JOHNS HOPKINS
UNIVERSITY

Engineering for Professionals

# Quantum Grep Implemented via Grover Search

May 9, 2022

615.781 Quantum Information Processing - Final Project

Sandun Panthangi
Ian DeTore
Rick Hewitt
Joshua Holcomb

Supervised by:
Dr. David Clader, Johns Hopkins University

## Contents

## Abstract

We outline here how the quantum implementation of the Grep function using Grover Search along with what our findings were.

## 1 Introduction

One of many capabilities that proves the power of a Quantum Computer is the ability to search through unstructured search space. This was demonstrated by Lov Grover using his Grover Search algorithm.[1]. Our aim here is to showcase the use of Grover Search algorithm to implement a Quantum version of GREP. We will discuss what GREP function is, how its implemented classically, then moving on to comparing the quantum version of it along with performance observed.

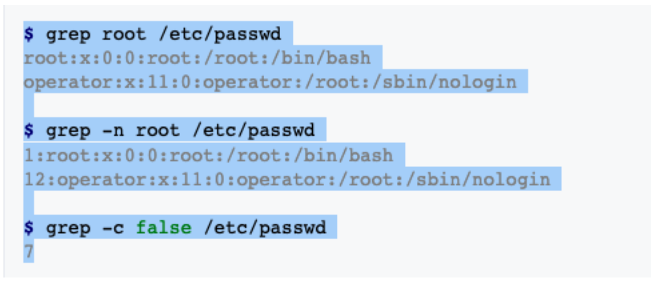
---

*No one really understands Quantum Mechanics – Richard Feynman. We are here to implement something great with what little we know*

---

## 2 What is GREP?

GREP is a command-line utility for text searching using regular expressions. Originally developed for the Unix operating system, GREP would allow a user to search for a line of text containing a regular expression. The name comes from the commands used for the Unix ed text editor. In ed, the command 'g', standing for 'global', followed by 're' for a 'regular expression' and 'p' for 'print', were combined in the editor as g/re/p. The command sequence globally searches for a regular expression and prints the matching lines. This function now is available for any Unix-Like operating systems.[2]



```
$ grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin

$ grep -n root /etc/passwd
1:root:x:0:0:root:/root:/bin/bash
12:operator:x:11:0:operator:/root:/sbin/nologin

$ grep -c false /etc/passwd
7
```

**Figure 1:** Output of GREP Command for different input arguments

A regular expression is a sequence of characters that specifies a search pattern in text. For example, in ed there is a regular expression command '\$' which matches the end of a line. Say we query abc\$, potential matches would include 123abc and theabc.

### 2.1 Classical Implementation of GREP

GREP operates partly via a set of rules that are designed for efficiency, and partly via a slower match function (a.k.a. matcher) that takes over when the fast matchers run into unusual features like back-references. When feasible, the Boyer–Moore fast string searching algorithm is used to match a single fixed pattern, and the Aho–Corasick algorithm is used to match multiple fixed patterns.[3]

The Boyer–Moore algorithm has worst-case running time of $O(n+m)$. This is for when the pattern does not appear in the text being searched.[4] In the event the pattern does appear in the text, we would see the running time of the algorithm to be $O(nm)$ in the worst case.

## 3 Grover Search

Grover's algorithm is a quantum search algorithm that enables unstructured data search with quadratic speedup on a quantum computer over classical implementations. This algorithm runs in $O(\sqrt{N})$ operations whereas in a classical computer it would take $O(N)$ [5]. Moreover, the Grover search algorithm is considered a *"general algorithm"* in that it can be applied far beyond search examples. This is known as the amplitude amplification trick.[6]

Grover's Search Algorithm is implemented in the following manner.

1. Introducing the Oracle - which simply is a function that takes in an argument then checks the target to see if the argument is present at target .
2. State Preparation - Apply the Hadamard transform $H^{\otimes n}$
3. Grover Iteration - Apply the Grover Operation
4. Measurement - Measure the state of the first $n$ qubits.

### 3.1 Introducing the Oracle

The oracle is a unitary operator $O$ defined as

$$O|x\rangle|q\rangle = |x\rangle|q \oplus f(x)\rangle$$

where $|x\rangle$ is the index register and $f(x) = 0$ for all $0 \le x \le 2^n$ except $x_0$. When $x_0$ then $f(x_0) = 1$ [1]

Here $O$ is defined by its actions on the computational basis. We will use the oracle to recognize the solution to our search problem.

### 3.2 State Preparation

State Preparation ensures the machine is in initial state as :

$$|0\rangle^{\otimes} |0\rangle$$

and subsequently apply a Hadamard Gate $H^{\otimes n}$

This can be defined as:

$$|0\rangle^{\otimes} |0\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=1}^{2^n-1} |x\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$$

Now the machine is in an uniform superposition state with all values of $x(modN)$ in the first factor, and in the state $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ in the second factor.

