Basic quantum subroutines: finding multiple marked elements and summing numbers

Joran van Apeldoorn[1], Sander Gribling[2], and Harold Nieuwboer[3]

[1]IViR and QuSoft, University of Amsterdam, The Netherlands
[2]Department of Econometrics and Operations Research, Tilburg University, Tilburg, The Netherlands
[3]Korteweg–de Vries Institute for Mathematics and QuSoft, University of Amsterdam, The Netherlands and Faculty of Computer Science, Ruhr University Bochum, Germany and Department of Mathematical Sciences, University of Copenhagen, Denmark

We show how to find all k marked elements in a list of size N using the optimal number $O(\sqrt{Nk})$ of quantum queries and only a polylogarithmic overhead in the gate complexity, in the setting where one has a small quantum memory. Previous algorithms either incurred a factor k overhead in the gate complexity, or had an extra factor $\log(k)$ in the query complexity.

We then consider the problem of finding a multiplicative δ-approximation of $s = \sum_{i=1}^{N} v_i$ where $v = (v_i) \in [0,1]^N$, given quantum query access to a binary description of v. We give an algorithm that does so, with probability at least $1 - \rho$, using $O(\sqrt{N \log(1/\rho)/\delta})$ quantum queries (under mild assumptions on ρ). This quadratically improves the dependence on $1/\delta$ and $\log(1/\rho)$ compared to a straightforward application of amplitude estimation. To obtain the improved $\log(1/\rho)$ dependence we use the first result.

1 Introduction

1.1 Finding multiple marked elements in a list

Grover's famous search algorithm [Gro96] can be used to find a marked element in a list quadratically faster than possible classically. Formally it can be used to solve the following problem: given a bit string $x \in \{0,1\}^N$, $x \neq 0$, find an index $i \in [N]$ such that $x_i = 1$.

In this work we consider the problem of finding *all* indices $i \in [N]$ for which $x_i = 1$. We give a query-optimal quantum algorithm with polylogarithmic gate overhead in the setting where one has a *small quantum memory*. We explain below why this last assumption makes the problem non-trivial. This improves over the previous state-of-the-art: previous algorithms were either query-optimal but with a polynomial gate overhead, or had a polylogarithmic gate overhead but also a logarithmic overhead in the query count.

A well-known query-optimal algorithm for the problem is as follows [dGdW02, Lem. 2]. Let k be the Hamming weight $|x| := \sum_{i=1}^{N} x_i$ of x. For ease of exposition, suppose the algorithm knows k. (For our results we will work with weaker assumptions such as knowing only an upper bound on k, or an estimate of it, see Section 3. We also ignore failure probabilities in this part of

Joran van Apeldoorn: work@bitofbytes.com
Sander Gribling: s.j.gribling@tilburguniversity.edu
Harold Nieuwboer: hani@math.ku.dk

the introduction.) A variant of Grover's algorithm [BBHT98] can find a single marked element using $O(\sqrt{N/k})$ quantum queries and $O(\sqrt{N/k}\log(N))$ additional single- and two-qubit gates. One can then find all k marked elements using

$$O\Big(\sqrt{N/k} + \sqrt{N/(k-1)} + \ldots + \sqrt{N}\Big) = O\Big(\sqrt{Nk}\Big)$$

quantum queries to x. The above complexity is obtained as follows. Suppose we have already found a set $J \subseteq [N]$ of marked elements. Then to find a new marked element, we replace x by the string $z \in \{0,1\}^N$ defined as

$$z_i = \begin{cases} x_i & \text{if } i \notin J, \\ 0 & \text{otherwise.} \end{cases}$$

A quantum query to z can be made using a single quantum query to x and quantum query to J (which on input $|i\rangle|b\rangle$ for $i \in [N], b \in \{0,1\}$ returns $|i\rangle|b \oplus \delta_{i\in J}\rangle$ where $\delta_{i\in J} \in \{0,1\}$ is one iff $i \in J$). In particular, if J can be stored in a quantum memory (i.e. queried and updated in unit time), then the query complexity will be $O(\sqrt{Nk})$ and the time complexity is $\widetilde{O}(\sqrt{Nk})$. We refer the interested reader to [GLM08] and [CHI+18, Sec. 5] for a discussion of quantum memory and its (dis)advantages.

However, when we cannot store J in a quantum memory, a naive implementation of the quantum queries to J is expensive in terms of gate complexity: if $|J| = s$, then one can use $O(s\ \log(N))$ quantum gates to implement a single query to J.[1] Since the size of J grows to k, the total gate complexity of finding all marked elements will scale as $\widetilde{O}(\sqrt{N}k^{3/2})$, which is a factor k larger than the query complexity. We show that this factor of k in the gate complexity can be avoided: we give an algorithm that finds, with large probability, all k indices using the optimal number of quantum queries to x, $O(\sqrt{Nk})$, while incurring only a polylogarithmic overhead in the gate complexity, in the case where we only have a small quantum memory. We state a simplified version of our main result below; for the full version, see Theorem 3.9 and the corresponding algorithm `GroverMultipleFast`.

Theorem 1.1. *Let $x \in \{0,1\}^N$ with $|x| = k \geq 2$, and let $\rho \in (0,1)$ be such that $k \in \Omega(\log(k/\rho)^3)$ (e.g. $\rho = \Omega(1/\mathrm{poly}(k))$). Then we can find, with probability $\geq 1-\rho$, all k indices $i \in [N]$ for which $x_i = 1$ using $O(\sqrt{Nk})$ quantum queries and $O(\sqrt{Nk}\log(k)^3\log(N))$ additional gates.*

We mention that by a simple coupon-collector argument one can already achieve both query- and gate-complexity $\sqrt{Nk}\,\mathrm{polylog}(N, 1/\rho)$, see Proposition 3.7. Our algorithm completely removes the polylog(N) factors in the query complexity and moreover has a much improved dependence on $\log(1/\rho)$: one can achieve $\rho = 1/\mathrm{poly}(k)$ without increasing the number of quantum queries made by the algorithm. In the same spirit, we mention that previous work had already shown that simply boosting a constant success probability is not optimal for finding a single marked element: one can do so with probability $\geq 1-\rho$ using $\sqrt{N\log(1/\rho)}$ quantum queries [BCdWZ99].

In a nutshell, our algorithm is a hybrid between the quantum coupon-collector and the query-optimal algorithm described above. First, we use the coupon collection strategy to find t marked indices $1 \leq i_1 < \cdots < i_t \leq n$, for t roughly $k/\log(k)^2$. A basic property of this strategy is that the resulting indices $\{i_1, \ldots, i_t\}$ yield a uniformly random subset of size t of the marked indices in x. Next, for every $j \in [t+1]$, we use the query-optimal algorithm to find all remaining marked elements in the interval $(i_{j-1}, i_j) \subseteq [n]$, where we write $i_0 = 0$ and $i_{t+1} = n+1$. With

[1] We ignore here the cost of maintaining a classical data structure for J, but comment on this again later.

high probability over the found indices $\{i_1, \ldots, i_t\}$, each of the intervals (i_{j-1}, i_j) contains few remaining marked indices, which reduces the effect of the high gate-complexity overhead of the query-optimal search algorithm.

1.2 Improved quantum summing algorithm

Given quantum query access to a binary description of $v \in [0,1]^N$, how difficult is it to obtain, with probability $\geq 1-\rho$, a multiplicative δ-approximation[2] of the sum $s = \sum_{i=1}^N v_i$? We provide an algorithm to do so whose complexity can be tuned by choosing a parameter $p \in (0,1)$; one special case of our second main result is as follows, see Theorem 4.3 for the full version. In the version below we have made very mild assumptions on the failure probability ρ and precision δ, which essentially correspond to the regime in which one makes at most $O(N)$ quantum queries.

Theorem 1.2 (Informal version of Theorem 4.3)**.** *Let $v \in [0,1]^N$. Let $\rho, \delta \in (0,1)$ be such that $\log(1/\rho)/\delta = O(N)$. Then we can find, with probability $\geq 1-\rho$, a multiplicative δ-approximation of $\sum_{i=1}^N v_i$ using*

$$O\left(\sqrt{\frac{N}{\delta}\log(1/\rho)}\right) \tag{1.1}$$

quantum queries to binary descriptions of the entries of v, and a gate complexity which is larger by a factor polylogarithmic in N, $1/\delta$ and $1/\rho$.

In a nutshell, our algorithm first finds all indices of "large enough" entries of the v using `GroverMultipleFast` and sums the corresponding elements classically. It then rescales the remaining "small enough" elements and uses amplitude estimation [BHMT02] to approximate their sum. To determine what "large enough" means, we use a recent quantum quantile estimation procedure from [Ham21]. Choosing the quantile carefully controls both the number of elements that need to be found in the first stage, as well as the size of the elements that remain to be summed in the second stage. Note that it is the above version of Grover's algorithm that allows us to obtain a query complexity with only a $\sqrt{\log(1/\rho)}$-dependence, and without additional polylogarithmic factors in N and δ. Indeed, the fact that the number of quantum queries required to find multiple marked elements does not depend on $\log(1/\rho)$ (for ρ not too small) allows us to balance the complexities of the two stages.[3]

The problem we consider can be viewed as a special case of the mean estimation problem, or as a generalization of the approximate counting problem for binary strings $x \in \{0,1\}^N$. We briefly discuss how our results compare to prior work on those problems.

Mean estimation algorithms. After multiplying the v_i by a factor $\frac{1}{N}$, we can interpret the problem of finding a multiplicative δ-approximation of the sum $s = \sum_{i=1}^N v_i$ as the problem of obtaining a multiplicative δ-approximation of the mean $\mu = \frac{1}{N}\sum_{i=1}^N v_i$ of the random variable that, for each $i \in [N]$, takes value v_i with probability $1/N$. Quantum algorithms for the mean estimation problem date back to the work of Grover [Gro97, Gro98]. A careful application of maximum finding and quantum amplitude estimation yields such an approximation of μ, with

[2] Here we use the convention that a multiplicative δ-approximation of a real number s is a real number $\tilde{s}$ for which $(1-\delta)s \leq \tilde{s} \leq (1+\delta)s$.

[3] For completeness we mention that if you *do* allow a large quantum memory, or a polynomial overhead in the gate complexity, then the same $\sqrt{\log(1/\rho)}$-dependence can be obtained using an analogous approach as the one we use to prove Theorem 4.3, but instead relying on, e.g., one of the versions of Grover's algorithm discussed in Section 1.1.

probability $\geq 1-\rho$, using $O(\frac{\sqrt{N}}{\delta}\log(1/\rho))$ quantum queries and polylogarithmic gate overhead, see Theorem 2.9. We improve the dependence on δ from $1/\delta$ to $1/\sqrt{\delta}$.

As for applications, we note that Theorem 2.9 was used to give quantum speedups for the matrix scaling problem in [vAGL+21, GN22], where it is used to approximate the row- and column sums of a matrix with non-negative entries. This is one of their main sources of quantum speedup, and the quality of this approximation directly affects the achievable precision for the matrix scaling problem. Using the improved quantum summing subroutine of Theorem 4.3, the dependence on the desired precision ε for the matrix scaling problem is further improved. More precisely, if $A \in \mathbb{R}_{\geq 0}^{N\times N}$ is an $N \times N$ matrix with non-negative entries, let $r(A) \in \mathbb{R}_{\geq 0}^{N}$ denote its *vector of row sums*, i.e., $r_i(A) = \sum_{j=1}^{N} A_{ij}$. Then given quantum query access to A, using the improved summing subroutine, one can with $\widetilde{O}(N^{1.5}/\sqrt{\delta})$ queries compute a vector $\hat{r} \in \mathbb{R}^n$ such that

$$\|\hat{r} - r(A)\|_1 \leq \delta \|r(A)\|_1.$$

Computing such an $\hat{r}$ with $\delta = \varepsilon^2$ is the bottleneck in the second-order method for matrix scaling presented in [GN22]. By reducing the complexity of this step, this method is improved to become better than the fastest *classical* first-order method (Sinkhorn's algorithm) for entrywise-positive matrices: the classical method finds an ε-ℓ^1-scaling of entrywise positive matrices in time $\widetilde{O}(N^2/\varepsilon)$, whereas the box-constrained Newton method now runs in time $\widetilde{O}(N^{1.5}/\varepsilon)$. Note that this gives an algorithm for matrix scaling whose runtime is sublinear in the input size when $1/\varepsilon = o(\sqrt{N})$, corresponding to $1/\delta = o(N)$, which is precisely the regime of δ for which the quantum subroutine improves over classical summing.

We remark that faster mean estimation algorithms have been developed for example for random variables with a small variance σ^2. Indeed, the current state of the art obtains a multiplicative δ-approximation, with probability $\geq 1-\rho$, using $\widetilde{O}((\frac{\sigma}{\delta\mu} + \frac{1}{\sqrt{\delta\mu}})\log(1/\rho))$ quantum queries in expectation [Ham21, KO23].[4] For comparison, we mention that $\sigma \leq \sqrt{\mu(1-\mu)}$ always holds, and when given binary access to the v_i, one may additionally assume (after maximum-finding and rescaling) that $\mu \in [1/N, 1]$. The second term in the complexity is then at most $\sqrt{N/\delta}\log(1/\rho)$ (i.e. at most our bound when we ignore the ρ-dependence). The first term, however, is larger than our complexity if and only if $\delta N \leq (\sigma/\mu)^2$ (again ignoring ρ). Our algorithm thus improves over prior mean estimation algorithms when the support is relatively small: when δN is at most $(\sigma/\mu)^2$.

