An efficient quantum algorithm for the Moebius function

Peter J. Love
Department of Physics, Haverford College, 370 Lancaster Avenue, Haverford, PA 19041

We give an efficient quantum algorithm for the Moebius function \( \mu(n) \) from the natural numbers to \( \{-1, 0, 1\} \). The cost of the algorithm is asymptotically quadratic in \( \log n \) and does not require the computation of the prime factorization of \( n \) as an intermediate step.

I. INTRODUCTION

There are many fascinating connections between number theory and physics. Moebius inversion \cite{1}, the primon gas \cite{2,3}, and the statistics of the nontrivial zeroes of the Riemann zeta function \cite{4} all provide points of contact between questions in number theory and in physics. In 1994 another connection was made when Peter Shor discovered quantum algorithms for order finding and the discrete logarithm, and hence provided a polynomial time quantum algorithm for factoring integers \cite{5}.

The quest for new quantum algorithms has been challenging. Shor’s algorithm was generalized by treating it as a special case of the hidden subgroup problem \cite{6}, and subsequent quantum algorithms were discovered for the hidden subgroup problem over the dihedral group \cite{7}, the Heisenberg group \cite{8}, for the computation of Gauss sums over finite fields and finite rings \cite{9} and of shifted quadratic characters \cite{10}. The determination of hidden structures was generalized to the determination of hidden nonlinear structures \cite{11}. Progress in the discovery of quantum algorithms for algebraic problems is reviewed in \cite{12}.

Rather than seeking specific algorithms one can try to elucidate the difficulty of classes of problems for a quantum computer. This is the goal of quantum complexity theory \cite{13}. Two important classes are BQP and QMA \cite{13,14,15}. These are the quantum equivalents, loosely speaking, of the classical complexity classes P and NP, which capture precisely the notions of the difficulty of finding a solution (P and BQP) versus the difficulty of verifying a solution (NP and QMA). A number of complete problems are known for BQP \cite{16,17} and QMA \cite{14,15,15,15}.

In the search for a new quantum algorithm for a particular problem one is therefore engaged in a war on two fronts. Firstly one must have some basis to believe the target problem is in BQP. Secondly one must utilize or discover quantum algorithmic techniques that can efficiently solve the problem at hand. It makes sense to seek new algorithms for problems that we know are quantumly efficient to address - such as problems in number theory whose solution can be obtained from the prime factorization of an integer. One example of such a problem is the determination of whether an integer is square free. Any positive integer may be written for problems that we know are quantumly efficient to address - such as problems in number theory whose solution can be obtained from the prime factorization of an integer. One example of such a problem is the determination of whether an integer is square free. Any positive integer may be written

\[
\mu(n) = \begin{cases} 
-1 & \text{if } n = p_1 p_2 \ldots p_t \\
0 & \text{otherwise}
\end{cases}
\tag{1}
\]

where the \( p_i \) are distinct primes. Evidently computation of the Moebius function is in BQP, as we can use Shor’s algorithm to determine the prime factorization of \( n \). The algorithm of \cite{19} provides partial information about the Moebius function. Testing square freeness corresponds to distinguishing whether \( \mu(n) = 0 \) from \( \mu(n) \neq 0 \).

Once one has an efficient quantum algorithm for a problem, to show quantum speedup requires evidence that the problem attacked is classically difficult - and this is itself a high barrier. Shor’s algorithm is not an example of a provable quantum speedup, because the classical complexity of factoring remains unknown. The best publicly known classical algorithm, the number field sieve, has complexity \( O(\exp(c(\log n)^{1/3}(\log \log n)^{2/3})) \) for some constant \( c \). Shor’s algorithm takes time \( O((\log n)^2(\log \log n)(\log \log \log n)) \) on a quantum computer. Hence, while we know that factoring is easy on a quantum computer, we do not know it is difficult on a classical computer. Likewise, the complexity of classical computation of the Moebius function is unknown.

Recently, an approach to providing partial information about the difficulty (or lack thereof) of factoring was proposed by Sarnak \cite{21,22}. What is referred to as the the \( AC^0 \) prime number theorem states that any function \( f \)
from the integers to \( \pm 1 \) that can be computed by a constant depth polynomial sized circuit has a correlation with the Mobius function that tends to zero \([21, 26]\). That is:

\[
\frac{1}{n} \sum_{x \leq n} f(n) \mu(n) = o(1). \tag{2}
\]

Sarnak’s conjecture therefore states that even determining the little information given about the prime factorization by the Mobius function is difficult. This conjecture was recently proved by Green \([26]\). It has also been shown that \([2]\) holds if \( f \) is a monotone Boolean function \([27]\). These results imply that the Mobius function cannot be computed by circuits in \( AC^0 \), or by monotone circuits. These results set lower bounds on computing the Mobius function, however, they do not exclude the possibility of an efficient classical algorithm for \( \mu(n) \), although no such algorithm is currently known. As noted above, one expects the computation of some functions with nonzero correlation with the Mobius function (in particular, the Mobius function itself) to be efficient quantum.

