Open Source Implementation of the Durr and Hoyer Algorithm

Mridul Sarkar

University of California-Davis

Chapter 1

Introduction

I began with an idea of quantum neural networks, a seemingly conceivable idea given the naive thought that each node of the classical neural network could be thought of as a particle in space, with quantum mechanics dictating the connections between particles. Though after review of many notable research papers I found the theory was not lacking in this area of quantum computing neither was application. I was amazed by the growth in the field.

Disheartened at first I continued to look for a way to contribute to knowledge in quantum computing. I found Unity Fund and their goals resonated with my own. They proposed a project for an open source package of the Durr and Hoyer Algorithm. It is an algorithm that excites me becasue just a year ago in my first data structures class I was taught O(N) is the best we can do for a list. I was very excited to see an algorithm that pushes this big-O to an awesome $O(\sqrt{N})$.

Chapter 2

Background

2.1 Quantum Computing

Quantum computing's backbone is undoubtedly quantum mechanics. Superposition, Wave Function Collapse, Entanglement, and Uncertainty are utilized by quantum computing to harness the smallest computational unit, a qubit, in order to improve computational effectiveness and efficiency. (1) The details of the quantum mechanic principles used in quantum computing will be explained in section 2.3. Through application of quantum principles we can harness a qubit and from there we can harness multiple qubits by extending quantum principles onto computational space.

2.2 Qubits and Quantum Gates

The classical computer utilizes two states, 0 and 1. Two possible states for a qubit are $|0\rangle$ and $|1\rangle$. Where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

In order to measure either state $|0\rangle$ or $|1\rangle$ we normalize the coefficients a_1 and a_2 into:

$$|a_1|^2 + |a_2|^2 = 1$$

Measuring the qubits we get either $|0\rangle$ with probability $|a_1|^2$ or $|1\rangle$ with probability $|a_2|^2$. To better under qubits one must examine larger systems that end up being much more useful in application. There is a more generalized form of ψ. Observe $a = a_1, a_2..., a_n$ as representation of an arbitrary vector.

$$\sum_{i=1}^{n} a_i \, |x_i\rangle$$

Following the rules of linear algebra we can extend this definition to apply an arbitrary transformation matrix A which can be understood as a quantum logic

gate.
To better understand an arbitrary qubit $|x\rangle = |x_1\rangle ... |x_n\rangle$ we need to understand how the transformation matrix A with i columns and j rows and the arbitrary vector a interact with our n-dimensional qubit. We generalize this to:

$$\psi = \sum_{i=1}^{n} (\sum_{j=1}^{n} A_{ij} a_j) |x_i\rangle$$

We will now examine some useful Quantum gates. Again quantum gates are analgous to the transformation matrix mentioned above, A. Review of Classical logic gates is recommended.

It is important to remember that any Unitary matrix is a quantum gate. (3) Not Gate:

2.3 Introduction to Quantum Mechanics and Algorithms

2.3.1 Quantum Mechanics Principles

Schrodinger equation

1-D page 15 (9) time independent (35) 2-D/3-D (135)

Uncertainty principle

page 32,122 of (9)

Identical Particles

page 191 (9)

2.3.2 Quantum Computing Principles

These postulates will help us understand how quantum mechanic principles translate into linear algebra. Allowing us to see numbers and no longer imagine quantum mechanics as some dark magic, as I did about two months ago. Additionally we are given information on how to use tools mathematicians are already familiar with in order to develop our own quantum algorithms.

Postulate 1

Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the

system's state space.

Postulate 2

The evolution of a closed quantum system is described by a unitary transformation. That is, the state—of the system at time t1 is related to the state —of the system at time t2 by a unitary operator U which depends only on the times t1 and t2

Postulate 2'

The time evolution of the state of a closed quantum system is described by the Schrodinger equation

Postulate 3

Quantum measurements are described by a collectionMm of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is—immediately before the measurement then the probability that result m occurs is
" 80-84, (3)

2.3.3 Useful Definitions

Taking the information from 2.3.2 we can understand quantum mechanics and their influence on computing we to help guide us while we build an algorithm. Using these basic definitions we can keep our classically trained brain in check.

