Formalization of Randomized Approximation Algorithms for Frequency Moments

Emin Karayel

March 17, 2025

Abstract

In 1999 Alon et. al. introduced the still active research topic of approximating the frequency moments of a data stream using randomized algorithms with minimal space usage. This includes the problem of estimating the cardinality of the stream elements—the zeroth frequency moment. But, also higher-order frequency moments that provide information about the skew of the data stream. (The k-th frequency moment of a data stream is the sum of the k-th powers of the occurrence counts of each element in the stream.) This entry formalizes three randomized algorithms for the approximation of F_0, F_2 and F_k for $k \geq 3$ based on [1, 2] and verifies their expected accuracy, success probability and space usage.

Contents

1 Preliminary Results

theory *Frequency-Moments-Preliminary-Results*
imports
HOL.Transcendental
HOL−Computational-Algebra.Primes
HOL−Library.Extended-Real
HOL−Library.Multiset
HOL−Library.Sublist
Prefix-Free-Code-Combinators.Prefix-Free-Code-Combinators
Bertrands-Postulate.Bertrand
Expander-Graphs.Expander-Graphs-Multiset-Extras
begin

This section contains various preliminary results.

lemma *card-ordered-pairs*:
fixes M :: ('a ::linorder) set
assumes finite M
shows $2 * card \{(x,y) \in M \times M.\ x < y\} = card\ M * (card\ M - 1)$
⟨proof⟩

lemma *ereal-mono*: $x \leq y \Longrightarrow ereal\ x \leq ereal\ y$
⟨proof⟩

lemma *abs-ge-iff*: $((x::real) \leq abs\ y) = (x \leq y \vee x \leq -y)$
⟨proof⟩

lemma *count-list-gr-1*:
$(x \in set\ xs) = (count\text{-}list\ xs\ x \geq 1)$
⟨proof⟩

lemma *count-list-append*: $count\text{-}list\ (xs@ys)\ v = count\text{-}list\ xs\ v + count\text{-}list\ ys\ v$
⟨proof⟩

lemma *count-list-lt-suffix*:
assumes *suffix a b*
assumes $x \in \{b\ !\ i |\ i.\ i < length\ b - length\ a\}$
shows $count\text{-}list\ a\ x < count\text{-}list\ b\ x$
⟨proof⟩

lemma *suffix-drop-drop*:
assumes $x \geq y$
shows *suffix (drop x a) (drop y a)*
⟨proof⟩

lemma *count-list-card*: *count-list xs x* = *card* {*k*. *k* < *length xs* ∧ *xs* ! *k* = *x*}
⟨*proof*⟩

lemma *card-gr-1-iff*:
  assumes *finite S* *x* ∈ *S* *y* ∈ *S* *x* ≠ *y*
  shows *card S* > *1*
  ⟨*proof*⟩

lemma *count-list-ge-2-iff*:
  assumes *y* < *z*
  assumes *z* < *length xs*
  assumes *xs* ! *y* = *xs* ! *z*
  shows *count-list xs* (*xs* ! *y*) > *1*
⟨*proof*⟩

Results about multisets and sorting

lemmas *disj-induct-mset* = *disj-induct-mset*

lemma *prod-mset-conv*:
  fixes *f* :: *'a* ⇒ *'b*::{*comm-monoid-mult*}
  shows *prod-mset* (*image-mset f A*) = *prod* (λ*x*. *f x*^(*count A x*)) (*set-mset A*)
⟨*proof*⟩

There is a version *sum-list-map-eq-sum-count* but it doesn't work if the function maps into the reals.

lemma *sum-list-eval*:
  fixes *f* :: *'a* ⇒ *'b*::{*ring*,*semiring-1*}
  shows *sum-list* (*map f xs*) = (∑ *x* ∈ *set xs*. *of-nat* (*count-list xs x*) ∗ *f x*)
⟨*proof*⟩

lemma *prod-list-eval*:
  fixes *f* :: *'a* ⇒ *'b*::{*ring*,*semiring-1*,*comm-monoid-mult*}
  shows *prod-list* (*map f xs*) = (∏ *x* ∈ *set xs*. (*f x*)^(*count-list xs x*))
⟨*proof*⟩

lemma *sorted-sorted-list-of-multiset*: *sorted* (*sorted-list-of-multiset M*)
  ⟨*proof*⟩

lemma *count-mset*: *count* (*mset xs*) *a* = *count-list xs a*
  ⟨*proof*⟩

lemma *swap-filter-image*: *filter-mset g* (*image-mset f A*) = *image-mset f* (*filter-mset* (*g* ∘ *f*) *A*)
  ⟨*proof*⟩

lemma *list-eq-iff*:
  assumes *mset xs* = *mset ys*
  assumes *sorted xs*

assumes *sorted ys*
shows *xs* = *ys*
⟨*proof*⟩

lemma *sorted-list-of-multiset-image-commute*:
assumes *mono f*
shows *sorted-list-of-multiset* (*image-mset f M*) = *map f* (*sorted-list-of-multiset M*)
⟨*proof*⟩

Results about rounding and floating point numbers

lemma *round-down-ge*:
$x \leq$ *round-down prec* x + *2 powr* (−*prec*)
⟨*proof*⟩

lemma *truncate-down-ge*:
$x \leq$ *truncate-down prec* x + *abs* $x * 2$ *powr* (−*prec*)
⟨*proof*⟩

lemma *truncate-down-pos*:
assumes $x \geq 0$
shows $x * (1 - 2$ *powr* (−*prec*)) ≤ *truncate-down prec x*
⟨*proof*⟩

lemma *truncate-down-eq*:
assumes *truncate-down r x* = *truncate-down r y*
shows *abs* (*x*−*y*) ≤ *max* (*abs x*) (*abs y*) ∗ *2 powr* (−*real r*)
⟨*proof*⟩

definition *rat-of-float* :: *float* ⇒ *rat* **where**
rat-of-float f = *of-int* (*mantissa f*) ∗
(*if exponent f* ≥ *0 then 2* ^ (*nat* (*exponent f*)) *else 1* / *2* ^ (*nat* (−*exponent f*)))

lemma *real-of-rat-of-float*: *real-of-rat* (*rat-of-float x*) = *real-of-float x*
⟨*proof*⟩

lemma *log-est*: *log 2* (*real n* + *1*) ≤ *n*
⟨*proof*⟩

lemma *truncate-mantissa-bound*:
abs (⌊*x* ∗ *2 powr* (*real r* − *real-of-int* ⌊*log 2* |*x*|⌋)⌋) ≤ *2* ^ (*r*+*1*) (**is** *?lhs* ≤ -)
⟨*proof*⟩

lemma *truncate-float-bit-count*:
bit-count (F_e (*float-of* (*truncate-down r x*))) ≤ *10* + *4* ∗ *real r* + *2*∗*log 2* (*2* + |*log 2* |*x*||)
(**is** *?lhs* ≤ *?rhs*)
⟨*proof*⟩

definition *prime-above* :: *nat* ⇒ *nat*
where *prime-above n* = (*SOME x. x* ∈ {*n..*(*2*∗*n*+*2*)} ∧ *prime x*)

The term *prime-above n* returns a prime between n and $2 * n + 2$. Because of Bertrand's postulate there always is such a value. In a refinement of the algorithms, it may make sense to replace this with an algorithm, that finds such a prime exactly or approximately.

The definition is intentionally inexact, to allow refinement with various algorithms, without modifying the high-level mathematical correctness proof.

lemma *ex-subset*:
assumes ∃ *x* ∈ *A*. *P x*
assumes *A* ⊆ *B*
shows ∃ *x* ∈ *B*. *P x*
⟨*proof*⟩

lemma
shows *prime-above-prime*: *prime* (*prime-above n*)
and *prime-above-range*: *prime-above n* ∈ {*n..*(*2*∗*n*+*2*)}
⟨*proof*⟩

lemma *prime-above-min*: *prime-above n* ≥ *2*
⟨*proof*⟩

lemma *prime-above-lower-bound*: *prime-above n* ≥ *n*
⟨*proof*⟩

lemma *prime-above-upper-bound*: *prime-above n* ≤ *2*∗*n*+*2*
⟨*proof*⟩

end

2 Frequency Moments

theory *Frequency-Moments*
imports
Frequency-Moments-Preliminary-Results
Finite-Fields.Finite-Fields-Mod-Ring-Code
Interpolation-Polynomials-HOL-Algebra.Interpolation-Polynomial-Cardinalities
begin

This section contains a definition of the frequency moments of a stream and a few general results about frequency moments..

definition *F* **where**
F k xs = (∑ *x* ∈ *set xs*. (*rat-of-nat* (*count-list xs x*)^*k*))

lemma *F-ge-0*: *F k as* ≥ *0*

⟨*proof*⟩

lemma *F-gr-0*:
assumes $as \neq []$
shows $F\ k\ as > 0$
⟨*proof*⟩

definition P_e :: *nat* ⇒ *nat* ⇒ *nat list* ⇒ *bool list option* **where**
P_e *p n f* = (*if p* > *1* ∧ *f* ∈ *bounded-degree-polynomials* (*ring-of* (*mod-ring p*)) *n then*
([*0*..<*n*] $\rightarrow_e$ Nb_e *p*) (λ*i* ∈ {..<*n*}. *ring.coeff* (*ring-of* (*mod-ring p*)) *f i*) *else None*)

lemma *poly-encoding*:
is-encoding (P_e *p n*)
⟨*proof*⟩

lemma *bounded-degree-polynomial-bit-count*:
assumes *p* > *1*
assumes *x* ∈ *bounded-degree-polynomials* (*ring-of* (*mod-ring p*)) *n*
shows *bit-count* (P_e *p n x*) ≤ *ereal* (*real n* ∗ (*log 2 p* + *1*))
⟨*proof*⟩

end

3 Ranks, *k* smallest element and elements

theory *K-Smallest*
imports
Frequency-Moments-Preliminary-Results
Interpolation-Polynomials-HOL-Algebra.Interpolation-Polynomial-Cardinalities
begin

This section contains definitions and results for the selection of the *k* smallest elements, the *k*-th smallest element, rank of an element in an ordered set.

definition *rank-of* :: *'a* :: *linorder* ⇒ *'a set* ⇒ *nat* **where** *rank-of x S* = *card* {*y* ∈ *S*. *y* < *x*}

The function *rank-of* returns the rank of an element within a set.

lemma *rank-mono*:
assumes *finite S*
shows *x* ≤ *y* ⟹ *rank-of x S* ≤ *rank-of y S*
⟨*proof*⟩

lemma *rank-mono-2*:
assumes *finite S*
shows *S'* ⊆ *S* ⟹ *rank-of x S'* ≤ *rank-of x S*
⟨*proof*⟩

lemma *rank-mono-commute*:
  assumes *finite S*
  assumes $S \subseteq T$
  assumes *strict-mono-on T f*
  assumes $x \in T$
  shows *rank-of x S* = *rank-of* (*f x*) (*f* ‘ *S*)
⟨*proof*⟩

definition *least* **where** *least k S* = {*y* ∈ *S*. *rank-of y S* < *k*}

The function *K-Smallest.least* returns the k smallest elements of a finite set.