A second reason that the quantum computation of the Mobius function is of interest is through its connection to quantum simulation. There is a broad toolkit of quantum simulation techniques, and experimental progress is encouraging. The Mobius function appears as a natural observable of the primon gas, a non-interacting quantum field theory whose elementary excitations correspond to the prime numbers \([2, 3]\). The Mobius function corresponds to the fermion operator \((-1)^F\) in the fermionic theory and the Witten index in the supersymmetric theory \([2]\). An efficient quantum simulation of the primon gas (by the methods of \([28]\) or developments thereof) should be able to compute the Mobius function. The complexity of such simulation algorithms could then be evaluated by comparison with the direct approach we shall develop in the remainder of the paper.

The subject of the present paper is a quantum algorithm for the Mobius function which does not rely on prime factorization as an intermediate step. This question is of independent interest, apart from the considerations described above, simply because the number of distinct efficient quantum algorithms remains small. The paper is organized as follows. In Section \( \text{II} \) we describe the algorithmic techniques that are used in the algorithm. In Section \( \text{III} \) we describe how the algorithm of \([19]\) is modified to enable us to separate cases where \( \mu(n) = 0 \) from cases where \( \mu(n) \neq 0 \). In Section \( \text{IV} \) we describe the algorithm for \( \mu(n) \neq 0 \). We close the paper with some discussion and directions for future work.

### II. METHODS

In this Section we introduce the various quantum algorithmic techniques that we shall use to compute the Mobius function. We briefly describe these transformations here - in the interest of making the present paper self contained further details are given in the Appendix. We shall make use of the fact that any classical computation may be performed reversibly and included as a coherent subroutine of a quantum algorithm. We define \( b = \lceil \log_2 n \rceil \) as the number of bits used to represent the argument \( n \).

We require the preparation of the uniform superposition of \( n \) logical basis states \( |l\rangle \) for \( 0 \leq l \leq n - 1 \).

\[
S_n |0\rangle^\otimes b = \frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} |l\rangle. \tag{3}
\]

The method we use is given in \([6]\) and described in detail in Appendix \( 1 \). The cost of this procedure is \( O(b^2) \).

The second technique we use is the quantum Fourier transform on a Hilbert space of arbitrary dimension \( n \).

\[
U_{\text{FT}} : |x\rangle |n\rangle \mapsto \frac{1}{\sqrt{n}} \sum_{y=0}^{n-1} e^{2\pi i xy/n} |y\rangle |n\rangle \tag{4}
\]

Methods for performing this transform efficiently are given in \([6]\), \([12]\), Section III C., and \([29]\). In Appendix \( 2 \) we describe the method of \([6]\) in some detail. The cost of the procedure of \([6]\) is \( O(b^2) \), this is reduced by the method of \([29]\), however this does not change the asymptotic scaling of our algorithm for the Mobius function.

We will compute two number-theoretic functions as intermediate steps in our computation of the Mobius function: the gcd and the Jacobi symbol. Both functions can be computed efficiently using standard techniques of reversible computation. The gcd can be computed using the extended Eulers algorithm \([19, 30]\). Improved quantum circuits for the GCD were given in \([31]\). The details of the quantum circuits for the gcd and for Jacobi symbol computation are given in \([32]\).
The Jacobi symbol is denoted \(( \frac{m}{n} )\), which for an integer with prime factorization \( n = p_1^{k_1} p_2^{k_2} p_3^{k_3} \ldots \) is given by

\[
( \frac{m}{n} ) = ( \frac{m}{p_1} )^{k_1} ( \frac{m}{p_2} )^{k_2} ( \frac{m}{p_1} )^{k_3} \ldots 
\]

(5)

where the \(( \frac{m}{p} )\) are Legendre symbols defined by:

\[
( \frac{m}{p} ) = \begin{cases} 
1 & \text{if } m \not\equiv 0 \mod p \text{ and } m \equiv x^2 \mod p \text{ for some } x \\
0 & \text{if } m \equiv 0 \mod p \\
-1 & \text{if } a \not\equiv x^2 \mod p \text{ for any } x
\end{cases}
\]

(6)

Both the Jacobi symbol and the GCD may be computed efficiently classically with a cost \( O(b^2) \). The classical reversible circuits (given in [32]) can then be used as coherent subroutines in our quantum algorithm. These two computations may be written:

\[
U_{\gcd} : \langle 0 \rangle \otimes b | x \rangle | n \rangle \mapsto | \gcd(x, n) \rangle | x \rangle | n \rangle
\]

\[
U_K : | 0 \rangle | x \rangle | n \rangle \mapsto \frac{1}{2} \left[ 1 + | \frac{x}{n} \rangle \right] | x \rangle | n \rangle \quad \text{for } \gcd(x, n) = 1
\]

(7)

where, \( q \in \{0, 1\} \).