Quantum Superposition

If a system can be in state A or state B, it can also being a "mixture" of the two states. If we measure it, we see either A or B, probabilistically. (1) (9)

Quantum Entanglement

If a system can be in state A or B, it has to be a mixture of both states. When measuring the system the independent components cannot be measured unless related to each other. (1) (9)

Wave Function Collapse

When the wave function, existing in the Hermitian space in superposition as multiple eigenstates, collapses to a single eigenstate due to interaction with the external world. (1)(9)

Uncertainty principle

Pairs of measurements where greater certainty of the outcome of one measurement implies greater uncertainty of the outcome of the other measurement.(1)(9)

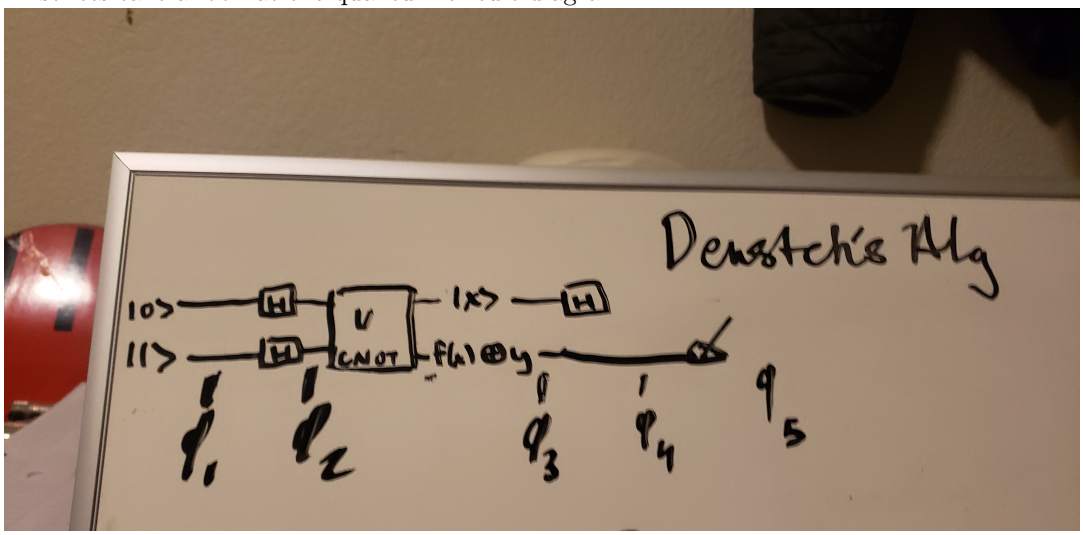
2.3.4 Deutsch's Algorithm

Introduction

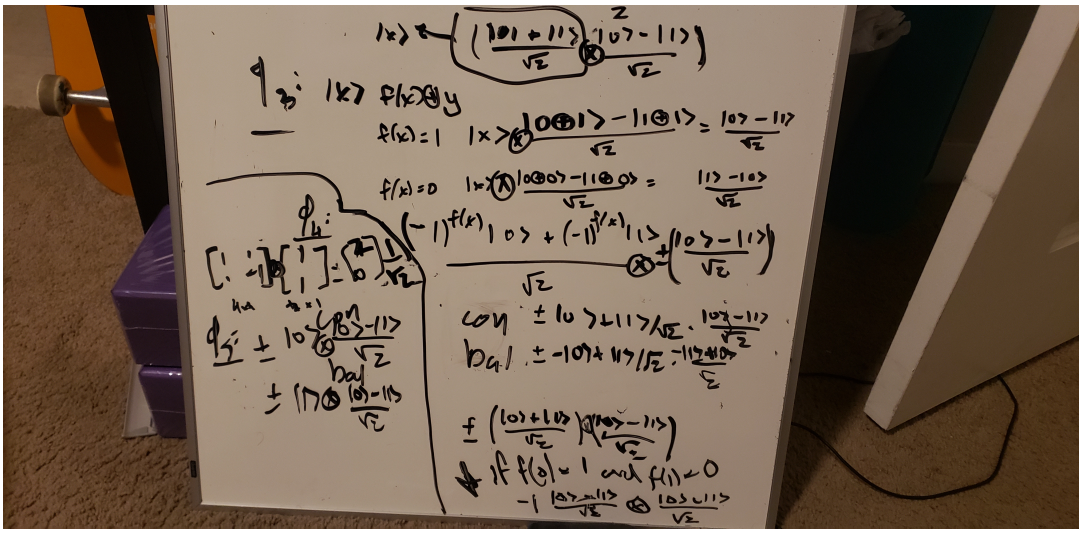
This algorithm though not that impressive at first glance showed the scientific community that quantum computing is the next step for computational efficiency. The problem to be solved is as follows: Given f(0)= 0 or 1 and f(1)= 0 or 1 find if f(0) equals f(1) or f(0) does not equal f(1).

Quantum Circuit

First lets take a look at the quantum circuit diagram:



Hadmard gates are the first step of our algorithm, rotating our tensors



At phi one we show entanglement. at phi 2 we show superposition and uncertainty phi 3 also shows uncertainty and entanglement phi 4 shows us wave function collapse and phi 5 lets us measure our system in order to determine whether it is balanced or constant

Efficiency

Classically this algorithm looks like this:

```
f(0) = 1 or 0
f(1) = 1 or 0

if f(0) = f(1) then constant
if f(0) ~= f(1) then balanced

if f(0) == 1:
    if f(1) == 1:
        constant
    else:
        balanced

else:
    if f(1) == 1:
        balanced
    else:
        constant
```

Two steps must be taken through the classical algorithm in order to determine first if f(0) is 1 or 0 and then to determine what f(1) is. As we saw When Understanding the Algorithm, at the end we have a state that once measured will let us see whether the equation is balanced or constant as we know what value will correspond to the constant or balanced state. Only one step versus two. This not only shows the computational efficiency but the sheer power of quantum principles in computing. Taking a problem that classically would have no room for improving efficiency and beautifully manipulating the problem into a quantum system.

