ k)) $\{\omega\}$
if $k<n$ for $\omega k$
proof (cases $\omega$ )
case True
then show ?thesis unfolding $q$-def by (simp add:measure-pmf-single)
next
case False
have $\{x \in$ space $M . X k x \in I\} \in$ events
using that m0 by (intro measurable-sets-Collect[OF m1]) auto
hence $\operatorname{prob}\{x \in \operatorname{space} M . X k x \notin I\}=1-\operatorname{prob}\{\omega \in \operatorname{space} M . X k \omega \in I\}$ by (subst prob-neg) auto
thus ?thesis using False unfolding $q$-def by (simp add:measure-pmf-single)
qed
hence 1: emeasure $M\{x \in$ space $M .(X k x \in I)=\omega\}=$ emeasure (bernoulli-pmf $(q k))\{\omega\}$
if $k<n$ for $\omega k$
using that unfolding emeasure-eq-measure measure-pmf.emeasure-eq-measure by $\operatorname{simp}$
interpret product-sigma-finite (bernoulli-pmf $\circ q$ )
by standard
have $\operatorname{distr} M($ count-space UNIV $)(\lambda \omega .(\lambda k \in\{. .<n\} . X k \omega \in I))=$ distr
(distr M (PiM $\{. .<n\}$ (bernoulli-pmf $\circ q))(\lambda \omega . \lambda k \in\{. .<n\} . X k \omega \in I))$
(count-space UNIV) id
by (subst distr-distr[OF m5 m4]) (simp add:comp-def)
also have $\ldots=\operatorname{distr}(\operatorname{PiM}\{. .<n\}(\lambda i .($ distr $M(($ bernoulli-pmf $\circ q) i)(\lambda \omega . X$ $i \omega \in I)$ ))
(count-space UNIV) id
using assms(1) indep atLeast0LessThan by (intro arg-cong2[where $f=\lambda x y$. distr $x$ y id]
iffD1 [OF indep-vars-iff-distr-eq-PiM] m3) auto
also have $\ldots=\operatorname{distr}(\operatorname{PiM}\{. .<n\}($ bernoulli-pmf $\circ q))($ count-space UNIV) id
using m3 1 by (intro distr-cong PiM-cong refl discrete-measure-eqI[where $\Omega=U N I V]$ )
(simp-all add:emeasure-distr vimage-def Int-def conj-commute)
also have $\ldots=$ ?pi (bernoulli-pmf $\circ q$ )
proof (rule discrete-measure-eq $[$ where $\Omega=$ extensional $\{. .<n\}]$, goal-cases)
case 1 show? case by simp
next
case 2 show ?case by simp
next
case 3 show ? case using countable-ext by simp
next
case (4 x)
have emeasure $\left(P i_{M}\{. .<n\}(\right.$ bernoulli-pmf $\left.\circ q)\right)\{x\}=$ emeasure $\left(P i_{M}\{. .<n\}(\right.$ bernoulli-pmf $\left.\circ q)\right)(\operatorname{PiE}\{. .<n\}(\lambda k .\{x k\}))$ using PiE-singleton[OF 4] by simp
also have $\ldots=\left(\prod i<n\right.$. emeasure ( measure-pmf (bernoulli-pmf $\left.\left.\left.(q i)\right)\right)\{x i\}\right)$ by (subst emeasure-PiM) auto
also have $\ldots=$ emeasure $($ Pi-pmf $\{. .<n\}$ undefined $($ bernoulli-pmf $\circ q))$ (PiE-dflt $\{. .<n\}$ undefined $(\lambda k .\{x k\}))$
unfolding measure-pmf.emeasure-eq-measure
by (subst measure-Pi-pmf-PiE-dftt) (simp-all add:prod-ennreal)
also have $\ldots=$ emeasure (Pi-pmf $\{. .<n\}$ undefined (bernoulli-pmf $\circ q$ )) $\{x\}$
using 4 by (intro arg-cong2[where $f=$ emeasure]) (auto simp add:PiE-dflt-def extensional-def)
