# Quantum computation and Grover's algorithm

Shahed Sharif

June 5, 2019

## 1 Introduction

The goal of this note is to give a sketch of quantum computing, and explain Grover's algorithm. Grover's algorithm works as follows. Say we have a set of $N$ objects, and $k$ of them satisfy a black box conditional $f$ (that is, $f(x) = 1$). We would like to find $x$ which satisfies $f$. Classically the best we can do is to try elements until we find one that works. This takes $O(N/k)$ trials. Grover's algorithm instead requires only $O(\sqrt{N/k})$ trials.

While still an exponential time algorithm, in practice Grover's vastly speeds up searches. For example, to reverse a 256-bit hash classically requires on average $2^{255}$ trials. With Grover's, it only requires $2^{128}$ trials. Thus the bit strength of hash functions is halved under Grover's algorithm. An analogous phenomenon holds for symmetric ciphers.

Observe that Grover's algorithm applies to any problem in NP—just let $f$ be the polynomial time certificate checker.

Lastly, Grover's algorithm also applies if the value of $k$ is unknown, but we will not discuss that implementation here.

## 2 Basics of computation

### 2.1 Classical computation

In classical computation, we have:

- bits,
- programs (or algorithms), and
- basic gates.

A *bit* is either one of the values $0, 1$, or a variable which holds such a value. An element of $\{0, 1\}^n$ is called a *bit string*.

A program is a function $f : \{0,1\}^* \to \{0,1\}^*$, where $\{0,1\}^* = \cup\{0,1\}^n$. We think of the function as a family $(f_n)$ where $f_n : \{0,1\}^n \to \{0,1\}^*$. Each individual $f_n$ is a *Boolean function* or *circuit*. In practice, when we talk about a program $f$, we usually set $f = f_n$ for some $n$; or equivalently, we "forget" the subscript.

We also specify a set of *universal basic gates*; these are functions $g_1, \ldots, g_m$ for which every $f_n$ can be written as a composition of direct sums of the $g_i$. We implicitly include the identity function as a basic gate. A standard set is AND, OR, and NOT. The industry standard consists of just a single (nonidentity) gate: NAND. Suppose a set of universal basic gates has been fixed; the specific choice won't matter for us.

For a function $f = (f_n)$, write each $f_n$ as a composition of direct sums of the basic gates. Let $c(n)$ be the number of nonidentity basic gates used to write $f_n$. We say that $f$ is *polynomial time* if there exists $c(n)$ as above and $k \in \mathbb{N}$ such that $c(n) \in O(n^k)$.

## 2.2 Quantum computation

In quantum computation, we have analogous constructions. Let $\mathbb{H}$ be the inner product space $\mathbb{C}^2$. Then a *qubit* is a norm 1 element $\psi \in \mathbb{H}$. An $n$-qubit or *state vector* is a norm 1 element $\psi \in \mathbb{H}^{\otimes n}$. If we let $N = 2^n$, then $\mathbb{H}^{\otimes n} \cong \mathbb{H}^N$, and in fact they are isomorphic as inner product spaces. We make this isomorphism explicit. To $\mathbb{H}$ we associate a distinguished orthonormal basis, written $|0\rangle$, $|1\rangle$. Then an orthonormal basis for $\mathbb{H}^N$ is furnished by the set of elements of the form

$$|x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$$

where $x_i \in \{0,1\}$ for all $i$. We write this element compactly as

$$|x_1 x_2 \ldots x_n\rangle .$$

Sometimes, we think of $x_1 x_2 \ldots x_n$ as a bit string. In particular, a bit (respectively bit string) can be viewed as a special case of a qubit (respectively $n$-qubit state); namely, as one of the standard basis vectors.

A common misconception about quantum computing is that since with (say) 3 qubits, our system is $2^3 = 8$-dimensional, the greater power comes from the additional dimensions. This is not true: Holevo's Theorem states that $n$ qubits can carry at most $n$ bits of information. One way to understand this is that we *cannot* actually access the value of a qubit. Instead, we have a *measurement* operation. What this does is take as input a state vector

$$\psi = \sum \alpha_i |x_i\rangle ,$$

sample $x_i$ with probability $|\alpha_i|^2$, replace $\psi$ with $|x_i\rangle$, and output the value of $x_i$. We say that $\psi$ *collapses* to $|x_i\rangle$ as a result of the measurement. For example, if

$$\psi = \frac{1}{\sqrt{2}} |01\rangle + \frac{1}{2} |10\rangle + \frac{1}{2} |11\rangle ,$$

once we measure, there is a $\frac{1}{4}$ chance that the measurement will yield 10; if it does, then $\psi$ has now become $|10\rangle$.