Conclusion and code

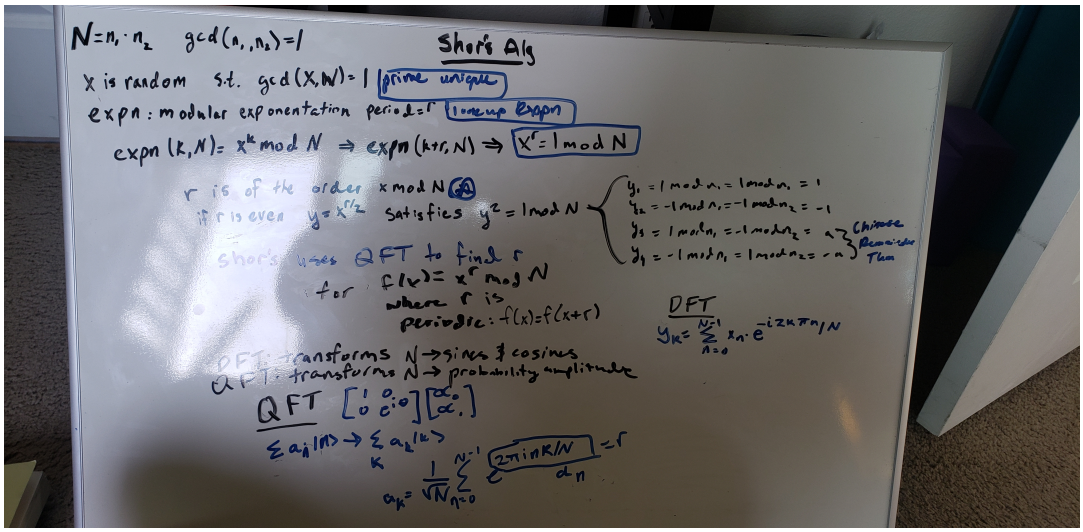
2.3.5 Shor's Algorithm

Introduction

At first I wasn't keen on understanding Shor's Algorithm. Truthfully I have never understood cryptography. I can definitely say after pushing myself to understand this algorithm, it is a perfect way to show someone the true power of quantum computing. Duestch's Algorithm let us a very simple and effective rationale for using quantum computers. Shor's Algorithm shakes our classically founded computational world with its power and efficiency. The problem seems simple but is very difficult for classical computers; find all prime factors of a very large number.