lemma *rank-strict-mono*:
  assumes *finite S*
  shows *strict-mono-on S* (λ*x*. *rank-of x S*)
⟨*proof*⟩

lemma *rank-of-image*:
  assumes *finite S*
  shows (λ*x*. *rank-of x S*) ‘ *S* = {*0*..<*card S*}
⟨*proof*⟩

lemma *card-least*:
  assumes *finite S*
  shows *card* (*least k S*) = *min k* (*card S*)
⟨*proof*⟩

lemma *least-subset*: *least k S* ⊆ *S*
  ⟨*proof*⟩

lemma *least-mono-commute*:
  assumes *finite S*
  assumes *strict-mono-on S f*
  shows *f* ‘ *least k S* = *least k* (*f* ‘ *S*)
⟨*proof*⟩

lemma *least-eq-iff*:
  assumes *finite B*
  assumes $A \subseteq B$
  assumes $\bigwedge x.\ x \in B \Longrightarrow$ *rank-of x B* $< k \Longrightarrow x \in A$
  shows *least k A* = *least k B*
⟨*proof*⟩

lemma *least-insert*:
  assumes *finite S*
  shows *least k* (*insert x* (*least k S*)) = *least k* (*insert x S*) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

definition *count-le* **where** *count-le x M* = *size* {#*y* ∈# *M*. *y* ≤ *x*#}
definition *count-less* **where** *count-less x M* = *size* {#*y* ∈# *M*. *y* < *x*#}

definition *nth-mset* :: *nat* ⇒ (*'a* :: *linorder*) *multiset* ⇒ *'a* **where**
  nth-mset k M = *sorted-list-of-multiset M* ! *k*

lemma *nth-mset-bound-left*:
  assumes *k* < *size M*
  assumes *count-less x M* ≤ *k*
  shows *x* ≤ *nth-mset k M*
⟨*proof*⟩

lemma *nth-mset-bound-left-excl*:
  assumes *k* < *size M*
  assumes *count-le x M* ≤ *k*
  shows *x* < *nth-mset k M*
⟨*proof*⟩

lemma *nth-mset-bound-right*:
  assumes *k* < *size M*
  assumes *count-le x M* > *k*
  shows *nth-mset k M* ≤ *x*
⟨*proof*⟩

lemma *nth-mset-commute-mono*:
  assumes *mono f*
  assumes *k* < *size M*
  shows *f* (*nth-mset k M*) = *nth-mset k* (*image-mset f M*)
⟨*proof*⟩

lemma *nth-mset-max*:
  assumes *size A* > *k*
  assumes ⋀*x*. *x* ≤ *nth-mset k A* ⟹ *count A x* ≤ *1*
  shows *nth-mset k A* = *Max* (*least* (*k*+*1*) (*set-mset A*)) **and** *card* (*least* (*k*+*1*) (*set-mset A*)) = *k*+*1*
⟨*proof*⟩

end

4 Landau Symbols

theory *Landau-Ext*
  imports
    HOL−Library.Landau-Symbols
    HOL.Topological-Spaces
begin

This section contains results about Landau Symbols in addition to "HOL-Library.Landau".

lemma *landau-sum*:
  assumes *eventually* (λ*x*. *g1 x* ≥ (*0*::*real*)) *F*
  assumes *eventually* (λ*x*. *g2 x* ≥ *0*) *F*
  assumes *f1* ∈ *O*[*F*](*g1*)
  assumes *f2* ∈ *O*[*F*](*g2*)
  shows (λ*x*. *f1 x* + *f2 x*) ∈ *O*[*F*](λ*x*. *g1 x* + *g2 x*)
⟨*proof*⟩

lemma *landau-sum-1*:
  assumes *eventually* (λ*x*. *g1 x* ≥ (*0*::*real*)) *F*
  assumes *eventually* (λ*x*. *g2 x* ≥ *0*) *F*
  assumes *f* ∈ *O*[*F*](*g1*)
  shows *f* ∈ *O*[*F*](λ*x*. *g1 x* + *g2 x*)
⟨*proof*⟩

lemma *landau-sum-2*:
  assumes *eventually* (λ*x*. *g1 x* ≥ (*0*::*real*)) *F*
  assumes *eventually* (λ*x*. *g2 x* ≥ *0*) *F*
  assumes *f* ∈ *O*[*F*](*g2*)
  shows *f* ∈ *O*[*F*](λ*x*. *g1 x* + *g2 x*)
⟨*proof*⟩

lemma *landau-ln-3*:
  assumes *eventually* (λ*x*. (*1*::*real*) ≤ *f x*) *F*
  assumes *f* ∈ *O*[*F*](*g*)
  shows (λ*x*. *ln* (*f x*)) ∈ *O*[*F*](*g*)
⟨*proof*⟩

lemma *landau-ln-2*:
  assumes *a* > (*1*::*real*)
  assumes *eventually* (λ*x*. *1* ≤ *f x*) *F*
  assumes *eventually* (λ*x*. *a* ≤ *g x*) *F*
  assumes *f* ∈ *O*[*F*](*g*)
  shows (λ*x*. *ln* (*f x*)) ∈ *O*[*F*](λ*x*. *ln* (*g x*))
⟨*proof*⟩

lemma *landau-real-nat*:
  fixes *f* :: *'a* ⇒ *int*
  assumes (λ*x*. *of-int* (*f x*)) ∈ *O*[*F*](*g*)
  shows (λ*x*. *real* (*nat* (*f x*))) ∈ *O*[*F*](*g*)
⟨*proof*⟩

lemma *landau-ceil*:
  assumes (λ-. *1*) ∈ *O*[*F'*](*g*)
  assumes *f* ∈ *O*[*F'*](*g*)
  shows (λ*x*. *real-of-int* ⌈*f x*⌉) ∈ *O*[*F'*](*g*)
⟨*proof*⟩

lemma *landau-rat-ceil*:

assumes $(\lambda\text{-}.\ 1) \in O[F'](g)$
assumes $(\lambda x.\ \textit{real-of-rat}\ (f\ x)) \in O[F'](g)$
shows $(\lambda x.\ \textit{real-of-int}\ \lceil f\ x \rceil) \in O[F'](g)$
⟨*proof*⟩

lemma *landau-nat-ceil*:
assumes $(\lambda\text{-}.\ 1) \in O[F'](g)$
assumes $f \in O[F'](g)$
shows $(\lambda x.\ \textit{real}\ (\textit{nat}\ \lceil f\ x \rceil)) \in O[F'](g)$
⟨*proof*⟩

lemma *eventually-prod1'*:
assumes $B \neq \textit{bot}$
assumes $(\forall_F\ x\ \textit{in}\ A.\ P\ x)$
shows $(\forall_F\ x\ \textit{in}\ A \times_F B.\ P\ (\textit{fst}\ x))$
⟨*proof*⟩

lemma *eventually-prod2'*:
assumes $A \neq \textit{bot}$
assumes $(\forall_F\ x\ \textit{in}\ B.\ P\ x)$
shows $(\forall_F\ x\ \textit{in}\ A \times_F B.\ P\ (\textit{snd}\ x))$
⟨*proof*⟩

lemma *sequentially-inf*: $\forall_F\ x\ \textit{in sequentially}.\ n \leq \textit{real}\ x$
⟨*proof*⟩

instantiation *rat* :: *linorder-topology*
begin

definition *open-rat* :: *rat set* ⇒ *bool*
where $\textit{open-rat} = \textit{generate-topology}\ (\textit{range}\ (\lambda a.\ \{..< a\}) \cup \textit{range}\ (\lambda a.\ \{a <..\}))$

instance
⟨*proof*⟩
end

lemma *inv-at-right-0-inf*:
$\forall_F\ x\ \textit{in at-right}\ 0.\ c \leq 1\ /\ \textit{real-of-rat}\ x$
⟨*proof*⟩

end

5 Probability Spaces

Some additional results about probability spaces in addition to "HOL-Probability".

theory *Probability-Ext*
imports
HOL−Probability.Stream-Space

Concentration-Inequalities.Bienaymes-Identity
Universal-Hash-Families.Carter-Wegman-Hash-Family
Frequency-Moments-Preliminary-Results
begin

context *prob-space*
begin

lemma *pmf-mono*:
assumes $M = measure\text{-}pmf\ p$
assumes $\bigwedge x.\ x \in P \Longrightarrow x \in set\text{-}pmf\ p \Longrightarrow x \in Q$
shows $prob\ P \leq prob\ Q$
⟨*proof*⟩

lemma *pmf-add*:
assumes $M = measure\text{-}pmf\ p$
assumes $\bigwedge x.\ x \in P \Longrightarrow x \in set\text{-}pmf\ p \Longrightarrow x \in Q \vee x \in R$
shows $prob\ P \leq prob\ Q + prob\ R$
⟨*proof*⟩

lemma *pmf-add-2*:
assumes $M = measure\text{-}pmf\ p$
assumes $prob\ \{\omega.\ P\ \omega\} \leq r1$
assumes $prob\ \{\omega.\ Q\ \omega\} \leq r2$
shows $prob\ \{\omega.\ P\ \omega \vee Q\ \omega\} \leq r1 + r2$ (**is** $?lhs \leq ?rhs$)
⟨*proof*⟩

end

end

6 Frequency Moment 0

theory *Frequency-Moment-0*
imports
Frequency-Moments-Preliminary-Results
Median-Method.Median
K-Smallest
Universal-Hash-Families.Carter-Wegman-Hash-Family
Frequency-Moments
Landau-Ext
Probability-Ext
Universal-Hash-Families.Universal-Hash-Families-More-Product-PMF
begin

This section contains a formalization of a new algorithm for the zero-th frequency moment inspired by ideas described in [2]. It is a KMV-type (k-minimum value) algorithm with a rounding method and matches the space complexity of the best algorithm described in [2].