finally have emeasure $\left(P i_{M}\{. .<n\}\right.$ (bernoulli-pmf $\left.\left.\circ q\right)\right)\{x\}=$
emeasure (Pi-pmf $\{. .<n\}$ undefined (bernoulli-pmf $\circ q)$ ) $\{x\}$
by $\operatorname{simp}$
thus ?case using 4
by (subst (1 2) emeasure-distr[OF m5]) (simp-all add:vimage-def space-PiM
PiE-def)
next
case 5
have $A E x$ in $P i_{M}\{. .<n\}$ (bernoulli-pmf $\circ q$ ). $x \in$ extensional $\{. .<n\}$
by (intro AE-I2) (simp add:space-PiM PiE-def)
then show ?case by (subst AE-distr-iff[OF m5]) simp-all
next
case 6
then show? ?case by (intro AE-pmfI) (simp add: set-Pi-pmf PiE-dftt-def ex-tensional-def)
qed
finally have 2: distr $M$ (count-space UNIV) $(\lambda \omega .(\lambda k \in\{. .<n\} . X k \omega \in I))=$ ?pi (bernoulli-pmfoq)
by $\operatorname{simp}$
have 3: $n<2 * \operatorname{card}\{k . k<n \wedge y k\}$ if $n<2 * \operatorname{card}\{k . k<n \wedge x k\} \bigwedge i . i<n \Longrightarrow x i \Longrightarrow y i$ for $x y$
proof -

```
    have 2 * card {k.k<n\wedgexk}\leq2 * card {k.k<n^yk}
        using that(2) by (intro mult-left-mono card-mono) auto
    thus ?thesis using that(1) by simp
    qed
    have 4:0\leqp^p\leqqi}\ qi\leq1 if i<n for 
    unfolding q-def using assms(2,5) that by auto
    have p-range: p\in{0..1}
    using 4[OF assms(1)] by auto
    have ?R = 1 - measure-pmf.prob (binomial-pmf n p) {k. 2* * <n}
        by (intro arg-cong2[where f=(-)] arg-cong2[where f=measure-pmf.prob])
auto
    also have ... = measure (binomial-pmf n p) {k.n<2*k}
        by (subst measure-pmf.prob-compl[symmetric]) (simp-all add:set-diff-eq not-le)
    also have ... = measure (?pi (bernoulli-pmf ○ (\lambda-. p))) {\omega. n<2 * card {k.k
< n\wedge\omega k}}
    using p-range by (subst binomial-pmf-altdef'[where A={..<n} and dflt=undefined])
auto
    also have ... \leq measure (?pi (bernoulli-pmf ○ q)) {\omega. n<2* card {k.k<n
\wedge\omegak}}
    using 3 4 by (intro prod-pmf-bernoulli-mono) auto
    also have ... =
    P}(\omega\mathrm{ in distr M (count-space UNIV) ( }\lambda\omega.\lambdak\in{..<n}. X k \omega\inI). n<2*card
{k. k<n\wedge\omega k})
    unfolding 2 by simp
    also have ... = \mathcal{P}(\omega\mathrm{ in M. n<2*card {k.k<n^Xk }|\inI})
    by (subst measure-distr[OF m6]) (simp-all add:vimage-def Int-def conj-commute
0)
    also have ... \leq?L
    using median-est[OF assms(3)] m0 m1
    by (intro finite-measure-mono measurable-sets-Collect[OF median-measurable[OF
n-ge-1]]) auto
    finally show ?R}\leq?L\mathrm{ by simp
qed
Cumulative distribution of the binomial distribution (contributed by Yong Kiam Tan):
lemma prob-binomial-pmf-upto:
assumes \(0 \leq p p \leq 1\)
shows measure-pmf.prob (binomial-pmf \(n\) p) \(\{. . m\}=\)
sum \((\lambda i\). real ( \(n\) choose \(i) * p \widehat{i} *(1-p) \uparrow(n-i))\{0 . . m\}\)
by (auto simp: pmf-binomial[OF assms] measure-measure-pmf-finite intro!: sum.cong)
A tail bound for the binomial distribution using Hoeffding's inequality:
lemma binomial-pmf-tail:
assumes \(p \in\{0 . .1\}\) real \(k \leq\) real \(n * p\)
```

shows measure (binomial-pmf $n p)\{. . k\} \leq \exp (-2 *$ real $n *(p-$ real $k /$ n) ${ }^{2}$ 2)
(is $? L \leq ? R$ )
proof (cases $n=0$ )
case True then show ?thesis by simp
next
case False
let ? $A=\{. .<n\}$
let ?pi $=$ Pi-pmf ?A undefined $(\lambda-$. bernoulli-pmf $p)$
define $\mu$ where $\mu=\left(\sum i<n\right.$. $\left(\int x\right.$. (of-bool $(x i)::$ real) $\partial$ ?pi) $)$
define $\varepsilon$ :: real where $\varepsilon=\mu-k$
have $\mu=\left(\sum i<n .\left(\int x\right.\right.$. (of-bool $x::$ real $) \partial($ map-pmf $(\lambda \omega . \omega i)$ ?pi) $\left.)\right)$
unfolding $\mu$-def by simp
also have $\ldots=\left(\sum i<n .\left(\int x .(\right.\right.$ of-bool $x::$ real $) \partial($ bernoulli-pmf $\left.\left.p)\right)\right)$
by (simp add: Pi-pmf-component)
also have $\ldots=$ real $n * p$ using assms(1) by simp
finally have $\mu$-alt: $\mu=$ real $n * p$
by $\operatorname{simp}$
have $\varepsilon-g e-0: \varepsilon \geq 0$
using assms(2) unfolding $\varepsilon$-def $\mu$-alt by auto
have indep: prob-space.indep-vars ?pi ( $\lambda$-. borel) $(\lambda k \omega$. of-bool $(\omega k))\{. .<n\}$
by (intro prob-space.indep-vars-compose2[OF prob-space-measure-pmf indep-vars-Pi-pmf]) auto
interpret Hoeffding-ineq ?pi $\{. .<n\} \lambda k \omega$. of-bool $(\omega k) \lambda-.0 \lambda-.1 \mu$
using indep unfolding $\mu$-def by (unfold-locales) simp-all
have ? $L=$ measure (map-pmf $(\lambda f$. card $\{x \in$ ?A. $f x\}$ ) ?pi) $\{. . k\}$
by (intro arg-cong2[where $f=$ measure-pmf.prob] binomial-pmf-altdef' assms(1))
auto
also have $\ldots=\mathcal{P}\left(\omega\right.$ in ?pi. $\left(\sum i<n\right.$. of-bool $\left.\left.(\omega i)\right) \leq \mu-\varepsilon\right)$
unfolding $\varepsilon$-def by (simp add:vimage-def Int-def)
also have $\ldots \leq \exp \left(-2 * \varepsilon^{2} /\left(\sum i<n .(1-0)^{2}\right)\right)$
using False by (intro Hoeffding-ineq-le $\varepsilon$-ge-0) auto
also have $\ldots=$ ? $R$
unfolding $\varepsilon$-def $\mu$-alt by (simp add:power2-eq-square field-simps)
finally show ?thesis by simp
qed
theorem (in prob-space) median-bound:
fixes $n::$ nat
fixes $I::(' b::\{$ linorder-topology, second-countable-topology $\})$ set
assumes interval $I$
assumes $\alpha>0$
assumes $\varepsilon \in\{0<. .<1\}$
assumes indep-vars ( $\lambda$-. borel) $X\{0 . .<n\}$

```
    assumes \(n \geq-\ln \varepsilon /\left(2 * \alpha^{2}\right)\)
    assumes \(\bigwedge \bar{i} . i<n \Longrightarrow \mathcal{P}(\omega\) in \(M . X i \omega \in I) \geq 1 / \mathcal{D}+\alpha\)
    shows \(\mathcal{P}(\omega\) in \(M\). median \(n(\lambda i\). \(X i \omega) \in I) \geq 1-\varepsilon\)
proof -
    have \(0<-\ln \varepsilon /\left(2 * \alpha^{2}\right)\)
    using assms by (intro divide-pos-pos) auto
    also have \(\ldots \leq\) real \(n\) using assms by simp