Approximate counting algorithms. As mentioned above, our algorithm improves the error-dependence for mean estimation (for random variables with small support). It therefore makes sense to compare our upper bound with the well-known lower bound for the approximate counting problem for binary strings $x \in \{0,1\}^N$. We first recall a precise statement. Let $x \in \{0,1\}^N$ and $k = |x|$, and U_x a unitary implementing quantum oracle access to x. Then for an integer $\Delta > 0$, any quantum algorithm which, with probability $\geq 2/3$, computes an additive Δ-approximation of k uses at least $\Omega(\sqrt{N/\Delta} + \sqrt{k(N-k)}/\Delta)$ applications of controlled-U_x [NW99, Thm. 1.10 and 1.11]. A matching upper bound is given in [BHMT02, Thm. 18], see Theorem 2.7 for a precise formulation. We can compare the complexity of our algorithm by converting multiplicative error into additive error, i.e., to achieve an additive error of ε we take $\delta = \varepsilon/k$ (or ε divided by a suitable multiplicative approximation of k). Then the key point is that if one

[4] In [Ham21, Proposition 6.4], a matching (up to log-factors) lower bound is shown for Bernoulli random variables. We remark that our algorithm does not break that lower bound since we parameterize the problem differently: the complexity of our algorithm depends also on the size of the support of the distribution.

considers Eq. (1.1) for $\varepsilon \leq \Delta$ and $k \geq 1$, then

$$\sqrt{\frac{Nk}{\varepsilon}} \geq \sqrt{\frac{Nk}{\Delta}} \geq \sqrt{\frac{1}{2}\frac{N}{\Delta} + \frac{1}{2}\frac{k(N-k)}{\Delta}} \geq \frac{1}{2}\left(\sqrt{\frac{N}{\Delta}} + \frac{\sqrt{k(N-k)}}{\Delta}\right)$$

where the last inequality follows from concavity of the square-root function and $\Delta \geq 1$. In other words, for all parameters N, k, Δ, the complexity of our algorithm (left hand side), is at least as large as the lower bound on approximate counting (right hand side), so we do not break the lower bound.

We highlight two ranges of parameters. On the one hand, when Δ is large, our upper bound is suboptimal for quantum counting. For example, when $\Delta = k/2$ (i.e., $\delta = 1/2$), our algorithm uses $O(\sqrt{N})$ queries whereas the approximate counting algorithm from [BHMT02, Thm. 18] uses only $O(\sqrt{N/k})$ queries. This is no surprise given that our algorithm finds all "large" elements, which in the counting setting amounts to finding all ones. On the other hand, when Δ is a small constant, say $\Delta = 1$, the approximate counting lower bound shows that our upper bound is essentially tight. To see this, note if one had a quantum algorithm for computing $(1 \pm \delta)$-multiplicative approximations of sums with quantum query complexity $O(\sqrt{N}/\delta^c)$ (that succeeds with probability $\geq 2/3$), this would give an upper bound of $O(\sqrt{N}k^c)$ for finding an additive $\Delta = 1$-approximation of k. The lower bound becomes $\Omega(\sqrt{k(N-k)})$ when $\Delta = 1$, and so we must have $\sqrt{N}k^c \geq \sqrt{k(N-k)}$, i.e., $c \geq 1/2$. We leave it as an open problem whether one can obtain a quantum algorithm for approximate summing of vectors $v \in [0,1]^N$ that matches the approximate counting complexity when applied to $v \in \{0,1\}^N$ for the entire range of parameters N, k, Δ.

Finally we highlight that our quantum upper bound for summing outperforms the classical lower bound for approximate counting for a certain range of parameters. The classical randomized query complexity of achieving a multiplicative δ-approximation of the Hamming weight of $x \in \{0,1\}^N$ is $\Theta(\min\{N, \frac{N}{\delta^2 k}\})$.[5] This classical bound exceeds our quantum upper bound of $\widetilde{O}(\sqrt{N/\delta})$ if $1/\delta \in O(N)$ and $1/\delta \in \Omega(k^{2/3}/N^{1/3})$ (ignoring logarithmic factors).

Organization of the paper

In Section 2 we discuss notation, the computational model, and some basic results we build upon. In Section 3 we provide our algorithm for searching for multiple marked elements. Lastly, in Section 4, we give our summation algorithm.

2 Preliminaries

2.1 Notation and assumptions

Throughout the paper, we will assume that $N \geq 1$ and $N = 2^n$ for some $n \geq 1$. We identify $\mathbb{C}^N$ with $\mathbb{C}^{2^n}$ by $|j\rangle \mapsto |j_1 \dots j_n\rangle$, where $(j_1, \dots, j_n) \in \{0,1\}^n$ is the standard binary encoding of $j - 1 \in \{0, \dots, 2^n - 1\}$. We write log for the logarithm with base 2 and ln for the natural logarithm. For a bit string $x \in \{0,1\}^N$ we write $|x| = \sum_{i \in [N]} x_i$. Throughout we will use k to denote the Hamming weight of x, i.e, $|x| = k$, and we write $k_{\text{est}}, k_{\text{lb}}, k_{\text{ub}}$ for various bounds on k: k_{est} will denote an integer such that $k/2 \leq k_{\text{est}} \leq 3k/2$, and k_{lb} and k_{ub} are lower- and upper bounds on k respectively.

[5] We believe this is well known: the upper bound (which was also conjectured in [BHMT02]) follows from the algorithm presented in [DKLR00], applied to a Bernoulli random variable with success probability k/N; the lower bound is claimed for instance in [AR20], but we could not locate a proof in the literature for cases other than $k = \Theta(N)$ [CEG95]. We therefore provide a (not entirely trivial) proof in Appendix A.

Procedure AmpEst(U, M)

Input: Access to controlled versions of unitary $U \in \mathrm{U}(2^q)$ and its inverse, an integer $M \geq 1$.

Output: Real number $\tilde{a} \in [0, 1]$.

Analysis: Lemma 2.3

2.2 Computational model

We express the cost of a quantum algorithm in terms of the number of one- and two-qubit gates it uses. Note that in particular we allow single-qubit rotations with arbitrary real angles. In Section 3, the angle will always be determined by classical data. In Section 4 we additionally apply controlled rotations where the control register is allowed to be in superposition; in this case we only use angles of the form $\pi/2^m$ and we carefully count the number of used gates. In the query setting, we separately count the number of quantum queries the algorithm makes, which means (controlled) applications of the query unitary or its inverse. We will use the following types of quantum queries to access either N-bit strings $x \in \{0,1\}^N$ or N-dimensional vectors $v \in [0,1]^N$ (specified in fixed-point format).

Definition 2.1. *A unitary $U_x \in \mathrm{U}(\mathbb{C}^N \otimes \mathbb{C}^2)$ is said to implement quantum oracle access to an N-bit string $x \in \{0,1\}^N$ if it acts as*

$$U_x |i\rangle |b\rangle = |i\rangle |b \oplus x_i\rangle$$

for all $i \in [N]$ and $b \in \{0,1\}$.

Definition 2.2. *A unitary $U_v \in \mathrm{U}(\mathbb{C}^N \otimes \mathbb{C}^{2^b})$ is said to implement quantum oracle access to $(0,b)$-fixed-point representations of $v \in [0,1]^N$ if it acts as*

$$U_v |i\rangle |0^b\rangle = |i\rangle |v_i\rangle$$

for all $i \in [N]$, where $|v_i\rangle = |(v_i)_1 \dots (v_i)_b\rangle$ satisfies $\sum_{j=1}^{b} (v_i)_j 2^{-j} = v_i$.

In both cases we allow the unitary to act on additional workspace registers, which we omit for notational convenience. Moreover, throughout the paper, every algorithm will use at most logarithmic number of additional ancillary qubits.

We additionally use a classical data structure to maintain sorted lists that supports both insertion and look-up in a time that scales logarithmically with the size of the list, see for example [Knu98, Sec. 6.2.3] or [CLRS22, Ch. 13]. We emphasize that we allow neither writing nor reading of such a data structure in superposition.

2.3 Various quantum subroutines

In this section we summarize the external results that we build upon, and in some cases give a quick proof of an aspect of the result that is not mentioned explicitly in the original source.

Lemma 2.3 (Amplitude estimation [BHMT02, Thm. 12])**.** *Let $U \in \mathbb{C}^{2^q \times 2^q}$ be a unitary that creates a state*

$$|\psi\rangle = U |0^q\rangle = \sqrt{a} |\phi_1\rangle |1\rangle + \sqrt{1-a} |\phi_0\rangle |0\rangle .$$

There is a quantum algorithm `AmpEst` *that, with probability $\geq \frac{8}{\pi^2}$, outputs an $\tilde{a} \in [0,1]$ such that*

$$|a - \tilde{a}| \leq 2\pi \frac{\sqrt{a(1-a)}}{M} + \frac{\pi^2}{M^2}$$

using M applications of controlled-U and M applications of controlled-$U^\dagger$. If M is a power of 2, the algorithm uses $O(qM)$ additional quantum gates, and the computation of the sine-squared function of the normalized phase.

Procedure GroverCertainty(U, k_0)

Input: Quantum oracle U_x to access $x \in \{0,1\}^N$, an integer $k_0 \geq 1$.
Output: An index $i \in [N]$.
Guarantee: If $|x| = k_0$, then $x_i = 1$ with certainty.
Analysis: Theorem 2.4

Procedure GroverExpectation(U_x)

Input: Quantum oracle U_x to access $x \in \{0,1\}^N$.
Output: An index $i \in [N]$.
Guarantee: If $|x| \geq 1$, then $x_i = 1$ with certainty.
Analysis: Theorem 2.5

Proof. This follows from the formulation in [BHMT02] by setting $k = 1$ and implementing the reflection through $|0^q\rangle$ using $O(q)$ gates, which needs to be performed M times. If M is a power of 2 we can implement the quantum Fourier transform on $m = \log_2(M)$ qubits using m Hadamard gates, and the QFT and its inverse need only be performed once; therefore, this cost is absorbed in the big-O. □

We note that the above formulation of `AmpEst` outputs a real number $\tilde{a}$ whereas we require a fixed-point encoded number for future uses. However, it suffices to use fixed-point arithmetic using $O(\log(M))$ bits; after all, the guarantee of `AmpEst` only gives a precision of $1/\text{poly}(M)$.

We also need a version of amplitude amplification where the success probability is 1 if one knows the amplitude of the "good" part of the state exactly. In a nutshell, the algorithm with success probability 1 is the usual amplitude amplification algorithm applied not to U but to U followed by a rotation of the last qubit to slightly reduce the amplitude a to $\bar{a}$. Carefully choosing $\bar{a}$ ensures that the success probability is exactly 1 after an integer number of rounds of amplitude amplification. This requires having access to gates which implement rotation by arbitrary angles, not just angles of the form $\pi/2^m$ for some integer m. We specialize the statement of this result to the search setting but remark that this works more generally. For exactly $N/4$ marked elements this observation was first made in [BBHT98].

Theorem 2.4 ([BHMT02, Thm. 4])**.** *Let $x \in \{0,1\}^N$ with $|x| = k \geq 1$. Then there is a quantum algorithm `GroverCertainty` that takes as input a quantum oracle U_x to access x and an integer $k_0 \in [N]$, and that outputs an index $i \in [N]$, such that $x_i = 1$ with certainty if $k_0 = k$, and uses $O(\sqrt{N/k_0})$ quantum queries to x, and $O(\sqrt{N/k_0}\log(N))$ additional gates.*

The other version of Grover that we need is the following, which is originally due to [BBHT98, Thm. 3], but we use a slightly different version from [BHMT02, Thm. 3]:

Theorem 2.5 ([BHMT02, Thm. 3])**.** *Let $x \in \{0,1\}^N$ with $|x| = k$, where k is not necessarily known. Then there is a quantum algorithm `GroverExpectation` that takes as input a quantum oracle U_x to access x, and if $k \geq 1$, outputs an index $i \in [N]$ such that $x_i = 1$. The number of quantum queries to x that it uses is a random variable Q, such that, if $k \geq 1$, then*

$$\mathbb{E}[Q] = O\left(\sqrt{N/k}\right),$$

and if $|x| = 0$, then $Q = \infty$ (i.e., the algorithm runs forever). The number of additional gates used is $O(Q\ \log(N))$. The index i which is output is uniformly random among all such indices, and independent of the value of Q.