Using our reversible computation of the Jacobi symbol we can implement the transformation

\[
U_J := | 0 \rangle | x \rangle | n \rangle \mapsto ( \frac{x}{n} ) | 0 \rangle | x \rangle | n \rangle
\]

(8)

using phase kickback:

\[
U_J = U_K (R_{\pi} \otimes 1) U_K
\]

(9)

where \( R_{\pi} \) is the transformation:

\[
R_{\pi} : | 0 \rangle \mapsto | 0 \rangle, \quad R_{\pi} : | 1 \rangle \mapsto - | 1 \rangle
\]

(10)

where we have used the fact that \( U_J \) is self inverse because the Jacobi symbols are \( \pm 1 \) when \( \gcd(x, n) = 1 \).

In the next section we will see how these algorithmic techniques are used in a modification of the method of [19] to discriminate arguments \( n \) for which \( \mu(n) = 0 \) from those for which \( \mu(n) \neq 0 \).

III. SQUARE-FREENESS

We can combine the computation of the gcd with the Fourier transform to prepare the so-called gcd state. First, we apply the algorithm given in [6] and described in Appendix 1 to prepare a uniform superposition of all values of \( x \) from 0 to \( n - 1 \). Applying \( U_{\gcd} \) to this superposition yields the state:

\[
| g_n \rangle = \frac{1}{\sqrt{n}} \left[ \sum_{\gcd(x,n)=1} | x \rangle | 1 \rangle + \frac{1}{\sqrt{n}} \sum_{\gcd(x,n)>1} | x \rangle | \gcd(x,n) \rangle \right].
\]

(11)

Where we use the fact that \( \gcd(0,n) = n \). Measuring the rightmost register gives a uniform superposition over all values of \( x \) co-prime to \( n \) with probability

\[
\frac{\phi(n)}{n} \geq \frac{1}{e^\gamma \log \log n + \frac{3}{\log \log n}}
\]

(12)

where \( \phi \) is the Euler totient function and \( \gamma \approx 0.577216 \) is the Euler-Mascheroni constant. A number of attempts scaling with \( \log \log n \) will yield this state with high probability.

We define Gauss sums over the integers modulo \( N \):

\[
S^w_n = \sum_{m=0}^{n-1} \left( \frac{m}{n} \right) e^{2\pi i m / n}
\]

(13)
FIG. 1: Quantum circuit for the preparation of the gcd state. Here $b = \lceil \log_2 n \rceil$ and we suppress showing any ancilla qubits needed for the calculation of the gcd by the extended Euclidean algorithm.

FIG. 2: Preparation of the state whose amplitudes are given by the Gauss sums defined in equation (13). We suppress display of the normalization of the states, and of any ancillas needed to compute the Jacobi symbols or to compute the gcd using the extended Euclidean algorithm. The key output of this stage of the algorithm is the upper ancilla qubit, which is one if and only if the input $n$ is square free. This bit can then be used to restrict computation of the Moebius function to values of $n$ where $\mu(n) \neq 0$

These Gauss sums have the dichotomous property that:

$$
S_n^y = 0 \quad \forall y \text{ such that } \gcd(y, n) > 1 \quad \text{if } n \text{ is squarefree} \\
S_n^y = 0 \quad \forall y \text{ such that } \gcd(y, N) = 1 \quad \text{if } n \text{ is not squarefree} \quad (14)
$$

As a first step in using this property to distinguish square-free values of $n$, on which the Moebius function is non-zero, we prepare the state

$$
|\chi\rangle = \frac{1}{\sqrt{\phi(n)}} \sum_{\gcd(x, n) = 1} (\frac{x}{n}) |x\rangle 
$$

first introduced in [19], where $\phi(n)$ is the Euler phi function [30]. This state is an example of a chi-state, used in [9] and other quantum algorithms for algebraic problems [12]. The preparation of the prime state, the algorithm for square freeness and for the computation of Gauss sums uses the same technique of probabilistic preparation of a quantum state by reversible function computation and measurement [9, 12, 19, 33, 34].