In addition to the Isabelle proof here, there is also an informal hand-written proof in Appendix A.

type-synonym *f0-state* = *nat* × *nat* × *nat* × *nat* × (*nat* ⇒ *nat list*) × (*nat* ⇒ *float set*)

definition *hash* **where** *hash p* = *ring.hash* (*ring-of* (*mod-ring p*))

fun *f0-init* :: *rat* ⇒ *rat* ⇒ *nat* ⇒ *f0-state pmf* **where**
f0-init δ ε n =
do {
let s = *nat* ⌈−*18* ∗ *ln* (*real-of-rat ε*)⌉;
let t = *nat* ⌈*80* / (*real-of-rat δ*)^2⌉;
let p = *prime-above* (*max n 19*);
let r = *nat* (*4* ∗ ⌈*log 2* (*1* / *real-of-rat δ*)⌉ + *23*);
h ← *prod-pmf* {..<*s*} (λ-. *pmf-of-set* (*bounded-degree-polynomials* (*ring-of* (*mod-ring p*)) *2*));
return-pmf (*s*, *t*, *p*, *r*, *h*, (λ- ∈ {*0*..<*s*}. {}))
}

fun *f0-update* :: *nat* ⇒ *f0-state* ⇒ *f0-state pmf* **where**
f0-update x (*s*, *t*, *p*, *r*, *h*, *sketch*) =
return-pmf (*s*, *t*, *p*, *r*, *h*, λ*i* ∈ {..<*s*}.
least t (*insert* (*float-of* (*truncate-down r* (*hash p x* (*h i*)))) (*sketch i*)))

fun *f0-result* :: *f0-state* ⇒ *rat pmf* **where**
f0-result (*s*, *t*, *p*, *r*, *h*, *sketch*) = *return-pmf* (*median s* (λ*i* ∈ {..<*s*}.
(*if card* (*sketch i*) < *t then of-nat* (*card* (*sketch i*)) *else*
rat-of-nat t∗ *rat-of-nat p* / *rat-of-float* (*Max* (*sketch i*)))
))

fun *f0-space-usage* :: (*nat* × *rat* × *rat*) ⇒ *real* **where**
f0-space-usage (*n*, *ε*, *δ*) = (
let s = *nat* ⌈−*18* ∗ *ln* (*real-of-rat ε*)⌉ *in*
let r = *nat* (*4* ∗ ⌈*log 2* (*1* / *real-of-rat δ*)⌉ + *23*) *in*
let t = *nat* ⌈*80* / (*real-of-rat δ*)^2 ⌉ *in*
6 +
2 ∗ *log 2* (*real s* + *1*) +
2 ∗ *log 2* (*real t* + *1*) +
2 ∗ *log 2* (*real n* + *21*) +
2 ∗ *log 2* (*real r* + *1*) +
real s ∗ (*5* + *2* ∗ *log 2* (*21* + *real n*) +
real t ∗ (*13* + *4* ∗ *r* + *2* ∗ *log 2* (*log 2* (*real n* + *13*)))))

definition *encode-f0-state* :: *f0-state* ⇒ *bool list option* **where**
encode-f0-state =
N_e ⋈$_e$ (λ*s*.
N_e ×$_e$ (
N_e ⋈$_e$ (λ*p*.
N_e ×$_e$ (

$([0..<s] \rightarrow_e (P_e\ p\ 2)) \times_e$
$([0..<s] \rightarrow_e (S_e\ F_e))))))$

lemma *inj-on encode-f0-state* (*dom encode-f0-state*)
⟨*proof*⟩

context
fixes ε δ :: *rat*
fixes *n* :: *nat*
fixes *as* :: *nat list*
fixes *result*
assumes ε-*range*: $\varepsilon \in \{0<..<1\}$
assumes δ-*range*: $\delta \in \{0<..<1\}$
assumes *as-range*: *set as* $\subseteq \{..<n\}$
defines *result* $\equiv$ *fold* (λ*a state. state* ⤜ *f0-update a*) *as* (*f0-init* δ ε *n*) ⤜ *f0-result*
begin

private definition *t* **where** $t = nat \lceil 80 / (real\text{-}of\text{-}rat\ \delta)^2 \rceil$
private lemma *t-gt-0*: $t > 0$ ⟨*proof*⟩ **definition** *s* **where** $s = nat \lceil -(18 * ln\ (real\text{-}of\text{-}rat\ \varepsilon)) \rceil$
private lemma *s-gt-0*: $s > 0$ ⟨*proof*⟩ **definition** *p* **where** *p* = *prime-above* (*max n 19*)

private lemma *p-prime*:*Factorial-Ring.prime p*
⟨*proof*⟩ **lemma** *p-ge-18*: $p \geq 18$
⟨*proof*⟩ **lemma** *p-gt-0*: $p > 0$ ⟨*proof*⟩ **lemma** *p-gt-1*: $p > 1$ ⟨*proof*⟩ **lemma** *n-le-p*: $n \leq p$
⟨*proof*⟩ **lemma** *p-le-n*: $p \leq 2*n + 40$
⟨*proof*⟩ **lemma** *as-lt-p*: $\bigwedge x.\ x \in set\ as \Longrightarrow x < p$
⟨*proof*⟩ **lemma** *as-subset-p*: *set as* $\subseteq \{..<p\}$
⟨*proof*⟩ **definition** *r* **where** $r = nat\ (4 * \lceil log\ 2\ (1 / real\text{-}of\text{-}rat\ \delta) \rceil + 23)$

private lemma *r-bound*: $4 * log\ 2\ (1 / real\text{-}of\text{-}rat\ \delta) + 23 \leq r$
⟨*proof*⟩ **lemma** *r-ge-23*: $r \geq 23$
⟨*proof*⟩ **lemma** *two-pow-r-le-1*: $0 < 1 - 2\ powr - real\ r$
⟨*proof*⟩

interpretation *carter-wegman-hash-family ring-of* (*mod-ring p*) *2*
rewrites *ring.hash* (*ring-of* (*mod-ring p*)) = *Frequency-Moment-0.hash p*
⟨*proof*⟩ **definition** *tr-hash* **where** *tr-hash x* ω = *truncate-down r* (*hash x* ω)

private definition *sketch-rv* **where**
sketch-rv ω = *least t* ((λ*x. float-of* (*tr-hash x* ω)) ‘ *set as*)

private definition *estimate*
where *estimate S* = (*if card S* < *t then of-nat* (*card S*) *else of-nat t* ∗ *of-nat p* / *rat-of-float* (*Max S*))

private definition *sketch-rv'* **where** *sketch-rv'* ω = *least t* ((λx. *tr-hash x* ω) ‘ *set as*)
private definition *estimate'* **where** *estimate' S* = (*if card S < t then real* (*card S*) *else real t* $*$ *real p* / *Max S*)

private definition Ω_0 **where** Ω_0 = *prod-pmf* {..<*s*} (λ-. *pmf-of-set space*)

private lemma *f0-alg-sketch*:
  defines *sketch* $\equiv$ *fold* (λa *state. state* $\gg=$ *f0-update a*) *as* (*f0-init* δ ε *n*)
  shows *sketch* = *map-pmf* (λx. (*s*,*t*,*p*,*r*, *x*, $\lambda i \in$ {..<*s*}. *sketch-rv* (*x i*))) Ω_0
  ⟨*proof*⟩ **lemma** *card-nat-in-ball*:
  fixes *x* :: *nat*
  fixes *q* :: *real*
  assumes $q \geq 0$
  defines $A \equiv$ {*k*. *abs* (*real x* − *real k*) $\leq q \wedge k \neq x$}
  shows *real* (*card A*) $\leq 2 * q$ **and** *finite A*
⟨*proof*⟩ **lemma** *prob-degree-lt-1*:
  prob {ω. *degree* $\omega < 1$} $\leq 1/$*real p*
⟨*proof*⟩ **lemma** *collision-prob*:
  assumes $c \geq 1$
  shows *prob* {ω. $\exists x \in$ *set as*. $\exists y \in$ *set as*. $x \neq y \wedge$ *tr-hash x* $\omega \leq c \wedge$ *tr-hash x* ω = *tr-hash y* ω} $\leq$
    $(5/2) * (\mathit{real}\ (\mathit{card}\ (\mathit{set}\ \mathit{as})))^2 * c^2 * 2\ \mathit{powr}\ -(\mathit{real}\ r)\ /\ (\mathit{real}\ p)^2 + 1/\mathit{real}\ p$
(**is** *prob* {ω. *?l* ω} $\leq$ *?r1* + *?r2*)
⟨*proof*⟩ **lemma** *of-bool-square*: $(\mathit{of\text{-}bool}\ x)^2$ = ((*of-bool x*)::*real*)
  ⟨*proof*⟩ **definition** *Q* **where** *Q y* ω = *card* {$x \in$ *set as*. *int* (*hash x* ω) < *y*}

private definition *m* **where** *m* = *card* (*set as*)

private lemma
  assumes $a \geq 0$
  assumes $a \leq$ *int p*
  shows *exp-Q*: *expectation* ($\lambda\omega$. *real* (*Q a* ω)) = *real m* $*$ (*of-int a*) / *p*
  and *var-Q*: *variance* ($\lambda\omega$. *real* (*Q a* ω)) $\leq$ *real m* $*$ (*of-int a*) / *p*
⟨*proof*⟩ **lemma** *t-bound*: $t \leq 81\ /\ (\mathit{real\text{-}of\text{-}rat}\ \delta)^2$
⟨*proof*⟩ **lemma** *t-r-bound*:
  $18 * 40 * (\mathit{real}\ t)^2 * 2\ \mathit{powr}\ (-\mathit{real}\ r) \leq 1$
⟨*proof*⟩ **lemma** *m-eq-F-0*: *real m* = *of-rat* (*F 0 as*)
  ⟨*proof*⟩ **lemma** *estimate'-bounds*:
  prob {ω. *of-rat* δ $*$ *real-of-rat* (*F 0 as*) < |*estimate'* (*sketch-rv'* ω) − *of-rat* (*F 0 as*)|} $\leq 1/3$
⟨*proof*⟩ **lemma** *median-bounds*:
  $\mathcal{P}$(ω *in measure-pmf* Ω_0. |*median s* (λi. *estimate* (*sketch-rv* (ω *i*))) − *F 0 as*| $\leq$ δ $*$ *F 0 as*) $\geq$ 1 − *real-of-rat* ε
⟨*proof*⟩

lemma *f0-alg-correct'*:
  $\mathcal{P}$(ω *in measure-pmf result*. |ω − *F 0 as*| $\leq$ δ $*$ *F 0 as*) $\geq$ 1 − *of-rat* ε
⟨*proof*⟩ **lemma** *f-subset*:

assumes g ‘ $A \subseteq h$ ‘ B
shows $(\lambda x.\ f\ (g\ x))$ ‘ $A \subseteq (\lambda x.\ f\ (h\ x))$ ‘ B
⟨*proof*⟩

lemma *f0-exact-space-usage′*:
defines $\Omega \equiv$ *fold* (λa *state. state* $\gg=$ *f0-update a*) *as* (*f0-init* $\delta\ \varepsilon\ n$)
shows *AE* ω *in* Ω. *bit-count* (*encode-f0-state* ω) $\leq$ *f0-space-usage* (n, ε, δ)
⟨*proof*⟩

end

Main results of this section:

theorem *f0-alg-correct*:
assumes $\varepsilon \in \{0<..<1\}$
assumes $\delta \in \{0<..<1\}$
assumes *set as* $\subseteq \{..<n\}$
defines $\Omega \equiv$ *fold* (λa *state. state* $\gg=$ *f0-update a*) *as* (*f0-init* $\delta\ \varepsilon\ n$) $\gg=$ *f0-result*
shows $\mathcal{P}(\omega$ *in measure-pmf* Ω. $|\omega - F\ 0\ as| \leq \delta * F\ 0\ as) \geq 1 -$ *of-rat* ε
⟨*proof*⟩

