QI: Integer Computation: Factoring, Multiplication

Gregory Croisdale, Cade Brown, and Rebecca Ryan May 6, 2020

This paper was written for PHYS494 at the University of Tennessee, Knoxville taught by George Siopsis in Spring, 2020.

Contents

1 Abstract

In this paper, we discuss some of the historically important quantum algorithms involving integers and their factors. We compare the feasibility and speed of some of the most impressive classical algorithms with our own implementation of the well-discussed "breaker of RSA": Shor's Algorithm[4,5]. We describe attempts and problems with implementing the Lucas-Lehmer Primality Test[6] and the Fourier Transform for Multiplication[7].

2 Accompanying Materials

The source code of all the algorithms we refer to in this paper can be found and ran from this GitHub Project:

https://github.com/gcrois/QFastInteger.

A static version of the demo page is available for viewing at the following address:

https://gcrois.github.io/QFastInteger/demo.html.

3 Introduction

4 Implementation of Shor's Algorithm for Factoring Integers

Shor's algorithm is perhaps one of the most anticipated quantum algorithms, for the risk it poses to encryption (see below in the document), as well as the interesting result that drastically improves the computational complexity of integer factorization, a very interesting problem.

This algorithm will be interesting to implement, because it uses elements such as QFT, as well as partial computation on classical and quantum systems, which combine to solve the problem in polynomial time $O((\log n)^2(\log\log n)(\log\log\log n)$.

While probabilistic algorithms already exist for primality testing on classical computers, finding out which numbers divide another number is still a very hard to solve problem using solely classical computers.

Shor's algorithm: https://qudev.phys.ethz.ch/static/content/QSIT15/Shors%20Algorithm.pdf

5 Breaking RSA

6 Lucas-Lehmer Primality Test

7 Fourier Transform for Multiplication

8 Conclusion

We will visually demonstrate the fastest classical computing methods to the aforementioned problems in both a Jupyter notebook and an HTML5 website for easy accessibility. We will animate the algorithms using Google Charts and a small input.