Quantum Algorithms

Analysis & Design of Algorithms

Computer and Information Sciences

05/12/2018

Alexander O'Hare & Caleb Sarikey

Supervised by Dr. Gentian Buzi

Summary

This project focuses on quantum algorithms and how they improve on standard algorithms. We begin by providing prerequisite and background information on quantum computers, qubits, superposition, entanglement, and amplitude amplification. These topics are essential in understanding how quantum algorithms work and why they are so effective. Specifically, this project analyzes Grover's quantum algorithm for searching an unsorted database in comparison to a standard linear search algorithm. These two algorithms were implemented and tested using C++ and Quantum Assembly Language. We conclude with our analysis and experimental results.

Prerequisite Knowledge

I. Quantum Computers

Unlike regular computers that use bits, quantum computers are based on quantum bits called qubits. A quantum computer operates on its qubits using quantum gates and measurements. Quantum computers use quantum-mechanical phenomena, such as superposition and entanglement, to process information. Because of this, quantum algorithms are probabilistic, not non-deterministic. This means that quantum programs output the probability of a something occuring, not if it occurs. To create human-readable output, quantum computers must be hybridized with a normal computer. When the qubit is measured, the result is output to a standard bit.

For this project, we made use of a 5-qubit computer through IBM's Quantum Experience to run Grover's quantum algorithm. Additionally through IBM's Quantum Experience, we ran simulations of a quantum computer utilizing standard computing technology. We did this due to limited access to the quantum computer, but also to see the differences in the results of the two.

II. Qubits

Qubits have two possible values (0, 1), but the laws of quantum mechanics allow for superposition states, which are values that exist anywhere between the extremes of 0 and 1. Because of this, a quantum computer with *n* qubits can be in an arbitrary superposition of up to 2^n different states simultaneously, whereas a normal computer can only be in one of these 2^n states at any time. However, only binary information can be extracted from a qubit. In other words, a qubit will always output either 0 or 1 when measured. The laws of physics prevent us from seeing the exact details of its superposition state. Since quantum algorithms are probabilistic, it is common practice to run a quantum program many times. This process then returns a list of all the outputs that were generated during these many samples and the number of times each output occurred.

III. Superposition

Superposition is the ability of a quantum system to be in multiple states at the same time. Much like waves in classical physics, quantum states can be added together (“superposed") and the result will be another valid quantum state. Conversely, every quantum state can be represented as a sum of two or more other distinct states. As previously stated, quantum computers have the advantage of being in up to 2^n superposition states. This means that a quantum computer with only 20 qubits has the ability to process 2^{20} (1,048,576) calculations simultaneously. Thus, certain difficult tasks that have long been thought impossible for classical computers may be achieved quickly and efficiently by a quantum computer.

IV. Entanglement

In brief, entanglement can be defined as an extremely strong correlation that exists between quantum particles. Two or more entangled particles are linked and remain in perfect unison, even if they are placed at opposite ends of the universe. The information in a quantum system exists in the correlations between its particles, and so a quantum state must be described for the system as a whole, not individually.

V. Amplitude Amplification

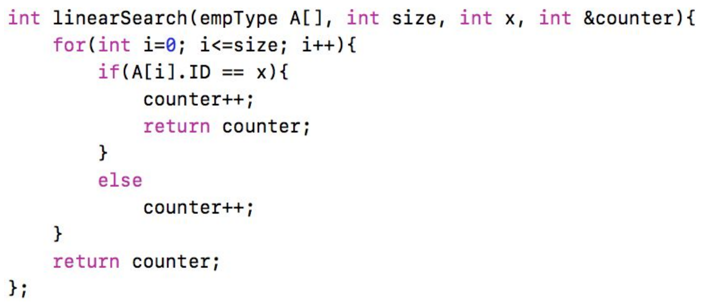
Amplitude amplification is a process that allows us to find a good solution after an expected number of applications, assuming that the algorithm makes no measurements. With each iteration, the correct solution is amplified. Amplitude amplification is used to obtain a quadratic speedup over standard algorithms, and is an essential part of Grover's quantum algorithm in doing so.