Because measurement is probabilistic, most quantum algorithms are probabilistic as well. We require that the error in such a program be less than $1/2$.

Programs are families of unitary operators $(U_n)$, where

$$U_n : \mathbb{H}^{\otimes n} \to \mathbb{H}^{\otimes n}.$$

Observe that unitary operators preserve length, so that the image of an $n$-qubit is an $n$-qubit.

We would like to define a set of universal basic gates; that is, a set of finitely many unitary operators so that any unitary can be written as products of tensor products of these operators. But this is impossible, since the set of unitaries is uncountable. There are two essentially equivalent ways around this: allow our products of basic gates to *approximate* a given unitary, or allow an uncountable set of basic gates. We will take the latter tack.

By a basic gate, we will mean a unitary operator that acts on at most 3 qubits. For example, an operator of the form $A \otimes I$ where $A$ is an $8 \times 8$ unitary matrix would be considered basic: it only acts on the first 3 qubits. But an operator which acts on, say, the 3rd, 7th, and 15th qubit of an $n$-qubit system would also qualify—we just can't write it as a pure tensor.

Given a program $(U_n)$ write it as a product of, say, $c(n)$ basic gates. If $c(n)$ can be chosen so that $c(n) = O(n^k)$ for some $k$, we say the algorithm is quantum polynomial time. (Formally, the algorithm is in *BQP*, for *bounded error quantum polynomial time*.)

# 3 P⊂BQP

We would like to show that any classical polynomial time algorithm can be implemented as a polynomial time quantum algorithm. However, we have to be careful: unitary operators are invertible, while classical algorithms, such as AND, are not! The standard way around this is as follows. Let $f : \{0,1\}^n \to \{0,1\}^m$ be a Boolean function. Let $N = 2^{n+m}$. We define

$$\begin{aligned} U_f : \mathbb{H}^N &\to \mathbb{H}^N \\ |xy\rangle &\mapsto |x, y \oplus f(x)\rangle . \end{aligned}$$

Here, $x \in \{0,1\}^n$, $y \in \{0,1\}^m$, and $\oplus$ is component-wise addition. Since $U_f$ just permutes basis elements, it is unitary. Furthermore, $\forall x \in \{0,1\}^n$,

$$U_f |x0\rangle = |xf(x)\rangle .$$

Thus, to say that P⊂BQP means the following:

**Theorem 3.1.** *If $f = (f_n)$ is in P, then $U = (U_{f_n})$ is in BQP.*

To prove this, one simply notes that $U_{\text{AND}}$, $U_{\text{OR}}$, and $U_{\text{NOT}}$ are themselves basic gates.

# 4 Grover's

## 4.1 Set-up

Let $V$ be a complex Hilbert space and $w \in V$. Let $P_w : V \to V$ be projection onto $w$. We define *reflection across* $w$ to be the operator

$$R_w = 2P_w - I.$$

For $N = 2^n$, let $j = j_n = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle$.

**Proposition 4.1.** $R_j$ *is in BQP.*

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Define the *Grover oracle* $G_f : \mathbb{H}^N \to \mathbb{H}^N$ by

$$G_f |x\rangle = \begin{cases} |x\rangle & f(x) = 0 \\ -|x\rangle & f(x) = 1. \end{cases}$$

Or more compactly, $G_f |x\rangle = (-1)^{f(x)} |x\rangle$.

**Proposition 4.2.** *If* $f$ *is in P, then* $G_f$ *is in BQP.*

Recall that to interpret "polynomial time," we think of $f$ has a family of functions $(f_n)$ and $G_f$ really means $(G_{f_n})$.

## 4.2 The algorithm

Let $f : \{0,1\}^n \to \{0,1\}$ be a (family of) polynomial time Boolean function(s). Let $S$ be the *truth set* of $f$:

$$S = \{x \in \{0,1\}^n | f(x) = 1\}.$$

We wish to find an element of $S$. Suppose $k = \#S$. We work in $\mathbb{H}^N$.

1. Prepare the $n$-qubit state $j = \frac{1}{\sqrt{N}} \sum |x\rangle$.
2. Compute the smallest $\alpha \geq 0$ so that $\sin \alpha = \sqrt{k/N}$. Set $t = \lfloor \pi/4\alpha \rfloor$.
3. Apply $(R_j \cdot G_f)^t$ to $j$.
4. Measure the resulting state and output the result $|x\rangle$.

