The *Oblivious Set* abstract data type

Alexander Towell

atowell@siue.edu

Abstract

We define the semantics of abstract data type for the *oblivious set*. We demonstrate a theoretical data structure, denoted the *Singular Hash Set*, that provides an optimal implementation of the oblivious set with respect to *entropy* and *space complexity*. Finally, we show how to use the *Bloom filter* and *Perfect Hash Filter* to implement oblivious sets and compare them to each other and the theoretically optimal *Singular Hash Set*.

Contents

1 Introduction

The *oblivious set*[1] is a fundamental data structure that may be used to construct other oblivious object types, like secure indices for Boolean Encrypted Search[2]

An *oblivious set* is an oblivious object type over $(2^{\mathcal{U}}, \in)$, which means that the oblivious set represents values in the set $2^{\mathcal{U}}$ over the binary *member-of* predicate $\in \colon \mathcal{U} \times 2^{\mathcal{U}} \mapsto \{0,1\}$.

Suppose we have data types $X_1, X_2, \ldots, X_n$ and a surjective pairing function

$$\mathrm{f} \colon \mathbb{N}^2 \mapsto \mathbb{N}\,. \tag{1}$$

Recursively, the n-tuple encoding function $\mathrm{g}_n \colon \mathbb{N}^n \mapsto \mathbb{N}$ may be defined as

$$\mathrm{g}_2(x_1, x_2) \coloneqq \mathrm{f}(x_1, x_2) \tag{2}$$
$$\mathrm{g}_n(x_1, \ldots, x_n) \coloneqq \mathrm{f}(x_1, \mathrm{g}_{n-1}(x_2, \ldots, x_n)) \,. \tag{3}$$

Assuming we have a serializer $\mathrm{h}_j \colon X_j \mapsto \mathbb{N}$ for $j = 1, \ldots, n$, an encoder for tuples of typle $X_1 \times \cdots \times X_n$ is given by

$$\mathrm{encode}(X_1, \ldots, X_n) \coloneqq \mathrm{g}_n(\mathrm{h}_1(X_1), \ldots, \mathrm{h}_n(X_n)) \,. \tag{4}$$

Now, we may use any random approximate set over the natural numbers to represent any n-tuple relation.

A *regular function* over some abstract data type X behaves the same way if given any data structure that implements the behavior of X.

1. Approximate membership tests of specific elements may be performed with a false positive rate ε and a false negative rate ω. Note that there is no way to efficiently iterate over the elements.

2. The cardinality may be estimated to be within some range. The the degree of uncertainty can be made arbitrarily large entropy at the expense of its space complexity.

3. Set-theoretic operations like union, intersection, and complement generate oblivious sets that approximate the true operation as a function of the false positive and false negative rates and the degree of similarity between the exact sets under consideration.

In section 2, we precisely define the oblivious set.

In section 3, we derive the probablistic model of *oblivious sets*. In section 4, we derive the *entropy* of *oblivious sets*. In section 5, we derive estimators of properties of *oblivious sets*. In section 6, we provide a theoretically optimal implementation of the oblivious set.

2 Oblivious sets

A set is given by the following definition.

Definition 2.1. *A set is an unordered collection of distinct elements from a universe of elements.*

A countable set is a *finite set* or a *countably infinite set*. A *finite set* has a finite number of elements. For example,

$$\mathcal{S} = \{1, 3, 5\}$$

is a finite set with three elements. A *countably infinite set* can be put in one-to-one correspondence with the set of natural numbers

$$\mathbb{N} = \{1, 2, 3, 4, 5, \ldots\} \,. \tag{5}$$

The cardinality of a set $\mathcal{S}$ is a measure of the number of elements in the set, denoted by

$$|\mathcal{S}| \,. \tag{6}$$

The cardinality of a *finite set* is a non-negative integer and counts the number of elements in the set, e.g.,

$$|\{1, 3, 5\}| = 3 \,.$$

Informally, an oblivious set of $\mathcal{S}$, denoted by $\check{\mathcal{S}}$, provides a *confidential* in-place binary representation such that very little information about $\mathcal{S}$ is disclosed. To be a minimally *useful*, $\check{\mathcal{S}}$ must permit *membership tests* with respect to $\mathcal{S}$ with specifiable false positive and false negative rates.

Note that an oblivious set where elements are from the universe $\mathcal{U}$ permits membership tests on elements in $\mathcal{U}$. These elements, and the universe $\mathcal{U}$, are *not* necessarily oblivious types. It is also possible to have an oblivious set over oblivious elements, where the elements have their own set of separate constraints, e.g., an oblivious type X in which only the less-than and equality predicates are possible. The oblivious set, of course, only allows *membership* tests to be performed on elements over $\check{\mathcal{U}}$.

In what follows, we provide a formal specification of the abstract data type of the oblivious set.

2.1 Oblivious object types

A *type* is a set and the elements of the set are called the *values* of the type. An *abstract data type* is a type and a set of operations on values of the type. For example, the *integer* abstract data type is defined by the set of integers and standard operations like addition and subtraction. A *data structure* is a particular way of organizing data and may implement one or more abstract data types.

Suppose we have an abstract data type denoted by T with a set of operators $\mathcal{F} = \{\mathrm{f}_1, \ldots, \mathrm{f}_\mathrm{n}\}$ that are (at least partially) functions of T. We denote that an *object* x in computer memory implements the abstract data type T by $T(x)$.

An *oblivious* object type[**?**] that implements the abstract data type T is a related type denoted by $\check{T}$ that provides guarantees about what can be learned about an object $\check{T}(x)$ by looking at the binary representation of $\check{x}$.

Optimally, the only information that can be learned about $\check{x}$ is given by the well-defined *behavior* of the the abstract data type on the set of operators $\mathcal{F}$. For instance, say an operator $\mathrm{g}\colon [T] \mapsto \{\mathbf{true}, \mathbf{false}\}$ is defined but not in F, and $\mathrm{g}(x) = \mathbf{true}$ for a particular object $T(x)$. Then, if the only information we have about x is given by $\check{x} = \check{T}(x)$, then $\mathrm{P}[\mathrm{g}(\check{x}) = \mathrm{g}(x)] = 0.5$, i.e., we can do no better than a random guess.

The *oblivious set* is an abstract data type which *confidentially* approximates sets with two types of errors, false positives and false negatives. Thus, the oblivious part consists of two parts. First, it is a type of *approximate set*[3]. Second, it provides additional confidentiality guarantees.

The abstract data type of the immutable approximate set[3] is given by the following definition.