Quantum-mechanical phenomena such as superposition, entanglement, and amplitude amplification give quantum computers the ability to process a vast number of calculations simultaneously and yield a probabilistic solution.

<u>Implementation and Analysis</u>

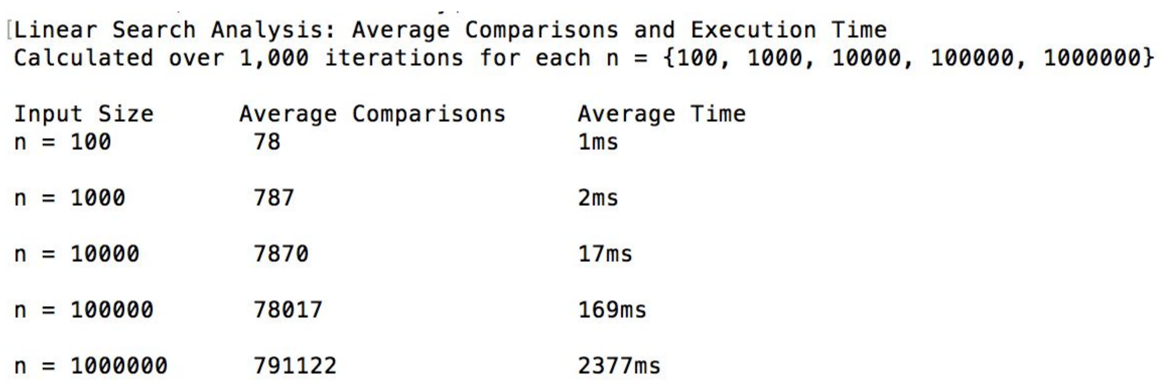
I. Linear Search

We implemented a standard linear search algorithm in C++, as seen in *Figure 1*. Our search function steps through the array and compares each element one-by-one to the randomly generated value being searched for, then returns a count of the total number of comparisons. The test program generates random numbers between 0 - 2*n* for arrays of size *n* = {100, 1,000, 10,000, 100,000, 1,000,000}. For each array size, the average number of comparisons and average execution time is calculated over 1,000 iterations. Results from two test runs are shown in *Figure 2.1* and *Figure 2.2* below.



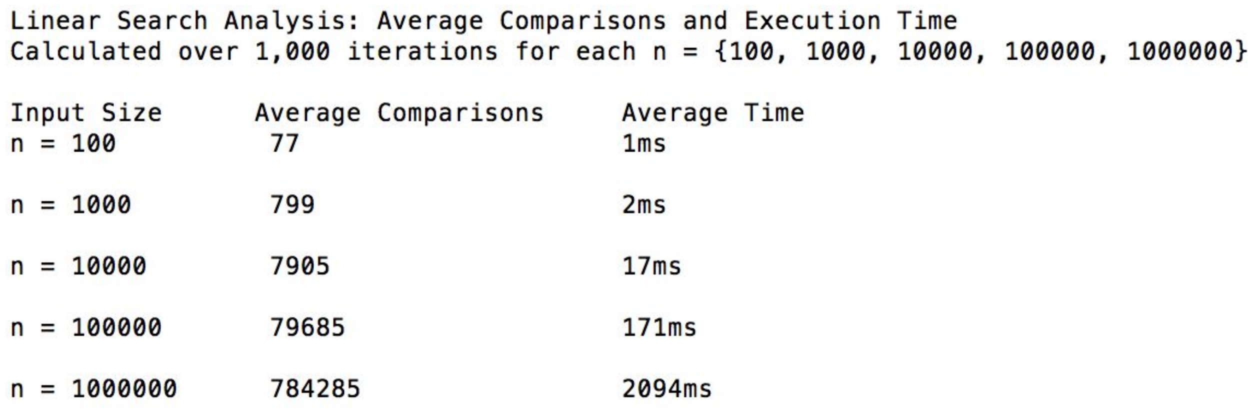
```cpp
int linearSearch(empType A[], int size, int x, int &counter){
    for(int i=0; i<=size; i++){
        if(A[i].ID == x){
            counter++;
            return counter;
        }
        else
            counter++;
    }
    return counter;
};
```