### 3.3 Grover Iteration

Grover Iteration is a subroutine here denoted as $G$, which when iterated gives the desired result for the search operation. The operation is as following: [1]

1. Call the Oracle
2. Apply the Hadamard transform $H^{\otimes n}$ to each qubit in the register.
3. Perform conditional phase shift, where $|x\rangle \rightarrow -(-1)^{\delta x_0} |x\rangle$ will give every computational basis state a phase shift of -1, except for $|0\rangle$.
4. Apply the Hadamard transform $H^{\otimes n}$ to each qubit in the register

Using only a single oracle call, we can show the effect of the steps 2-4 as following:

$$H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} = 2|\psi\rangle\langle\psi| - I$$

where $|\psi\rangle \rightarrow$ equally weighted superposition states of $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$

Therefore we can show $G$, our Grover iteration as:

$$G = (2|\psi\rangle\langle\psi| - I)O$$

where $2|\psi\rangle\langle\psi| - I$ is the inversion about mean operation.

Grover iteration can be regarded as a rotation in the two dimensional space spanned by the starting vector $|\psi\rangle$ and the state consisting of a uniform superposition of solutions to the search problem.[1]

If we define two normalized states as:

$$|\alpha\rangle \equiv \frac{1}{\sqrt{N-M}} \sum_x'' |x\rangle$$

$$|\beta\rangle \equiv \frac{1}{\sqrt{M}} \sum_x' |x\rangle$$

where $\sum_x^{''|x\rangle}$ sums all valid solutions and $\sum_x^{'|x\rangle}$ sums all else, we can rewrite $|\psi\rangle$ as:

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}\,|\alpha\rangle + \sqrt{\frac{M}{N}}\,|\beta\rangle$$

Now if we consider

$$G = (2\,|\psi\rangle\,\langle\psi| - I)O$$

the Grover iteration, the effect of its transformation can be described the following way:

1. Oracle $O$ performs a reflection about the $|\alpha\rangle$ vector in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$
2. $(2\,|\psi\rangle\,\langle\psi| - I)$ performs a reflection about the $|\psi\rangle$ which is also in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$
3. If we take the product of the above 2 reflections, we will get a single complete rotation.[1]

Visually we can show it as following:[1]

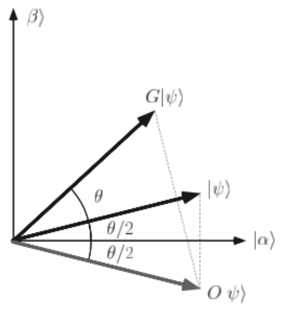


**Figure 2:** Single Grover Iteration

Now we can summarize the Grover Iteration to be:

$$\left[(2\,|\psi\rangle\,\langle\psi| - I)O\right]^R \frac{1}{\sqrt{2^n}} \sum_{x=1}^{2^n-1} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

When we apply the Grover Iteration repeatedly the state vector will rotate closer to $|\beta\rangle$ in the above diagram. This will allow us an observation, with high probability, in the computational basis, giving us the answer to the search problem.

## 4 QGREP Implementation

### 4.1 Overview

Our *QGREP* Code can be summarized in several steps:

1. Convert files into arrays of strings and store metadata on line numbers and files.
2. Cast strings into arrays of integers using ASCII.

3. Apply Grover's search across array to find candidate solutions for matching characters starting with first character of target string.

4. Repeat steps 1-3 until entire target string is searched

5. Convert back into human-readable formats and print information to user.

Our code is openly accessible on GitHub [7].

## 4.2 Unicode generation

In order to provide a simple mapping to generate an oracle with for Grover's search, we require a mapping of characters to quantum states. We choose to use Unicode to convert characters in files to a matrix of integers, where the binary encoding can be treated as the sequence of quantum states of qubits. Simulation of the quantum circuit then allows for assigning probabilities of character-matching to each character in the file, where the most probable characters are generally matches.

## 4.3 Adjustable Grover Circuit

As mentioned in *Section 3*, the circuit for Grover's Search is comprised of each qubits in superposition via Hadamard gates, along with a unique oracle for the desired state, and by applying conditional phase shifting with the inversion about the mean.

By implementing Qiskit and utilizing simple logic in Python, the Python function can automatically determine how many qubits are needed for the given string query. To give more flexibility to the search, the user is able to determine how many iterations are required. An iteration is defined as the number of cycles of the combined oracle and mean inversion module are repeated, including the first one. As shown in Figure 5, the number of iterations is periodic and depends on the number of qubits. Since the code follows the ASCII structure, which contains 7-8 bits, queries related to string-searches reach their highest probabilities between 12-13 iterations.