    finally have real \(n>0\) by simp
    hence \(n-g e-0: n>0\) by \(\operatorname{simp}\)
    have d0: real-of-int \(\lfloor\) real \(n / 2\rfloor * 2 /\) real \(n \leq 1\)
    using \(n\)-ge-0 by simp linarith
    hence d1: real (nat \(\lfloor\) real \(n / 2\rfloor) \leq\) real \(n *(1 / 2)\)
    using \(n\)-ge- 0 by (simp add:field-simps)
    also have \(\ldots \leq\) real \(n *(1 / 2+\alpha)\)
    using assms(2) by (intro mult-left-mono) auto
    finally have d1: real (nat \(\lfloor\) real \(n / 2\rfloor) \leq\) real \(n *(1 / 2+\alpha)\) by simp
    have \(1 / 2+\alpha \leq \mathcal{P}(\omega\) in \(M . X 0 \omega \in I)\) using \(n\)-ge-0 by (intro assms(6))
    also have \(\ldots \leq 1\) by simp
    finally have \(d 2: 1 / 2+\alpha \leq 1\) by \(\operatorname{simp}\)
    have d3: nat \(\lfloor\) real \(n / 2\rfloor=n\) div 2 by linarith
    have \(1-\varepsilon \leq 1-\exp \left(-2 *\right.\) real \(\left.n * \alpha^{2}\right)\)
        using assms \((2,3,5)\) by (intro diff-mono order.refl iffD1[OF ln-ge-iff]) (auto
simp:field-simps)
    also have \(\ldots \leq 1-\exp (-2 *\) real \(n *((1 / 2+\alpha)-\) real \((\) nat \(\lfloor\) real \(n / 2\rfloor) /\) real
\(n)^{2}\) )
    using d0 n-ge-0 assms(2)
            by (intro diff-mono order.refl iffD2[OF exp-le-cancel-iff] mult-left-mono-neg
power-mono) auto
    also have \(\ldots \leq 1-\) measure (binomial-pmf \(n(1 / 2+\alpha))\{\)..nat \(\lfloor\) real \(n / 2\rfloor\}\)
        using assms(2) d1 d2 by (intro diff-mono order.refl binomial-pmf-tail) auto
    also have \(\ldots=1\) - measure (binomial-pmf \(n(1 / 2+\alpha))\{\)..n div 2\(\}\) by (simp
add:d3)
    also have \(\ldots \leq \mathcal{P}(\omega\) in M. median \(n(\lambda i . X i \omega) \in I)\)
    using assms(2) by (intro median-bound-raw n-ge-0 assms (1,4,6) add-nonneg-nonneg)
auto
    finally show?thesis by simp
qed
```

This is a specialization of the above to closed real intervals.

```
corollary (in prob-space) median-bound-1:
    assumes \alpha>0
    assumes }\varepsilon\in{0<..<1
    assumes indep-vars ( }\lambda\mathrm{ -. borel) X {0..<n}
    assumes n\geq-ln\varepsilon/(2*\mp@subsup{\alpha}{}{2})
```

```
assumes \(\forall i \in\{0 . .<n\} . \mathcal{P}(\omega\) in \(M . X i \omega \in(\{a . . b\}::\) real set \()) \geq 1 / \mathcal{D}+\alpha\)
shows \(\mathcal{P}(\omega\) in \(M\). median \(n(\lambda i\). \(X i \omega) \in\{a . . b\}) \geq 1-\varepsilon\)
using \(\operatorname{assms}(5)\) by (intro median-bound \([O F-\operatorname{assms}(1,2,3,4)]\) ) (auto simp:interval-def)
```

This is a specialization of the above, where $\alpha=\frac{1}{6}$ and the interval is described using a mid point $\mu$ and radius $\delta$. The choice of $\alpha=\frac{1}{6}$ implies a success probability per random variable of $\frac{2}{3}$. It is a commonly chosen success probability for Monte-Carlo algorithms (cf. [2, §4] or [3, §1]).