Fig 1. *Linear Search implementation in C++*



```
Linear Search Analysis: Average Comparisons and Execution Time
Calculated over 1,000 iterations for each n = {100, 1000, 10000, 100000, 1000000}

Input Size      Average Comparisons     Average Time
n = 100          78                     1ms

n = 1000         787                    2ms

n = 10000        7870                   17ms

n = 100000       78017                  169ms

n = 1000000      791122                 2377ms
```

Fig 2.1. *Linear Search results for test run 1*



```
Linear Search Analysis: Average Comparisons and Execution Time
Calculated over 1,000 iterations for each n = {100, 1000, 10000, 100000, 1000000}

Input Size      Average Comparisons     Average Time
n = 100          77                     1ms

n = 1000         799                    2ms

n = 10000        7905                   17ms

n = 100000       79685                  171ms

n = 1000000      784285                 2094ms
```

Fig 2.2. *Linear Search results for test run 2*

As expected, our linear search algorithm results indicate an efficiency class of O(*n*), where *n* is the number of elements in the input set.

II. Quantum Assembly Language (QASM)

Of the few quantum programming languages available, we chose to implement Grover's algorithm in Quantum Assembly Language. The syntax of QASM has elements of C and Assembly Language. Registers and gates are an essential part of the code. All quantum programs require a quantum register as well as a binary register to display human-readable information. Specifically, Grover's quantum algorithm requires the use of h gates, s gates, cx gates, tdg gates, t gates, and x gates, which all perform different linear transformations on qubits.

There are also operations that allow entanglement of two qubits. Linear transformations done on entangled bits also perform the operation on the corresponding entangled bit. Each qubit register is measured when no more operations on that specific bit need to be done. Once measured, the qubit can no longer be operated upon. The results of the measurement are copied to a standard binary bit which may be output to the user**.**

III. Grover's Algorithm

Grover's quantum algorithm is an improvement on searching unsorted sets. Grover's algorithm uses superposition, entanglement, and amplitude amplification to find a state with probability of success > 50%. Explanation and analysis of the algorithm follows:

Input: $n + 1$ qubits

Algorithm:

1. Start with the qubits in a superposition of all possible states, all with the same phase.
2. Apply the oracle function only once.
    a. *Note*: *Because the system begins in a superposition of all possible states, the oracle function is effectively applied to all states, but it will only flip the phase of the state for which we are searching. If a measurement were to be made at this point, all possible outcomes would still be equally likely. The following steps use quantum mechanics to find the correct answer.*
3. Apply the Hadamard gate to n qubits.
    a. Note: Applying a Hadamard gate creates equally weighted superposition of all 2^n states. The importance of this is that a superposition containing exponentially many terms can be prepared using only a polynomial number of operations.

$$\boxed{H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}$$

$$\hat{H}|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

$$\hat{H}|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Fig 3. *A Hadamard gate as applied to one qubit*

4. Apply the Diffusion operation.
    a. *Note*: *This gate is a vital piece of Grover's algorithm that performs a flip of qubits with regard to the average value. This is where amplitude amplification plays a role. First, the amplitude of the desired state is flipped, then the amplitudes are inverted about their mean. This results in the amplitude of the desired state being slightly larger than all of the other amplitudes. Iterating this process results in the desired state having a significantly larger amplitude.*

$$D = \left\{\begin{matrix} -1+\frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & -1+\frac{2}{N} & \cdots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & -1+\frac{2}{N} \end{matrix}\right\}$$

Fig 4. *The Diffusion operator*

5. Apply the Hadamard gate to *n* qubits again.

It can be shown that if steps 3-5 are performed for O(√n) iterations, a measurement will result in the correct answer with a strong probability. As a result, the total runtime of a single Grover iteration is Θ(2n), from applying the two Hadamard gates, plus the O(n) cost of applying gates to perform phase shifts. Thus, the runtime of Grover's entire algorithm, performing O(√n) = O($\sqrt{2^n}$) = O($2^{n/2}$) iterations each with a runtime of O(n) is O($2^{n/2}$). Unlike other quantum algorithms which may provide exponential speedup over their standard counterparts, Grover's algorithm provides only a quadratic speedup. However, even quadratic speedup is considerable when N is very large. For example, Grover's algorithm could brute-force a 128-bit symmetric cryptographic key in roughly 264 iterations, or a 256-bit key in roughly 2128 iterations, raising security concerns for future quantum attacks.