# 5 Results

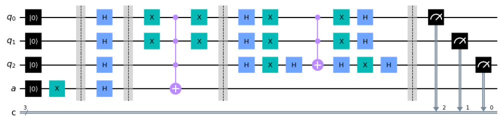


**Figure 3:** 3-Qubit Grover Search; State 1 Oracle

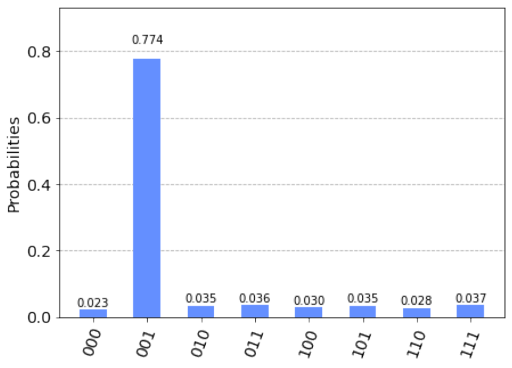


**Figure 4:** Oracle 1 (001) Simulation Results

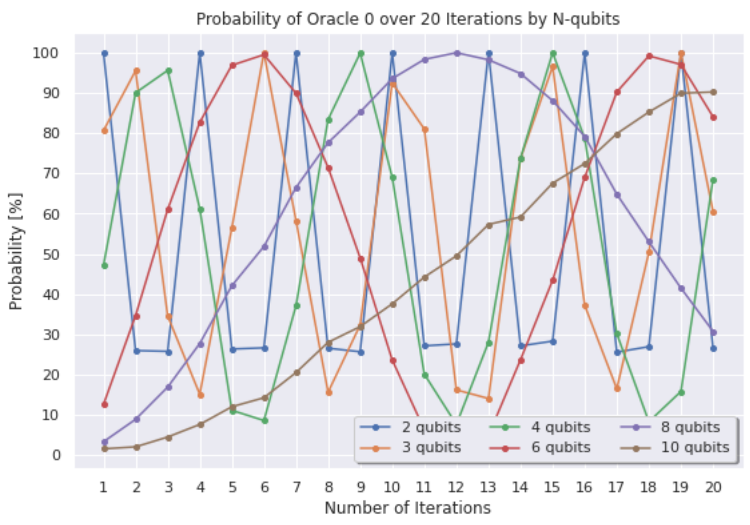
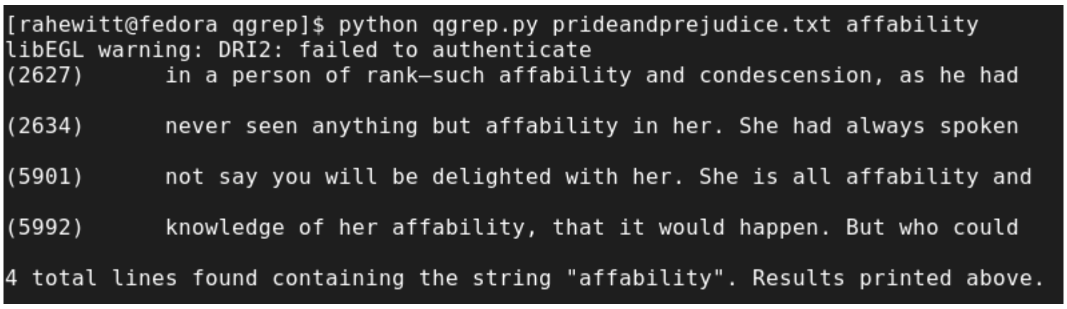


**Figure 5:** Data for Oracle 0 accuracy with different numbers of qubits.



```
[rahewitt@fedora qgrep]$ python qgrep.py prideandprejudice.txt affability
libEGL warning: DRI2: failed to authenticate
(2627)      in a person of rank—such affability and condescension, as he had

(2634)      never seen anything but affability in her. She had always spoken

(5901)      not say you will be delighted with her. She is all affability and

(5992)      knowledge of her affability, that it would happen. But who could

4 total lines found containing the string "affability". Results printed above.
```

**Figure 6:** Example of a string search in the text of Pride and Prejudice using qgrep.

## 6 Discussion

Overall, we've found that *grep* can be implemented using quantum circuits in place of classical operations. Our implemented code demonstrates the capability to read full-text excerpts and accurately extract a user-inputted string. For this particular use case we would anticipate that "qgrep" via Grover Search algorithm fully implemented on a quantum machine would allow a faster search result than its classical counter part given the superior complexity $O(\sqrt{N})$ of the quantum algorithm.

In practical use cases we can apply qgrep to search library data, research databases, or any other applications for which pattern-matching strings over large pieces of text is required.

## References

[1] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. English. Cambridge: Cambridge University Press, 2000. ISBN: 0-521-63503-9; 0-521-63235-8.

[2] Brian W. Kernighan. *The Unix Programming Environment*. English. Prentice-Hall, 1984. ISBN: 978-0139376818.

[3] GNU. *grep*. URL: http://www.gnu.org/software/grep/manual/grep.pdf. (accessed: 04.04.2022).

[4] Zvi Galil. "On Improving the Worst Case Running Time of the Boyer-Moore String Matching Algorithm". In: *Communications of the ACM* 22-9 (1979), pp. 505–508. DOI: https://doi.org/10.1145/359146.359148.

[5] Kapil Kumar and Akhtar Rasool. "Pattern Matching: A Quantum Oriented Approach". In: *Procedia Computer Science* 167 (2020), pp. 1991–2002. DOI: https://doi.org/10.1016/j.procs.2020.03.230.

[6] Qiskit. *Grover's Algorithm*. URL: https://qiskit.org/textbook/ch-algorithms/grover.html. (accessed: 04.11.2022).

[7] Ian DeTore and Rick Hewitt. *QGrep GitHub Repo*. URL: https://github.com/idetoreStarry/qgrep.