Primality Testing in Quantum Domain

Kaushik Chakraborty*[1], Anupam Chattopadhyay†[2] and Pratyay Poddar‡[3]

[1]Indian Statistical Institute, Kolkata 700108, India.
[2]UMIC Research Centre, RWTH Aachen University, 52074 Aachen, Germany.
[3]Hitachi Cambridge Laboratory, Cambridge CB3 0HE, Great Britain.

Abstract

In this paper, we have studied the efficiency of Quantum algorithms towards the primality testing algorithm. Our contribution here is to point out where massive computation power of quantum turing machines can be directed in the existing classical primality testing algorithm for significant increase in computational speed.

1 Introduction & Motivation

In terms of efficiency quantum algorithms outperforms the best known classical algorithms in several scenarios. The performance of quantum algorithms has been studied in many domains of both computer science and mathematics. In recent years, researchers have started to solve the mystries of prime numbers using the computation power of quantum turing machine [9]. This paper is also another approach in that direction.

Prime numbers always baffled mathematicians around the globe. In modern times, prime numbers are found to be very important for classical cryptography. It is necessary to find a large prime number for several cryptographic techniques viz. public key cryptography (RSA) [2]. Conforming to that necessity opens up a crucial question - how do we know if that large number n is a prime? Therefore, having found a large enough number, the next step is to find out if that is a prime number. This process is known as primality testing (PT). There are several randomized polynomial time algorithm techniques available for PT viz. *Miller-Rabin Test*, *Solovay-Strassen Test* [2], [4] etc.

*kaushik.chakraborty9@gmail.com
†Anupam.Chattopadhyay@ice.rwth-aachen.de
‡pp374@cam.ac.uk

Until 2002 existence of a deterministic polynomial time algorithm for primality testing was not confirmed. In that year, Agarwal *et. al* [1] has come up with an affirmative answer to that. This algorithm is known as AKS primality testing algorithm, after the name of the inventors.

AKS algorithm is based upon Fermat's little theorem [2]. The main step of AKS algorithm [1] is following,

Theorem 1 *Given an integer $n > 1$, let r be an integer such that $o_r(n) > \log^2 n$. Suppose*

$$(x+a)^n \equiv x^n + a \ (mod\ x^r - 1, n) \quad for\ a = 1, ..., \lfloor \sqrt{\phi(r)} \log n \rfloor. \tag{1}$$

Then, n has a prime factor $\leq r$ or n is a prime number.

The running time of AKS algorithm is $O(r^{1.5} \log^3 n)$, where r is a parameter in the algorithm. In AKS [1] , *Agarwal et al.* showed that the value of $\sqrt{\phi(r)}$ is of $O(\log^5 n)$. So, running time of that algorithm is $O(\log^{10.5} n)$. Many improvements were proposed on the value of r latter on. The detailed improvements of the AKS algorithm is described in the given references [3].[5]. In every improvement of the AKS algorithm there is loop, for testing some condition λ is satisfied for each $1 \leq a \leq g(r)$, where g is a real valued function. This paper points out those iterations and used amplitude amplification algorithm assuming the availability of the oracle U for performing the conditioning operation λ. Use of amplitude amplification algorithm reduces the number of times the oracle operator needs to be called or the condition checking operator for the operation λ quadratic times. In this article, we have used the original AKS algorithm as an example and shown how we get quadratic improvements from classical AKS algorithm.

2 Quantum Algorithm for Primality Testing

In this section we have described our approach for solving the primality test problem. As described in the previous sections, our algorithm is based upon the classical AKS algorithm. Here we have used quantum amplitude amplification technique as a subroutine for the main computational step of AKS algorithm. The structure of AKS algorithm allows us to apply the quantum amplitude amplification algorithm directly. The main computational task of the AKS algorithm is given in Theorem 1. In Theorem 1 the parameter $o_r(n)$ is the smallest number j for which,

$$n^j \equiv 1 \ (mod\ r) \tag{2}$$

$\phi(r)$ is the Euler's totient function of r [2].