We implemented Grover's algorithm using IBM's quantum computer with 3 qubits with the goal to find the state 011. We then tested the algorithm on an IBM's actual quantum computer, as well as the simulator. Due to the nature of quantum algorithms and the limitations of QASM there was no mechanism to measure exact execution time or number of operations. However, we were able to obtain results from many test runs to further analyze.

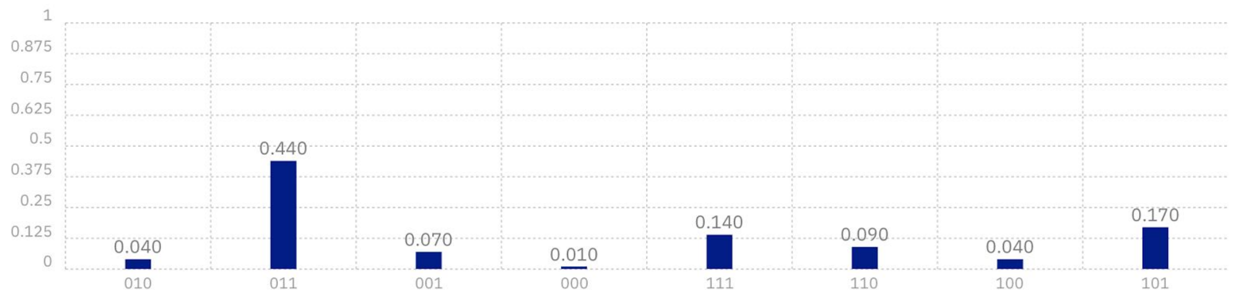


Fig 5.1 *Simulation results of Grover's algorithm with 011 having 44% probability*

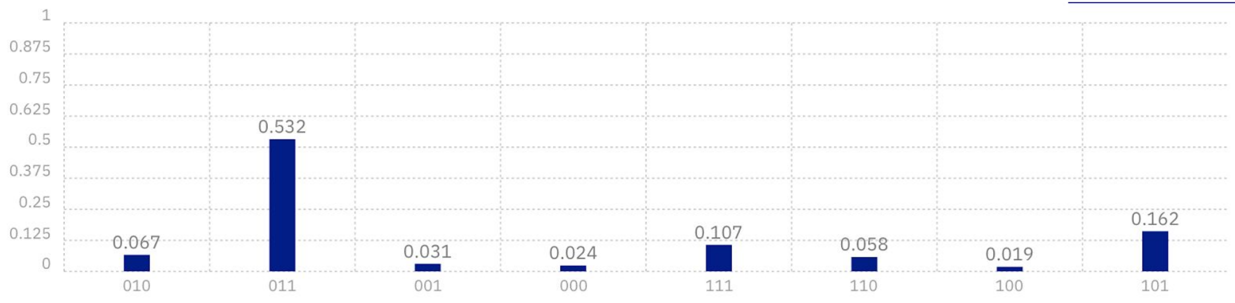


Fig 5.2 *Simulation results of Grover's algorithm with 011 having 53% probability*