Procedure $\mathrm{Grover}_{2/3}(U_x, k_{\mathrm{lb}})$

Input: Quantum oracle U_x to access $x \in \{0,1\}^N$ and a lower bound k_{lb} on $|x|$.
Output: An index $i \in [N]$.
Guarantee: If $|x| \geq 1$, then with probability $\geq 2/3$, $x_i = 1$.
Analysis: Lemma 2.6

Procedure $\mathrm{ApproxCount}(U_x, \varepsilon, \rho)$

Input: Quantum oracle U_x to access $x \in \{0,1\}^N$, rational number $\varepsilon > 0$ such that $\frac{1}{3N} < \varepsilon \leq 1$, failure probability $\rho > 0$.
Output: Integer $\tilde{k} \in \{0, \ldots, N\}$.
Guarantee: If $|x| = k \geq 1$, with probability $\geq 1 - \rho$, $|\tilde{k} - k| \leq \varepsilon k$, and if $k = 0$ then $\tilde{k} = 0$ with certainty.
Analysis: Theorem 2.7

Lemma 2.6. *Let $x \in \{0,1\}^N$. Then there is a quantum algorithm $\mathtt{Grover}_{2/3}$ that takes as input a quantum oracle U_x to access x and a lower bound $k_{\mathrm{lb}} \geq 1$ on $|x|$. With probability $\geq 2/3$, it outputs an index $i \in [N]$ such that $x_i = 1$. It uses $O(\sqrt{N/k_{\mathrm{lb}}})$ quantum queries to x, and $O(\sqrt{N/k_{\mathrm{lb}}}\log(N))$ additional gates.*

Proof. The algorithm $\mathtt{GroverExpectation}$ finds an index i such that $x_i = 1$. Its number of applications of controlled-U_x is a random variable Q and the number of additional gates is $O(Q \cdot \log(N))$. By Theorem 2.5 we have $\mathbb{E}[Q] = O(\sqrt{N/|x|})$. Markov's inequality shows that if we terminate $\mathtt{GroverExpectation}$ after at most $C\sqrt{N/|x|}$ quantum queries for a suitable constant $C > 0$, then it finds an index i such that $x_i = 1$ with probability at least $2/3$. The procedure $\mathtt{Grover}_{2/3}$ uses the lower bound k_{lb} on $|x|$ to decide when to terminate $\mathtt{GroverExpectation}$. For the same constant $C > 0$ as before, it terminates after at most $C\sqrt{N/k_{\mathrm{lb}}}$ quantum queries. Since $C\sqrt{N/k_{\mathrm{lb}}} \geq C\sqrt{N/|x|}$, the success probability of $\mathtt{Grover}_{2/3}$ is also at least $2/3$. □

Let us make some remarks about the complexity of finding a single marked element. First, to find such an element with certainty one can essentially remove the $\log(N)$ factor in the gate complexity: $O(\sqrt{N}\log(\log^*(N)))$ gates suffice [AdW17]. Second, by cleverly combining $\mathtt{GroverCertainty}$ and $\mathtt{Grover}_{2/3}$, one can find a marked element (among an unknown number of solutions) with probability $\geq 1 - \rho$ using $\sqrt{N\log(1/\rho)}$ quantum queries [BCdWZ99]. This shows that the standard way of boosting the success probability of $\mathtt{Grover}_{2/3}$ is not optimal.

Next, we recall a well-known result on approximate counting.

Theorem 2.7 ([BHMT02, Thm. 18])**.** *Let $x \in \{0,1\}^N$ and write $|x| = k$. Let $\frac{1}{3N} < \varepsilon \leq 1$. Then there is a quantum algorithm that, with probability at least $2/3$, that outputs an estimate $\tilde{k}$ such that*

$$|\tilde{k} - k| \leq \varepsilon k$$

using an expected number of

$$\Theta\left(\sqrt{\frac{N}{\lfloor \varepsilon k \rfloor + 1}} + \frac{\sqrt{k(N-k)}}{\lfloor \varepsilon k \rfloor + 1}\right)$$

quantum queries to x. If $k = 0$, then the algorithm outputs $\tilde{k} = 0$ with certainty, using $\Theta(\sqrt{N})$ quantum queries to x. In both cases, the algorithm uses a number of gates which is $O(\log(N))$ times the number of quantum queries. To boost the success probability to $1 - \rho$, repeat the procedure $O(\log(1/\rho))$ many times and output the median of the returned values.

We often use the special case $\varepsilon = 1/2$ of the above theorem, hence we record it here for future use. (Note that the proof of Theorem 2.7 given in [BHMT02] in fact starts by obtaining a constant factor approximation of $|x|$.)

Corollary 2.8. *Let $x \in \{0,1\}^N$ and write $|x| = k$. Then there is a quantum algorithm that outputs a k_{est} such that, with probability $\geq 1-\rho$, we have $k/2 \leq k_{\text{est}} \leq 3k/2$, and uses $O(\sqrt{N/(k+1)}\log(1/\rho))$ quantum queries and $O(\sqrt{N/(k+1)}\log(1/\rho)\log(N))$ gates.*

We now discuss known extensions of the above results on counting the Hamming weight of a bit string to the problem of *mean estimation*: given a vector $v \in [0,1]^N$, one is interested in approximating $\bar{v} = \frac{1}{N}\sum_{i=1}^N v_i$. This was first studied in [Gro97] and later in [Gro98] where in the latter it was shown that one can find an additive ε-approximation of $\bar{v}$ using $\widetilde{O}(1/\varepsilon)$ quantum queries to a unitary that prepares a state encoding the entries of v in its amplitudes, and a similar number of additional gates (also dependent on N). Using amplitude amplification techniques one can reduce the query dependence to $O(1/\varepsilon)$ with $O(\log(N)/\varepsilon)$ additional gates. This result may be easily recovered from Lemma 2.3 with $M = \Theta(1/\varepsilon)$, applied to a unitary preparing

$$\frac{1}{\sqrt{N}}\sum_{i=1}^N |i\rangle\left(\sqrt{1-v_i}\,|0\rangle + \sqrt{v_i}\,|1\rangle\right).$$

It is well-known that when one has quantum oracle access to fixed point representations of the entries of v (cf. Definition 2.2), rather than just a state encoding its entries in the amplitudes, one can give an algorithm whose complexity depends only on N and δ, with guarantees as given below.

Theorem 2.9. *Let $v \in [0,1]^N$ be a vector with each entry v_i encoded in $(0,b)$-fixed-point format, and let U_v be a unitary implementing binary oracle access to v (cf. Definition 2.2). Let $\rho \in (0,1)$. Then with $O(\frac{\sqrt{N}}{\delta}\log(1/\rho))$ applications of controlled-U_v, controlled-$U_v^\dagger$, and a polylogarithmic gate overhead, one can find with probability $\geq 1-\rho$ a multiplicative δ-approximation of $\frac{1}{N}\sum_i v_i$.*

We give an informal description of the algorithm here, and refer the interested reader to [vAGL+21] for a careful implementation along with a bit complexity analysis. By using quantum maximum finding [DH96], with $O(\sqrt{N})$ quantum queries and $O(b\sqrt{N}\log(N))$ other gates, one may find $v_{\max} = \max_i v_i$. If $v_{\max} = 0$ one may output $\bar{v}_{\text{est}} = 0$ as an estimate of $\bar{v}$. Note that having binary access here makes it easy to compare elements. Next, set $w_i = v_i/v_{\max}$, and let U be a unitary preparing a state

$$\frac{1}{\sqrt{N}}\sum_{i=1}^N |i\rangle\left(\sqrt{1-w_i}\,|0\rangle + \sqrt{w_i}\,|1\rangle\right).$$

Then Lemma 2.3 with $M = 8\sqrt{N}/\delta$ outputs an estimate $\bar{w}_{\text{est}}$ of $\bar{w}$, such that

$$|\bar{w}_{\text{est}} - \bar{w}| \leq 2\pi\frac{\sqrt{\bar{w}(1-\bar{w})}}{M} + \frac{\pi^2}{M^2} \leq \frac{2\pi}{8}\delta\bar{w}\sqrt{1-\bar{w}} + \frac{\pi^2}{64}\delta^2\bar{w} \leq \delta\bar{w}$$

because $1/N \leq \bar{w}$, so $\bar{w}_{\text{est}}$ is a multiplicative δ-approximation of $\bar{w}$. Therefore $\bar{w}_{\text{est}} \cdot v_{\max}$ is a multiplicative δ-approximation of $\bar{v}$. We note that in this step the binary access to the entries of v enables the "binary amplification" by ensuring the largest entry of w is 1.

3 Fast Grover search for multiple items, without quantum memory

In this section we give a version of Grover's search algorithm for the problem of, given a string $x \in \{0,1\}^N$, finding *all* k indices $i \in [N]$ such that $x_i = 1$. For $\rho \in (0,1)$ with $\rho = \Omega(1/\text{poly}(k))$, our

algorithm finds all such indices with probability $\geq 1-\rho$, and uses $O(\sqrt{Nk})$ quantum queries and $\widetilde{O}(\sqrt{Nk})$ single- and two-qubit gates. The contribution here is that the query complexity is optimal and the time complexity is only polylogarithmically worse than the query complexity, without using a QRAM.

3.1 Deterministic Grover for multiple elements

We first recall the well-known result [dGdW02, Lem. 2], that it is possible to find all solutions with probability 1 using $O(\sqrt{Nk})$ quantum queries, which is optimal, but suffers from a too-high gate complexity in terms of k. The algorithm is given in `GroverCertaintyMultiple`. We first define for each $j \in [N]$ a gate C_j, referred to as the "control-on-j-NOT"-gate, and describe how to implement it with a standard gate-set. The point of this gate is that if one has quantum oracle access U_x to $x \in \{0,1\}^N$, then $C_j U_x$ implements quantum oracle access to the bit string $y \in \{0,1\}^N$ which agrees with x on all indices, except on the j-th index, where the bit is flipped.

Lemma 3.1. *Let $N = 2^n$. For $j \in [N]$ define the "control-on-j-NOT"-gate $C_j \in \mathrm{U}(\mathbb{C}^N \otimes \mathbb{C}^2)$ by*

$$C_j |i\rangle |b\rangle = \begin{cases} |i\rangle |b \oplus 1\rangle & \text{if } i = j, \\ |i\rangle |b\rangle & \text{otherwise.} \end{cases} \tag{3.1}$$

Then the C_j-gate can be implemented with $O(n)$ standard gates and $n-1$ ancillary qubits.

Proof. Let $|j\rangle = |j_1 \dots j_n\rangle$ be the binary encoding of $j-1$. Then:

1. For each $l \in [n]$ such that $j_l = 0$, apply a NOT gate on the l-th qubit of the index register.
2. Apply a NOT-gate to the output register containing b, controlled on all n qubits of the index register. This can be implemented using $O(n)$ Toffoli gates, one CNOT gate, and $n-1$ ancilla qubits, see [NC02, Fig. 4.10].
3. Apply the NOT gates from the first step again. □

Lemma 3.2. *Let $x \in \{0,1\}^N$, U_x a quantum oracle to access x, and $k_{\mathrm{ub}} \geq 1$. If $|x| = k \leq k_{\mathrm{ub}}$, then `GroverCertaintyMultiple`$(U_x, k_{\mathrm{ub}})$ finds, with probability 1, all k indices i such that $x_i = 1$. The algorithm uses*

$$O\Big(\sqrt{Nk_{\mathrm{ub}}}\Big)$$

applications of U_x, and

$$O\Big(\sqrt{Nk_{\mathrm{ub}}}(k+1)\log(N)\Big)$$

additional non-query gates.

Proof. We first establish correctness of `GroverCertaintyMultiple`. For $m \in [k_{\mathrm{ub}}]$, let $J_m \subseteq [N]$ be the index set and U_{J_m} the unitary in the algorithm at the m-th step. Then by the definition of C_j, U_{J_m} implements oracle access to the bit string y^m which agrees with x on $[N] \setminus J_m$, and is zero on the indices in J_m (whereas $x_j = 1$ for $j \in J_m$). Clearly $j \in J_0$ implies that $x_j = 1$. It remains to show that in k_{ub} iterations we find *all* marked elements. To do so, observe that there can be at most $k_{\mathrm{ub}} - k$ iterations in which one fails to find a new $j \in [N]$ such that $x_j = 1$: indeed, as soon as this happens, we have $m = |y^m|$, and every iteration afterwards we find a new index with certainty by the guarantees of `GroverCertainty`.

Procedure GroverCertaintyMultiple(U_x, k_{ub})

Input: Quantum oracle U_x to access $x \in \{0,1\}^N$, an integer $k_{\text{ub}} \geq 1$.
Output: Classical list of indices $J \subseteq [N]$.
Guarantee: If $|x| \leq k_{\text{ub}}$, then for every $j \in [N]$, $j \in J$ if and only if $x_j = 1$.
Analysis: Lemma 3.2

$J_{k_{\text{ub}}} \leftarrow \emptyset$;
$U_{J_{k_{\text{ub}}}} \leftarrow U_x$;
$m \leftarrow k_{\text{ub}}$;
while $m > 0$ **do**
  use `GroverCertainty`(U_{J_m}, m) to find a $j \in [N] \setminus J_m$;
  if $x_j = 1$ **then**
    $J_{m-1} \leftarrow J_m \cup \{j\}$;
    $U_{J_{m-1}} \leftarrow C_j U_{J_m}$, where C_j is defined in Lemma 3.1;
  else
    $J_{m-1} \leftarrow J_m$;
    $U_{J_{m-1}} \leftarrow U_{J_m}$;
  end if
  $m \leftarrow m - 1$;
end while
return J_0;

In total, this procedure uses $\sum_{m=1}^{k_{\text{ub}}} O(\sqrt{N/m}) = O(\sqrt{Nk_{\text{ub}}})$ applications of U_x. The number of auxiliary gates for a single query in the m-th iteration is $O(|J_m| \cdot \log(N))$, and `GroverCertainty` itself uses an additional $O(\sqrt{N/m}\log(N))$ additional gates. Therefore the total number of gates in the m-th iteration is

$$O\left(\sqrt{N/m} \cdot |J_m| \cdot \log(N) + \sqrt{N/m}\log(N)\right) = O\left(\sqrt{N/m}(k+1)\log(N)\right)$$

Summing this over all iterations yields a total gate complexity of $O(\sqrt{Nk_{\text{ub}}}(k+1)\log(N))$. □

3.2 Coupon collecting Grover

We next give another simple version of Grover which can be used to find a large fraction of the marked elements in a time-efficient manner, but does not yield a query-optimal bound when the fraction is close to 1. The algorithm is given in `GroverCoupon`, and is analyzed in Proposition 3.7.