Definition 2.2. *The abstract data type of the* approximate set *over a universe $\mathcal{U}$ has values given by the set $\mathcal{P}(\mathcal{U})$. At a minimum, a set must provide some way to test whether particular elements in $\mathcal{U}$ are members of a particular set,*

$$\in\colon \mathcal{U} \times \mathcal{P}(\mathcal{U}) \mapsto \{\mathbf{true}, \mathbf{false}\}\,, \tag{7}$$

$$\tag{8}$$

Let an element that is selected uniformly at random from the universe $\mathcal{U}$ be denoted by X*. A set $\mathcal{S}^{\pm}$ is a* approximate set *of a set $\mathcal{S}$ with a false positive rate ε and false negative rate ω if the following conditions hold:*

(i) If X *is a member of $\mathcal{S}$, it is not a member of $\mathcal{S}^{\pm}$ with a probability ω,*

$$\mathrm{P}\Big[\mathrm{X} \notin \mathcal{S}^{\pm} \,\Big|\, \mathrm{X} \in \mathcal{S}\Big] = \omega\,. \tag{9}$$

(ii) If X *is* not *a member of $\mathcal{S}$, it is a member of $\mathcal{S}^{-}$ with a probability ε,*

$$\mathrm{P}\Big[\mathrm{X} \in \mathcal{S}^{\pm} \,\Big|\, \mathrm{X} \notin \mathcal{S}\Big] = \varepsilon\,. \tag{10}$$

The optimal space complexity of *countably infinite* approximate sets is given by the following postulate.

Postulate 2.1. *The* optimal *space complexity of a data structure implementing the* approximate set *over a* countably infinite *universe is independent of the type of elements and depends only the false positive rate ε and false negative rate ω as given by*

$$-(1-\omega)\log_2 \varepsilon \text{ bits/element}\,. \tag{11}$$

The abstract data type of the *immutable* oblivious set is given by the following definition.

Definition 2.3 (Oblivious set)**.** *Assuming that the only information about a set of interested $\mathcal{S} \subset \mathcal{U}$ is given by another set $\check{\mathcal{S}}$, $\check{\mathcal{S}}$ is an* oblivious set *of a set $\mathcal{S}$ if the following conditions are hold:*

(i) There is no efficient way to enumerate the elements in $\check{\mathcal{S}}$.[1]

(ii) Any estimator of the cardinality of $\mathcal{S}$ may only be able determine an approximate upper and lower bound, uniformly distributed, where the uncertainty may be traded for space-efficiency.

Definition 2.4 (Approximate oblivious set)**.** *Assuming the conditions specified for oblivious sets hold in definition 2.3, an oblivious set $\check{\mathcal{S}}$ is an approximate oblivious set of $\mathcal{S}$ with a false positive rate ε and a false negative rate ω if the following additional conditions hold:*

1. *Each negative element tests positive with a probability ε and tests negative with a probability $1-\varepsilon$. That is, each test is Bernoulli distributed, which is the* maximum entropy *distribution given that the false positive rate is ε.*[2]

   Assuming we do not have access to $\mathcal{S}$, the most accurate prediction possible when predicting whether an element is negative is ε.

$$\mathrm{P}[\mathrm{X} \in \mathcal{S}] = \mathrm{P}[\mathrm{X} \textit{ is a false negative or } \mathrm{X} \textit{ is a true positive}]\,. \tag{12}$$

2. *Each positive element tests negative with a probability ω and tests positive with a probability $1-\omega$. That is, each test is Bernoulli distributed, which is the* maximum entropy *distribution given that the false negative rate is ω.*[3]

3. *By items 1 and 2, $\check{\mathcal{S}}$ is an approximate set[3] of $\mathcal{S}$ with a false positive rate ε and false negative rate ω.*

A *oblivious positive set* is a special case given by the following definition.

Definition 2.5. *An* oblivious set *$\check{\mathcal{S}}$ with a false negative rate equal to zero is a* oblivious positive set *denoted by $\check{\mathcal{S}}^-$. By this definition, $\check{\mathcal{S}}^-$ is a* superset *of $\mathcal{S}$.*

The *complement* of a *oblivious positive set* is given by the following definition.

Definition 2.6. *An oblivious set $\check{\mathcal{S}}$ with a false positive rate equal to zero is a* oblivious negative set *denoted by $\check{\mathcal{S}}^-$. By this definition, $\check{\mathcal{S}}^-$ is a* subset *of $\mathcal{S}$.*

[1] That is, the true positives and false positives.

[2] If the universe is finite and there are n negatives, the number of false positives is binomially distributed with a mean εn.

[3] If there are p positives, the number of false positives is binomially distributed with a mean εn.

The absolute space efficiency of a data structure Y implementing an oblivious set consisting of m positives with a false positive rate ε, false negative rate ω, and an entropy β is given by

$$\mathtt{E}(\varepsilon, \omega, m, \beta) = \mathrm{E}\left[\frac{-(1-\omega)(m+\mathrm{X})\log_2 \varepsilon}{\mathtt{BL}(Y)}\right], \tag{13}$$

where

$$\mathrm{X} \sim \mathrm{DU}(0, 2^\beta - 1) \tag{14}$$

and Y is a function of the random variable X.

The *Singular Hash Set* is an *optimal* implementation of the oblivious set abstract data type.

The relative efficiency of the *optimal* oblivious set with entropy β to the *optimal* approximate set ($\beta = 0$) has an expectation given by

$$\mathtt{RE}(\,\cdot\,, m, \beta) = 2^{-\beta} \sum_{k=0}^{2^\beta - 1} \left(1 + \frac{k}{m}\right)^{-1}. \tag{15}$$

For a fixed β, as $m \to \infty$ the relative efficiency goes to 1.

See **??** to see how a C++ interface for the approximate set abstract data type may be defined.

3 Probabilistic model

The uncertain number of false positives is given by the following theorem.

Theorem 3.1. *Given a set $\mathcal{S}$ with m positives from a universe of u elements, the number of* false positives *in an approximate set $\mathcal{S}^\pm$ with a false positive rate ε is a random variable denoted by* FP_m *with a distribution given by*

$$\mathrm{FP}_m \sim \mathrm{BIN}(u - m, \varepsilon)\,. \tag{16}$$

The number of false negatives is given by the following theorem.

Theorem 3.2. *Given a set $\mathcal{S}$ with m positives, the number of* false negatives *with respect to an approximate set $\mathcal{S}^\pm$ with a false negative rate ω is a random variable denoted by* FN_m *with a distribution given by*

$$\mathrm{FN}_m \sim \mathrm{BIN}(m, \omega)\,. \tag{17}$$

The *expected* cardinality is given by the following theorem.