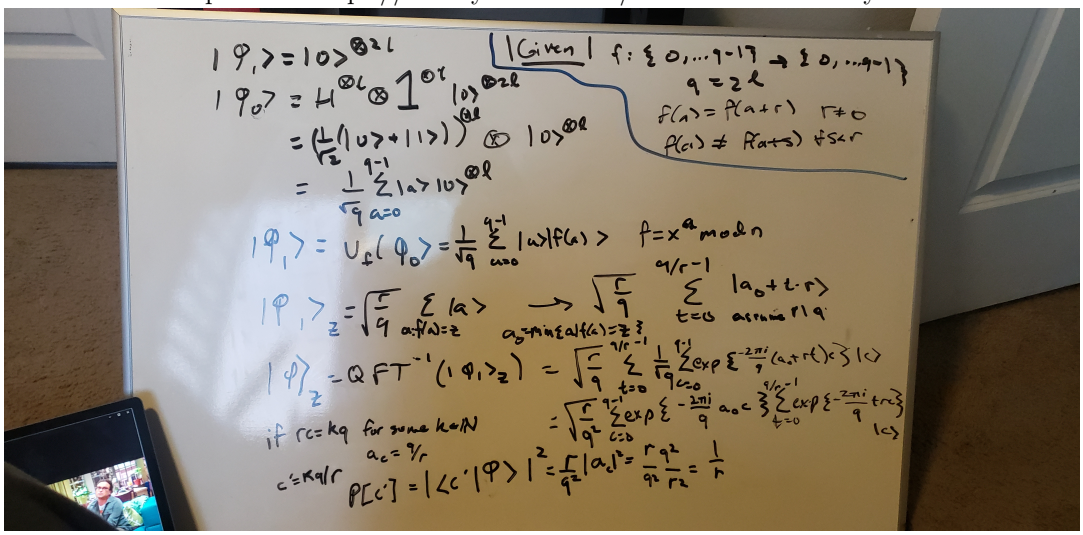
Quantum Circuit

Understanding the Algorithm

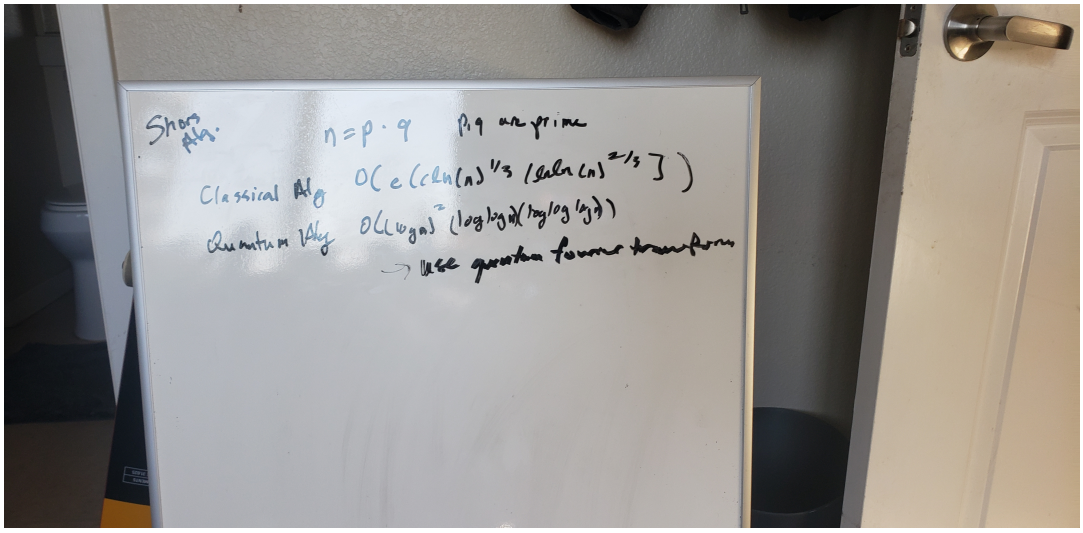


Take the number you want to be factored, use number theory to find the period

of very long sequence. Then utilize the quantum computer as computational interferometer, using light through diffraction grading, we get a pattern that gives us the space between the different gradings. The period is found through using the quantum interferometer. The final step is to use number theory again to solve for our primes. https://www.youtube.com/watch?v=hOlOY7NyMfs