**Theorem 4.3.** *Grover's algorithm outputs* $x \in S$ *with probability at least* $1/2$.

Since $f$ is in P, we can check if $x \in S$ by computing $f(x)$. If $f(x) = 0$, we repeat the algorithm. The probability of error approaches $0$ exponentially as we iterate, independent of $N$.

One edge case is when $k > N/2$. Then $\alpha > \pi/4$, and so $t = 0$. But then we are just measuring $j$, which is equivalent to choosing a bit string $x \in \{0,1\}^n$ uniformly randomly. This is exactly the classical algorithm! But $k > N/2$, so we end up trivially satisfying the theorem.

The proof, strangely enough, depends on the following classical fact:

**Proposition 4.4.** *Let $\ell_1$ and $\ell_2$ be lines in $\mathbb{R}^2$ passing through the origin. Let $\alpha$ be the (signed) angle from $\ell_1$ to $\ell_2$. Then reflection across $\ell_1$ followed by reflection across $\ell_2$ is the same as rotation by $\alpha$.*

We are now ready to prove our theorem.

*Proof of Theorem 4.3.* Let

$$h = \frac{1}{\sqrt{k}} \sum_{x \in S} |x\rangle .$$

Let $V \subset \mathbb{H}^N$ be the real span of $j$ and $h$; thus $V$ is isometric to $\mathbb{R}^2$. Define $m$ by

$$m = \frac{1}{\sqrt{N-k}} \sum_{x \notin S} |x\rangle .$$

The vectors $h, m$ stand for "hit" and "miss", respectively. Observe that if $\alpha m + \beta h$ is a state vector, then if we measure, the probability of obtaining a "hit" ($x \in S$) is $|\beta|^2$, and the probability of obtaining a miss is $|\alpha|^2$.

Notice that $m \in V$ and $h, m$ are orthonormal. Orient $V$ so that the angle from $m$ to $h$ is $+\pi/2$. Furthermore, let us think of the positive span of $m$ as the positive $x$-axis, and the positive span of $h$ as the positive $y$-axis; this allows us to assign angle measures to vectors in $V$. Specifically, given a unit vector $v \in V$ with angle measure $\theta$ (as compared to the positive $x$-axis), we have

$$v = (\cos\theta)m + (\sin\theta)h = \langle m, v\rangle\, m + \langle h, v\rangle\, h.$$

Using this formula, one checks that the angle of our initial vector $j$ is $\alpha$.

Let $\ell_2$ be the line spanned by $j$. Observe that $G_f$, $R_j$ act on $V$ as the reflections across the $x$-axis and $\ell_2$, respectively. Since $j \in V$, we have that $(R_j G_f)^r j \in V$ for all $r \in \mathbb{Z}$. Since $V$ is isometric to $\mathbb{R}^2$, Proposition 4.4 tells us that $R_j G_f$ acts on $V$ as a rotation by $2\alpha$. We have

$$\frac{\pi}{4\alpha} - 1 \leq t \leq \frac{\pi}{4\alpha}$$

so that

$$\frac{\pi}{2} - 2\alpha \leq 2\alpha t \leq \frac{\pi}{2}.$$

Therefore step 3 of Grover's algorithm rotates $j$ by $2\alpha t$. Since our start vector had angle $\alpha$, the vector at the end of step 3 has angle $\theta$ satisfying

$$\frac{\pi}{2} - \alpha \leq \theta \leq \frac{\pi}{2} + \alpha.$$

The associated $n$-qubit state is

$$(\cos\theta)m + (\sin\theta)h.$$

When measuring, the probability of obtaining $x \in S$ is thus $\sin^2\theta$. By the remarks after Grover's algorithm, we may assume that $\alpha \leq \pi/4$, from which $\sin^2\theta \geq 1/2$. The theorem follows. □

We would like to give an estimate of the running time. To do so, we must estimate $t$. Via $\sin x \leq x$, we must have $\alpha \geq \sqrt{k/N}$, whence $t = O(\sqrt{N/k})$, as claimed earlier. Since the probability of success is at least $1/2$, the expected number of times we need to run the entire algorithm before finding $x \in S$ is less than or equal to 2. The oracle $f$ is polynomial time, and so the expected total run time is $O(\sqrt{N/k}) \cdot n^{O(1)}$.