theorem *f0-exact-space-usage*:
assumes $\varepsilon \in \{0<..<1\}$
assumes $\delta \in \{0<..<1\}$
assumes *set as* $\subseteq \{..<n\}$
defines $\Omega \equiv$ *fold* (λa *state. state* $\gg=$ *f0-update a*) *as* (*f0-init* $\delta\ \varepsilon\ n$)
shows *AE* ω *in* Ω. *bit-count* (*encode-f0-state* ω) $\leq$ *f0-space-usage* (n, ε, δ)
⟨*proof*⟩

theorem *f0-asymptotic-space-complexity*:
f0-space-usage $\in O[$*at-top* $\times_F$ *at-right 0* $\times_F$ *at-right 0*$](\lambda(n, \varepsilon, \delta).$ *ln* (*1* / *of-rat* ε) $*$
(*ln* (*real n*) + *1* / (*of-rat* $\delta)^2 *$ (*ln* (*ln* (*real n*)) + *ln* (*1* / *of-rat* δ))))
(**is** - $\in O[?F](?rhs)$)
⟨*proof*⟩

end

7 Frequency Moment 2

theory *Frequency-Moment-2*
imports
Universal-Hash-Families.Carter-Wegman-Hash-Family
Equivalence-Relation-Enumeration.Equivalence-Relation-Enumeration
Landau-Ext
Median-Method.Median
Probability-Ext
Universal-Hash-Families.Universal-Hash-Families-More-Product-PMF
Frequency-Moments
begin

hide-const (**open**) *Discrete-Topology.discrete*
hide-const (**open**) *Isolated.discrete*

This section contains a formalization of the algorithm for the second frequency moment. It is based on the algorithm described in [1, §2.2]. The only difference is that the algorithm is adapted to work with prime field of odd order, which greatly reduces the implementation complexity.

fun *f2-hash* **where**
f2-hash p h k = (*if even* (*ring.hash* (*ring-of* (*mod-ring p*)) *k h*) *then int p* − *1 else* − *int p* − *1*)

type-synonym *f2-state* = *nat* × *nat* × *nat* × (*nat* × *nat* ⇒ *nat list*) × (*nat* × *nat* ⇒ *int*)

fun *f2-init* :: *rat* ⇒ *rat* ⇒ *nat* ⇒ *f2-state pmf* **where**
f2-init δ ε *n* =
  do {
    let s_1 = *nat* $\lceil 6 / \delta^2 \rceil$;
    let s_2 = *nat* $\lceil -(18 * ln\ (real\text{-}of\text{-}rat\ \varepsilon)) \rceil$;
    let p = *prime-above* (*max n 3*);
    h ← *prod-pmf* ($\{..<s_1\} \times \{..<s_2\}$) (λ-. *pmf-of-set* (*bounded-degree-polynomials* (*ring-of* (*mod-ring p*)) *4*));
    return-pmf (s_1, s_2, *p*, *h*, (λ- ∈ $\{..<s_1\} \times \{..<s_2\}$. (*0* :: *int*)))
  }

fun *f2-update* :: *nat* ⇒ *f2-state* ⇒ *f2-state pmf* **where**
f2-update x (s_1, s_2, *p*, *h*, *sketch*) =
  return-pmf (s_1, s_2, *p*, *h*, λ*i* ∈ $\{..<s_1\} \times \{..<s_2\}$. *f2-hash p* (*h i*) *x* + *sketch i*)

fun *f2-result* :: *f2-state* ⇒ *rat pmf* **where**
f2-result (s_1, s_2, *p*, *h*, *sketch*) =
  return-pmf (*median* s_2 (λi_2 ∈ $\{..<s_2\}$.
    ($\sum i_1 \in \{..<s_1\}$. (*rat-of-int* (*sketch* (i_1, i_2)))^2) / (((*rat-of-nat p*)^2−*1*) * *rat-of-nat* s_1)))

fun *f2-space-usage* :: (*nat* × *nat* × *rat* × *rat*) ⇒ *real* **where**
f2-space-usage (*n*, *m*, ε, δ) = (
  let s_1 = *nat* $\lceil 6 / \delta^2 \rceil$ *in*
  let s_2 = *nat* $\lceil -(18 * ln\ (real\text{-}of\text{-}rat\ \varepsilon)) \rceil$ *in*
  3 +
  2 * *log 2* (s_1 + *1*) +
  2 * *log 2* (s_2 + *1*) +
  2 * *log 2* (*9* + *2* * *real n*) +
  s_1 * s_2 * (*5* + *4***log 2* (*8* + *2* * *real n*) + *2* * *log 2* (*real m* * (*18* + *4* * *real n*) + *1*)))

definition *encode-f2-state* :: *f2-state* ⇒ *bool list option* **where**
encode-f2-state =

$N_e \bowtie_e (\lambda s_1.$
$N_e \bowtie_e (\lambda s_2.$
$N_e \bowtie_e (\lambda p.$
$(List.product\ [0..<s_1]\ [0..<s_2] \rightarrow_e P_e\ p\ 4) \times_e$
$(List.product\ [0..<s_1]\ [0..<s_2] \rightarrow_e I_e))))$

lemma *inj-on encode-f2-state* (*dom encode-f2-state*)
⟨*proof*⟩

context
fixes $\varepsilon\ \delta$ :: *rat*
fixes n :: *nat*
fixes *as* :: *nat list*
fixes *result*
assumes *ε-range*: $\varepsilon \in \{0<..<1\}$
assumes *δ-range*: $\delta > 0$
assumes *as-range*: *set as* $\subseteq \{..<n\}$
defines *result* $\equiv$ *fold* (λa *state*. *state* $\ggg$ *f2-update a*) *as* (*f2-init* $\delta\ \varepsilon\ n$) $\ggg$ *f2-result*
begin

private definition s_1 **where** $s_1 = nat\ \lceil 6\ /\ \delta^2 \rceil$

lemma *s1-gt-0*: $s_1 > 0$
⟨*proof*⟩ **definition** s_2 **where** $s_2 = nat\ \lceil -(18 * \ln\ (real\text{-}of\text{-}rat\ \varepsilon)) \rceil$

lemma *s2-gt-0*: $s_2 > 0$
⟨*proof*⟩ **definition** p **where** $p = prime\text{-}above\ (max\ n\ 3)$

lemma *p-prime*: *Factorial-Ring.prime p*
⟨*proof*⟩

lemma *p-ge-3*: $p \geq 3$
⟨*proof*⟩

lemma *p-gt-0*: $p > 0$ ⟨*proof*⟩

lemma *p-gt-1*: $p > 1$ ⟨*proof*⟩

lemma *p-ge-n*: $p \geq n$ ⟨*proof*⟩

interpretation *carter-wegman-hash-family ring-of* (*mod-ring p*) *4*
⟨*proof*⟩

definition *sketch* **where** *sketch* = *fold* (λa *state*. *state* $\ggg$ *f2-update a*) *as* (*f2-init* $\delta\ \varepsilon\ n$)
private definition Ω **where**Ω = *prod-pmf* ($\{..<s_1\} \times \{..<s_2\}$) ($\lambda$-. *pmf-of-set space*)
private definition Ω_p **where**Ω_p = *measure-pmf* Ω

private definition *sketch-rv* **where** *sketch-rv* ω = *of-int* (*sum-list* (*map* (*f2-hash* *p* ω) *as*))^*2*
private definition *mean-rv* **where** *mean-rv* ω = (λi_2. ($\sum i_1$ = *0*..<s_1. *sketch-rv* (ω (i_1, i_2))) / (((*of-nat p*)^2 − *1*) ∗ *of-nat* s_1))
private definition *result-rv* **where** *result-rv* ω = *median* s_2 ($\lambda i_2 \in \{..<s_2\}$. *mean-rv* ω i_2)

lemma *mean-rv-alg-sketch*:
sketch = Ω ≫= ($\lambda\omega$. *return-pmf* (s_1, s_2, *p*, ω, $\lambda i \in \{..<s_1\} \times \{..<s_2\}$. *sum-list* (*map* (*f2-hash p* (ω *i*)) *as*)))
⟨*proof*⟩

lemma *distr*: *result* = *map-pmf result-rv* Ω
⟨*proof*⟩ **lemma** *f2-hash-pow-exp*:
assumes *k* < *p*
shows
expectation ($\lambda\omega$. *real-of-int* (*f2-hash p* ω *k*) ^*m*) =
((*real p* − *1*) ^ *m* ∗ (*real p* + *1*) + (− *real p* − *1*) ^ *m* ∗ (*real p* − *1*)) / (*2* ∗ *real p*)
⟨*proof*⟩

lemma
shows *var-sketch-rv*:*variance sketch-rv* ≤ *2*∗(*real-of-rat* (*F 2 as*)^*2*) ∗ ((*real p*)^2−*1*)^2 (**is** *?A*)
and *exp-sketch-rv*:*expectation sketch-rv* = *real-of-rat* (*F 2 as*) ∗ ((*real p*)^2−*1*) (**is** *?B*)
⟨*proof*⟩

lemma *space-omega-1* [*simp*]: *Sigma-Algebra.space* Ω_p = *UNIV*
⟨*proof*⟩

interpretation Ω: *prob-space* Ω_p
⟨*proof*⟩

lemma *integrable-*Ω:
fixes *f* :: ((*nat* × *nat*) ⇒ (*nat list*)) ⇒ *real*
shows *integrable* Ω_p *f*
⟨*proof*⟩

lemma *sketch-rv-exp*:
assumes $i_2 < s_2$
assumes $i_1 \in \{0..<s_1\}$
shows Ω.*expectation* ($\lambda\omega$. *sketch-rv* (ω (i_1, i_2))) = *real-of-rat* (*F 2 as*) ∗ ((*real p*)^2 − *1*)
⟨*proof*⟩

lemma *sketch-rv-var*:
assumes $i_2 < s_2$
assumes $i_1 \in \{0..<s_1\}$

shows Ω.*variance* ($\lambda\omega$. *sketch-rv* (ω (i_1, i_2))) $\leq$ *2* $*$ (*real-of-rat* (*F 2 as*))^2 $*$ ((*real p*)^2 $-$ *1*)^2
⟨*proof*⟩

lemma *mean-rv-exp*:
assumes $i < s_2$
shows Ω.*expectation* ($\lambda\omega$. *mean-rv* ω *i*) = *real-of-rat* (*F 2 as*)
⟨*proof*⟩

lemma *mean-rv-var*:
assumes $i < s_2$
shows Ω.*variance* ($\lambda\omega$. *mean-rv* ω *i*) $\leq$ (*real-of-rat* (δ $*$ *F 2 as*))^2 / *3*
⟨*proof*⟩

lemma *mean-rv-bounds*:
assumes $i < s_2$
shows Ω.*prob* {ω. *real-of-rat* δ $*$ *real-of-rat* (*F 2 as*) < |*mean-rv* ω *i* $-$ *real-of-rat* (*F 2 as*)|} $\leq$ *1/3*
⟨*proof*⟩