```
corollary (in prob-space) median-bound-2:
    fixes }\mu\delta::\mathrm{ real
    assumes }\varepsilon\in{0<..<1
    assumes indep-vars (\lambda-. borel) X {0..<n}
    assumes n\geq-18* ln \varepsilon
    assumes }\bigwedgei.i<n\Longrightarrow\mathcal{P}(\omega\mathrm{ in M. abs (Xi }\omega-\mu)>\delta)\leq1/
    shows }\mathcal{P}(\omega\mathrm{ in M. abs (median n (גi.X i }\omega)-\mu)\leq\delta)\geq\overline{1}-
proof -
    have b:\i. i<n\Longrightarrow space M - {\omega { space M. X i\omega 
{\omega\in space M. abs (Xi\omega-\mu)>\delta}
    apply (rule order-antisym, rule subsetI, simp, linarith)
    by (rule subsetI, simp, linarith)
    have }\i.i<n\Longrightarrow1-\mathcal{P}(\omega\mathrm{ in M. X i }\omega\in{\mu-\delta..\mu+\delta})\leq1/
    apply (subst prob-compl[symmetric])
        apply (measurable)
        using assms(2) apply (simp add:indep-vars-def)
    apply (subst b, simp)
    using assms(4) by simp
    hence a:\i. i<n\Longrightarrow\mathcal{P}(\omega\mathrm{ in M. Xi }\omega\in{\mu-\delta..\mu+\delta})\geq2/3 by simp
    have 1-\varepsilon\leq\mathcal{P}(\omega in M. median n (\lambdai.X i \omega) \in{\mu-\delta..\mu+\delta})
    apply (rule median-bound-1[OF - assms(1) assms(2), where \alpha=1/6], simp)
        using assms(3) apply (simp add:power2-eq-square)
        using a by simp
    also have ... = \mathcal{P}(\omega\mathrm{ in M. abs (median n (\i. Xi }\omega)-\mu)\leq\delta)
    apply (rule arg-cong2[where f=measure], simp)
    apply (rule order-antisym, rule subsetI, simp, linarith)
    by (rule subsetI, simp, linarith)
    finally show ?thesis by simp
qed
```


## 4 Some additional results about the median

lemma sorted-mono-map:
assumes sorted xs
assumes mono $f$
shows sorted (map fxs)
using assms apply (simp add:sorted-wrt-map)
apply (rule sorted-wrt-mono-rel $[$ where $P=(\leq)]$ ) by (simp add:mono-def, simp)

This could be added to HOL.List:

```
lemma map-sort:
    assumes mono \(f\)
    shows sort (map fxs) \(=\operatorname{map} f(\) sort \(x s)\)
    using assms by (intro properties-for-sort sorted-mono-map) auto
lemma median-cong:
    assumes \(\bigwedge i . i<n \Longrightarrow f i=g i\)
    shows median \(n f=\) median \(n g\)
    apply (cases \(n=0\), simp add:median-def)
    apply ( simp add:median-def)
    apply (rule arg-cong2 [where \(f=(!)]\) )
    apply (rule arg-cong[where \(f=\) sort], rule map-cong, simp, simp add:assms)
    by \(\operatorname{simp}\)
lemma median-restrict:
    median \(n(\lambda i \in\{0 . .<n\} . f i)=\) median \(n f\)
    by (rule median-cong, simp)
lemma median-commute-mono:
    assumes \(n>0\)
    assumes mono \(g\)
    shows \(g(\) median \(n f)=\) median \(n(g \circ f)\)
    apply (simp add: median-def del:map-map)
    apply (subst map-map[symmetric])
    apply (subst map-sort[OF assms(2)])
    apply (subst nth-map, simp) using assms apply fastforce
    by \(\operatorname{simp}\)
lemma median-rat:
    assumes \(n>0\)
    shows real-of-rat (median \(n f\) ) \(=\) median \(n(\lambda i\).real-of-rat \((f i))\)
    apply (subst (2) comp-def[where \(g=f\), symmetric])
    apply (rule median-commute-mono[OF assms(1)])
    by (simp add: mono-def of-rat-less-eq)
lemma median-const:
    assumes \(k>0\)
    shows median \(k(\lambda i \in\{0 . .<k\} . a)=a\)
proof -
    have b: sorted (map ( \(\lambda\)-. a) \([0 . .<k]\) )
    by (subst sorted-wrt-map, simp)
    have \(a\) : sort \((\operatorname{map}(\lambda-. a)[0 . .<k])=\operatorname{map}(\lambda-. a)[0 . .<k]\)
    by (subst sorted-sort-id[OF b], simp)
    have median \(k(\lambda i \in\{0 . .<k\} . a)=\) median \(k(\lambda-. a)\)
    by (subst median-restrict, simp)
```

```
    also have ... = a using assms by (simp add:median-def a)
    finally show ?thesis by simp
qed
end
```


## References

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. Journal of Computer and System Sciences, 58(1):137-147, 1999.
[2] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In Randomization and Approximation Techniques in Computer Science, pages 1-10. Springer Berlin Heidelberg, 2002.
[3] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '10, pages 41-52, New York, 2010.