Theorem 3.3 (Cardinality)**.** *Given a set $\mathcal{S}$ of cardinality m from a universe of u elements, an approximate set $\mathcal{S}^\pm$ has an* expected *cardinality given by*

$$u\varepsilon + m(1 - \varepsilon - \omega)\,, \tag{18}$$

where ε is the false positive rate and ω is the false negative rate.

3.1 Positives and negatives

The distribution of false positives and false negatives are Bernoulli distributed random variables conditioned on a particular number of positives. The distribution of positives (and negatives) is given by the following definition.

Definition 3.1. *The number of* positives *in a universe of u elements is uncertain. We model the uncertainty as a discrete random variable, denoted by* P*, with a probability mass function*[4]

$$\mathrm{f}_{\mathrm{P}}(p \mid u) \tag{19}$$

and a support $\{0, \ldots, u\}$*. Conversely, the distribution of negatives is a random variable* $\mathrm{N} = u - \mathrm{P}$ *with a probability mass function*

$$\mathrm{f}_{\mathrm{N}}(n \mid u) = \mathrm{f}_{\mathrm{P}}(u - n \mid u)\,. \tag{20}$$

The form the probability mass function $\mathrm{f}_{\mathrm{P}}(\,\cdot\,)$ takes cannot be a priori specified, although it may be estimated with an empirical probability.

Modeling the distribution of positives provides a complete specification for the distribution of false positives and false negatives (and true positives and true negatives).

Example 1 *The* expected *number of false positives is give by the expectation*

$$\mathrm{E}[\mathrm{FP}] = \sum_{p=0}^{u} \sum_{f_p=0}^{u-p} f_p \cdot \mathrm{f}_{\mathrm{P}}(p \mid u)\, \mathrm{f}_{\mathrm{FP}}(f_p \mid p, u, \varepsilon) \tag{a}$$

$$= \sum_{p=0}^{u} \mathrm{f}_{\mathrm{P}}(p \mid u)\, \mathrm{E}[\mathrm{FP}_p \mid u] = \sum_{p=0}^{u} \mathrm{f}_{\mathrm{P}}(p \mid u)(u - p)\varepsilon \tag{b}$$

$$= \varepsilon \left(u - \sum_{p=0}^{u} m\, \mathrm{f}_{\mathrm{P}}(p \mid u) \right) = \varepsilon \left(u - \mathrm{E}[\mathrm{P}] \right)\,. \tag{c}$$

Note that $\mathrm{E}[\mathrm{N}] = u - \mathrm{E}[\mathrm{P}]$*, thus*

$$\mathrm{E}[\mathrm{FP}] = \varepsilon\, \mathrm{E}[\mathrm{N}]\,. \tag{d}$$

The joint probability mass function of positives, false positives, and false negatives is given by

$$\mathrm{f}(p, f_p, f_n \mid u, \varepsilon, \omega) = \mathrm{f}_{\mathrm{P}}(p \mid u)\, \mathrm{f}_{\mathrm{FP}}(f_p \mid p, u, \varepsilon)\, \mathrm{f}_{\mathrm{FN}}(f_n \mid p, u, \omega)\,. \tag{21}$$

4 Entropy

Theorem 4.1. *The* entropy *of the uncertain number of false positives and false negatives is given by*

$$c + \frac{1}{2} \log_2\big((u - m) m \varepsilon (1 - \varepsilon) \omega (1 - \omega)\big) + \mathcal{O}\left(\frac{u}{m(u - m)} \right), \tag{22}$$

where $c = \log_2(2\pi e)$*.*

Proof. The entropy of the joint distribution of false positives and false negatives given that m are positive is given by

$$\mathcal{H}(\mathrm{FP}_m, \mathrm{FN}_m)\,. \tag{a}$$

[4] The probability mass function of a random variable X is denoted by $\mathrm{f}_{\mathrm{X}}(\,\cdot\,)$.

By **??**, FP_m and FN_m are independent. Thus,

$$\mathcal{H}(\mathrm{FP}_m, \mathrm{FN}_m) = \mathcal{H}(\mathrm{FP}_m) + \mathcal{H}(\mathrm{FN}_m)\,. \tag{b}$$

The entropy of FP_m is defined as

$$\mathcal{H}(\mathrm{FP}_m) = -\sum_{f_p=0}^{u-m} \log_2 \mathrm{f}_{\mathrm{FP}_m}(f_p \mid u, \varepsilon)\, \mathrm{f}_{\mathrm{FP}_m}(f_p \mid u, \varepsilon)\,. \tag{c}$$

$$\mathcal{H}(\mathrm{FP}_m) = c + \log_2 \sqrt{u-m} + \log_2 \sqrt{\varepsilon} + \log_2 \sqrt{1-\varepsilon} + \mathcal{O}\left(\frac{1}{u-m}\right)\,, \tag{d}$$

$$\mathcal{H}(\mathrm{FN}_m) = c + \log_2 \sqrt{m} + \log_2 \sqrt{\omega} + \log_2 \sqrt{1-\omega} + \mathcal{O}\left(\frac{1}{m}\right)\,, \tag{e}$$

and $c = \log_2 \sqrt{2\pi e}$. Summing these and simplifying yields the result

$$\mathcal{H}(\mathrm{FP}_m, \mathrm{FN}_m) = c + \frac{1}{2}\log_2\big((u-m)m\varepsilon(1-\varepsilon)\omega(1-\omega)\big) + \mathcal{O}\left(\frac{u}{m(u-m)}\right)\,, \tag{f}$$

where $c = \log_2(2\pi e)$. □

4.1 Positives and negatives

The distribution of false positives and false negatives are Bernoulli distributed random variables conditioned on a particular number of positives. The distribution of positives (and negatives) is given by the following definition.

Definition 4.1. *The number of* positives *in a universe of u elements is uncertain. We model the uncertainty as a discrete random variable, denoted by* P*, with a probability mass function*[5]

$$\mathrm{f}_{\mathrm{P}}(p \mid u) \tag{23}$$

and a support $\{0, \ldots, u\}$*. Conversely, the distribution of negatives is a random variable* $\mathrm{N} = u - \mathrm{P}$ *with a probability mass function*

$$\mathrm{f}_{\mathrm{N}}(n \mid u) = \mathrm{f}_{\mathrm{P}}(u - n \mid u)\,. \tag{24}$$

The form the probability mass function $\mathrm{f}_{\mathrm{P}}(\,\cdot\,)$ takes cannot be a priori specified, although it may be estimated with an empirical probability.

Modeling the distribution of positives provides a complete specification for the distribution of false positives and false negatives (and true positives and true negatives).