lemma *f2-alg-correct′*:
$\mathcal{P}$(ω *in measure-pmf result*. |ω $-$ *F 2 as*| $\leq$ δ $*$ *F 2 as*) $\geq$ *1*$-$*of-rat* ε
⟨*proof*⟩

lemma *f2-exact-space-usage′*:
AE ω *in sketch* . *bit-count* (*encode-f2-state* ω) $\leq$ *f2-space-usage* (*n*, *length as*, ε, δ)
⟨*proof*⟩

end

Main results of this section:

theorem *f2-alg-correct*:
assumes $\varepsilon \in \{0<..<1\}$
assumes $\delta > 0$
assumes *set as* $\subseteq$ {..<*n*}
defines $\Omega \equiv$ *fold* (λ*a state*. *state* ⤜ *f2-update a*) *as* (*f2-init* δ ε *n*) ⤜ *f2-result*
shows $\mathcal{P}$(ω *in measure-pmf* Ω. |ω $-$ *F 2 as*| $\leq$ δ $*$ *F 2 as*) $\geq$ *1*$-$*of-rat* ε
⟨*proof*⟩

theorem *f2-exact-space-usage*:
assumes $\varepsilon \in \{0<..<1\}$
assumes $\delta > 0$
assumes *set as* $\subseteq$ {..<*n*}
defines *M* $\equiv$ *fold* (λ*a state*. *state* ⤜ *f2-update a*) *as* (*f2-init* δ ε *n*)
shows *AE* ω *in M*. *bit-count* (*encode-f2-state* ω) $\leq$ *f2-space-usage* (*n*, *length as*, ε, δ)
⟨*proof*⟩

theorem *f2-asymptotic-space-complexity*:
f2-space-usage ∈ *O*[*at-top* $\times_F$ *at-top* $\times_F$ *at-right 0* $\times_F$ *at-right 0*](λ (*n*, *m*, ε, δ).
(*ln* (*1* / *of-rat* ε)) / (*of-rat* δ)^2 ∗ (*ln* (*real n*) + *ln* (*real m*)))
(**is** - ∈ *O*[*?F*](*?rhs*))
⟨*proof*⟩

end

8 Frequency Moment k

theory *Frequency-Moment-k*
imports
Frequency-Moments
Landau-Ext
Lp.Lp
Median-Method.Median
Probability-Ext
Universal-Hash-Families.Universal-Hash-Families-More-Product-PMF
begin

This section contains a formalization of the algorithm for the k-th frequency moment. It is based on the algorithm described in [1, §2.1].

type-synonym *fk-state* = *nat* × *nat* × *nat* × *nat* × (*nat* × *nat* ⇒ (*nat* × *nat*))

```
fun fk-init :: nat ⇒ rat ⇒ rat ⇒ nat ⇒ fk-state pmf where
  fk-init k δ ε n =
    do {
      let s1 = nat ⌈3 ∗ real k ∗ n powr (1−1/real k) / (real-of-rat δ)^2⌉;
      let s2 = nat ⌈−18 ∗ ln (real-of-rat ε)⌉;
      return-pmf (s1, s2, k, 0, (λ- ∈ {0..<s1} × {0..<s2}. (0,0)))
    }

fun fk-update :: nat ⇒ fk-state ⇒ fk-state pmf where
  fk-update a (s1, s2, k, m, r) =
    do {
     coins ← prod-pmf ({0..<s1} × {0..<s2}) (λ-. bernoulli-pmf (1/(real m+1)));
      return-pmf (s1, s2, k, m+1, λi ∈ {0..<s1} × {0..<s2}.
        if coins i then
          (a,0)
        else (
          let (x,l) = r i in (x, l + of-bool (x=a))
        )
      )
    }

fun fk-result :: fk-state ⇒ rat pmf where
  fk-result (s1, s2, k, m, r) =
    return-pmf (median s2 (λi2 ∈ {0..<s2}.
```

($\sum$ i_1∈{*0..<*s_1}. *rat-of-nat* (*let t* = *snd* (*r* (i_1, i_2)) + *1 in m* ∗ (*t^k* − (*t* − *1*)^*k*))) / (*rat-of-nat* s_1))
)

lemma *bernoulli-pmf-1*: *bernoulli-pmf 1* = *return-pmf True*
⟨*proof*⟩

fun *fk-space-usage* :: (*nat* × *nat* × *nat* × *rat* × *rat*) ⇒ *real* **where**
fk-space-usage (*k*, *n*, *m*, ε, δ) = (
let s_1 = *nat* ⌈*3*∗*real k*∗ (*real n*) *powr* (*1*−*1*/ *real k*) / (*real-of-rat* δ)^2 ⌉ *in*
let s_2 = *nat* ⌈−(*18* ∗ *ln* (*real-of-rat* ε))⌉ *in*
4 +
2 ∗ *log 2* (s_1 + *1*) +
2 ∗ *log 2* (s_2 + *1*) +
2 ∗ *log 2* (*real k* + *1*) +
2 ∗ *log 2* (*real m* + *1*) +
s_1 ∗ s_2 ∗ (*2* + *2* ∗ *log 2* (*real n*+*1*) + *2* ∗ *log 2* (*real m*+*1*)))

definition *encode-fk-state* :: *fk-state* ⇒ *bool list option* **where**
encode-fk-state =
N_e $\bowtie_e$ (λs_1.
N_e $\bowtie_e$ (λs_2.
N_e $\times_e$
N_e $\times_e$
(*List.product* [*0*..<s_1] [*0*..<s_2] $\rightarrow_e$ (N_e $\times_e$ N_e))))

lemma *inj-on encode-fk-state* (*dom encode-fk-state*)
⟨*proof*⟩

This is an intermediate non-parallel form *fk-update* used only in the correctness proof.

fun *fk-update-2* :: *'a* ⇒ (*nat* × *'a* × *nat*) ⇒ (*nat* × *'a* × *nat*) *pmf* **where**
fk-update-2 a (*m*,*x*,*l*) =
do {
coin ← *bernoulli-pmf* (*1*/(*real m*+*1*));
return-pmf (*m*+*1*,*if coin then* (*a*,*0*) *else* (*x*, *l* + *of-bool* (*x*=*a*)))
}

definition *sketch* **where** *sketch as i* = (*as* ! *i*, *count-list* (*drop* (*i*+*1*) *as*) (*as* ! *i*))

lemma *fk-update-2-distr*:
assumes *as* ≠ []
shows *fold* (λ*x s*. *s* ⤜ *fk-update-2 x*) *as* (*return-pmf* (*0*,*0*,*0*)) =
pmf-of-set {..<*length as*} ⤜ (λ*k*. *return-pmf* (*length as*, *sketch as k*))
⟨*proof*⟩

context
fixes ε δ :: *rat*
fixes *n k* :: *nat*

fixes *as*
assumes *k-ge-1*: $k \geq 1$
assumes *ε-range*: $\varepsilon \in \{0<..<1\}$
assumes *δ-range*: $\delta > 0$
assumes *as-range*: $set\ as \subseteq \{..<n\}$
begin

definition s_1 **where** $s_1 = nat\ \lceil 3 * real\ k * (real\ n)\ powr\ (1-1/real\ k)\ /\ (real\text{-}of\text{-}rat\ \delta)^2 \rceil$
definition s_2 **where** $s_2 = nat\ \lceil -(18 * ln\ (real\text{-}of\text{-}rat\ \varepsilon)) \rceil$

definition $M_1 = \{(u, v).\ v < count\text{-}list\ as\ u\}$
definition $\Omega_1 = measure\text{-}pmf\ (pmf\text{-}of\text{-}set\ M_1)$

definition $M_2 = prod\text{-}pmf\ (\{0..<s_1\} \times \{0..<s_2\})\ (\lambda\text{-}.\ pmf\text{-}of\text{-}set\ M_1)$
definition $\Omega_2 = measure\text{-}pmf\ M_2$

interpretation *prob-space* Ω_1
⟨*proof*⟩

interpretation Ω_2:*prob-space* Ω_2
⟨*proof*⟩

lemma *split-space*: $(\sum a \in M_1.\ f\ (snd\ a)) = (\sum u \in set\ as.\ (\sum v \in \{0..<count\text{-}list\ as\ u\}.\ f\ v))$
⟨*proof*⟩

lemma
assumes $as \neq []$
shows *fin-space*: $finite\ M_1$
and *non-empty-space*: $M_1 \neq \{\}$
and *card-space*: $card\ M_1 = length\ as$
⟨*proof*⟩

lemma
assumes $as \neq []$
shows *integrable-1*: $integrable\ \Omega_1\ (f :: - \Rightarrow real)$ **and**
integrable-2: $integrable\ \Omega_2\ (g :: - \Rightarrow real)$
⟨*proof*⟩

lemma *sketch-distr*:
assumes $as \neq []$
shows $pmf\text{-}of\text{-}set\ \{..<length\ as\} \gg\!= (\lambda k.\ return\text{-}pmf\ (sketch\ as\ k)) = pmf\text{-}of\text{-}set\ M_1$
⟨*proof*⟩

lemma *fk-update-distr*:
$fold\ (\lambda x\ s.\ s \gg\!= fk\text{-}update\ x)\ as\ (fk\text{-}init\ k\ \delta\ \varepsilon\ n) =$
$prod\text{-}pmf\ (\{0..<s_1\} \times \{0..<s_2\})\ (\lambda\text{-}.\ fold\ (\lambda x\ s.\ s \gg\!= fk\text{-}update\text{-}2\ x)\ as\ (return\text{-}pmf$

(0,0,0)))
  ≫= (λ*x*. *return-pmf* (*s*$_1$,*s*$_2$,*k*, *length as*, λ*i*∈{*0*..<*s*$_1$}×{*0*..<*s*$_2$}. *snd* (*x i*)))
⟨*proof*⟩

lemma *power-diff-sum*:
  fixes *a b* :: *'a* :: {*comm-ring-1*,*power*}
  assumes *k* > *0*
  shows *a*^*k* − *b*^*k* = (*a*−*b*) ∗ (∑ *i* = *0*..<*k*. *a* ^ *i* ∗ *b* ^(*k* − *1* − *i*)) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *power-diff-est*:
  assumes *k* > *0*
  assumes (*a* :: *real*) ≥ *b*
  assumes *b* ≥ *0*
  shows *a*^*k* −*b*^*k* ≤ (*a*−*b*) ∗ *k* ∗ *a*^(*k*−*1*)
⟨*proof*⟩

Specialization of the Hoelder inquality for sums.