The algorithm is simple: the idea is to repeatedly call $\texttt{Grover}_{2/3}$ to sample marked elements. The analysis is based on the observation that the required number of calls to $\texttt{Grover}_{2/3}$ is a sum of geometrically distributed random variables: for $1 \leq i \leq t$, the number of calls to obtain the i-th distinct marked element is a geometrically distributed random variable with success probability $p_i' \geq \frac{2}{3}p_i$, where $p_i = (k - i + 1)/k$ is the probability of observing a new element after $i - 1$ distinct elements have been found. This is because $\texttt{Grover}_{2/3}$ succeeds with probability $\geq 2/3$, and the fact that if $\texttt{Grover}_{2/3}$ successfully finds a marked element, then it is uniformly random among the marked elements. The number of calls can then by bounded using a general tail bound on sums of geometrically distributed random variables given in [Jan18, Thm. 2.3] (see Lemma 3.4).

The analysis is based on tail bounds of sums of geometrically distributed random variables. These tail bounds in turn are stated in terms of the harmonic numbers, for which we recall some

basic properties in the following lemma.

Lemma 3.3. *The k-th harmonic number H_k is defined by $H_k = \sum_{j=1}^{k} \frac{1}{j}$, and we shall use the convention $H_0 = 0$. For $k \geq 1$ it satisfies*

$$H_k - \gamma - \ln(k) \in \left[\frac{1}{2(k+1)}, \frac{1}{2k}\right],$$

where $\gamma \approx 0.577$ is the Euler–Mascheroni constant. Furthermore, for $0 \leq t < k$, this implies

$$H_k - H_{k-t} \leq \ln\left(\frac{k}{k-t}\right) + \frac{2k-t+1}{2k(k-t+1)},$$

which in turn for $t \leq k/2 < k$ implies

$$H_k - H_{k-t} \leq \frac{2(t+1)}{k}.$$

Proof. The bounds on $H_k - \gamma - \ln(k)$ are well-known, see [You91] for an elementary proof. For the estimate on $H_k - H_{k-t}$ we have

$$H_k - H_{k-t} \leq \ln(k) - \ln(k-t) + \frac{1}{2k} + \frac{1}{2(k-t+1)} = \ln\left(\frac{k}{k-t}\right) + \frac{2k-t+1}{2k(k-t+1)}.$$

Furthermore, if $t \leq k/2 < k$, then

$$\ln\left(1 + \frac{t}{k-t}\right) + \frac{2k-t+1}{2k(k-t+1)} \leq \frac{2t}{k} + \frac{2k+1}{2k(k/2+1)} \leq \frac{2t}{k} + \frac{2}{k} = \frac{2(t+1)}{k}. \qquad \square$$

We use the following tail bound for geometrically distributed variables.

Lemma 3.4 ([Jan18, Thm. 2.3])**.** *For $i \in [n]$ assume $X_i \sim \mathrm{Geo}(p_i)$ for $p_i \in (0,1]$. Let $X = \sum_{i\in[n]} X_i$ and write $\mu = \mathbb{E}[X]$, $p_* = \min_{i\in[n]} p_i$. Then for any $\lambda \geq 1$ we have*

$$\Pr[X \geq \lambda\mu] \leq \lambda^{-1}(1-p_*)^{(\lambda-1-\ln(\lambda))\mu}.$$

Corollary 3.5. *For $i \in [n]$ assume $X_i \sim \mathrm{Geo}(p_i)$ for $p_i \in (0,1]$. Let $X = \sum_{i\in[n]} X_i$ and write $\mu = \mathbb{E}[X]$, $p_* = \min_{i\in[n]} p_i$. Let $\rho \in (0,1)$. Then $\Pr[X \geq T] \leq \rho$ whenever*

$$T \geq 2\ln(2)\mu + 2\frac{\ln(1/\rho)}{\ln(1/(1-p_*))}.$$

Proof. We apply Lemma 3.4 with $\lambda \geq 1$ to obtain

$$\Pr[X \geq \lambda\mu] \leq \lambda^{-1}(1-p_*)^{(\lambda-1-\ln(\lambda))\mu} \leq (1-p_*)^{(\lambda-1-\ln(\lambda))\mu}.$$

By the first-order characterization of convexity of $\lambda \mapsto \lambda - 1 - \ln(\lambda)$ at $\lambda = 2$, we have

$$\lambda - 1 - \ln(\lambda) \geq \left(1 - \frac{1}{2}\right)(\lambda - 2) + 2 - 1 - \ln(2) = \frac{1}{2}\lambda - \ln(2).$$

Therefore

$$\Pr[X \geq \lambda\mu] \leq e^{\ln(1-p_*)(\lambda/2-\ln(2))\mu},$$

and so to ensure that this is at most ρ, it suffices to take

$$\lambda\mu \geq 2\ln(2)\mu + \frac{2\ln(\rho)}{\ln(1-p_*)}.$$

Note that such λ also satisfies $\lambda \geq 1$, because $2\ln(2) \geq 1$ and $\ln(\rho)/\ln(1-p_*) \geq 0$. Therefore we have shown that $\Pr[X \geq T] \leq \rho$ whenever $T \geq 2\ln(2)\mu + 2\frac{\ln(\rho)}{\ln(1-p_*)}$. $\square$

Applying the above tail bound with $p_i \geq \frac{2}{3}(k-i+1)/k$ yields the following lemma.

Lemma 3.6. *Let $1 \leq t \leq k \leq N$ and subset $I \subseteq [N]$ of size k, and let $\rho \in (0,1)$. Consider a procedure in which at each step with probability $\geq 2/3$, one obtains a uniformly random sample from I. The outputs of*

$$r \geq 3\ln(2)\, k(H_k - H_{k-t}) + \frac{2\ln(1/\rho)}{\ln(3k/(k+2(t-1)))} =: R_{t,k,\rho}$$

repetitions of this procedure suffice to, with probability $\geq 1-\rho$, obtain t distinct samples from I.

Proof. We apply Corollary 3.5 to $X = \sum_{i\in[t]} X_i$ where $X_i \sim \mathrm{Geo}(p_i)$ with $p_i \geq \frac{2}{3}(k-i+1)/k$. We then have $p_* \coloneqq \min_{i\in[t]} p_i \geq \frac{2}{3}(k-t+1)/k$, and thus $\frac{1}{1-p_*} \leq \frac{1}{1-\frac{2}{3}(k-t+1)/k} = \frac{3k}{k+2(t-1)}$, and $\mu \coloneqq \mathbb{E}[X] = \sum_{i\in[t]} \frac{1}{p_i} \leq \sum_{i\in[t]} \frac{3}{2}\frac{k}{k-t+1} = \frac{3}{2}k(H_k - H_{k-t})$. Corollary 3.5 thus shows that we obtain t distinct samples from I with probability at least $1-\rho$, whenever we use at least $R_{t,k,\rho}$ repetitions of this procedure. □

We briefly emphasize the value of this lemma. For general $t \leq k$, we can use the simple bound $\ln(k/(t-1)) \geq \ln(k/(k-1)) \geq 1/k$ and the estimate $H_k - H_{k-t} \approx \ln(k/(k-t))$, to obtain that $r \in \Omega(k\log(k) + k\ln(1/\rho)) = \Omega(k\log(k/\rho))$ samples suffice. By contrast, an application of Markov's inequality only yields a sample complexity upper bound of $k\log(k)\log(1/\rho)$. In later applications (cf. Theorem 3.9), we apply this with t at most $k/2$, in which case we can give tighter estimates. Indeed, the factor $1/\ln(3k/(k+2(t-1)))$ is then at most a constant and $H_k - H_{k-t} \leq \frac{2(t+1)}{k}$ by Lemma 3.3, and thus $r \in \Omega(t + \ln(1/\rho))$ samples suffice. Therefore the bound is an improvement over the sample complexity of $\Omega(t\ln(1/\rho))$ one would obtain from a simple application of Markov's inequality – in particular, one can now "for free" choose ρ to be exponentially small in t (and similar above).

By using $\texttt{Grover}_{2/3}$ to obtain the samples required for Lemma 3.6, we obtain the following algorithmic result.

Proposition 3.7. *Let $x \in \{0,1\}^N$ with $|x| = k$ unknown, let $R \geq 1$, let $k_{\mathrm{lb}} \geq 1$ be such that $k_{\mathrm{lb}} \leq k$, let $t \geq 1$, and $\rho \in (0,1)$. Assume $1 \leq t \leq k$. Then* `GroverCoupon` *called with a quantum oracle U_x to access x, and additional inputs R, k_{lb}, and t, uses $O(\sqrt{N/k_{\mathrm{lb}}}\, r)$ quantum queries to x and $O(\sqrt{N/k_{\mathrm{lb}}}\, r\log(N))$ additional quantum gates. Here, r is a random variable such that $r \leq R$ with certainty, and with probability $\geq 1-\rho$, one has*

$$r \leq R_{t,k,\rho} = 3\ln(2)k(H_k - H_{k-t}) + 2\frac{\ln(1/\rho)}{\ln(3k/(k+2(t-1)))}.$$

If $R \geq R_{t,k,\rho}$, then with probability $\geq 1-\rho$, it finds a set of t distinct marked elements, uniformly at random from the set of all sets of k marked elements.

Proof. We first analyze the complexity of `GroverCoupon`. Let $r \in [R]$ be the number of times the algorithm repeats line 3 through line 8. By Lemma 2.6, the application of $\texttt{Grover}_{2/3}$ in line 3 uses $O(\sqrt{N/k_{\mathrm{lb}}})$ quantum queries and $O(\sqrt{N/k_{\mathrm{lb}}}\log(N))$ additional gates. With one additional query we can verify if the index $j \in [N]$ that is returned by $\texttt{Grover}_{2/3}$ is such that $x_j = 1$. If indeed $x_j = 1$, then we add j to J. As mentioned in Section 2.2, we can insert an element in the sorted list J in (classical) time $O(\log(N))$. We can verify line 7 in time $O(\log(N))$ by maintaining a counter for $|J|$. The above shows that `GroverCoupon` indeed uses $O(\sqrt{N/k_{\mathrm{lb}}}\, r)$ quantum queries and $O(\sqrt{N/k_{\mathrm{lb}}}\, r\log(N))$ additional quantum gates.

Procedure GroverCoupon(U_x, R, k_{lb}, t)

Input: Quantum oracle U_x to access $x \in \{0,1\}^N$, an integer $R \geq 1$, an integer k_{lb} such that $k_{\text{lb}} \leq |x|$, an integer $t \geq 1$ such that $1 \leq t \leq |x|$.

Output: Classical sorted list of indices $J \subseteq [N]$.

Guarantee: If $R \geq R_{t,k,\rho} = 3\ln(2)k(H_k - H_{k-t}) + 2\frac{\ln(1/\rho)}{\ln(3k/(k+2(t-1)))}$, then, with probability $\geq 1-\rho$, we have $|J| = t$ and $x_j = 1$ for all $j \in J$.

Analysis: Proposition 3.7

1 $J \leftarrow \emptyset$;
2 **for** $r = 1, \ldots, R$ **do**
3 | use $\texttt{Grover}_{2/3}$ with arguments (U_x, k_{lb}) to find a $j \in [N]$ such that $x_j = 1$ with probability $\geq 2/3$;
4 | **if** $j \notin J$ *and* $x_j = 1$ **then**
5 | | add j to J;
6 | **end if**
7 | **if** $|J| = t$ **then**
8 | | **return** J ;
9 | **end if**
10 **end for**
11 **return** J;

We now establish correctness. By construction $r \leq R$ with certainty. Lemma 2.6 shows that, with probability $\geq 2/3$, the index returned by $\texttt{Grover}_{2/3}$ in line 3 is a uniformly random marked element. Hence Lemma 3.6 shows that after obtaining

$$R_{t,k,\rho} = 3\ln(2)\, k(H_k - H_{k-t}) + \frac{2\ln(1/\rho)}{\ln(3k/(k+2(t-1)))}$$

such indices, we have obtained t distinct indices with probability at least $1-\rho$. In other words, if $R \geq R_{t,k,\rho}$, then, with probability at least $1-\rho$, `GroverCoupon` terminates at line 8 with a sorted list $J \subseteq [N]$ of t distinct marked indices. □

3.3 Grover for multiple elements, fast

In this section we improve the complexity of finding all marked indices by combining the two previously discussed algorithms, `GroverCoupon` and `GroverCertaintyMultiple`. The structure of our algorithm, `GroverMultipleFast`, is as follows. As before, suppose we are given query access to an $x \in \{0,1\}^N$. Let the (unknown) number of marked indices be $k \geq 1$, i.e., $k = |x|$. We first use `GroverCoupon` to find a (large) fraction of the marked elements. That is, we find a uniformly random subset $J_0 \subseteq [N]$ of τk marked elements, where $0 < \tau < 1$ is a parameter we can use to tune the complexity of the algorithm. This subset J_0 partitions $[N]$ into intervals. We then use `GroverCertaintyMultiple` to find all marked indices in each interval separately.

The following lemma upper bounds the probability that when we draw a set $S \subseteq [k]$ of size t uniformly at random, there exists an interval of length $\geq \ell$ in the set $[k] \setminus S$. In the analysis of `GroverMultipleFast` (see Theorem 3.9), we will use this bound to control the number of elements that are in between any two elements of the previously sampled indices J_0.