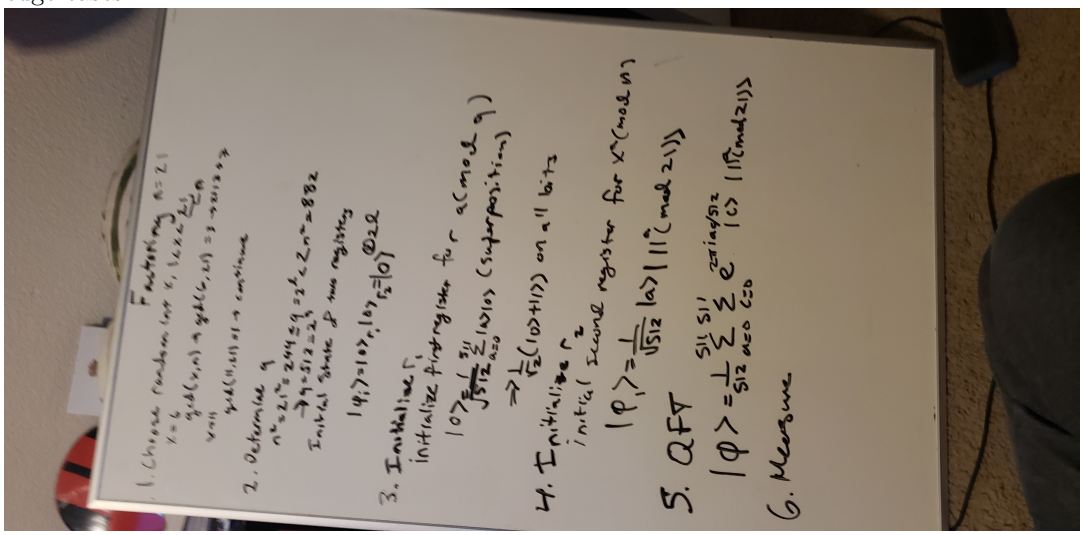
Efficiency



Conclusion

In conclusion we will examine the use of Shors when we have N =21. Lets observe how our algorithms handles factoring primes and observe how to handle

edge cases.



2.3.6 Grovers Algorithm

Introduction

Grovers algorithm lets us search through a database unlike any other search algorithm. Using the power of the probabilistic wave function. Entangling and superimposing all the data we are able to effectively determine where our desired data is within our structure, with much more efficiency than any classical search algorithm.