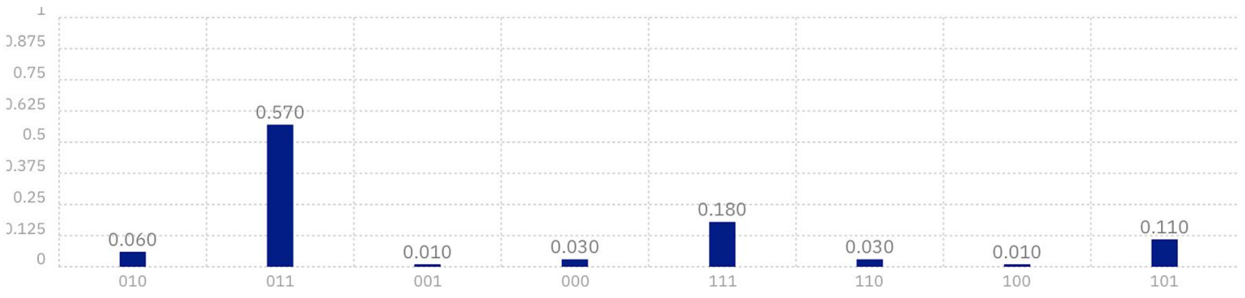


Fig 5.3 *Simulation results of Grover's algorithm with 011 having 57% probability*

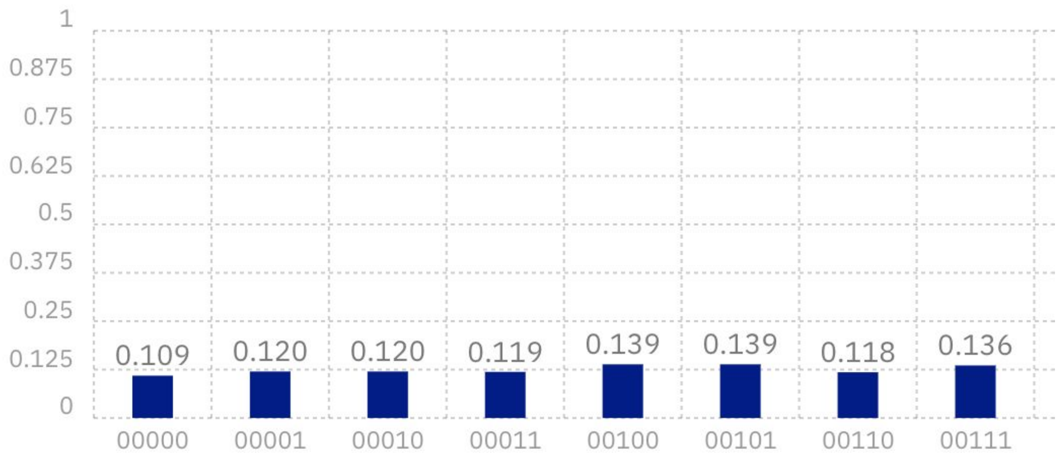


Fig 6. *Erroneous results of Grover's algorithm when tested on an actual quantum computer*

<u>Results</u>

We received different results on the simulator than we did on the actual Quantum Computer. On the simulator, we were able to identify the existence of a specific value with a probability as little as 44% and as high as 57%. However, on the Quantum Computer, the probability of values were almost all equal to one another. We are not exactly sure of what these results indicate, or why the Quantum Computer's results deviated so far from the simulator's. We have concluded that Quantum Computers are not yet prepared to solve real world problems. If they were, solutions to problems such as Prime Factorization, Pell's equation, and breaking current cryptosystems would be available to us today.

References

Brassard, G., Høyer, P., Mosca, M., & Tapp, A. (2002). Quantum amplitude amplification and estimation. *Contemporary Mathematics Quantum Computation and Information,*53-74.

Cross, A. W., Bishop, L. S., Smolin, J. A., & Gambetta, J. M. (2017). Open Quantum Assembly Language. *Quantum Physics*.

Grover's algorithm. (2018, April 30). Retrieved from https://en.wikipedia.org/wiki/Grover's_algorithm

IBM. (n.d.). IBM Quantum Experience. Retrieved from https://quantumexperience.ng.bluemix.net/qx/editor

Kemp, D. (2017, December). An Interactive Introduction To Quantum Computing. Retrieved from http://davidbkemp.github.io/QuantumComputingArticle/

Strubell, E. (2011). An Introduction to Quantum Algorithms.