Lemma 3.8. *Let $S \subseteq [k]$ be a uniformly random t-element set, and let $1 \leq \ell \leq k - t$. The probability that $[k]\setminus S$ contains a contiguous subset I of length $\geq \ell$, i.e., $I = \{a, a+1, \ldots, a+\ell-1\}$ for $1 \leq a \leq k-\ell+1$, is at most $(k-\ell+1)(1-\frac{t}{k})^\ell$.*

Proof. The probability that $[k] \setminus S$ contains a contiguous subset I of length at least ℓ is the same as the probability that it contains a contiguous subset of length *exactly* ℓ. This is in turn given by

$$\Pr[\exists\, a \in \{1, \ldots, k-\ell+1\} : \{a, \ldots, a+\ell-1\} \cap S = \emptyset].$$

By a union bound, this is at most

$$\sum_{a=1}^{k-\ell+1} \Pr[\{a, \ldots, a+\ell-1\} \cap S = \emptyset].$$

By the uniform randomness of S, each of the latter probabilities is the same, and given by

$$\Pr[\{a, \ldots, a+\ell-1\} \cap S = \emptyset] = \frac{\binom{k-\ell}{t}}{\binom{k}{t}} = \frac{(k-t)(k-t-1)\cdots(k-t-l+1)}{k(k-1)\cdots(k-l+1)} \leq \left(\frac{k-t}{k}\right)^{\ell}.$$

We conclude that the probability that $[k] \setminus S$ contains a contiguous subset I of at least ℓ is at most $(k-\ell+1)(1-\frac{t}{k})^{\ell}$. □

Procedure GroverMultipleFast(U_x, k_{est}, ρ, λ)

Input: Quantum oracle U_x to access $x \in \{0,1\}^N$, an integer $k_{\text{est}} \geq 1$ such that $|x|/2 \leq k_{\text{est}} \leq 3|x|/2$, a failure probability $\rho > 0$, threshold parameter $\lambda \in [6, k_{\text{est}}]$.

Output: Classical list of indices $J \subseteq [N]$.

Guarantee: If λ and ρ are such that $\log(6k_{\text{est}}/\rho) \leq \lceil k_{\text{est}}/\lambda \rceil$, then, with probability $\geq 1-\rho$, we have $|J| = |x|$ and $x_j = 1$ for all $j \in J$.

Analysis: Theorem 3.9

$J \leftarrow \emptyset$;
$t \leftarrow \lceil k/\lambda \rceil$;
$R \leftarrow 6\ln(2)(t+1) + 2\ln(1/\rho)\ln(3/2)$;
use `GroverCoupon`$(U_x, R, \frac{2}{3}k_{\text{est}}, t)$ to find, with probability $\geq 1-\rho/3$, a sorted list $J_0 \subseteq [N]$ with $x_j = 1$ for all $j \in J_0$, $|J_0| = t$;
set $J \leftarrow J_0$ and write $J_0 = \{a_1 < a_2 < \cdots < a_t\}$;
set $a_0 = 0$ and $a_{t+1} = N+1$;
for $i = 0, \ldots, t$ **do**
    If $a_{i+1} = a_i + 1$, continue with next loop; otherwise, let $b_i = 2^{\lceil \log(a_{i+1}-1-a_i) \rceil}$;
    construct from U_x an oracle U_y which implements access to the bit string $y \in \{0,1\}^{b_i}$ given by $y_j = x_{a_i+j}$ if $a_i + j < a_{i+1}$, and 0 otherwise;
    $(k_j)_{\text{est}} \leftarrow$ `ApproxCount`$(U_y, \frac{1}{2}, \frac{\rho}{3(t+1)})$;
    use `GroverCertaintyMultiple`$(U_y, 2(k_j)_{\text{est}})$ to find all $j \in (a_i, a_{i+1})$ such that $x_j = 1$, and add these to J;
end for
return J;

Theorem 3.9. *Let $x \in \{0,1\}^N$ with $|x| = k \geq 2$, and assume one knows $k_{\text{est}} \geq 1$ such that $k/2 \leq k_{\text{est}} \leq 3k/2$. Let $0 < \rho < 1$ and $6 \leq \lambda \leq k_{\text{est}}$ be such that $t := \lceil k_{\text{est}}/\lambda \rceil \geq \log(6k_{\text{est}}/\rho)$. Then*

$$O\left(\sqrt{Nk}\left(1 + \frac{1}{\sqrt{\lambda}}\log(k/\rho\lambda)\right)\right)$$

quantum queries to x suffice to, with probability $\geq 1-\rho$, find all k indices i s.t. $x_i = 1$. The algorithm uses an additional

$$O\Big(\sqrt{Nk}\lambda \log(k/\rho)\log(N)\Big)$$

non-query gates.

We remark here that `GroverMultipleFast` takes a multiplicative estimate k_{est} of k as additional input, which can be found with $O(\sqrt{N/k}\log(1/\rho))$ quantum queries and $O(\sqrt{N/k}\log(1/\rho)\log(N))$ additional gates; see Corollary 2.8. Both of these costs are dominated by that of finding the actual elements. The above theorem also includes a parameter λ that allows for a trade-off between query complexity and gate complexity. Before we provide the proof of Theorem 3.9, let us highlight the two extremal cases that follow from taking λ either as large as useful or as small as possible.

Corollary 3.10. *Let $x \in \{0,1\}^N$ with $|x| = k \geq 2$. Assume one knows k_{est} such that $k/2 \leq k_{\text{est}} \leq 3k/2$. Let $1 > \rho > 0$. Then we can find, with probability $\geq 1-\rho$, all k indices i for which $x_i = 1$ using either:*

- *$O(\sqrt{Nk})$ quantum queries and time complexity $O(\sqrt{Nk}\min\{\log^3(k/\rho), k\}\log(N))$, via Theorem 3.9 with $\lambda = \min\{k_{\text{est}}/\log(6k_{\text{est}}/\rho), \log^2(k_{\text{est}}/\rho)\}$,[6] or,*
- *$O(\sqrt{Nk}\log(k/\rho))$ quantum queries and time complexity $O(\sqrt{Nk}\log(k/\rho)\log(N))$, via Theorem 3.9 with $\lambda = 6$.*

Proof of Theorem 3.9. Let $t = \lceil k_{\text{est}}/\lambda \rceil$. Note that because $\lambda \geq 6$ and $k_{\text{est}} \leq 3k/2$, we have $t \leq k/2$. Therefore we can find t of the solutions using the procedure of Proposition 3.7 with probability $\geq 1-\rho/3$, using

$$O\left(\sqrt{\frac{N}{k}}(t + \log(1/\rho))\right) = O\left(\sqrt{\frac{N}{k}}\left(\frac{k}{\lambda} + \log(1/\rho)\right)\right) \tag{3.2}$$

queries and

$$O\left(\sqrt{\frac{N}{k}}\left(\frac{k}{\lambda} + \log(1/\rho)\right)\log(N)\right) \tag{3.3}$$

gates. We remark here that these upper bounds hold because $t \leq k/2 < k$. Indeed, under that assumption on t and k we have $k(H_k - H_{k-t}) \leq 2(t+1)$ by Lemma 3.3, and moreover the factor $1/\ln(3k/(k+2(t-1)))$ is $\Theta(1)$ (it lies between $1/\ln(3)$ and $1/\ln(3/2)$). This shows that calling `GroverCoupon` with $R = 6\ln(2)(t+1) + 2\ln(1/\rho)\ln(3/2) \in \Theta(t + \log(1/\rho))$ has the desired behaviour.

Let $a_1 < a_2 < \cdots < a_t$ denote the found indices for which $x_{a_j} = 1$ and define the intervals $I_0 = \{1, \ldots, a_1 - 1\}$, $I_t = \{a_t + 1, \ldots, N\}$, and, for $j \in [t-1]$, $I_j = \{a_j + 1, \ldots, a_{j+1} - 1\}$. We use k_j to denote the (unknown) number of marked elements in I_j, so in particular $\sum_{j=0}^{t} k_j \leq k - t$. Then by Lemma 3.8, the probability that there is a k_j larger than $\ell := \frac{k}{t}(\log(k) + \log(3/\rho))$ is at most

$$(k-\ell+1)\left(1-\frac{t}{k}\right)^{\ell} \leq 2^{\log(k)}\left(1-\frac{t}{k}\right)^{\frac{k}{t}(\log(k)+\log(3/\rho))} \leq 2^{\log(k)}\left(\frac{1}{2}\right)^{\log(k)+\log(3/\rho)} = \rho/3.$$

[6] Strictly speaking, this choice of λ could be smaller than 6, but in that case `GroverCertaintyMultiple` already has the stated complexity.

Here we used that $\ell \geq 1$, $(1-\frac{t}{k})^{k/t} \leq \frac{1}{e} \leq \frac{1}{2}$, and $\log(k)+\log(3/\rho) \geq 0$.[7] For the rest of the argument we may thus assume that there is no interval with more than ℓ not-yet-found marked elements.

In the next step of our algorithm we search for all marked elements in each interval. To do so for the jth interval, we search over the elements from $[2^{\lceil \log(|I_j|)\rceil}]$ marking an element $i \in [2^{\lceil \log(|I_j|)\rceil}]$ if $x_{i+a_j} = 1$ and $i \leq |I_j|$ (letting $a_0 = 0$). One can implement this unitary using $O(1)$ quantum queries and $O(\log(N))$ gates (to implement the addition and comparison). For each interval, we first compute an estimate $(k_j)_{\text{est}}$ of k_j that satisfies $k_j/2 \leq (k_j)_{\text{est}} \leq 3k_j/2$ using Corollary 2.8, with success probability $\geq 1-\rho/(3(t+1))$. The associated query cost is $O(\sqrt{|I_j|/(k_j+1)}\log(t/\rho))$, and it uses $O(\sqrt{|I_j|/(k_j+1)}\log(t/\rho)\log(N))$ additional gates. Then Lemma 3.2 shows that we can find all marked elements in the j-th interval with probability 1 using $O(\sqrt{|I_j|\,(k_j)_{\text{est}}})$ quantum queries and $O(\sqrt{|I_j|}\,(k_j)_{\text{est}}^{3/2}\log(N))$ additional gates. By a union bound, with probability $\geq 1-\rho/3$, all $(k_j)_{\text{est}}$ are correct, and this step has a total query complexity of

$$O\left(\sum_{j=0}^{t}\sqrt{|I_j|k_j}+\sum_{j=0}^{t}\sqrt{|I_j|}\log(t/\rho)\right) = O\Big(\sqrt{Nk}+\sqrt{Nt}\log(t/\rho)\Big) = O\Big(\sqrt{Nk}(1+\frac{\log(k/\rho\lambda)}{\sqrt{\lambda}})\Big), \tag{3.4}$$

where the first step uses Cauchy–Schwarz for both terms (reading $\sqrt{|I_j|}$ as $\sqrt{|I_j|\cdot 1}$ for the second term) and $\sum_{j=0}^{t}|I_j| \leq N$, $\sum_{j=0}^{t}k_j = k$. To analyze the gate complexity of this step, we first bound $\sum_{j=0}^{t}k_j^3$. We have $\|\mathbf{k}^2\|_\infty \leq \ell^2 = O(\lambda^2\log^2(3k/\rho))$ where $\mathbf{k}$ is the vector with entries k_j and $\mathbf{k}^2$ is the entrywise square of $\mathbf{k}$. As we also have $\|\mathbf{k}\|_1 \leq k$ we get $\sum_{j=0}^{t}k_j^3 = \langle \mathbf{k}, \mathbf{k}^2\rangle \leq \|\mathbf{k}\|_1\|\mathbf{k}^2\|_\infty = O(k\ell^2)$. Then the gate complexity of the final search steps becomes:

$$\begin{aligned}
&O\left(\left(\sum_{j=0}^{t}\sqrt{|I_j|k_j^3}+\sum_{j=0}^{t}\sqrt{|I_j|}\log(t/\rho)\right)\log(N)\right)\\
&= O\left(\sqrt{N}\left(\sqrt{\sum_{j=0}^{t}k_j^3}+\sqrt{t}\log(t/\rho)\right)\log(N)\right)\\
&= O\Big(\sqrt{N}\left(\sqrt{k}\ell+1+\sqrt{t}\log(t/\rho)\right)\log(N)\Big)\\
&= O\Big(\sqrt{N}\left(\sqrt{k}\lambda\log(k/\rho)+\sqrt{k/\lambda}\log(k/\rho\lambda)\right)\log(N)\Big)\\
&= O\Big(\sqrt{Nk}\left(\lambda\log(k/\rho)+\sqrt{1/\lambda}\log(k/\rho\lambda)\right)\log(N)\Big)\\
&= O\Big(\sqrt{Nk}\lambda\log(k/\rho)\log(N)\Big),
\end{aligned} \tag{3.5}$$

where we again used Cauchy–Schwarz in the first step, and the total error probability is bounded by $\rho/3+\rho/3+(t+1)\cdot\frac{\rho}{3(t+1)} = \rho$.