For the computation of equation 1 using quantum amplitude amplification algorithm, we have called a subroutine named **QPrime**, in which we have created an equal superposition state of all a's such that $a \in \{0, \lfloor\sqrt{\phi(r)}\log(n)\rfloor\}$. Then we have assumed that there exists an oracle operator U_P, which has the following characteristics,

$$\begin{aligned} U_P|a\rangle &= -|a\rangle \qquad if \;\; (X+a)^n \not\equiv X^n + a \; (mod \; X^r - 1, n) \;\&\; a \neq 0 \\ &= |a\rangle \qquad otherwise \end{aligned}$$

This subroutine will return 0, *iff* $\nexists a \in \{1, \lfloor\sqrt{\varphi(r)}\log(n)\rfloor\}$ such that $(X+a)^n \not\equiv X^n + a \; (mod \; X^r - 1, n)$. The subroutine uses quantum amplitude amplification technique for checking the existance of such a. Using generalized version of quantum amplitude amplification [7], [8] we have solved this problem. The detailed step by step operation has been shown in the detailed version.

QPrime subroutine is a typical amplitude amplification algorithm setting and our goal here is to test whether the function P corresponding U_P is a constant or not. This will give us quadratic speed up from classical sequential searching. So, in quantum setting it will take $O(\lceil\phi(r))^{\frac{1}{4}}(\log n)^{\frac{1}{2}}\rceil)$ many evaluations of U_P instead of $O(\lfloor\sqrt{\phi(r)}\log n\rfloor)$, which is a quadratic speed up. The analysis of the query complexity of **QPrime** subroutine is given in the detailed version of this paper. One can use BBHT [7] algorithm instead of **QPrime** for finding the existance of such a.

3 Conclusion

The quantum reversible circuit realization of the operation $(X + a)^n \equiv X^n + a \; (mod \; X^r - 1, n)$ with optimal number of elimentary gates may also reduce the actual time complexity of the original AKS algorithm. Our future work includes the study of the realization of U_P operator for AKS algorithm and also the circuit realization of the corresponding oracle operators for implementing the condition operation λ.

References

[1] Agrawal Manindra and Kayal Neeraj and Saxena Nitin, PRIMES is in P, 2004 doi=10.4007/annals.2004.160.781

[2] Stinson Douglas, In*Cryptography: Theory and Practice,Second Edition, 2006*, year = 2006, isbn = 1584882069, edition = 3rd, publisher = CRC/C&H

[3] Tsz-wo Sze, A Potentially Fast Primality Test, 2007

[4] Goldwasser Shafi and Kilian Joe, Primality Testing using Elliptic Curves In*J. ACM,* 46(4):450-472,1999

[5] Lenstra, W. Hendrik Jr. and Pomerance Carl, Primality Testing with Gaussian Periods Available at `http://www.math.dartmouth.edu/~carlp/PDF/complexity12.pdf` 2005.

[6] Brassard, Gilles and Høyer, Peter and Mosca, Michele and Tapp, Alain, Quantum amplitude amplification and estimation, In *Quantum computation and information (Washington, DC, 2000)*, volume 305, pages 53–74, Amer. Math. Soc., 2002.

[7] Boyer, Michel and Brassard, Gilles and Høyer, Peter and Tapp, Alain, Tight Bounds on Quantum Serching, In*Fortsch. Phys. 46:493-506,1998* Available at `http://arxiv.org/abs/quant-ph/9605034v1`

[8] Chakraborty, Kaushik and Maitra, Subhamoy Quantum algorithm to check Resiliency of a Boolean function, In*IACR Cryptology ePrint Archive,2013*, volume 2013, pages 232.

[9] Latorre I. Jose and Sierra, German, Available at `http://arxiv.org/abs/1302.6245v2`