By taking the Fourier transform of our chi state we obtain

$$
U_{FT} : |\chi\rangle |n\rangle \mapsto \frac{1}{\sqrt{n\phi(n)}} \sum_{\gcd(x, n) = 1} \sum_{m=0}^{n-1} \left( \frac{x}{n} \right) e^{2\pi imx/n} |m\rangle |n\rangle = \frac{1}{\sqrt{n\phi(n)}} \sum_{m=0}^{n-1} S_m^n |m\rangle . 
$$

Here we depart from the procedure of [19]. We add a register and apply $U_{\gcd}$ again:

$$
U_{\gcd} : \frac{1}{\sqrt{n\phi(n)}} \sum_{m=0}^{n-1} S_m^n |m\rangle |0\rangle \mapsto |\text{disc}\rangle = \frac{1}{\sqrt{n\phi(n)}} \sum_{m=0}^{n-1} S_m^n |m\rangle |\gcd(m, n)\rangle . 
$$

Now the dichotomic property of the Gauss sum enables us to discriminate square free $n$. We have:

$$
|\text{disc}\rangle = \begin{cases} 
\frac{1}{\sqrt{n\phi(n)}} \sum_{\gcd(m, n) = 1} S_m^n |m\rangle |1\rangle & \text{if } n \text{ squarefree} \\
\frac{1}{\sqrt{n\phi(n)}} \sum_{\gcd(m, n) > 1} S_m^n |m\rangle |\gcd(m, n)\rangle & \text{if } n \text{ not squarefree}
\end{cases} 
$$

(18)
FIG. 3: Preparation of the ancilla state and the gcd state by uncomputation of the chi state. The gcd state is the input to the next stage of the Moebius function computation. This is obtained by applying the inverse Fourier transform to the output of the circuit shown in Figure 2, and the applying \( U_J \) again, using the fact that \( U_J \) is self inverse.

Therefore operations conditioned on the ancilla qubit being in state \( |1\rangle \) will act on the register storing \( n \) only when \( n \) is square-free, i.e. only when the Moebius function \( \mu(n) \) is nonzero. Conversely, a measurement of any value of the ancilla register other than \( n \) shows that \( n \) is not square free, and hence \( \mu(n) \neq 0 \).

IV. THE MOEBIUS FUNCTION

We can now move beyond the results of [19] and define an algorithm to compute \( \mu(n) \) for values of \( n \) such that \( \mu(n) \neq 0 \). The Moebius function can be defined as a sum of primitive roots:

\[
\mu(n) = \sum_{1 \leq x \leq n, \gcd(x,n)=1} e^{2\pi i \frac{x}{n}}
\]  

(19)

We define a generalized version of the Moebius function as follows:

\[
\mu(y)(n) = \sum_{1 \leq x \leq n, \gcd(x,n)=1} e^{2\pi i \frac{xy}{n}}
\]  

(20)

This function has the following property:

\[
\mu(ya)(n) = \mu(a)(n) \text{ if } \gcd(y,n) = 1
\]

(21)

which is due to the fact that if \( \gcd(y,n) = 1 \) then \( y \) is an element of the multiplicative group of integers modulo \( n \), and hence \( \mu(y)(n) \) differs from \( \mu(n) \) only by a reordering of the terms in the sum. This is specific case of a more general property of Gauss sums over finite fields and rings that is exploited in the efficient quantum algorithms for computing them [9]. For general \( y \) this function is related to the Moebius function as follows:

\[
\mu(y)(n) = \frac{\phi(n)}{\phi(n/\gcd(n,y))} \mu(n/\gcd(y,n)).
\]

(22)

where \( \phi \) is the Euler totient function.

We may define an algorithm for the Moebius function as follows. First we prepare the gcd state using the method described above:

\[
|g^n\rangle |n\rangle = \frac{1}{\sqrt{\phi(n)}} \sum_{1 \leq x \leq n, \gcd(x,n)=1} |x\rangle |n\rangle
\]

(23)

Next we apply the Fourier transform to this state:

\[
\text{FFT} : |g^n\rangle |n\rangle \mapsto \frac{1}{\sqrt{n\phi(n)}} \sum_{1 \leq y \leq n} \left( \sum_{1 \leq x \leq n, \gcd(x,n)=1} e^{2\pi i xy/n} \right) |y\rangle |n\rangle = \frac{1}{\sqrt{n\phi(n)}} \sum_{1 \leq y \leq n} \mu(y)(n) |y\rangle |n\rangle
\]