To conclude, the upper bound on the total query complexity follows by combining Eqs. (3.2)

[7] Note also that $\ell \leq k$ because $t \geq \log(6k_{\text{est}}/\rho) \geq \log(3k/\rho)$ by assumption; if $\ell > k$, then the probability of having an interval of length $\geq \ell$ is of course 0, and in this regime one may just as well run `GroverCertainty-Multiple` on the whole string (and have zero failure probability).

and (3.4):

$$O\left(\underbrace{\sqrt{\frac{N}{k}}\left(\frac{k}{\lambda}+\log(1/\rho)\right)}_{\text{sample } t \text{ elements}}+\underbrace{\sqrt{Nk}\left(1+\frac{1}{\sqrt{\lambda}}\log(k/\rho\lambda)\right)}_{\text{find remaining elements}}\right)$$
$$=O\left(\sqrt{Nk}\left(1+\frac{1}{\sqrt{\lambda}}\log(k/\rho\lambda)\right)+\sqrt{\frac{N}{k}}\log(1/\rho)\right)=O\left(\sqrt{Nk}\left(1+\frac{1}{\sqrt{\lambda}}\log(k/\rho\lambda)\right)\right).$$

Here the first equality uses that $\sqrt{\frac{N}{k}}\frac{k}{\lambda}\le\sqrt{Nk}$ since $\lambda\ge 1$. The second equality follows since $\log(1/\rho)\le\log(6k_{\text{est}}/\rho)$ and, by assumption, $\log(6k_{\text{est}}/\rho)\le\lceil k_{\text{est}}/\lambda\rceil=t\le k$. A similar argument using Eqs. (3.3) and (3.5) and $\lambda\ge 1$, establishes the desired gate complexity:

$$O\left(\underbrace{\sqrt{\frac{N}{k}}\left(\frac{k}{\lambda}+\log(1/\rho)\right)\log(N)}_{\text{sample } t \text{ elements}}+\underbrace{\sqrt{Nk}\lambda\log(k/\rho)\log(N)}_{\text{find remaining elements}}\right)$$
$$=O\left(\sqrt{Nk}\left(\frac{1}{\lambda}+\lambda\log(k/\rho)\right)\log(N)+\sqrt{\frac{N}{k}}\log(1/\rho)\log(N)\right)$$
$$=O\left(\sqrt{Nk}\lambda\log(k/\rho)\log(N)\right).$$ □

4 Improved query complexity for approximate summation

In this section, we provide an algorithm `ApproxSum`, which given quantum query access to a binary description of $v\in[0,1]^N$, in the sense of Definition 2.2, finds a multiplicative δ-approximation of $s=\sum_{i=1}^N v_i$ with probability $\ge 1-\rho$ using

$$O\left(\sqrt{\frac{N}{\delta}\log(1/\rho)}\right) \tag{4.1}$$

quantum queries and a similar gate complexity (with only a polylogarithmic overhead). In the above (4.1) we have made very mild assumptions on the value of ρ and δ; a precise statement is given in Theorem 4.3. The algorithm is given in `ApproxSum`. By slightly perturbing the entries of v, we may assume without loss of generality that all entries of v are distinct; we shall make this assumption throughout this section, and have made this assumption in the description of the algorithm as well.

We briefly explain the overall strategy. Recall from the proof of Theorem 2.7 that it is useful to preprocess the vector v by using quantum maximum finding to find $v_{\max}=\max_{i\in[N]}v_i$, and then to use amplitude estimation on the vector $w=v/v_{\max}$. We take this approach slightly further: we first find the largest k entries $z_1,\dots,z_k$ of v, where $k=\Theta(pN)$ for $p\in(0,1)$, and sum their values classically. Let $\tilde{z}$ be the smallest value among the $z_1,\dots,z_k$.[8] For the next part, we treat the corresponding entries of v as zero: checking whether one exceeds the threshold $\tilde{z}$ is a binary comparison, hence can be done in superposition without explicitly using their indices,

[8]We actually first compute a good value of $\tilde{z}$ using a quantile estimation subroutine [Ham21, Thm. 3.4] and then find all the z_j's. Alternatively, one could use [DHHM06, Thm. 3.4] to find all $\Theta(pN)$ largest elements directly, but our approach has the advantage of being able to use the better ρ-dependence of our version of Grover search.

Procedure ApproxSum(U_v, δ, p, λ, ρ)

Input: Quantum query access U_v to $(0, b)$-fixed point representations of $v \in [0,1]^N$, $\delta \in (0,1)$, $p \in (0,1)$, $\lambda \geq 6$, failure probability $\rho > 0$.
Output: A real number $\tilde{s}$.
Guarantee: With probability $\geq 1 - \rho$, $\tilde{s}$ is a multiplicative δ-approximation of s.
Analysis: Theorem 4.3

use Theorem 4.2 to compute $\tilde{z} \in [0,1]$ such that with probability $\geq 1 - \rho/4$, $Q(p) \leq \tilde{z} \leq Q(cp)$, where $c < 1$ is a universal constant and Q is defined in Eq. (4.2);
let $x \in \{0,1\}^N$ be defined by $x_i = 1$ if $v_i \geq \tilde{z}$ and $x_i = 0$ otherwise;
let U_x implement quantum query access to x by applying U_v, comparing to $\tilde{z}$, and uncomputing U_v;
compute estimate k_{est} of $k = |x|$ satisfying $\frac{k}{2} \leq k_{\text{est}} \leq \frac{3k}{2}$ with probability $\geq 1 - \rho/4$ using Corollary 2.8;
use `GroverMultipleFast`$(U_x, k_{\text{est}}, \rho/4, \lambda)$ to find all indices $i_1, \ldots, i_k$ such that $x_{i_j} = 1$;
if $\tilde{z} = 0$ **then**
    return $\sum_{j=1}^{k} v_{i_j}$;
else
    construct unitary U_w for query access to $w \in [0,1]^N$ where $w_i = 0$ if $v_i \geq \tilde{z}$ and $w_i = v_i/\tilde{z}$ otherwise;
    let U be a unitary such that $U|0\rangle = |\psi\rangle$ given by

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle \left(\sqrt{\tilde{w}_i}\,|1\rangle + \sqrt{1 - \tilde{w}_i}\,|0\rangle\right)$$

    where α_i is a $\lceil \log(4N/\delta) \rceil$-bit approximation of $\arcsin(\sqrt{w_i})$, and $\sqrt{\tilde{w}_i} = \sin(\alpha_i)$;
    use `AmpEst`(U, M) with $M = \lceil 12\pi\sqrt{\delta^2 pc} \rceil$, increased to the next power of 2 if necessary, with $c < 1$ from Theorem 4.2, to compute $\tilde{a} \approx \sum_i \tilde{w}_i/N$, and repeat $O(\log(1/\rho))$ times and take the mean of the outputs to achieve success probability $\geq 1 - \rho/4$;
    return $\sum_{j=1}^{k} v_{i_j} + N\tilde{z}\tilde{a}$;
end if

and so with one query to v we can implement quantum oracle access to the vector $w \in [0,1]^N$ defined by

$$w_i = \begin{cases} \frac{v_i}{\tilde{z}} & \text{if } v_i < \tilde{z} \\ 0 & \text{else.} \end{cases}$$

This has the effect of amplifying the small elements in v at no extra cost. We then use amplitude estimation to compute $\sum_{i=1}^{N} w_i$ with additive precision $O(\delta s/\tilde{z})$ (without knowing s). This yields an additive δs-approximation of $\sum_{i=1}^{N} v_i$ (i.e., a multiplicative δ-approximation), where we use that

$$\sum_{i=1}^{N} v_i = \sum_{i=1}^{k} z_i + \tilde{z} \sum_{i=1}^{N} w_i$$

To balance the costs of these two stages we need to carefully choose $\tilde{z}$. We do so by estimating the p-th quantile of the vector. We first give an algorithm `ApproxSum` whose complexity depends on the quantile p and then give a suitable choice for p that allows us to obtain (4.1), see Theorem 4.3 and Corollary 4.4.

We use the following lemma to derive a bound on the required precision for certain arithmetic operations.

Lemma 4.1 ([BHMT02, Lem. 7]). *If $a = \sin^2(\theta_a)$ and $\tilde{a} = \sin^2(\tilde{\theta}_a)$ for $\theta_a, \tilde{\theta}_a \in [0, 2\pi]$, then $|\tilde{\theta}_a - \theta_a| \leq \delta$ implies $|\tilde{a} - a| \leq 2\delta\sqrt{a(1-a)} + \delta^2$.*

For the quantile estimation, we use a subroutine from [Ham21]. Let $v \in [0,1]^N$. Then for $p \in (0,1)$, we define the p-quantile $Q(p) \in [0,1]$ by

$$Q(p) = \sup\{z \in [0,1] : |\{i \in [N] : v_i \geq z\}| \geq pN\}. \tag{4.2}$$

In words, $Q(p)$ is the largest value $z \in [0,1]$ such that there are at least pN entries of v which are larger than z. The subroutine we invoke allows one to produce an estimate for $Q(p)$, in the following sense:

Theorem 4.2 ([Ham21, Thm. 3.4]). *There exists a universal constant $c \in (0,1)$ such that the following holds: Let $v \in [0,1]^N$ and let U_v be a unitary implementing quantum oracle access to v. Then $O(\log(1/\rho)/\sqrt{p})$ applications of controlled-U_v and controlled-$U_v^\dagger$ suffice to find, with probability $\geq 1-\rho$, a value $\tilde{z}$ such that $Q(p) \leq \tilde{z} \leq Q(cp)$. The algorithm uses an additional $O((\log(1/\rho)/\sqrt{p})\, b \log(b) \log(N))$ gates.*

The actual access model for which the above theorem holds is more general, but we have instantiated it for our setting. The gate complexity overhead follows from having to implement their access model from ours, which involves arithmetic and comparisons on the fixed point representations we use, and the fact that the underlying technique is amplitude amplification. We now get to the main theorem of this section, which proves the correctness of `ApproxSum` and analyzes its complexity.

Theorem 4.3. *Let $v \in [0,1]^N$, let U_v be a unitary implementing quantum query access to $(0,b)$-fixed point representations of v, and let $\delta \in (0,1)$. Let $p, \rho \in (0,1)$ and choose $6 \leq \lambda \leq \min\{cpN/\log(pN/\rho), \log(cpN/\rho)^2\}$. Then* `ApproxSum` *computes, with probability $\geq 1-\rho$, a multiplicative δ-approximation of $s = \sum_{i=1}^N v_i$. It uses*

$$O\left(\frac{\log(1/\rho)}{\sqrt{p}} + \sqrt{\frac{N}{Np+1}}\log(1/\rho) + N\sqrt{p}\left(1 + \frac{1}{\sqrt{\lambda}}\log(Np/\lambda\rho)\right) + \frac{1}{\delta\sqrt{p}}\log(1/\rho)\right)$$

quantum queries, and the number of additional gates is bounded by

$$O\left(\frac{\log(1/\rho)}{\sqrt{p}}\, b\log(b)\log(N) + \sqrt{\frac{N}{Np+1}}\log(1/\rho)\log(N) \right.$$
$$\left. + N\sqrt{p}\lambda\log(pN/\rho)\log(N) + \frac{1}{\delta\sqrt{p}} b\log(b)\log(N/\delta)\log^2\log(N/\delta)\log(1/\rho)\right).$$

Before we give the proof, we discuss two useful regimes for p and λ:

Corollary 4.4. *Let $v \in [0,1]^N$, let U_v be a unitary implementing quantum oracle access to $(0,b)$-fixed point representations of v, and let $\delta \in (0,1)$. Then we can find, with probability $\geq 1-\rho$, a multiplicative δ-approximation of $s = \sum_{i=1}^N v_i$, using:*

- *$O(\sqrt{N\log(1/\rho)/\delta})$ quantum queries, when $p = \Theta(\log(1/\rho)/(\delta N)) < 1$ and we choose $\lambda = \min\{cpN/\log(6pN/\rho)), \log(cpN/\rho)^2\} \geq 6$, and using $\sqrt{N/\delta}\,\mathrm{poly}(\log(1/\rho), b, \log(N), \log(1/\delta))$ additional gates, or*

- $O(\sqrt{N/\delta}\log(1/\rho))$ *quantum queries when* $p = \Theta(1/(\delta N)) < 1$ *and we choose* $\lambda = 6$*, and using* $\sqrt{N/\delta}\ \mathrm{poly}(\log(1/\rho), b, \log(N), \log(1/\delta))$ *additional gates.*

Proof of Theorem 4.3. We assume without loss of generality that all the entries of v are distinct. If this is not the case, one can perturb the i-th entry of v by $i2^{-\ell}$ for some sufficiently large $\ell = \Omega(\log(N) + b)$, where we recall that b is the number of bits describing v_i, and discarding these trailing bits from the output value $\tilde{s}$.

We use Theorem 4.2 to find a value $\tilde{z}$ such that the number of elements of v that are at least as large as $\tilde{z}$, is at most pN and at least cpN. The number of quantum queries is

$$O\left(\frac{\log(1/\rho)}{\sqrt{p}}\right),$$

and the number of additional gates used is

$$O\left(\frac{\log(1/\rho)}{\sqrt{p}}\, b\log(b)\log(N)\right).$$

Let $k = |\{i \in [N] : v_i \geq \tilde{z}\}|$. By the assumption that the v_i are all distinct, $cpN \leq k \leq pN$. We next compute a multiplicative $\frac{1}{2}$-approximation of k using Corollary 2.8. This uses

$$O\left(\sqrt{N/(k+1)}\log(1/\rho)\right)$$

quantum queries and

$$O\left(\sqrt{N/(k+1)}\log(1/\rho)\log(N)\right)$$

additional gates. The next step is to find all k such elements using `GroverMultipleFast` (Theorem 3.9). This uses

$$O\left(\sqrt{Nk}\left(1 + \frac{1}{\sqrt{\lambda}}\log(k/(\lambda\rho))\right)\right)$$

quantum queries and

$$O\left(\sqrt{Nk\lambda}\log(k/\rho)\log(N)\right)$$

additional gates.