Example 2 *The* expected *number of false positives is give by the expectation*

$$\mathrm{E}[\mathrm{FP}] = \sum_{p=0}^{u} \sum_{f_p=0}^{u-p} f_p \cdot \mathrm{f}_{\mathrm{P}}(p \mid u)\, \mathrm{f}_{\mathrm{FP}}(f_p \mid p, u, \varepsilon) \tag{a}$$

$$= \sum_{p=0}^{u} \mathrm{f}_{\mathrm{P}}(p \mid u)\, \mathrm{E}[\mathrm{FP}_p \mid u] = \sum_{p=0}^{u} \mathrm{f}_{\mathrm{P}}(p \mid u)(u-p)\varepsilon \tag{b}$$

$$= \varepsilon\left(u - \sum_{p=0}^{u} m\, \mathrm{f}_{\mathrm{P}}(p \mid u)\right) = \varepsilon\left(u - \mathrm{E}[\mathrm{P}]\right)\,. \tag{c}$$

[5] The probability mass function of a random variable X is denoted by $\mathrm{f}_{\mathrm{X}}(\,\cdot\,)$.

Note that $\mathrm{E}[\mathrm{N}] = u - \mathrm{E}[\mathrm{P}]$*, thus*

$$\mathrm{E}[\mathrm{FP}] = \varepsilon\,\mathrm{E}[\mathrm{N}]\,. \tag{d}$$

The joint probability mass function of positives, false positives, and false negatives is given by

$$\mathrm{f}\big(p, f_p, f_n \mid u, \varepsilon, \omega\big) = \mathrm{f_P}(p \mid u)\,\mathrm{f_{FP}}\big(f_p \mid p, u, \varepsilon\big)\,\mathrm{f_{FN}}(f_n \mid p, u, \omega)\,. \tag{25}$$

Since FP and FN are independent, the joint entropy of P, FP, and FN is given by

$$\mathcal{H}(\mathrm{P}, \mathrm{FP}, \mathrm{FN})` = \mathcal{H}(\mathrm{P}) + \mathcal{H}(\mathrm{FP} \mid \mathrm{P}) + \mathcal{H}(\mathrm{FN} \mid u - \mathrm{P}) \tag{26}$$

$$= \mathcal{H}(\mathrm{P}) + \sum_{p=0}^{u} \mathcal{H}(\mathrm{FP} \mid p) + \sum_{n=0}^{u} \mathcal{H}(\mathrm{FN} \mid n)\,. \tag{27}$$

5 Set estimators

Given an approximate set $\mathcal{S}^{\pm}$ with a false positive rate ε and a false negative rate ω, the *method of moments* estimator of the cardinality of $\mathcal{S}$ is given by

$$\hat{m} = \frac{|\mathcal{S}^{\pm}| - \varepsilon u}{1 - \varepsilon - \omega}\,. \tag{28}$$

Since the optimal space complexity is $-\log_2 \varepsilon$ per element, any data structure that implements an approximate set with a false positive rate ε obtains the maximum entropy, i.e., the maximum entropy per element is given by

$$-\log_2 \varepsilon \text{ bits/element}\,. \tag{29}$$

Thus, the *entropy* of an implementation of the approximate set is given by

$$\frac{H(\mathcal{S}^{-})}{m} = -\log_2 \varepsilon \tag{30}$$

Theorem 5.1. *An unbiased estimator of the cardinality of a* countably infinite *approximate set* $\mathcal{S}^{-}$ *of a set* $\mathcal{S}$ *with a false positive rate* ε *is given by*

$$\hat{m} = \frac{\mathtt{BL}\big(\mathcal{S}^{-}\big)}{b}\,, \tag{31}$$

were `BL` *is the bit length function and* b *is the* expected *bits per element.*

Proof. The *expected* bit length is given by

$$-mb$$

where m is the cardinality of $\mathcal{S}$. Thus, the *method of moments* estimator is given by assuming the bit length realizes the expected value,

$$\mathtt{BL}(\mathcal{S}^{-}) = mb\,. \tag{a}$$

Solving for m results in the estimator

$$\hat{m} = \frac{\mathtt{BL}(\mathcal{S}^{-})}{b}\,. \tag{b}$$

□

Given an approximate set $\mathcal{S}^{\pm}$ with a false positive rate ε and a false negative rate ω, the *method of moments* estimator of the cardinality of $\mathcal{S}$ is given by

$$\hat{m} = \frac{|\mathcal{S}^{\pm}| - \varepsilon u}{1 - \varepsilon - \omega}\,. \tag{32}$$

NOTE: The exact oblivious set bit length reveals nothing about the size of set $\mathcal{S}$ since it only depends on $|\mathcal{U}|$.

By Kerckhoffs's principle, we assume the algorithms are known. For instance, we assume the *space complexity* with respect to the cardinality of the *exact* set is known.

Thus, the *cardinality*, a unary function of $\check{\mathcal{S}}$, may be estimated by using the information about the *expected* bit length. If the expected bit length as a function of m is given by $\mathrm{f}(m)$, where m is the cardinality of the exact set, then a *method of moments* estimator is given by assuming the bit length realizes the expected value,

$$\mathtt{BL}(\check{\mathcal{S}}) = \mathrm{f}(m)\,, \tag{33}$$

and solving for m, resulting in the estimator

$$\hat{m} = \mathrm{f}^{-1}\Big(\mathtt{BL}(\check{\mathcal{S}})\Big)\,. \tag{34}$$

Consider the following example of the *Perfect Hash Filter*[4].

Example 3 *The* optimal Perfect Hash Filter *has a space complexity given by*

$$\mathtt{BL}(\check{\mathcal{S}}) = m \log_2 \frac{e}{\varepsilon}\,. \tag{a}$$

Solving for m results in the estimator

$$\hat{m} = \frac{\mathtt{BL}(\check{\mathcal{S}})}{\log_2 \frac{e}{\varepsilon}}\,. \tag{b}$$

Depending on the entropy of the random bit length of the oblivious set object type, cardinality estimators with very low variance may be obtainable. Thus, in order to increase the uncertainty, we must artificially inflate the bit length of $\check{\mathcal{S}}$. One simple way of achieving this is to randomly sample a positive integer from 0 to N, and insert

6 The *Singular Hash Set*

In what follows, we provide a theoretical implementation of the *oblivious set* that obtains optimality in the following ways:

1. The space complexity obtains the theoretical lower-bound of a positive random approximate set with a false positive rate ε.

2. The entropy obtains the upper bound. This is necessarily the case since it obtains the optimal space complexity.

Definition 6.1 (Cartesian product)**.** *Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ denote n sets. The set $\mathcal{X}_1 \times \cdots \times \mathcal{X}_n = \big\{(x_1, \ldots, x_n)\colon x_1 \in \mathcal{X}_1 \wedge \cdots \wedge x_n \in \mathcal{X}_n\big\}$ is called the Cartesian product of sets $\mathcal{X}_1, \ldots, \mathcal{X}_n$.*

A shorthand notation for the Cartesian product $\mathcal{X} \times \mathcal{X} \times \mathcal{X}$ is denoted by $\mathcal{X}^3$.