lemma *Holder-inequality-sum*:
  assumes *p* > (*0*::*real*) *q* > *0 1*/*p* + *1*/*q* = *1*
  assumes *finite A*
  shows |∑ *x*∈*A*. *f x* ∗ *g x*| ≤ (∑ *x*∈*A*. |*f x*| *powr p*) *powr* (*1*/*p*) ∗ (∑ *x*∈*A*. |*g x*| *powr q*) *powr* (*1*/*q*)
⟨*proof*⟩

lemma *real-count-list-pos*:
  assumes *x* ∈ *set as*
  shows *real* (*count-list as x*) > *0*
  ⟨*proof*⟩

lemma *fk-estimate*:
  assumes *as* ≠ []
  shows *length as* ∗ *of-rat* (*F* (*2*∗*k*−*1*) *as*) ≤ *n powr* (*1* − *1* / *real k*) ∗ (*of-rat* (*F k as*))^*2*
  (**is** *?lhs* ≤ *?rhs*)
⟨*proof*⟩

definition *result*
  where *result a* = *of-nat* (*length as*) ∗ *of-nat* (*Suc* (*snd a*) ^ *k* − *snd a* ^ *k*)

lemma *result-exp-1*:
  assumes *as* ≠ []
  shows *expectation result* = *real-of-rat* (*F k as*)
⟨*proof*⟩

lemma *result-var-1*:
  assumes *as* ≠ []
  shows *variance result* ≤ (*of-rat* (*F k as*))^2 ∗ *k* ∗ *n powr* (*1* − *1* / *real k*)

⟨proof⟩

theorem *fk-alg-sketch*:
assumes $as \neq []$
shows $fold\ (\lambda a\ state.\ state \ggg= fk\text{-}update\ a)\ as\ (fk\text{-}init\ k\ \delta\ \varepsilon\ n) =$
$map\text{-}pmf\ (\lambda x.\ (s_1,s_2,k,length\ as,\ x))\ M_2$ (**is** *?lhs = ?rhs*)
⟨proof⟩

definition *mean-rv*
where $mean\text{-}rv\ \omega\ i_2 = (\sum i_1 = 0..<s_1.\ result\ (\omega\ (i_1,\ i_2)))\ /\ of\text{-}nat\ s_1$

definition *median-rv*
where $median\text{-}rv\ \omega = median\ s_2\ (\lambda i_2.\ mean\text{-}rv\ \omega\ i_2)$

lemma *fk-alg-correct'*:
defines $M \equiv fold\ (\lambda a\ state.\ state \ggg= fk\text{-}update\ a)\ as\ (fk\text{-}init\ k\ \delta\ \varepsilon\ n) \ggg= fk\text{-}result$
shows $\mathcal{P}(\omega\ in\ measure\text{-}pmf\ M.\ |\omega - F\ k\ as| \leq \delta * F\ k\ as) \geq 1 - of\text{-}rat\ \varepsilon$
⟨proof⟩

lemma *fk-exact-space-usage'*:
defines $M \equiv fold\ (\lambda a\ state.\ state \ggg= fk\text{-}update\ a)\ as\ (fk\text{-}init\ k\ \delta\ \varepsilon\ n)$
shows $AE\ \omega\ in\ M.\ bit\text{-}count\ (encode\text{-}fk\text{-}state\ \omega) \leq fk\text{-}space\text{-}usage\ (k,\ n,\ length\ as,\ \varepsilon,\ \delta)$
(**is** $AE\ \omega\ in\ M.\ (\text{-} \leq ?rhs)$)
⟨proof⟩

end

Main results of this section:

theorem *fk-alg-correct*:
assumes $k \geq 1$
assumes $\varepsilon \in \{0<..<1\}$
assumes $\delta > 0$
assumes $set\ as \subseteq \{..<n\}$
defines $M \equiv fold\ (\lambda a\ state.\ state \ggg= fk\text{-}update\ a)\ as\ (fk\text{-}init\ k\ \delta\ \varepsilon\ n) \ggg= fk\text{-}result$
shows $\mathcal{P}(\omega\ in\ measure\text{-}pmf\ M.\ |\omega - F\ k\ as| \leq \delta * F\ k\ as) \geq 1 - of\text{-}rat\ \varepsilon$
⟨proof⟩

theorem *fk-exact-space-usage*:
assumes $k \geq 1$
assumes $\varepsilon \in \{0<..<1\}$
assumes $\delta > 0$
assumes $set\ as \subseteq \{..<n\}$
defines $M \equiv fold\ (\lambda a\ state.\ state \ggg= fk\text{-}update\ a)\ as\ (fk\text{-}init\ k\ \delta\ \varepsilon\ n)$
shows $AE\ \omega\ in\ M.\ bit\text{-}count\ (encode\text{-}fk\text{-}state\ \omega) \leq fk\text{-}space\text{-}usage\ (k,\ n,\ length\ as,\ \varepsilon,\ \delta)$
⟨proof⟩

theorem *fk-asymptotic-space-complexity*:

fk-space-usage ∈
O[*at-top* $\times_F$ *at-top* $\times_F$ *at-top* $\times_F$ *at-right* (*0*::*rat*) $\times_F$ *at-right* (*0*::*rat*)](λ (*k*, *n*, *m*, ε, δ).
real k ∗ *real n powr* (*1*−*1*/ *real k*) / (*of-rat* δ)^2 ∗ (*ln* (*1* / *of-rat* ε)) ∗ (*ln* (*real n*) + *ln* (*real m*)))
(**is** - ∈ *O*[*?F*](*?rhs*))
⟨*proof*⟩

end

9 Tutorial on the use of Pseudorandom-Objects

theory *Tutorial-Pseudorandom-Objects*
imports
Universal-Hash-Families.Pseudorandom-Objects-Hash-Families
Expander-Graphs.Pseudorandom-Objects-Expander-Walks
Equivalence-Relation-Enumeration.Equivalence-Relation-Enumeration
Median-Method.Median
Concentration-Inequalities.Bienaymes-Identity
Frequency-Moments.Frequency-Moments
begin

This section is a tutorial for the use of pseudorandom objects. Starting from the approximation algorithm for the second frequency moment by Alon et al. [1], we will improve the solution until we achieve a space complexity of $\mathcal{O}(\ln n + \varepsilon^{-2} \ln(\delta^{-1}) \ln m)$, where n denotes the range of the stream elements, m denotes the length of the stream, ε denotes the desired accuracy and δ denotes the desired failure probability.

The construction relies on a combination of pseudorandom object, in particular an expander walk and two chained hash families.

hide-const (**open**) *topological-space-class.discrete*
hide-const (**open**) *Abstract-Rewriting.restrict*
hide-fact (**open**) *Abstract-Rewriting.restrict-def*
hide-fact (**open**) *Henstock-Kurzweil-Integration.integral-cong*
hide-fact (**open**) *Henstock-Kurzweil-Integration.integral-mult-right*
hide-fact (**open**) *Henstock-Kurzweil-Integration.integral-diff*

The following lemmas show a one-side and two-sided Chernoff-bound for $\{0, 1\}$-valued independent identically distributed random variables. This to show the similarity with expander walks, for which similar bounds can be established: *expander-chernoff-bound-one-sided* and *expander-chernoff-bound*.

lemma *classic-chernoff-bound-one-sided*:
fixes *l* :: *nat*
assumes *AE x in measure-pmf p. f x* ∈ {*0*,*1*::*real*}
assumes (∫ *x*. *f x* ∂*p*) ≤ μ *l* > *0* γ ≥ *0*
shows *measure* (*prod-pmf* {*0*..<*l*} (λ-. *p*)) {*w*. (∑ *i*<*l*. *f* (*w i*))/*l*−μ≥γ} ≤ *exp* (− *2* ∗ *real l* ∗ γ^*2*)

(**is** $?L \le ?R$)
⟨*proof*⟩

lemma *classic-chernoff-bound*:
assumes $AE\ x\ in\ measure\text{-}pmf\ p.\ f\ x \in \{0,1::real\}\ l > (0::nat)\ \gamma \ge 0$
defines $\mu \equiv (\int x.\ f\ x\ \partial p)$
shows $measure\ (prod\text{-}pmf\ \{0..<l\}\ (\lambda\text{-}.\ p))\ \{w.\ |(\sum i<l.\ f\ (w\ i))/l-\mu| \ge \gamma\} \le 2*exp\ (-2*real\ l*\gamma\hat{\ }2)$
(**is** $?L \le ?R$)
⟨*proof*⟩

Definition of the second frequency moment of a stream.

definition $F2 :: 'a\ list \Rightarrow real$ **where**
$F2\ xs = (\sum\ x \in set\ xs.\ (of\text{-}nat\ (count\text{-}list\ xs\ x)\hat{\ }2))$

lemma *prime-power-ls*: $is\text{-}prime\text{-}power\ (pro\text{-}size\ (\mathcal{L}\ [-\ 1,\ 1]))$
⟨*proof*⟩

lemma *prime-power-h2*: $is\text{-}prime\text{-}power\ (pro\text{-}size\ (\mathcal{H}\ 4\ n\ (\mathcal{L}\ [-\ 1,\ 1::real])))$
⟨*proof*⟩

abbreviation Ψ **where** $\Psi \equiv pmf\text{-}of\text{-}set\ \{-1,1::real\}$

lemma *f2-exp*:
assumes $finite\ (set\text{-}pmf\ p)$
assumes $\bigwedge I.\ I \subseteq \{0..<n\} \Longrightarrow card\ I \le 4 \Longrightarrow map\text{-}pmf\ (\lambda x.\ (\lambda i\in I.\ x\ i))\ p = prod\text{-}pmf\ I\ (\lambda\text{-}.\ \Psi)$
assumes $set\ xs \subseteq \{0..<n::nat\}$
shows $(\int h.\ (\sum x \leftarrow xs.\ h\ x)\hat{\ }2\ \partial p) = F2\ xs$ (**is** $?L = ?R$)
⟨*proof*⟩

lemma *f2-exp-sq*:
assumes $finite\ (set\text{-}pmf\ p)$
assumes $\bigwedge I.\ I \subseteq \{0..<n\} \Longrightarrow card\ I \le 4 \Longrightarrow map\text{-}pmf\ (\lambda x.\ (\lambda i\in I.\ x\ i))\ p = prod\text{-}pmf\ I\ (\lambda\text{-}.\ \Psi)$
assumes $set\ xs \subseteq \{0..<n::nat\}$
shows $(\int h.\ ((\sum x\leftarrow xs.\ h\ x)\hat{\ }2)\hat{\ }2\ \partial p) \le 3 * F2\ xs\hat{\ }2$ (**is** $?L \le ?R$)
⟨*proof*⟩

lemma *f2-var*:
assumes $finite\ (set\text{-}pmf\ p)$
assumes $\bigwedge I.\ I \subseteq \{0..<n\} \Longrightarrow card\ I \le 4 \Longrightarrow map\text{-}pmf\ (\lambda x.\ (\lambda i\in I.\ x\ i))\ p = prod\text{-}pmf\ I\ (\lambda\text{-}.\ \Psi)$
assumes $set\ xs \subseteq \{0..<n::nat\}$
shows $measure\text{-}pmf.variance\ p\ (\lambda h.\ (\sum x\leftarrow xs.\ h\ x)\hat{\ }2) \le 2* F2\ xs\hat{\ }2$
(**is** $?L \le ?R$)
⟨*proof*⟩