Quantum Circuit

Understanding the Algorithm

Efficiency

Conclusion and Code

]

Chapter 3

Durr and Hoyer Algorithm

3.1 Introduction

3.2 Algorithm

A brief introduction to the Durr and Hoyer Algorithm can be derived from the Quantum Minimum Searching Algorithm outlined in 'A quantum algorithm for finding the minimum' [(1)]:

Durr and Hoyer propose a computationally efficient routine to find the minimum of a table of distinct unsorted integers. The algorithm is as follow:

QUANTUM MINIMUM SEARCHING ALGORITHM

1. Choose threshold index $0 \leq y \leq N-1$ uniformly at random.
2. Repeat the following and interrupt it when the total running time is more than $22.5\sqrt{N} + 1.4\lg^2 N$.[1] Then go to stage 2(2c).
   (a) Initialize the memory as $\sum_j \frac{1}{\sqrt{N}}|j\rangle|y\rangle$. Mark every item j for which $T[j] < T[y]$.
   (b) Apply the quantum exponential searching algorithm of [2].
   (c) Observe the first register: let y' be the outcome. If $T[y'] < T[y]$, then set threshold index y to y'.
3. Return y.

Pseudocode

```
t = table.flatten()
N = len(t)
y = randomuniform[1..N] #N is a integer
time_limit = (22.5*sqrt(N)=1.4*log(N)^2)
q = preprocess.simulate(t,y)

while time.clock() < time_limit:
    for i in range(N):
        while t[i] < t[y]:
            j = N-j
            if N is even
                q=Algorithm with Hadamard gates[0...j]
                y'=q[0]
            else
                t=Algorithm with QFT instead of H[0...j]
                y'=q[0]
    if t[y'] < t[y]:
        y = y'
        return
return y
```

It is easy to understand the conceptual implementation in python. It is important to note steps: (a,b)

(a)

Part a is probably the most interesting part of this algorithm from my perspective. After a brief chat with Dr. Hoyer I realized this aspect of the algorithm is a field of research in itself. The creation of qubits from a table of unique integers is an interesting question. An even more interesting question is if we create a function to map values onto integers and must ponder how to create a state of qubits to capture this classical data.

For now, the more prevalent question is what sort of initialization we can use to verify the integrity and efficiency of this algorithm. My most immediate thought is entangle all the qubits and set them as unique basis. The classical

value of the integer is not represented here, only the index of the integer is captured at a quantum level on the table assuming an input datatype of BigEndian. If the goal is to test the algorithm's probabilistic integrity this plan of action will suffice.

The register can be initiated as follows:

$\sum_0^{i-1} \frac{1}{\sqrt{N}} |j\rangle |y\rangle$

$= \frac{1}{\sqrt{N}} |j_0\rangle |y\rangle + \frac{1}{\sqrt{N}} |j_1\rangle |y\rangle + ... + \frac{1}{\sqrt{N}} |j_{i-1}\rangle |y\rangle$

$= \frac{i}{\sqrt{N}} |y\rangle (|j_0\rangle + ... + |j_{i-1}\rangle)$

Define a flattened table t as 0....N, assume it is ordered for the example.
Assume N = i and take y = 1
Take i = 1

Set $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$\sum_0^{i-1} \frac{1}{\sqrt{N}} |j\rangle |y\rangle$

$= \frac{1}{\sqrt{1}} (|0\rangle \otimes |1\rangle + |1\rangle \otimes |1\rangle)$

$= |01\rangle + |11\rangle$

$= \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Take i = 3

Set $|0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} |2\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} |3\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

$\sum_0^{i-1} \frac{1}{\sqrt{N}} |j\rangle |y\rangle$

$= \frac{3}{\sqrt{3}}(|0\rangle \otimes |1\rangle + |1\rangle \otimes |1\rangle + |2\rangle \otimes |1\rangle + |3\rangle \otimes |1\rangle)$

$= \frac{3}{\sqrt{3}}(|01\rangle + |11\rangle + |21\rangle + |31\rangle)$

This can be generalized for any i.
In order to mark the indices we simply mark all indices that come before i.