The binary set $\{0, 1\}$ is denoted by $\mathcal{B}$. The set of all bit (binary) strings of length n is therefore $\mathcal{B}^n$. The cardinality of $\mathcal{B}^n$ is

$$|\mathcal{B}^n| = 2^n \,. \tag{35}$$

The set of all bit strings of length n or less is denoted by $\mathcal{B}^{\leq n}$, which has a cardinality $2^{n+1} - 1$. The countably infinite set of all bit strings, $\mathcal{B}^{\leq \infty}$, is also denoted by $\mathcal{B}^*$.

The bit length of an object x is denoted by

$$\texttt{BL}(x) \,, \tag{36}$$

e.g., the bit length of any $x \in \mathcal{B}^n$ is $\texttt{BL}(x) = n$.

A (convenient) one-to-one correspondence between $\mathcal{B}$ and $\mathbb{N}$ is given by the following definition.

Definition 6.2. *Let the set of bit strings $\mathcal{B}^*$ and the set of natural numbers $\mathbb{N}$ have the bijection given by*

$$(b_1\, b_2\, \cdots\, b_m) \longleftrightarrow 2^m + \sum_{j=1}^{m} 2^{m-j} b_j \,. \tag{37}$$

We denote the mapping of a bit string (or natural number) x by x'.

An important observation of this mapping is that a natural number n maps to a bit string n' of length $\texttt{BL}(n') = \lfloor \log_2 n \rfloor$.

Example 4 *Consider the number* 100 *(base* 10*). To determine the bit string* $100'$*, we perform the following steps:*

1. *The length of* $100'$ *is* $\lfloor \log_2 100 \rfloor = 6$.
2. *Subtract* $2^6 = 64$ *from* 100 *which results in* 36.
3. 36_{10} *in binary is* 100100_2.
4. $100' = (1\,0\,0\,1\,0\,0)$.

The Singular Hash Set (SHS) is a data type that implements the *oblivious* random positive approximate set abstract data type. The implementation consists of a product data structure (tuple) $\mathcal{B}^k \times \mathcal{B}^*$, an algorithm that *generates* the data structure, and an algorithm that implements the *member-of* function by appropriately *querying* the data structure.

A *hash function* is related to countable sets $\mathcal{B}^*$ and $\mathcal{B}^n$ and is given by the following definition.

Definition 6.3. *A* hash function $\texttt{h} \colon \mathcal{B}^* \mapsto \mathcal{B}^n$ *is a function such that all bit strings of arbitrary-length are mapped (hashed) to bit strings of fixed-length n. For a given $x \in \mathcal{B}$, $y = \texttt{h}(x)$ is denoted the* hash *of x.*

The SHS assumes that random oracles are available.

Definition 6.4. *A random oracle, denoted by $\texttt{h}^* \colon \mathcal{B}^* \mapsto \mathcal{B}^\infty$, is a theoretical* hash function *whose output is uniformly distributed over the elements of $\mathcal{B}^\infty$.*

The implementation of the algorithm that generates the data structure for the SHS that implements an *approximate oblivious set* is given by the following theorem.

Theorem 6.1 (SHS)**.** *Algorithm 1 implements the regular function*

$$\texttt{make\_singular\_hash\_set}\colon\ \mathcal{P}(\mathcal{U}) \times [\varepsilon] \mapsto \mathcal{B}^k \times \mathcal{B}^*\,, \tag{38}$$

where $[\varepsilon] = \left\{2^{-k} \colon k \in \mathbb{N}\right\}$*, and the member-of function*

$$\texttt{contains}\colon\ \mathcal{B}^k \times \mathcal{B}^* \mapsto \{\mathbf{true}, \mathbf{false}\} \tag{39}$$

has an implementation given by **??***, a generator of* positive approximate oblivious sets *over the universe* $\mathcal{U}$*, i.e.,*

The product type *generated by t*

$$\texttt{make\_singular\_hash\_set}(\mathcal{S}, \varepsilon) \tag{40}$$

is a positive approximate oblivous set of $\mathcal{S}$ *with a* false positive rate

Proof. In order for the Singular Hash Set to be a positive oblivious set $\check{\mathcal{S}}^-(\varepsilon)$, it must also be a random positive approximate set (RAS^+) $\mathcal{S}^-(\varepsilon)$ and thus must satisfy the two conditions given by definition 2.2:

1. $\mathcal{S}$ is a subset of $\check{\mathcal{S}}^-$. This condition guarantees that no *false negatives* may occur.
2. An element in $\mathcal{U}$ that is not a member of $\mathcal{S}$ is a member of $\check{\mathcal{S}}^-$ with a probability ε, denoted the false positive rate, i.e.,
$$\mathrm{P}\left[x \in \check{\mathcal{S}}^- \,\middle|\, x \notin \mathcal{S}\right] = \varepsilon \tag{a}$$
for any $x \in \mathcal{U}$.

To prove the first condition, note that algorithm 3 tests any element x for membership in $\mathcal{S}^-$ by computing the hash of x concatenated with the bit string b_n and returning **true** if the hash is h_k where algorithm 1 finds bit strings b_n and h_k such that each element of $\mathcal{S}$ concatenated with b_n hashes to h_k.

To prove the second condition, suppose we have a set $\mathcal{S} = \{x_1, \ldots, x_m\}$ and each element in $\mathcal{S}$ hashes to $y = \mathtt{h}(x_1)$. By **??**, $\mathtt{h}\colon \mathcal{B}^* \mapsto \mathcal{B}^k$ approximates a random oracle and thus uniformly distributes over its domain of 2^k possibilities. Since y is a particular element in $\mathcal{B}^k$, the probability that an element not in $\mathcal{S}$ hashes to y is 2^{-k}. $\square$

Condition for maxium entropy:

The countably infinite intersection of separate instances of an approximate oblivious set of $\mathcal{S}$ where $\omega > 0$ is the empty set,

$$\bigcap_j \check{\mathcal{S}}^j = \varnothing\,, \tag{41}$$

where $\check{\mathcal{S}}^1, \check{\mathcal{S}}^2, \ldots$ are random instances of an approximate oblivious set of $\mathcal{S}$.

TODO: this directly follows from the sampling distribution and taking the limit.