(24)
we may divide this sum as follows:

$$\frac{1}{\sqrt{n\phi(n)}} \sum_{1 \leq y \leq n} \mu(y) |y\rangle = \frac{\mu(n)}{\sqrt{n\phi(n)}} \sum_{1 \leq y \leq n} |y\rangle + \frac{1}{\sqrt{n\phi(n)}} \sum_{\gcd(y,n) > 1} \mu(y) |y\rangle$$ \hspace{1cm} (25)

Adding another ancilla register we act with $U_{\gcd}$ and obtain:

$$U_{\gcd} \frac{1}{\sqrt{n\phi(n)}} \sum_{1 \leq y \leq n} \mu(y) |y\rangle |0\rangle^b = \frac{\mu(n)}{\sqrt{n}} |g^n\rangle |1\rangle + \frac{1}{\sqrt{n\phi(n)}} \sum_{\gcd(y,n) > 1} \mu(y) |y\rangle |\gcd(y,n)\rangle$$ \hspace{1cm} (26)

Measurement of the second register yields the result $|1\rangle$ with probability $\phi(n)/n$. The state in the first register after the measurement is $(-1)^t |\chi\rangle$, where $t$ is defined as in eq. \[1\].

By running the above procedure as a controlled operation with a single ancilla bit initially prepared in the state $(|0\rangle + |1\rangle)/\sqrt{2}$ we obtain:

$$\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) |g^n\rangle \rightarrow |0\rangle |g^n\rangle + (-1)^t |1\rangle |g^n\rangle = \frac{1}{\sqrt{2}} (|0\rangle + (-1)^t |1\rangle) |g^n\rangle$$ \hspace{1cm} (27)

performing a Hadamard operation on the control bit then results in the state:

$$H : \frac{1}{\sqrt{2}} (|0\rangle + (-1)^t |1\rangle) |g^n\rangle \rightarrow \left(\frac{1 + (-1)^t}{2} |0\rangle + \frac{1 - (-1)^t}{2} |1\rangle\right) |g^n\rangle = |t \mod 2\rangle |g^n\rangle.$$ \hspace{1cm} (28)

Measuring the ancilla bit then yields the value of $t$ modulo two with probability one, from which the value of the Moebius function may be inferred.
V. DISCUSSION AND CONCLUSIONS

We have defined an efficient quantum algorithm for the computation of the Moebius function. The number of qubits required for an argument \( n \) represented by \( b = \lceil \log n \rceil \) qubits is linear in \( b \) and the number of gates is quadratic in \( b \). The cost of this algorithm can be compared to both Shor’s algorithm and the algorithm for detecting square-freeness of \([19]\), which our algorithm uses as a subroutine. The cost of Shor’s algorithm is dominated by the cost of modular exponentiation. For a \( b \)-bit integer this is the cost of \( O(b^2) \) squarings and multiplications of \( b \) bit numbers. The cost therefore scales as \( b \) times the cost of multiplication. For small integers primary-school multiplication algorithms are optimal, and multiplication of two \( b \)-bit numbers requires \( O(b^3) \) operations, and so using this multiplication algorithm the cost of Shor’s algorithm is \( O(b^3) \). The Schönhage-Strassen algorithm has the best known asymptotic scaling for multiplication and uses \( O(b \log b \log \log b) \) time, and so employing this multiplication method Shor’s algorithm requires \( O(b^3 \log b \log \log b) \) time and \( O(b \log b \log \log b) \) space. To completely factor an integer using Shor’s algorithm, and hence compute \( \mu(n) \) we require \( O(b^3 \log b \log \log b) \) time.

The cost of the algorithm given here is given by the cost of uniform superposition preparation, GCD computation, Jacobi symbol computation and Fourier transform. Each of these is \( O(b^2) \) and hence the overall algorithm is \( O(b^2) \), a significant improvement over repeated use of Shor’s algorithm. This advantage is derived from the fact that the algorithm does not require modular exponentiation, and hence avoids the need for repeated squarings and multiplication. The part of our algorithm that computes \( \mu(n) \) for \( \mu(n) = 0 \) is slightly simpler than the full algorithm of \([19]\) because we do not compute the square part of the integer - we only need to detect square-freeness.

Beyond evaluation of the Moebius function one would of course like to apply the Moebius transformation \([1]\) to a function supplied as quantum data. For the very general setting of Moebius transforms of functions in terms of a set and its power set such transforms have been defined, realizing a Grover speedup over the classical cost \([37]\). The existence of an efficient algorithm for the Moebius function should enable the efficient application of the Moebius transform.