lemma

assumes $s \in$ *set-pmf* ($\mathcal{H}_P$ *4 n* ($\mathcal{L}$ [−*1*,*1*]))
assumes *set xs* $\subseteq$ {*0*..<*n*}
shows *f2-exp-hp*: ($\int$ *h*. ($\sum$ *x* ← *xs*. *h x*)^*2* ∂*sample-pro s*) = *F2 xs* (**is** *?T1*)
and *f2-exp-sq-hp*: ($\int$ *h*. (($\sum$ *x* ← *xs*. *h x*)^*2*)^*2* ∂*sample-pro s*) ≤ *3*∗ *F2 xs*^*2* (**is** *?T2*)
and *f2-var-hp*: *measure-pmf*.*variance s* (λ*h*. ($\sum$ *x* ← *xs*. *h x*)^*2*) ≤ *2*∗ *F2 xs*^*2* (**is** *?T3*)
⟨*proof*⟩

lemmas *f2-exp-h* = *f2-exp-hp*[*OF hash-pro-in-hash-pro-pmf*[*OF prime-power-ls*]]
lemmas *f2-var-h* = *f2-var-hp*[*OF hash-pro-in-hash-pro-pmf*[*OF prime-power-ls*]]

lemma *F2-definite*:
assumes *xs* ≠ []
shows *F2 xs* > *0*
⟨*proof*⟩

The following algorithm uses a completely random function, accordingly it requires a lot of space: $\mathcal{O}(n + \ln m)$.

fun *example-1* :: *nat* ⇒ *nat list* ⇒ *real pmf*
where *example-1 n xs* =
do {
h ← *prod-pmf* {*0*..<*n*} (λ-. *pmf-of-set* {−*1*,*1*::*real*});
return-pmf (($\sum$ *x* ← *xs*. *h x*)^*2*)
}

lemma *example-1-correct*:
assumes *set xs* $\subseteq$ {*0*..<*n*}
shows
measure-pmf.*expectation* (*example-1 n xs*) *id* = *F2 xs* (**is** *?L1* = *?R1*)
measure-pmf.*variance* (*example-1 n xs*) *id* ≤ *2* ∗ *F2 xs*^*2* (**is** *?L2* ≤ *?R2*)
⟨*proof*⟩

This version replaces a the use of completely random function with a pseudorandom object, it requires a lot less space: $\mathcal{O}(\ln n + \ln m)$.

fun *example-2* :: *nat* ⇒ *nat list* ⇒ *real pmf*
where *example-2 n xs* =
do {
h ← *sample-pro* ($\mathcal{H}$ *4 n* ($\mathcal{L}$ [−*1*,*1*]));
return-pmf (($\sum$ *x* ← *xs*. *h x*)^*2*)
}

lemma *example-2-correct*:
assumes *set xs* $\subseteq$ {*0*..<*n*}
shows
measure-pmf.*expectation* (*example-2 n xs*) *id* = *F2 xs* (**is** *?L1* = *?R1*)
measure-pmf.*variance* (*example-2 n xs*) *id* ≤ *2* ∗ *F2 xs*^*2* (**is** *?L2* ≤ *?R2*)
⟨*proof*⟩

The following version replaces the deterministic construction of the pseudo-random object with a randomized one. This algorithm is much faster, but the correctness proof is more difficult.

fun *example-3* :: *nat* ⇒ *nat list* ⇒ *real pmf*
  where *example-3 n xs* =
    do {
      h ← *sample-pro* =<< $\mathcal{H}_P$ *4 n* ($\mathcal{L}$ [−*1*,*1*]);
      return-pmf ((∑ *x* ← *xs*. *h x*)^*2*)
    }

lemma
  assumes *set xs* ⊆ {*0*..<*n*}
  shows
    measure-pmf.expectation (*example-3 n xs*) *id* = *F2 xs* (**is** *?L1* = *?R1*)
    measure-pmf.variance (*example-3 n xs*) *id* ≤ *2* ∗ *F2 xs*^*2* (**is** *?L2* ≤ *?R2*)
⟨*proof*⟩

context
  fixes *ε δ* :: *real*
  assumes *ε-gt-0*: *ε* > *0*
  assumes *δ-range*: *δ* ∈ {*0*<..<*1*}
begin

By using the mean of many independent parallel estimates the following algorithm achieves a relative accuracy of ε, with probability $\frac{3}{4}$. It requires $\mathcal{O}(\varepsilon^{-2}(\ln n + \ln m))$ bits of space.

fun *example-4* :: *nat* ⇒ *nat list* ⇒ *real pmf*
  where *example-4 n xs* =
    do {
      let s = *nat* ⌈*8* / *ε*^*2*⌉;
      h ← *prod-pmf* {*0*..<*s*} (λ-. *sample-pro* ($\mathcal{H}$ *4 n* ($\mathcal{L}$ [−*1*,*1*])));
      return-pmf ((∑ *j* < *s*. (∑ *x* ← *xs*. *h j x*)^*2*)/*s*)
    }

lemma *example-4-correct-aux*:
  assumes *set xs* ⊆ {*0*..<*n*}
  defines *s* ≡ *nat* ⌈*8* / *ε*^*2*⌉
  defines *R* ≡ (λ*h* :: *nat* ⇒ *nat* ⇒ *real*. (∑*j*<*s*. (∑*x*←*xs*. *h j x*)^*2*)/*real s*)
  assumes *fin*: *finite* (*set-pmf p*)
  assumes *indep*: *prob-space.k-wise-indep-vars* (*measure-pmf p*) *2* (λ-. *discrete*) (λ*i x*. *x i*) {..<*s*}
  assumes *comp*: ⋀*i*. *i* < *s* ⟹ *map-pmf* (λ*x*. *x i*) *p* = *sample-pro* ($\mathcal{H}$ *4 n* ($\mathcal{L}$ [−*1*,*1*]))
  shows *measure p* {*h*. |*R h* − *F2 xs*| > *ε* ∗ *F2 xs*} ≤ *1*/*4* (**is** *?L* ≤ *?R*)
⟨*proof*⟩

lemma *example-4-correct*:
  assumes *set xs* ⊆ {*0*..<*n*}

shows $\mathcal{P}(\omega$ *in example-4 n xs.* $|\omega - F2\ xs| > \varepsilon * F2\ xs) \leq 1/4$ (**is** *?L* ≤ *?R*)
⟨*proof*⟩

Instead of independent samples, we can choose the seeds using a second pair-wise independent pseudorandom object. This algorithm requires only $\mathcal{O}(\ln n + \varepsilon^{-2} \ln m)$ bits of space.

```
fun example-5 :: nat ⇒ nat list ⇒ real pmf
  where example-5 n xs =
    do {
      let s = nat ⌈8 / ε^2⌉;
      h ← sample-pro (ℋ 2 s (ℋ 4 n (ℒ [−1,1])));
      return-pmf ((∑ j < s. (∑ x ← xs. h j x)^2)/s)
    }
```

```
lemma example-5-correct-aux:
  assumes set xs ⊆ {0..<n}
  defines s ≡ nat ⌈8 / ε^2⌉
  defines R ≡ (λh :: nat ⇒ nat ⇒ real. (∑j<s. (∑x←xs. h j x)^2)/real s)
  shows measure (sample-pro (ℋ 2 s (ℋ 4 n (ℒ [−1,1])))) {h. |R h − F2 xs| > ε
∗ F2 xs} ≤ 1/4
⟨proof⟩
```

```
lemma example-5-correct:
  assumes set xs ⊆ {0..<n}
  shows 𝒫(ω in example-5 n xs. |ω − F2 xs| > ε ∗ F2 xs) ≤ 1/4 (is ?L ≤ ?R)
⟨proof⟩
```

The following algorithm improves on the previous one, by achieving a success probability of δ. This works by taking the median of $\mathcal{O}(\ln(\delta^{-1}))$ parallel independent samples. It requires $\mathcal{O}(\ln(\delta^{-1})(\ln n + \varepsilon^{-2} \ln m))$ bits of space.

```
fun example-6 :: nat ⇒ nat list ⇒ real pmf
  where example-6 n xs =
    do {
      let s = nat ⌈8 / ε^2⌉; let t = nat ⌈8 ∗ ln (1/δ)⌉;
      h ← prod-pmf {0..<t} (λ-. sample-pro (ℋ 2 s (ℋ 4 n (ℒ [−1,1]))));
      return-pmf (median t (λi. ((∑ j < s. (∑ x ← xs. h i j x)^2)/ s)))
    }
```

```
lemma example-6-correct:
  assumes set xs ⊆ {0..<n}
  shows 𝒫(ω in example-6 n xs. |ω − F2 xs| > ε ∗ F2 xs) ≤ δ (is ?L ≤ ?R)
⟨proof⟩
```

The following algorithm uses an expander random walk, instead of independent samples. It requires only $\mathcal{O}(\ln n + \ln(\delta^{-1})\varepsilon^{-2} \ln m)$ bits of space.

```
fun example-7 :: nat ⇒ nat list ⇒ real pmf
  where example-7 n xs =
    do {
```

```
    let s = nat ⌈8 / ε^2⌉; let t = nat ⌈32 ∗ ln (1/δ)⌉;
    h ← sample-pro (ℰ t (1/8) (ℋ 2 s (ℋ 4 n (ℒ [−1,1]))));
    return-pmf (median t (λi. ((∑j < s. (∑x ← xs. h i j x)^2)/ s)))
  }
```

lemma *example-7-correct*:
  assumes *set xs* ⊆ {*0..<n*}
  shows 𝒫(*ω in example-7 n xs.* |*ω* − *F2 xs*| > *ε* ∗ *F2 xs*) ≤ *δ* (**is** *?L* ≤ *?R*)
⟨*proof*⟩

end

end

A Informal proof of correctness for the F_0 algorithm

This appendix contains a detailed informal proof for the new Rounding-KMV algorithm that approximates F_0 introduced in Section 6 for reference. It follows the same reasoning as the formalized proof.

Because of the amplification result about medians (see for example [1, §2.1]) it is enough to show that each of the estimates the median is taken from is within the desired interval with success probability $\frac{2}{3}$. To verify the latter, let $a_1, \ldots, a_m$ be the stream elements, where we assume that the elements are a subset of $\{0, \ldots, n-1\}$ and $0 < \delta < 1$ be the desired relative accuracy. Let p be the smallest prime such that $p \geq \max(n, 19)$ and let h be a random polynomial over $GF(p)$ with degree strictly less than 2. The algoritm also introduces the internal parameters t, r defined by:

$$t := \lceil 80\delta^{-2} \rceil \qquad\qquad r := 4\log_2 \lceil \delta^{-1} \rceil + 23$$

The estimate the algorithm obtains is R, defined using:

$$H := \{\lfloor h(a) \rfloor_r | a \in A\} \qquad R := \begin{cases} tp\,(\min_t(H))^{-1} & \text{if } |H| \geq t \\ |H| & \text{othewise,} \end{cases}$$

where $A := \{a_1, \ldots, a_m\}$, $\min_t(H)$ denotes the t-th smallest element of H and $\lfloor x \rfloor_r$ denotes the largest binary floating point number smaller or equal to x with a mantissa that requires at most r bits to represent.[1] With these definitions, it is possible to state the main theorem as:

$$P(|R - F_0| \leq \delta |F_0|) \geq \frac{2}{3}.$$

which is shown separately in the following two subsections for the cases $F_0 \geq t$ and $F_0 < t$.