In the case of a positive approximate oblivious set, where $\omega = 0$, the countably infinite intersection of separate instances of a positive approximate oblivious set of $\mathcal{S}$ is $\mathcal{S}$,

$$\bigcap_j \check{\mathcal{S}}_j^- = \mathcal{S}\,. \tag{42}$$

Algorithm 1: Implementation of `make_singular_hash_set` over a universal set $\mathcal{U}$

in : $\mathcal{S}$ is a subset of a universal set $\mathcal{U}$. ε is the false positive rate.

out : An *oblivious* positive approximate set of $\mathcal{S}$.

function `make_singular_hash_set`($\mathcal{S}$, ε)
    $\mathcal{S}_{\mathcal{B}} \leftarrow \{\texttt{encode}_{\mathcal{U}\mapsto\mathcal{B}}(x) \colon x \in \mathcal{S}\}$
    for $n \leftarrow 0$ **to** ∞ **do**
        for $j \leftarrow 1$ **to** 2^n **do**
            found $\leftarrow$ **true**
            `// To maximize` *entropy* `we try bit strings of length` n `in random order.`
            $b_n \leftarrow$ a bit string of length n randomly drawn from $\mathcal{B}_n$ without replacement
            $h_k \leftarrow$ **null**
            for $x \in \mathcal{S}_{\mathcal{B}}$ **do**
                if $h_k =$ **null then**
                    $h_k \leftarrow \texttt{h}(x + b_n) \mod (k+1)$
                end
                else if $h \neq \texttt{h}(x + b_n) \mod (k+1)$ **then**
                    found $\leftarrow$ **false**
                end
            end
            if found **then**
                `// This tuple is the data structure of the Singular Hash Set.`
                return (h_k, b_n)
            end
        end
    end

Algorithm 2: Implementation of `make_singular_hash_set` over a universal set $\mathcal{U}$

param : k is any number in the set $\{1, 2, \ldots\}$.
in : $\mathcal{S}$ is a subset of a finite universal set $\mathcal{U}$.
out : An *oblivious* exact set of $\mathcal{S}$.

```
function make_singular_hash_set(S; k)
    S_B ← {encode_{U↦B}(x) : x ∈ S}
    S̄_B ← {encode_{U↦B}(x) : x ∈ S̄}
    // To find the smallest bit string, search for a bit string of length n in ascending order.
    for n ← 0 to ∞ do
        for j ← 1 to 2^n do
            found ← true
            // To maximize entropy we try bit strings of length n in random order.
            b_n ← a bit string of length n randomly drawn from B_n without replacement
            h_k ← null
            for x ∈ S_B do
                if h_k = null then
                    h_k ← h(x ++ b_n) mod (k + 1)
                end
                else if h_k ≠ h(x ++ b_n) mod (k + 1) then
                    found ← false
                end
            end
            for x ∈ S̄_B do
                if h_k = h(x ++ b_n) mod (k + 1) then
                    found ← false
                end
            end
            if found then
                return (h_k, b_n)
            end
        end
    end
```

A countable union of separate instances of a negative approximate oblivious set of $\mathcal{S}$ is the complement of $\mathcal{S}$

$$\bigcap_j \check{\mathcal{S}}_j^- = \overline{\mathcal{S}}\,. \tag{43}$$

Exact oblivious set:

$$k(\rho u - 1) + (u \log_2(1 - 2^{-k})(\rho - 1) \tag{44}$$

$$k(\rho u - 1)/m + (u \log_2(1 - 2^{-k})(\rho - 1)/m \tag{45}$$

$$k\left(1 - \frac{1}{u\rho}\right) + \log_2\left(1 - 2^{-k}\right)\left(1 - \frac{1}{\rho}\right) \tag{46}$$

$$k(\rho - \frac{1}{u}) + \log_2(1 - 2^{-k}(\rho - 1) \text{ bits/elementinuniversalset} \tag{47}$$

Theorem 6.2.

$$\min_k k(\rho - \frac{1}{u}) + \log_2\Big(1 - 2^{-k}(\rho - 1)\Big) \tag{48}$$

bits per element in the universal set $\mathcal{U}$ *where* $u = |\mathcal{U}|$.

The above algortihm for generating exact oblivious sets has a space complexity of u bits. Assuming m is not too much smaller than u, i.e., $\mathcal{S}$ is dense, this is the theoretical lower-bound for sets over finite universes.

If m is not dense, then trying larger singular hash lengths will result in a better lower-bound.

Suppose you are interested in creating exact oblivious sets over the universe of bit strings up to length n, denoted by $\mathcal{B}_{\leq n}$. Then, there are $|\mathcal{B}_{\leq n}| = 2^{n+1} - 1$ bit strings in the universe. In this case, the set has a lower-bound given by $\mathcal{O}(n)$.

Algorithm 3: Implementation of `contains`

in : $\check{\mathcal{S}}^-$ is the Singular Hash Set (product type) to query and x is the element to test for membership.

out : **true** if $x \in \check{\mathcal{S}}^-$ otherwise **false**. If we rephrase this with respect to $\mathcal{S} \subset \check{\mathcal{S}}^-$, then **true** if $x \in \mathcal{S}$ and otherwise **true** with probability ε and **false** with probability $1 - \varepsilon$, where ε is the false positive rate of $\check{\mathcal{S}}^-$.

function `contains(`$\check{\mathcal{S}}^-$,x`)`
- // The Singular Hash Set $\check{\mathcal{S}}^-$ is coded by the tuple (b_n, h_k), where b_n is a bit string of length n and h_k is the singular hash.
- $k = \texttt{BL}(h_k)$
- **if** $\texttt{h}(x + b_n) \mod k = h_k$ **then**
  - // False positives occur with probability $\varepsilon = 2^{-k}$.
  - **return true**
- **else**
  - **return false**

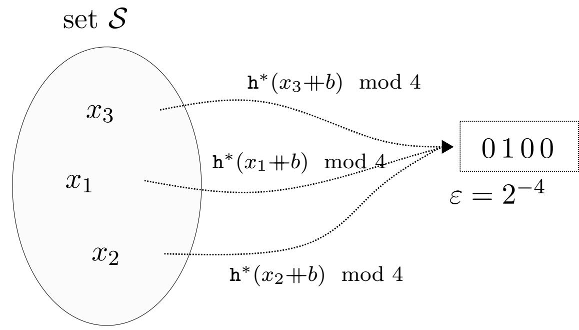

Figure 1: *Singular Hash Set* over a *countably infinite* universe

6.1 Space complexity

The probability that every element of $\mathcal{S}$ collides for a particular bit string in **??** is given by the following theorem.

Theorem 6.3. *The probability that a bit string $b \in \mathcal{B}$ results in perfect collision in* **??** *is given by*

$$\varepsilon^{m-1}\,, \tag{49}$$

where $m = |\mathcal{S}|$ and ε is the specified false positive rate.