Another recent strand of quantum information connected to number theory is the definition and study of so-called sequence states \([33–35]\). These are states in which the only basis states that have non-zero amplitudes are labelled by a sequence of integers with a particular number theoretic property. The prime state, for example, is a superposition of all the computational basis states labelled by prime numbers \([34]\). The properties of this state, in particular its entanglement, were studied in \([33]\). Because primality testing is classically efficient in general \([36]\) (and for restricted values of \( n \) tests of pseudoprimality such as Miller-Rabin may be used \([24]\), these states can be prepared efficiently. In cases where efficient classical computation is not possible one may use Groverization techniques to obtain a polynomial speedup over the classical cost \([35]\). In \([33]\) the Moebius state is defined as:

\[
\sum_{\alpha<2^n} \mu(a) \ket{a}
\]

The preparation of such a superposition would require coherently computing the value of \( \mu(n) \) for many values of \( n \) in quantum parallel. In this case the success probability of the algorithm becomes an issue, and amplification techniques would be needed to increase the probability of simultaneous success of the computation of each value of \( \mu(n) \).

Our algorithm utilizes the quantum algorithm for the detection of square-freeness of an integer \([19]\). This is an area where there has been recent progress in classical algorithms \([38]\), and it remains an interesting question to compare the classical and quantum algorithms for detection of square-freeness.

We leave these areas of inquiry to future work.
Acknowledgements

I would like to thank Peter Sarnak, Borzu Toloui, Andrew White and Alan Aspuru-Guzik for productive discussions. This project is supported by NSF award PHY-0955518 and by AFOSR award no FA9550-12-1-0046.

[1] Nanxian, C. (2010) Möbius inversion in physics. (World Scientific).
[2] Spector, D. (1990) Communications In Mathematical Physics 127, 239–252.
[3] Julia, B. L. (1990) Statistical theory of numbers, Springer Proceedings in Physics eds. Luck, J. M, Moussa, P, & Waldschmidt, M. (Springer-Verlag), Vol. 47, p. 276293.
[4] Schumayer, D & Hutchinson, D. A. W. (2011) SIAM Journal on Computing 30, 1484–1509.
[5] Kitaev, A. Y. (2008) eprint arXiv:quant-ph/0506254.
[6] Kuperberg, G. (2005) SIAM Journal on Computing 35, 170–188.
[7] Bacon, D. (2008) Quantum Info. Comput. 8, 438–467.
[8] Shor, P. W. (1997) SIAM Journal on Computing 26, 1484–1509.
[9] Sarnak, P. (2010). Three Lectures on the Mobius Function Randomness and Dynamics. Available at http://publications.ias.edu/node/506.
[10] Sarnak, P. (2014). Mobius lectures Summer 2010. Available at http://www.math.princeton.edu/sarnak/Mobius lectures Summer 2010.pdf.
[11] Kalai, G. (2011). The AC0 Prime Number Conjecture, blog post (2011). Available at http://gilkalai.wordpress.com/2011/02/21/the-ac0-prime-number-conjecture/.
[12] Kalai, G. (2011). Walsh Fourier transform of the Mobius function, Math Overflow question (2011). Available at http://mathoverflow.net/questions/57543/walsh-fourier-transform-of-mobius-functions.
[13] Lipton, R. J. (2011). The depth of the Mobius function, blog post (2011). Available at http://rjlipton.wordpress.com/2011/02/23/the-depth-of-the-mobius-function/.
[14] Green, B. (2012) Combinatorics, Probability and Computing 21, 942–951.
[15] Bourgain, J. (2013) Israel Journal of Mathematics 197, 215–235.
[16] Li, J, Peng, X, Du, J, & Suter, D. (2012) Scientific Reports 2.
[17] Jordan, S. P, Lee, K. S. M, & Preskill, J. (2012) Science 336, 1130–1133.
[18] Stein, W. (2011) Elementary Number Theory: Primes, Congruences, and Secrets. pp. 1–172.
[19] Saeedi, M & Markov, I. L. (2013) arXiv:1304.7516 [cs.ET].
Algorithmic techniques

In this Appendix we give details of the algorithmic techniques used above. These techniques are well known in the literature, however, we describe them here in the interest of making the present paper self contained. We represent controlled operations such that if $U$ is a unitary operator, then $\Lambda_0(U)$ and $\Lambda_1(U)$ are controlled unitary operations applied when the control bit is 0 or 1, respectively. This means:

$$\begin{align*}
\Lambda_0(U)[0] \otimes |\psi_0\rangle + [1] \otimes |\psi_1\rangle &= [0] \otimes (U |\psi_0\rangle) + [1] \otimes |\psi_1\rangle \\
\Lambda_1(U)[0] \otimes |\psi_0\rangle + [1] \otimes |\psi_1\rangle &= [0] \otimes |\psi_0\rangle + [1] \otimes (U |\psi_1\rangle)
\end{align*}$$