Let $z_1, \ldots, z_k$ be the entries of v that are $\geq \tilde{z}$. Then

$$\sum_{i=1}^{N} v_i = \sum_{j=1}^{k} z_j + \tilde{z}\sum_{i=1}^{N} w_i$$

where

$$w_i = \begin{cases} \frac{v_i}{\tilde{z}} & \text{if } v_i < \tilde{z} \\ 0 & \text{otherwise.} \end{cases}$$

As we have found all the z_j's, we can compute their sum exactly; therefore, to determine a multiplicative δ-approximation of s, we must produce an additive δs-approximation of $\tilde{z}\sum_{i=1}^{N} w_i$. Let $\varepsilon := \delta s$; note that we do not know s as we do not know δ. Then we have to approximate $\frac{1}{N}\sum_{i=1}^{N} w_i$ with precision $\varepsilon/(N\tilde{z})$. For this, we use amplitude estimation as follows. First, one can implement query access to U_w by using two quantum queries to v and $O(b\log(b))$ non-query gates, by querying an entry, comparing the entry to $\tilde{z}$, and conditional on the comparison

uncomputing the query, and lastly performing the division by $\tilde{z}$. From this, we can construct a unitary U with $U|0\rangle = |\psi\rangle$ satisfying

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle \left(\sqrt{\tilde{w}_i}\,|1\rangle + \sqrt{1-\tilde{w}_i}\,|0\rangle \right),$$

where $\tilde{w}_i$ is close to w_i. One can implement such a unitary as follows. First, set up a uniform superposition over the index register using $O(\log(N))$ gates. Use U_w to load binary descriptions of the entries of w. Calculate a $\lceil \log_2(4N/\delta) \rceil$-bit approximation α_i of $\arcsin(\sqrt{w_i})$ using $O(\log(bN/\delta)\log^2\log(bN/\delta))$ gates [BZ10, Ch. 4]. Then conditionally rotate the last qubit from 0 to 1 over angles $\pi/4$, $\pi/8$, et cetera, depending on the bits of α_i. Lastly, we uncompute α_i and U_w to return work registers to the zero state, and we have obtained the desired state $|\psi\rangle$, where $\sqrt{\tilde{w}_i} = \sin(\alpha_i)$. We now show that $\tilde{w}_i = \sin(\alpha_i)^2$ is close to w_i, and hence

$$a := \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_i = \| |\psi_1\rangle \|^2$$

is close to $\frac{1}{N}\sum_{i=1}^N w_i$. Lemma 4.1 shows that if $|\alpha_i - \arcsin(\sqrt{w_i})| \le \xi$, then

$$|\tilde{w}_i - w_i| = |\sin^2(\alpha_i) - w_i| \le 2\xi\sqrt{w_i(1-w_i)} + \xi^2 \le \xi + \xi^2.$$

Since α_i is a $\lceil \log_2(4N/\delta) \rceil$-bit approximation of $\arcsin(\sqrt{w_i})$, we may apply the above with $\xi = \delta/(4N)$ for every $i \in [N]$. Because $s \ge \tilde{z}$, $\delta = \varepsilon/s \le \varepsilon/\tilde{z}$, and $\delta \le 1$, so the total error satisfies

$$|a - \frac{1}{N}\sum_{i=1}^N w_i| \le \frac{1}{N}\sum_{i=1}^N |\tilde{w}_i - w_i| = \xi + \xi^2 \le \frac{\delta}{4N} + \frac{\delta^2}{16N^2} \le \frac{\varepsilon}{2N\tilde{z}}.$$

Next, we use this to derive an upper bound on a:

$$a \le \frac{\varepsilon}{2N\tilde{z}} + \frac{1}{N}\sum_{i=1}^N w_i = \frac{\varepsilon}{2N\tilde{z}} + \frac{1}{N}\sum_{i:v_i<\tilde{z}} \frac{v_i}{\tilde{z}} \le \frac{2s}{N\tilde{z}},$$

where the last inequality uses $\varepsilon = \delta s \le s$ and $\sum_{i:v_i<\tilde{z}} v_i \le s$. Therefore, using `AmpEst` with M applications of U yields a number $\tilde{a} \in [0,1]$ with

$$|\tilde{a} - a| \le 2\pi \frac{\sqrt{a(1-a)}}{M} + \frac{\pi^2}{M^2} \le 2\pi \frac{\sqrt{2s/(N\tilde{z})}}{M} + \frac{\pi^2}{M^2}$$

by Lemma 2.3. We now determine an appropriate number of rounds M to be used for amplitude estimation. We will choose M such that $|\tilde{a} - a| \le \frac{1}{2}\varepsilon/(N\tilde{z})$; if we do so, then by the triangle inequality $|\tilde{a} - \frac{1}{N}\sum_{i=1}^N w_i| \le \varepsilon/(N\tilde{z})$. The claim is that any $M \ge 12\pi\sqrt{N\tilde{z}/(\varepsilon\delta)}$ suffices, as then

$$2\pi \frac{\sqrt{2s/(\tilde{z}N)}}{M} \le \frac{2\pi\sqrt{2}}{12\pi} \frac{\sqrt{s/(N\tilde{z})}}{\sqrt{N\tilde{z}/(\varepsilon\delta)}} = \frac{\sqrt{2}}{6}\frac{\varepsilon}{N\tilde{z}} \le \frac{1}{4}\frac{\varepsilon}{N\tilde{z}},$$

and, using $\delta \le 1$,

$$\frac{\pi^2}{M^2} \le \frac{\varepsilon\delta}{144N\tilde{z}} \le \frac{1}{4}\frac{\varepsilon}{N\tilde{z}}.$$

Even though we do not know ε, by choosing p carefully, we can enforce upper bounds on $\tilde{z}$ and give a safe choice for M. We use that the number of entries k which are at least $\tilde{z}$ satisfies $k \ge cpN$, so that

$$cp\,N\,\tilde{z} \le \sum_{i:v_j \ge \tilde{z}} v_j \le s,$$

i.e., $\tilde{z} \leq s/(cpN)$. Therefore it suffices to take $M = 12\pi/\sqrt{\delta^2 pc}$, as this satisfies

$$M = 12\pi\sqrt{\frac{1}{\delta^2 pc}} = 12\pi\sqrt{\frac{s}{\delta\varepsilon pc}} \geq 12\pi\sqrt{\frac{N\tilde{z}}{\delta\varepsilon}},$$

This guarantees that $|\tilde{a} - \frac{1}{N}\sum_{i=1}^{N} w_i| \leq \varepsilon/(N\tilde{z})$, and the output value $\tilde{s} = \sum_{j=1}^{k} z_j + N\tilde{z}\tilde{a}$ satisfies

$$|\tilde{s} - s| \leq \varepsilon = \delta s.$$

The number of quantum queries used for this step is therefore $O(M) = O(1/(\delta\sqrt{p}))$, and the number of additional gates used is $O(Mb\log(b)\log(N/\delta)\log^2\log(N/\delta))$. To amplify the success probability to $1-\rho$, we repeat the above procedure $\log(1/\rho)$ many times and output the median of the individual estimates. The query- and gate complexity of the entire algorithm follow by combining those of the four parts: the quantile estimation, the approximate counting, Grover search for finding all large elements, and amplitude estimation for approximating the sum of the small elements. □

Acknowledgements

We would like to thank Ronald de Wolf for helpful discussions and comments on an early version of this work, and Yassine Hamoudi for helpful discussion regarding [Ham21]. We also thank anonymous referees for their feedback. HN acknowledges support by the Dutch Research Council (NWO grant OCENW.KLEIN.267), by the European Research Council (ERC) through ERC Starting Grant 101040907-SYMOPTIC and ERC Grant Agreement No. 81876432, and by VILLUM FONDEN via the QMATH Centre of Excellence (Grant No. 10059). JvA was supported by the Dutch Research Council (NWO/OCW), as part of QSC (024.003.037) and by QuantumDelta NL.

References

[AdW17] Srinivasan Arunachalam and Ronald de Wolf. Optimizing the number of gates in quantum search. *Quantum Info. Comput.*, 17(3-4):251–261, 2017. doi:10.26421/qic17.3-4.

[AJ06] José A. Adell and P. Jodrá. Exact Kolmogorov and total variation distances between some familiar discrete distributions. *Journal of Inequalities and Applications*, 2006(1):1–8, 2006. doi:10.1155/JIA/2006/64307.

[vAGL+21] Joran van Apeldoorn, Sander Gribling, Yinan Li, Harold Nieuwboer, Michael Walter, and Ronald de Wolf. Quantum algorithms for matrix scaling and matrix balancing. In *Proceedings of 48th International Colloquium on Automata, Languages, and Programming (ICALP'21)*, volume 198, pages 110:1–110:17, 2021. arXiv:2011.12823, doi:10.4230/LIPIcs.ICALP.2021.110.

[AR20] Scott Aaronson and Patrick Rall. Quantum approximate counting, simplified. In *Symposium on Simplicity in Algorithms*, pages 24–32, 2020. doi:10.1137/1.9781611976014.5.

[BBHT98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998. Earlier version in Physcomp'96. arXiv:quant-ph/9605034.

[BCdWZ99] Harry Buhrman, Richard Cleve, Ronald de Wolf, and Christof Zalka. Bounds for small-error and zero-error quantum algorithms. In *40th Annual Symposium on Foundations of Computer Science (FOCS'99)*, pages 358–368. IEEE Computer Society, 1999.

[BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *Contemporary Mathematics*, pages 53–74. American Mathematical Society, 2002. doi:10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P.

[BZ10] Richard Brent and Paul Zimmermann. *Modern Computer Arithmetic*, volume 18. Cambridge University Press, 2010.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53(1):17–25, January 1995. doi:10.1016/0020-0190(94)00171-T.

[CHI+18] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, jan 2018. doi:10.1098/rspa.2017.0551.

[CLRS22] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 4th edition, 2022.

[DF80] P. Diaconis and D. Freedman. Finite Exchangeable Sequences. *The Annals of Probability*, 8(4):745–764, 1980. URL: https://www.jstor.org/stable/2242823.

[DH96] Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum, 1996. doi:10.48550/arXiv.quant-ph/9607014.

[DHHM06] Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum Query Complexity of Some Graph Problems. *SIAM Journal on Computing*, 35(6):1310–1328, January 2006. doi:10.1137/050644719.

[DKLR00] Paul Dagum, Richard Karp, Michael Luby, and Sheldon Ross. An Optimal Algorithm for Monte Carlo Estimation. *SIAM Journal on Computing*, 29(5):1484–1496, January 2000. doi:10.1137/S0097539797315306.

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16), apr 2008. doi:10.1103/physrevlett.100.160501.

[GN22] Sander Gribling and Harold Nieuwboer. Improved quantum lower and upper bounds for matrix scaling. In *Proceedings of 39th International Symposium on Theoretical Aspects of Computer Science (STACS'22)*, volume 219, pages 35:1–35:23, 2022. arXiv:2109.15282, doi:10.4230/LIPIcs.STACS.2022.35.

[dGdW02] Mart de Graaf and Ronald de Wolf. On Quantum Versions of the Yao Principle. In *19th Symposium on Theoretical Aspects of Computer Science (STACS'02)*, volume 2285 of *Lecture Notes in Computer Science*, pages 347–358, Berlin, Heidelberg, 2002. Springer. doi:10.1007/3-540-45841-7_28.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 38th Annual ACM Symposium on Theory of Computing (STOC'96)*, pages 212–219, 1996. arXiv:quant-ph/9605043, doi:10.1145/237814.237866.

[Gro97] Lov K. Grover. Quantum telecomputation, 1997. Bell Labs Technical Memorandum ITD97-31630F. doi:10.48550/arXiv.quant-ph/9704012.

[Gro98] Lov K. Grover. A framework for fast quantum mechanical algorithms. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC'98)*, pages 53–62, 1998. arXiv:quant-ph/9711043, doi:10.1145/276698.276712.

[Ham21] Yassine Hamoudi. Quantum Sub-Gaussian Mean Estimator. In *29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:17, 2021. doi:10.4230/LIPIcs.ESA.2021.50.

[Jan18] Svante Janson. Tail bounds for sums of geometric and exponential variables. *Statistics & Probability Letters*, 135:1–6, 2018. doi:10.1016/j.spl.2017.11.017.

[Knu98] Donald Ervin Knuth. *The Art of Computer Programming*, volume III. Addison-Wesley, 2nd edition, 1998. URL: https://www.worldcat.org/oclc/312994415.

[KO23] Robin Kothari and Ryan O'Donnell. Mean estimation when you have the source code; or, quantum Monte Carlo methods. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'23)*, pages 1186–1215, 2023. doi:10.1137/1.9781611977554.ch44.

[NC02] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2002.

[NW99] Ashwin Nayak and Felix Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the 31st Annual ACM SIGACT Symposium on Theory of Computing (STOC'99)*, pages 384–393, 1999. arXiv:quant-ph/9804066, doi:10.1145/301250.301349.

[Roo01] B. Roos. Binomial Approximation to the Poisson Binomial Distribution: The Krawtchouk Expansion. *Theory of Probability & Its Applications*, 45(2):258–272, 2001. doi:10.1137/S0040585X9797821X.

[You91] Robert M. Young. 75.9 Euler's Constant. *The Mathematical Gazette*, 75(472):187–190, 1991. doi:10.2307/3620251.

A Classical lower bound for approximate counting

Let $x \in \{0,1\}^N$ be a bit string. We show in this appendix the (well-known) result that any randomized classical algorithm that computes with probability $\geq 5/6$ a δ-multiplicative approximation of the Hamming weight $k = |x|$, must make at least $\Omega(\min(N, \frac{N}{\delta^2 k}))$ queries to k. Although a matching upper bound on the sample complexity follows from an application of Chebyshev's inequality, and the lower bound is claimed in [AR20], we did not succeed in locating a proof for non-constant δ in the literature. We therefore include a proof of the following statement.