Proof. Suppose we have a set $\mathcal{S} = \{x_1, \ldots, x_m\}$ and x_1 hashes to $y = \mathtt{h}_{\mathcal{S}}(x_1)$, where $\mathtt{h}_{\mathcal{S}} \colon \mathcal{B}^* \mapsto \mathcal{B}^k$ is a random hash function that uniformly distributes over its domain of 2^k possibilities. Since y is a particular element in $\mathcal{B}^k$, the probability that x_j for $j = 2, \ldots, m$ hashes to y is given by

$$\frac{1}{2^k} = \varepsilon\,. \tag{a}$$

Since $\mathtt{h}_{\mathcal{S}}$ is a random hash function, the hashes of $x_1, \ldots, x_m$ are independent. Thus, the joint probability that $x_2, \ldots, x_m$ hash to y is given by the product of their marginal probabilities

$$\varepsilon^{m-1}\,. \tag{b}$$

□

Theorem 6.4. *The expected bit length of the Singular Hash Set obtains the information-theoretic lower-bound given by*

$$-\log_2 \varepsilon \text{ bits/element}\,, \tag{50}$$

where ε is the false positive rate.

Proof. The space required for the Singular Hash Set found by algorithm 1 is of the order of the length n of the bit string b_n. Therefore, for space efficiency, the algorithm exhaustively searches for a bit string in the order of increasing size n.

We are interested in the first case when a perfect collision occurs, which is a geometric distribution with probability of success ε^{m-1} as given by the discrete random variable

$$\mathrm{Q} \sim \mathrm{GEO}\Big(\varepsilon^{m-1}\Big)\,. \tag{a}$$

By definition 6.2, the n^{th} trial uniquely maps to a bit string of length $m = \lfloor \log_2 n \rfloor$. Thus, the bit string is a random length given approximately by

$$\mathrm{N} = \log_2 \mathrm{Q} \text{ bits}\,. \tag{b}$$

This is a slight *overestimate* since we are simplifying by avoiding the floor function.

We approximate the logarithm with a second-order Taylor series around the *expected* value of Q as given by

$$\mathrm{N} \approx \log_2 \mathrm{E}[\mathrm{Q}] - \frac{\log_2 e}{\mathrm{E}[\mathrm{Q}]} \left(\mathrm{Q} - \mathrm{E}[\mathrm{Q}]\right)^2 \text{ bits}\,. \tag{c}$$

We are interested in the *expected* value of N,

$$\mathrm{E}[\mathrm{N}] \approx \log_2 \mathrm{E}[\mathrm{Q}] - \frac{\log_2 e}{\mathrm{E}[\mathrm{Q}]} \mathrm{E}\left[\mathrm{Q} - \mathrm{E}[\mathrm{Q}]\right]^2 \text{ bits}\,. \tag{d}$$

The variance of Q is given by

$$\mathrm{var}[\mathrm{Q}] = \mathrm{E}\left[\mathrm{Q} - \mathrm{E}[\mathrm{Q}]\right]^2 , \tag{e}$$

and thus we may rewrite eq. (d) as

$$\mathrm{E}[\mathrm{N}] \approx \log_2 \mathrm{E}[\mathrm{Q}] - \frac{\log_2 e}{\mathrm{E}[\mathrm{Q}]} \mathrm{var}[\mathrm{Q}] \text{ bits}\,. \tag{f}$$

Since Q is geometrically distributed with a probability of success ε^{m-1}, the expectation and variance of Q is known to be

$$\mathrm{E}[\mathrm{Q}] = \varepsilon^{-(m-1)} \tag{g}$$

and

$$\mathrm{var}[\mathrm{Q}] = \frac{1 - \varepsilon^{m-1}}{\left(\varepsilon^{m-1}\right)^2}\,. \tag{h}$$

Plugging these values into eq. (f) yields

$$\mathrm{E}[\mathrm{N}] \approx \log_2 \varepsilon^{-(m-1)} - \frac{\log_2 e}{\varepsilon^{-(m-1)}} \frac{1 - \varepsilon^{(m-1)}}{\left(\varepsilon^{m-1}\right)^2} \tag{i}$$

$$= -(m-1) \log_2 \varepsilon + \left(1 - \varepsilon^{-(m-1)}\right) \log_2 e \text{ bits}\,. \tag{j}$$

We are interested in the bits per element. There are m elements, so dividing by m results in

$$-\frac{m-1}{m} \log_2 \varepsilon + \frac{1 - \varepsilon^{-(m-1)}}{m} \text{ bits/element}\,. \tag{k}$$

Asymptotically, as $m \to \infty$, the expected bits per element goes to

$$-\log_2 \varepsilon\,. \tag{l}$$

□

By **??**, **??** has an expected time complexity that grows exponentially as m grows The algorithm is intended to illustrate theoretical properties, not necessarily be used in practice.

6.2 Entropy

Proof.

$$\mathcal{H}(\mathrm{N}) = -\,\mathrm{E}\big[\log_2 \mathrm{f_N}(\mathrm{N})\big]\,. \tag{a}$$

$$\mathcal{H}(\mathrm{N}) = -\sum_{n=0}^{\infty} \log_2 \mathrm{f_N}(n)\, \mathrm{f_N}(n) \tag{b}$$

$$= -\sum_{n=0}^{\infty} \log_2\left(q^{2^n-1}\left(1-q^{2^n}\right)\right) \mathrm{f_N}(n)\,. \tag{c}$$

$$\begin{aligned}\mathcal{H}(\mathrm{N}) = -\,\log_2 q \sum_{n=0}^{\infty}(2^n-1)\,\mathrm{f_N}(n) - \\ \sum_{n=0}^{\infty}\log_2\left(1-q^{2^n}\right)\mathrm{f_N}(n)\,.\end{aligned} \tag{d}$$

$$\begin{aligned}\mathcal{H}(\mathrm{N}) = -\,\log_2 q \left[\sum_{n=0}^{\infty}(2^n\,\mathrm{f_N}(n)\right] - \\ \sum_{n=0}^{\infty}\log_2\left(1-q^{2^n}\right)\mathrm{f_N}(n)\,.\end{aligned} \tag{e}$$

□

Theorem 6.5. *The random bit string that codes the Singular Hash Set is a* maximum entropy *coder for the* approximate set *abstract data type.*

Proof. The result immediately follows from the fact that the bit string is *incompressible*, and thus obtains maximum entropy. The hash h_k is the result of a random oracle and is thus uniformly distributed and, given that a bit string of length n is found, the probability that a particular b_n is found is uniformly distributed also. The only part of this that does not obtain maximum entropy is the particular bit length n of b_n. □

The only information contained in the encoding of the approximate set is given by the probability mass function of the random bit length N, which is a function of the cardinality of the objective set being approximated and the false positive rate ε.