(30)

1. Preparation of Uniform superposition states

The first technique we require is the preparation of the uniform superposition of $n$ logical basis states $|l\rangle$ for $0 \leq l \leq n - 1$. That is, we need to obtain a quantum circuit for the operator $S_n$:

$$S_n |0\rangle^\otimes b = \frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} |l\rangle .$$

(31)

If $n = 2^b$ it is well known that this can be accomplished by the application of $b$ Hadamard gates [39]:

$$\frac{1}{\sqrt{2^b}} \sum_{l=0}^{2^b-1} |l\rangle = (H |0\rangle)^\otimes b .$$

(32)

The superposition for $n \neq 2^b$ can also be prepared efficiently by the method of [4]. First, we take $b = \lceil \log_2 n \rceil$. Write $n = 2^{b-1} + n_1$ and proceed recursively:

$$\frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} |l\rangle = \frac{1}{\sqrt{n}} \sum_{l=0}^{2^{b-1}-1} |l\rangle + \frac{1}{\sqrt{2^{b-1}}} \sum_{l=0}^{n_1-1} |l\rangle .$$

(33)

the highest of the $b$ bits is always zero in the first sum, and always one in the second sum. Hence we may write:

$$\frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} |l\rangle = \frac{1}{\sqrt{n}} \sum_{l=0}^{2^{b-1}-1} |l\rangle + \sum_{l=0}^{n_1-1} \frac{1}{\sqrt{n}} |l\rangle + \frac{1}{\sqrt{2^{b-1}}} \sum_{l=0}^{n_1-1} |l\rangle .$$

(34)

This gives a recursive procedure for the definition of $S_n$ in terms of $S_{n_1}$:

$$S_n |0\rangle^\otimes b = \sqrt{\frac{2^{b-1}}{n}} |0\rangle (H |0\rangle)^\otimes b-1 + \sqrt{\frac{n_1}{n}} |1\rangle S_{n_1} |0\rangle^\otimes b-1$$

$$= [\Lambda_0(H^\otimes b-1) + \Lambda_1(S_{n_1})] \left[ \sqrt{\frac{2^{b-1}}{n}} |0\rangle + \sqrt{\frac{n_1}{n}} |1\rangle \right] |0\rangle^\otimes b-1$$

(35)

So that we may write:

$$S_n = \left[ \Lambda_0(H^\otimes b-1) + \Lambda_1(S_{n_1}) \right] \left[ R_n \otimes 1^\otimes b-1 \right]$$

(36)

where the single qubit rotation $R_n$ is defined by:

$$R_n = \frac{1}{\sqrt{n}} \left( \sqrt{\frac{2^{b-1}}{n_1}} \sqrt{\frac{n_1}{2^{b-1}}} \right)$$

(37)

This procedure, which is represented by the circuit in Figure 7, is repeated until $n_1$ is a power of two, in which case the method of eq. [32] is used. The cost of this procedure is maximal when $n = 2^a - 1$, in which case at each stage of the recursion $n_1$ is also of the form $2^{a'} - 1$, and asymptotically the cost scales as $O(b^2)$. In addition to this, if $n$ is provided in a second register, there will be an additional ancilla cost due to the need to coherently compute the angles of the rotations $R_{n_1}$, so that they can be applied by phase kickback.
FIG. 7: The quantum circuits showing the recursive definition of the algorithm to prepare a uniform superposition of the first $n$ logical basis states. Here $n_1 = n - 2^{\lfloor \log_2 n \rfloor}$, $R_{n_1}$ is the rotation defined in equation 37 where $n$ is a parameter for the rotation, and $H$ is the Hadamard gate. This recursive procedure is applied until $n_1$ is a power of two.

2. The Fourier transform of dimension $n$

The central algorithmic technique we shall use, as in many quantum algorithms, is the quantum Fourier transform. In our case the transform must be applied so that the order of the transform $n$ is arbitrary and is provided coherently via a separate register:

$$U_{FT} : |x\rangle |n\rangle \mapsto \frac{1}{\sqrt{n}} \sum_{y=0}^{n-1} e^{2\pi i xy/n} |y\rangle |n\rangle$$

(38)

This is the case for the quantum algorithm for square-freeness [19], and it is well known how to perform this transform, (See [6], or [12], Section III C., or [29]). We describe the method of [6].