Theorem A.1. *There exists a universal constant $C > 0$ such that the following holds. Let $\delta > 0$ be such that $k\delta$ is an integer and $k(1+\delta) \leq N$. Suppose $\mathcal{A}$ is a randomized query algorithm such that for all $x \in \{0,1\}^N$ with $|x| \in \{k, k(1+\delta)\}$,*

$$\Pr_{r \sim \mathrm{Unif}(\{0,1\}^R)}[\mathcal{A}(x,r) = |x|] \geq \frac{5}{6},$$

where R is the number of used random bits, and $\mathrm{Unif}(\{0,1\}^R)$ *refers to the uniform distribution on them. Assume that $\mathcal{A}$ makes $t \geq 0$ queries, independent of the input x, or the randomness r used by the algorithm. Then*

$$t \geq C \min\{N, \frac{N}{\delta^2 |x|}\}$$

At a high level, our proof boils down to showing that if a t-query algorithm $\mathcal{A}$ succeeds with high probability, then the total variation distance between $\mathrm{Hyp}(N, k, t)$ and $\mathrm{Hyp}(N, k(1+\delta), t)$ must be $\Omega(1)$. Here $\mathrm{Hyp}(N, \ell, t)$ is the distribution on the number of observed marked elements if one draws t elements from a set of size N of which ℓ elements are marked, without replacement. The lower bound on t then follows from a t-dependent upper bound on this total variation distance, for which we state the necessary ingredients from the literature first. We let $\mathrm{Bin}(t, p)$ the binomial distribution with parameters $t \geq 1$ and $p \in (0, 1)$, corresponding to t independent Bernoulli trials, each of which succeeds with probability p. First, we state the following bound [DF80, Thm. 3] which shows that when the number of samples t is small compared to N, sampling with and without replacement yield approximately the same distribution.

Theorem A.2. *The total variation distance between* $\mathrm{Hyp}(N, \ell, t)$ *and* $\mathrm{Bin}(t, \ell/N)$ *is at most* $4t/N$.

Next, we use the following estimate on the total variation distance between two binomial distributions with the same t, but distinct success probability [Roo01, AJ06]:

Theorem A.3. *Let $t \geq 1$, $p \in (0, 1)$ and $\delta \in (0, 1)$ such that $p(1+\delta) < 1$. The total variation distance between* $\mathrm{Bin}(t, p)$ *and* $\mathrm{Bin}(t, p(1+\delta))$ *is at most*

$$\frac{\sqrt{e}}{2} \frac{\tau}{(1-\tau)^2},$$

assuming $\tau < 1$, where

$$\tau = \delta \sqrt{\frac{p(t+2)}{2(1-p)}}.$$

Proof. This follows from [Roo01, eq. (15)]. We apply their bound on the distance to $\mathrm{Bin}(t, p)$ with the following parameters: $s = 0$, n is our t, the random variable S_n is the sum of t Bernoulli random variables with success probability $p + x$ with $x = \delta p$, hence its distribution P^{S_n} is $\mathrm{Bin}(t, p + x)$, and

$$\gamma_1(p) = tx,\ \gamma_2(p) = tx^2,\ \eta(p) = 2tx^2 + t^2x^2,\ \theta(p) = \frac{2tx^2 + t^2x^2}{2tp(1-p)}.$$

The upper bound given is then $d_{\mathrm{TV}}(P^{S_n}, \mathrm{Bin}(t, p)) \leq \frac{\sqrt{e}}{2} \frac{\sqrt{\theta(p)}}{(1-\sqrt{\theta(p)})^2}$. The claimed bound in the theorem then follows from $\theta(p) = \tau$, using $x = p\delta$. □

Via the triangle inequality, the above two theorems suffice to upper bound the total variation distance between $\mathrm{Hyp}(N, k, t)$ and $\mathrm{Hyp}(N, k(1+\delta), t)$ (for a precise statement, see the proof below). We are now ready to prove Theorem A.1.

Proof of Theorem A.1. For an integer ℓ, let $X_\ell \subseteq \{0,1\}^N$ be the set of bit strings with Hamming weight ℓ. We use Yao's minimax principle to lower bound the number of queries required by a randomized algorithm that outputs $|x|$ with probability $\geq 5/6$ on every input $x \in X_k \cup X_{k(1+\delta)}$. That is, we exhibit a distribution $\mathcal{D}$ on $X_k \cup X_{k(1+\delta)}$ for which every deterministic algorithm that computes $|x|$ on a $5/6$ fraction of the inputs, weighted according to $\mathcal{D}$, requires at least

$C \min\{N, N/(\delta^2 k)\}$ queries for some universal constant C. Consider the distribution $\mathcal{D}$ on inputs that with probability $1/2$ samples a uniformly random element from X_k, and with probability $1/2$ samples a uniformly random element from $X_{k(1+\delta)}$. Suppose $\mathcal{A}$ is a deterministic t-query algorithm that on input $x \sim \mathcal{D}$ correctly returns $|x|$ with probability at least $5/6$ (where the probability is over the sample from $\mathcal{D}$). Note that we allow $\mathcal{A}$ to know k and δ. We show the desired lower bound on t. Let $A(x) = a$ denote the substring $(x_{i_1}, \ldots, x_{i_t})$ of x that corresponds to the t queried indices $i_1, \ldots, i_t$. Note that the output $\mathcal{A}(x) \in \{k, k(1+\delta)\}$ of the algorithm is deterministic and *only* a function of a (for a fixed algorithm). If one thinks of $\mathcal{A}$ as a decision tree, then the first index to be queried does not depend on a, and after every subsequent query, the next index to be queried is deterministic as a function of the previous queried indices and outcomes. It is also the case that the queried indices $i_1, \ldots, i_t$ are a function of the query outcomes a! Therefore, we may view $\mathcal{A}(x)$ as a just a function of $a = A(x)$. Let $B \subset X_k \cup X_{k(1+\delta)}$ be the set of $a \in \{0,1\}^t$ on which $\mathcal{A}(a)$ outputs $k(1+\delta)$.

Let P_ℓ be the distribution on $a = A(x) \in \{0,1\}^t$ induced by $x \sim \mathrm{Unif}(X_\ell)$, where the latter refers to the uniform distribution on X_ℓ. By assumption, A can distinguish (with constant success probability) the distributions P_k and $P_{k(1+\delta)}$. Therefore, the total variation distance between these distributions is large. Indeed, the probability, with respect to $\mathcal{D}$, that $\mathcal{A}$ outputs the wrong value of $|x|$ is at most $1/6$, therefore $\mathcal{A}$ fails with probability at most $1/3$ when $x \sim \mathrm{Unif}(X_\ell)$ for both $\ell = k$ and $\ell = k(1+\delta)$, and hence

$$\begin{aligned}
\frac{1}{3} + \frac{1}{3} &\geq \Pr_{x \in_R X_{k(1+\delta)}}[\mathcal{A}(x) = k] + \Pr_{x \in_R X_k}[\mathcal{A}(x) = k(1+\delta)] \\
&= \Pr_{a \sim P_{k(1+\delta)}}[\mathcal{A}(a) = k] + \Pr_{a \sim P_k}[\mathcal{A}(a) = k(1+\delta)] \\
&= 1 + (P_k(B) - P_{k(1+\delta)}(B)) \\
&\geq 1 - d_{\mathrm{TV}}(P_k, P_{k(1+\delta)}),
\end{aligned}$$

so $d_{\mathrm{TV}}(P_k, P_{k(1+\delta)}) \geq 1/3$.

We now relate P_ℓ to the hypergeometric distribution $\mathrm{Hyp}(N, \ell, t)$, so that we can upper bound the above total variation distance as a function of t. We prove that

$$P_\ell(a) = \Pr_{x \in X_\ell}[A(x) = a] = \frac{1}{\binom{t}{|a|}} \Pr[W_\ell = |a|]$$

where $W_\ell \sim \mathrm{Hyp}(N, \ell, t)$ is hypergeometrically distributed, i.e.,

$$\Pr[W_\ell = |a|] = \frac{\binom{\ell}{|a|}\binom{N-\ell}{t-|a|}}{\binom{N}{t}}.$$

We prove this by exploiting the permutation symmetry of the distribution on X_ℓ, along with an iterative conditioning argument. Let $i_1, \ldots, i_t$ denote the sequence of indices of x queried, so that $A(x) = (x_{i_1}, \ldots, x_{i_t})$. Recall that the $i_1, \ldots, i_t$ may be chosen adaptively, but i_{j+1} is determined completely from $i_1, \ldots, i_j$ and $x_{i_1}, \ldots, x_{i_j}$. Then

$$\Pr_{x \in X_\ell}[x_{i_1} = 1] = \frac{\ell}{N}.$$

Moreover, one has

$$\Pr_{x \in X_\ell}[x_{i_{j+1}} = 1 | x_{i_1}, \ldots, x_{i_j}] = \frac{\ell - \sum_{s=1}^{j} x_{i_s}}{N - j},$$

because conditioned on the values of x at the indices $i_1, \ldots, i_j$, the distribution of x becomes uniform among bit strings of Hamming weight $\ell - \sum_{s=1}^{j} x_{i_s}$ of length $N - j$ in the remaining position. Finally

$$\Pr_{x \in X_\ell}[A(x) = a] = \prod_{j=1}^{t} \Pr_{x \in X_\ell}[x_{i_j} = a_j | x_{i_1} = a_1, \ldots, x_{i_{j-1}} = a_{j-1}]$$

which after careful consideration is seen to be equal to

$$\begin{aligned}\frac{\ell \cdots (\ell - |a| + 1) \cdot (N - \ell) \cdots (N - \ell - (t - |a|) + 1)}{N \cdots (N - t + 1)} &= \frac{\ell!}{(\ell - |a|)!} \frac{(N - \ell)!}{(N - \ell - (t - |a|))!} \frac{(N - t)!}{N!} \\ &= \frac{\binom{N-t}{\ell - |a|}}{\binom{N}{\ell}}\end{aligned}$$

Now because

$$\Pr[W_\ell = |a|] = \frac{\binom{\ell}{|a|}\binom{N-\ell}{t-|a|}}{\binom{N}{t}}$$

we see that we indeed have

$$\Pr_{x \in X_\ell}[A(x) = a] = \frac{1}{\binom{t}{|a|}} \Pr[W_\ell = |a|].$$

Therefore we obtain

$$\begin{aligned} d_{\mathrm{TV}}(P_k, P_{k(1+\delta)}) &= \frac{1}{2} \sum_{a \in \{0,1\}^t} |P_k(a) - P_{k(1+\delta)}(a)| \\ &= \frac{1}{2} \sum_{a \in \{0,1\}^t} \frac{1}{\binom{t}{|a|}} |\Pr[W_k = |a|] - \Pr[W_{k(1+\delta)} = |a|]| \\ &= \frac{1}{2} \sum_{s=0}^{t} |\Pr[W_k = s] - \Pr[W_{k(1+\delta)} = s]| \\ &= d_{\mathrm{TV}}(\mathrm{Hyp}(N, k, t), \mathrm{Hyp}(N, k(1+\delta), t)). \end{aligned}$$

We now give a t-dependent upper bound on this total variation distance; combined with the assumption that $d_{\mathrm{TV}}(P_k, P_{k(1+\delta)}) \geq 1/3$, this will lead to the right lower bound on t. Let

$$\tau = \frac{\delta k}{N} \sqrt{\frac{t+2}{2\frac{k}{N}(1 - \frac{k}{N})}} = \delta \sqrt{\frac{k(t+2)}{2(N-k)}}.$$

If $\tau \geq 1/2$, then

$$t + 2 \geq \frac{(N-k)}{2\delta^2 k}.$$

and so in this case $t = \Omega(N/(\delta^2 k))$ (unless the latter is $O(1)$, in which case the query lower bound we aim for is constant and uninteresting). Otherwise, by the triangle inequality,

$$\begin{aligned} d_{\mathrm{TV}}(\mathrm{Hyp}(N, k, t), \mathrm{Hyp}(N, k(1+\delta), t)) \leq\ & d_{\mathrm{TV}}(\mathrm{Hyp}(N, k, t), \mathrm{Bin}(t, k/N)) \\ &+ d_{\mathrm{TV}}(\mathrm{Bin}(t, k/N), \mathrm{Bin}(t, k(1+\delta)/N)) \\ &+ d_{\mathrm{TV}}(\mathrm{Bin}(t, k(1+\delta)/N), \mathrm{Hyp}(N, k(1+\delta), t)) \end{aligned}$$

$$\leq \frac{8t}{N} + \frac{\sqrt{e}}{2}\frac{\tau}{(1-\tau)^2}$$
$$\leq \frac{8t}{N} + 2\tau\sqrt{e},$$

where we applied Theorems A.2 and A.3 (note that we needed $\tau < 1$). Since the left-hand side is at least $1/3$ as shown before, we have

$$\frac{1}{3} \leq \frac{8t}{N} + 2\tau\sqrt{e},$$

so at least one of the two terms must be $1/6$ or greater. If $8t/N \geq 1/6$, then $t \geq N/48$ and we are done; otherwise, $\tau \geq 1/24$ and so

$$t + 2 \geq \frac{2(N-k)}{24^2\delta^2 k}.$$

If $k \leq N/2$, then $N - k \geq N/2$ and one deduces $t + 2 \geq N/(144\delta^2 k)$. □