(b)

Part b must be understood through another paper 'Tight bounds on quantum searching' [(2)]. It details the use of the quantum exponential searching algorithm in three cases. Those three cases being: finding one solution, multiple solutions, or an unknown number of solutions. For each case there is different prepration of qubits and algorithms used in unison to find the solution(s). The Durr Hoyer Algorithm is supposed to push the computational efficiency versus computational accuracy. The statement earlier mentioning the fifty percent success rate for finding the minimum is bounded by the Big-O can be explored further now. [3]

It is intuitive to see that in the case for finding a minimum of an unsorted table is finding one unique solution. The details are mentioned under "Implementation Considerations".

6 Implementation considerations

Grover's algorithm consists of a number of iterations followed by a measurement. In his original article [4] Grover shows that the unitary transform G, defined below, efficiently implements what we called an iteration in §2.

For every $A \subset \mathbb{Z}_N$, let S_A be the conditional phase shift transform given by

$$S_A|i\rangle = \begin{cases} -|i\rangle & \text{if } i \in A \\ |i\rangle & \text{otherwise.} \end{cases}$$

For every $i \in \mathbb{Z}_N$, denote $S_{\{i\}}$ by S_i. Let T be the Walsh-Hadamard transform

$$T|j\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{i \cdot j} |i\rangle,$$

where $i \cdot j$ denotes the bitwise dot product of the two strings i and j. Then the transform G is given by

$$G = -TS_0TS_{i_0}.$$

Grover considers only the case when N is a power of 2 since the transform T is well-defined only in this case. However, the assumption on N can be removed by observing that G is just one of many transforms that efficiently implements an iteration. Let T' be any unitary transform satisfying

$$T'|0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle. \tag{5}$$

Then one may easily verify that the transform $T'S_0T'^{-1}S_{i_0}$ works just as well, and, more interestingly, that

$$T'S_0T'^{-1}S_A$$

implements the general iteration analysed in §3. Any transform T' satisfying (5) can thus be used in the algorithm.

When N is a power of 2, the Walsh-Hadamard transform is indeed the simplest possible choice for T'. When N is not a power of two, the approximate Fourier transform given by Kitaev [5] can be used.

A thorough read of this will supply any clarification to the python code posted above. 'G' mentioned above is the 'Algorithm' in the python pseudo code. It will take in the random value, the table in quantum information, It is important to note that if the table has odd elements a different form of the algorithm must be used. Being new to quantum computing I have found myself wading through darkness at times. Reading this even now leaves me a bit unsure and skeptical of my own implementation, redoing my calculations. After research and conversations with quantum computing aficionados I have reached the following conclusions:

1. T is just a Hadamard Gate
2. S_A is a variation of the CNOT Gate
3. S_0 will will change the sign of the qubit if it is in null space, otherwise it will give the same result back

(1)
:https://arxiv.org/pdf/quant-ph/9607014.pdf
(2)
:https://arxiv.org/pdf/quant-ph/9605034.pdf

: Essentialy, the number of times we call upon the algorithm in step (b) the greater our Big-0 and probabilistic success rate. Durr Hoyer answers to a 'sweet spot' of sorts. The algoirthm will run for the alloted while loop in the python code above to achieve a 50 percent success rate. This proof is done in detail in [(1)]. An important feature of this open source library will be the user's manipulation of how many times step (b) is applied. In addition, future implementation can include manipulation of the algorithm in order to implement the different 3 cases mentioned for the quantum exponential searching algorithm. Though this means the algorithm is technically no longer Durr Hoyer it will be an exciting feature.

3.2.1 Implementation

3.2.2 Efficiency

3.3 Applications

Chapter 4

Conclusion and Future Work

Chapter 5

Acknowledgments