The first step is to use the algorithm of Section 1 to create the state:

$$|0, x\rangle \mapsto |x\rangle \frac{1}{\sqrt{n}} \sum_{y=0}^{n-1} e^{2\pi i xy/n} |y\rangle .$$

(39)

First, a $b$ bit approximation to $x/n$:

$$\bar{x}_n = \sum_{l=1}^{b-1} \left(\frac{x}{n} 2^l \mod 2\right) 2^{-l},$$

(40)

is computed in the ancilla register by coherent reversible computation:

$$|0, x\rangle \mapsto |\bar{x}, x\rangle$$

(41)

next, a uniform superposition of the first $n$ logical basis states is created in the right register using the method of Section 1

$$|\bar{x}, x\rangle \mapsto \frac{1}{\sqrt{n}} \sum_{y=0}^{n-1} |x, y\rangle .$$

(42)

The transformation:

$$|\bar{x}, y\rangle \mapsto \exp (2\pi i \bar{x} y) |x, y\rangle$$

(43)

is applied to this state. This computation proceeds by phase kickback, resulting in:

$$|\bar{x}\rangle \frac{1}{\sqrt{n}} \sum_{y=0}^{n-1} e^{2\pi i \bar{x} y} |y\rangle .$$

(44)

This is a close approximation to the desired transformation, except that the value $\bar{x}$ remains in the first register. We are therefore faced with the problem of uncomputing $\bar{x}$ to return this register to the state $|0\rangle$. This problem is easier to address if we consider approximating the inverse transformation, that is:

$$|0\rangle |e_x\rangle \mapsto |\bar{x}\rangle |e_x\rangle .$$

(45)
Where we have defined:

$$|e_x⟩ = \frac{1}{\sqrt{n}} \sum_{y=0}^{n-1} e^{2\pi i xy/n} |y⟩.$$  \hspace{1cm} (46)

This transformation is easy to interpret if we note that the states $|e_x⟩$ are eigenstates of the shift operator $\sigma |x⟩ → |x + 1 \mod n⟩$ with eigenvalue $e^{-2\pi i x/n}$. We may therefore interpret the inverse of our desired operation as the construction of a pointer state, in which the first register encodes information about the eigenvalue corresponding to the eigenvector. The construction of such states is accomplished, with high probability, by phase estimation as follows.

$$|0⟩ |e_x⟩ ↦ \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} |x⟩ |e_x⟩ ↦ \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} |x⟩ \sigma_y |e_x⟩ = \frac{1}{\sqrt{n}} \sum_{y=0}^{n-1} |y⟩ e^{-2\pi i y x/n} |e_x⟩.$$  \hspace{1cm} (47)

The first stage here is accomplished by the algorithm of Section 1, whereas the second step is performed by the application of the operator:

$$\sum_{y=0}^{2^b-1} |y⟩ ⟨y| \otimes \sigma^y.$$  \hspace{1cm} (48)

This operator can be performed efficiently by using the binary representation of $y$, and using the fact that:

$$\sigma^{2p} |x⟩ = |x + 2p \mod n⟩.$$  \hspace{1cm} (49)

Hence the operation given in eq. 48 that can be implemented by $b$ controlled coherent addition operations, each of which requires a single ancilla qubit and a gate cost linear in $b$ 40. The overall asymptotic cost of this transformation is therefore $O(b^2)$.

The final stage of this algorithm is the Fourier transform of the ancilla register,

$$\frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} |y⟩ e^{-2\pi i y x/n} |e_x⟩ \rightarrow \frac{1}{\sqrt{n}} \sum_{x=0}^{2^b-1} \left( \frac{1}{\sqrt{2^b}} \sum_{y=0}^{n-1} e^{2\pi i y (p/2^b - x/n)} \right) |p⟩ |e_x⟩.$$  \hspace{1cm} (50)

The coefficient in brackets is maximal when $p \approx 2^b x/n$, i.e. when $p \approx \bar{x}_n$. Hence the transformation results in:

$$|0, x⟩ \rightarrow \frac{1}{\sqrt{n}} \sum_{p=0}^{2^b-1} \left( \frac{1}{\sqrt{2^b}} \sum_{y=0}^{n-1} e^{2\pi i y (p/2^b - x/n)} \right) |p⟩ |e_x⟩ \approx |\bar{x}⟩ |e_x⟩.$$  \hspace{1cm} (51)

Hence inverting these operations results, with high probability, in the required transformation. While this is the most simple algorithm to describe, an alternative approach offers superior scaling of $O(\log b \log \log b)$ 29. However, while the approach of 29 is optimal for practical implementations it does not affect the overall asymptotic scaling of our algorithm for the Moebius function.