Given a set $\mathcal{S}$, the random bit length N of bit string b_n has a probability mass concentrated around the theoretical lower-bound. Therefore, a *method-of-moments* estimator of the cardinality of the set $\mathcal{S}$ is given by

$$|\hat{\mathcal{S}}| = -\frac{\mathtt{BL}\left(\mathcal{S}^-\right)}{\log_2 \varepsilon}\,, \tag{51}$$

were `BL` is the bit length function and ε is the *false positive rate*.

We may trade space complexity for entropy if desired. For instance, if the policy is to search for a bit string of a length t much larger than the expected bit length, $-m\log_2 \varepsilon$, then

$$m < -\frac{t}{\log_2 \varepsilon}\,. \tag{52}$$

Thus, an estimator of the upper-bound on the cardinality is given by

$$\hat{m}_{\max} \approx -\frac{t}{\log_2 \varepsilon} \tag{53}$$

and an estimate of the *lower-bound* is 0 (the empty set) where the cardinality is uniformly distributed (maximum entropy) between 0 and $\hat{m}_{\max}$, which has an entropy given by

$$\log_2(1 + \hat{m}_{\max})\,. \tag{54}$$

The absolute space efficiency is now given by

$$\frac{m}{\hat{m}_{\max}}\,, \tag{55}$$

which is the *maximum* efficiency possible for an approximate set with a cardinality uniformly distributed between 0 and $\hat{m}_{\max}$.

Example 5 *Suppose $t = rm, r > 1$, then*

$$\hat{m}_{\max} \approx -\frac{rm}{\log_2 \varepsilon} \tag{a}$$

which has an entropy given by

$$\log_2\left(1 - \frac{rm}{\log_2 \varepsilon}\right) \approx \log_2 r + \log_2 m + \log_2 \frac{1}{\varepsilon} \tag{b}$$

and an absolute space efficiency r.

Appendices

A Probability mass of random bit length

For a particular n, each $b_n \in \mathcal{B}^n$ may fail to result in a perfect collision, therefore n is uncertain and takes on particular values with probabilities given by the following theorem.

Definition A.1. *The Singular Hash Set generator given by* **??** *finds a random string of a random bit length given by*

$$\mathrm{N} = \texttt{bit\_length\_sampler}(m, \varepsilon)\,, \tag{56}$$

conditioned on a random set of cardinality m and a false positive rate ε.

Theorem A.1. *The random bit length* N *has a probability mass function given by*

$$\mathrm{f}_{\mathrm{N}}(n \mid m, \varepsilon) = q^{2^n - 1}\left(1 - q^{2^n}\right)\,, \tag{57}$$

where $q = 1 - \varepsilon^{m-1}$, m is the cardinality of the random set, and ε is the false positive rate.

Algorithm 4: Bit length sampler of Singular Hash Set

in : m is the cardinality of the random set to approximate and ε is the false positive rate.

out : A *minimum bit length* n of a Singular Hash Set conditioned on a random set with cardinality m and a false positive rate ε.

function `bit_length_sampler(`m,ε`)`:
- $\mathcal{S} \leftarrow \emptyset$;
- **for** $i \leftarrow 1$ **to** m **do**
  - $x \leftarrow$ randomly draw a bit string from $\mathcal{B}^*$ without replacement;
  - $\mathcal{S} \leftarrow \mathcal{S} \cup \{x\}$;
- **end**
- $(h_k, b_n) \leftarrow$ `make_singular_hash_set(`$\mathcal{S}$, ε`)`;
- **return** $k + n + \mathcal{O}(1)$;

Proof. Each iteration of the loop in **??** has a collision test which is Bernoulli distributed with a probability of success ε^{m-1}, where success denotes a perfect collision. We are interested in the random length N of the bit string when this outcome occurs.

For the random variable N to realize a value n, every bit string smaller than length n must fail and a bit string of length n must succeed. There are $2^n - 1$ bit strings smaller than length n and each one fails with probability q, and so by the product rule the probability that they all fail is given by

$$q^{2^n-1}\,. \tag{a}$$

Given that every bit string smaller than length n fails, what is the probability that every bit string of length n fails? There are 2^n bit strings of length n, each of which fails with probability q as before and thus by the product rule the probability that they all fail is q^{2^n}, whose complement, the probability that not all bit strings of length n fail, is given by

$$1 - q^{2^n}\,. \tag{b}$$

By the product rule, the probability that every bit string smaller than length n fails and a bit string of length n succeeds is given by the product of (a) and (b),

$$q^{2^n-1}\left(1 - q^{2^n}\right)\,. \tag{c}$$

For Equation (c) to be a probability mass function, two conditions must be met. First, its range must be a subset of $[0, 1]$. Second, the summation over its domain must be 1.

The first case is trivially shown by the observation that q is a positive number between 0 and 1 and therefore any non-negative power of q is positive number between 0 and 1.

The second case is shown by calculating the infinite series

$$S = \sum_{n=0}^{\infty} q^{2^n-1}\left(1 - q^{2^n}\right) \tag{d}$$

$$= \sum_{n=0}^{\infty} q^{2^n-1} - q^{2^{n+1}-1}\,. \tag{e}$$

Explicitly evaluating this series for the first 4 terms reveals a telescoping sum given by

$$S = (1 - q) + (q - q^3) + (q^3 - q^7) + (q^7 - q^{15}) + \cdots\,, \tag{f}$$

where everything cancels except 1.

Note that a simpler proof is given by

$$\mathrm{N} = \log_2 \mathrm{Q}\,. \tag{g}$$

Thus, the probability mass function is given by

$$\begin{aligned}
\mathrm{f_N}(n \mid m, \varepsilon) &= \mathrm{P}[\mathrm{N} = n] && \text{(h)}\\
&= \mathrm{P}[\log_2 \mathrm{Q} = n] && \text{(i)}\\
&= \mathrm{P}[\mathrm{Q} = 2^n] && \text{(j)}\\
&= \mathrm{f_Q}(2^n \mid m, \varepsilon) && \text{(k)}\\
&= \varepsilon^{m-1}(1 - \varepsilon^{m-1})^{2^n} && \text{(l)}
\end{aligned}$$

□

References

[1] Alexander Towell. The oblivious set abstract data type. . URL ?

[2] Alexander Towell. Secure index abstract data types for boolean encrypted search. . URL ?

[3] Alexander Towell. The algebraic model of the random approximate set. 2018.

[4] Alexander Towell. The perfect hash filter: an efficient, immutable implementation of the positive random approximate set abstract data type. 2018.