[1] This rounding operation is called *truncate-down* in Isabelle, it is defined in HOL-Library.Float.

A.1 Case $F_0 \geq t$

Let us introduce:

$$H^* := \{h(a)|a \in A\}^{\#} \qquad\qquad R^* := tp\left(\min_t^{\#}(H^*)\right)^{-1}$$

These definitions are modified versions of the definitions for H and R: The set H^* is a multiset, this means that each element also has a multiplicity, counting the number of *distinct* elements of A being mapped by h to the same value. Note that by definition: $|H^*| = |A|$. Similarly the operation $\min_t^{\#}$ obtains the t-th element of the multiset H (taking multiplicities into account). Note also that there is no rounding operation $\lfloor\cdot\rfloor_r$ in the definition of H^*. The key reason for the introduction of these alternative versions of H, R is that it is easier to show probabilistic bounds on the distances $|R^* - F_0|$ and $|R^* - R|$ as opposed to $|R - F_0|$ directly. In particular the plan is to show:

$$P\left(|R^* - F_0| > \delta' F_0\right) \leq \frac{2}{9}, \text{ and} \tag{1}$$

$$P\left(|R^* - F_0| \leq \delta' F_0 \wedge |R - R^*| > \frac{\delta}{4} F_0\right) \leq \frac{1}{9} \tag{2}$$

where $\delta' := \frac{3}{4}\delta$. I.e. the probability that R^* has not the relative accuracy of $\frac{3}{4}\delta$ is less that $\frac{2}{9}$ and the probability that assuming R^* has the relative accuracy of $\frac{3}{4}\delta$ but that R deviates by more that $\frac{1}{4}\delta F_0$ is at most $\frac{1}{9}$. Hence, the probability that neither of these events happen is at least $\frac{2}{3}$ but in that case:

$$|R - F_0| \leq |R - R^*| + |R^* - F_0| \leq \frac{\delta}{4} F_0 + \frac{3\delta}{4} F_0 = \delta F_0. \tag{3}$$

Thus we only need to show Equation 1 and 2. For the verification of Equation 1 let

$$Q(u) = |\{h(a) < u \mid a \in A\}|$$

and observe that $\min_t^{\#}(H^*) < u$ if $Q(u) \geq t$ and $\min_t^{\#}(H^*) \geq v$ if $Q(v) \leq t - 1$. To see why this is true note that, if at least t elements of A are mapped by h below a certain value, then the t-smallest element must also be within them, and thus also be below that value. And that the opposite direction of this conclusion is also true. Note that this relies on the fact that H^* is a multiset and that multiplicities are being taken into account, when computing the t-th smallest element. Alternatively, it is also possible to write $Q(u) = \sum_{a \in A} 1_{\{h(a)<u\}}$[2], i.e., Q is a sum of pairwise independent $\{0,1\}$-valued random variables, with expectation $\frac{u}{p}$ and variance $\frac{u}{p} - \frac{u^2}{p^2}$.

[2]The notation 1_A is shorthand for the indicator function of A, i.e., $1_A(x) = 1$ if $x \in A$ and 0 otherwise.

[3] Using lineariy of expectation and Bienaymé's identity, it follows that $\operatorname{Var} Q(u) \leq \operatorname{E} Q(u) = |A|up^{-1} = F_0 up^{-1}$ for $u \in \{0, \ldots, p\}$.
For $v = \left\lfloor \frac{tp}{(1-\delta')F_0} \right\rfloor$ it is possible to conclude:

$$t - 1 \leq {}^{4} \frac{t}{(1-\delta')} - 3\sqrt{\frac{t}{(1-\delta')}} - 1 \leq \frac{F_0 v}{p} - 3\sqrt{\frac{F_0 v}{p}} \leq \operatorname{E}Q(v) - 3\sqrt{\operatorname{Var}Q(v)}$$

and thus using Tchebyshev's inequality:

$$\begin{aligned} P\left(R^* < (1-\delta')\,F_0\right) &= P\left(\operatorname{rank}_t^{\#}(H^*) > \frac{tp}{(1-\delta')F_0}\right) \\ &\leq P(\operatorname{rank}_t^{\#}(H^*) \geq v) = P(Q(v) \leq t-1) \\ &\leq P\left(Q(v) \leq \operatorname{E}Q(v) - 3\sqrt{\operatorname{Var}Q(v)}\right) \leq \frac{1}{9}. \end{aligned} \tag{4}$$

Similarly for $u = \left\lceil \frac{tp}{(1+\delta')F_0} \right\rceil$ it is possible to conclude:

$$t \geq \frac{t}{(1+\delta')} + 3\sqrt{\frac{t}{(1+\delta')}} + 1 + 1 \geq \frac{F_0 u}{p} + 3\sqrt{\frac{F_0 u}{p}} \geq \operatorname{E}Q(u) + 3\sqrt{\operatorname{Var}Q(v)}$$

and thus using Tchebyshev's inequality:

$$\begin{aligned} P\left(R^* > (1+\delta')\,F_0\right) &= P\left(\operatorname{rank}_t^{\#}(H^*) < \frac{tp}{(1+\delta')F_0}\right) \\ &\leq P(\operatorname{rank}_t^{\#}(H^*) < u) = P(Q(u) \geq t) \\ &\leq P\left(Q(u) \geq \operatorname{E}Q(u) + 3\sqrt{\operatorname{Var}Q(u)}\right) \leq \frac{1}{9}. \end{aligned} \tag{5}$$

Note that Equation 4 and 5 confirm Equation 1. To verfiy Equation 2, note that

$$\min_t(H) = \lfloor \min_t^{\#}(H^*) \rfloor_r \tag{6}$$

if there are no collisions, induced by the application of $\lfloor h(\cdot) \rfloor_r$ on the elements of A. Even more carefully, note that the equation would remain true, as long as there are no collision within the smallest t elements of H^*. Because Equation 2 needs to be shown only in the case where $R^* \geq (1-\delta')F_0$, i.e., when $\min_t^{\#}(H^*) \leq v$, it is enough to bound the probability of a collision in the range $[0; v]$. Moreover Equation 6 implies $|\min_t(H) - \min_t^{\#}(H^*)| \leq \max(\min_t^{\#}(H^*), \min_t(H))2^{-r}$ from which it is possible to derive $|R^* - R| \leq \frac{\delta}{4} F_0$. Another important fact is that h is injective with probability $1 - \frac{1}{p}$,

[3] A consequence of h being chosen uniformly from a 2-independent hash family.
[4] The verification of this inequality is a lengthy but straightforward calculcation using the definition of δ' and t.

this is because h is chosen uniformly from the polynomials of degree less than 2. If it is a degree 1 polynomial it is a linear function on $GF(p)$ and thus injective. Because $p \geq 18$ the probability that h is not injective can be bounded by 1/18. With these in mind, we can conclude:

$$
\begin{aligned}
& P\left(|R^* - F_0| \leq \delta' F_0 \wedge |R - R^*| > \frac{\delta}{4} F_0\right) \\
\leq\ & P\left(R^* \geq (1-\delta')F_0 \wedge \min_t^{\#}(H^*) \neq \min_t(H) \wedge h \text{ inj.}\right) + P(\neg h \text{ inj.}) \\
\leq\ & P\left(\exists a \neq b \in A. \lfloor h(a) \rfloor_r = \lfloor h(b) \rfloor_r \leq v \wedge h(a) \neq h(b)\right) + \frac{1}{18} \\
\leq\ & \frac{1}{18} + \sum_{a \neq b \in A} P\left(\lfloor h(a) \rfloor_r = \lfloor h(b) \rfloor_r \leq v \wedge h(a) \neq h(b)\right) \\
\leq\ & \frac{1}{18} + \sum_{a \neq b \in A} P\left(|h(a) - h(b)| \leq v 2^{-r} \wedge h(a) \leq v(1 + 2^{-r}) \wedge h(a) \neq h(b)\right) \\
\leq\ & \frac{1}{18} + \sum_{a \neq b \in A} \sum_{\substack{a', b' \in \{0, \ldots, p-1\} \wedge a' \neq b' \\ |a' - b'| \leq v 2^{-r} \wedge a' \leq v(1 + 2^{-r})}} P(h(a) = a') P(h(b) = b') \\
\leq\ & \frac{1}{18} + \frac{5 F_0^2 v^2}{2p^2} 2^{-r} \leq \frac{1}{9}.
\end{aligned}
$$

which shows that Equation 2 is true.

A.2 Case $F_0 < t$

Note that in this case $|H| \leq F_0 < t$ and thus $R = |H|$, hence the goal is to show that: $P(|H| \neq F_0) \leq \frac{1}{3}$. The latter can only happen, if there is a collision induced by the application of $\lfloor h(\cdot) \rfloor_r$. As before h is not injective with probability at most $\frac{1}{18}$, hence:

$$\begin{aligned}
& P\left(|R - F_0| > \delta F_0\right) \leq P\left(R \neq F_0\right) \\
\leq\ & \frac{1}{18} + P\left(R \neq F_0 \wedge h \text{ inj.}\right) \\
\leq\ & \frac{1}{18} + P\left(\exists a \neq b \in A. \lfloor h(a) \rfloor_r = \lfloor h(b) \rfloor_r \wedge h \text{ inj.}\right) \\
\leq\ & \frac{1}{18} + \sum_{a \neq b \in A} P\left(\lfloor h(a) \rfloor_r = \lfloor h(b) \rfloor_r \wedge h(a) \neq h(b)\right) \\
\leq\ & \frac{1}{18} + \sum_{a \neq b \in A} P\left(|h(a) - h(b)| \leq p2^{-r} \wedge h(a) \neq h(b)\right) \\
\leq\ & \frac{1}{18} + \sum_{a \neq b \in A} \sum_{\substack{a',b' \in \{0,\ldots,p-1\} \\ a' \neq b' \wedge |a'-b'| \leq p2^{-r}}} P(h(a) = a')P(h(b) = b') \\
\leq\ & \frac{1}{18} + F_0^2 2^{-r+1} \leq \frac{1}{18} + t^2 2^{-r+1} \leq \frac{1}{9}.
\end{aligned}$$

Which concludes the proof. □

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 J. D. P. Rolim and S. Vadhan, editors, *Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer Berlin Heidelberg, 2002.