Learners' Space

Introduction to Quantum Computing

"Quantum computation will be the first technology that allows useful tasks to be performed in collaboration between parallel universes."

— David Deutsch

Pranav | **Advay** | **Soham** | **Darsh** | **Nitanshu**

Contents

Week 3

Quantum Algorithms

The essential task of a quantum algorithm, as we shall discuss here, is to use the laws of the quantum world to perform a specific computational task (often classical) in a more efficient or cost-effective manner. We will qualify what we mean by "cost-effective" later. But as of now, the key aspect of a quantum algorithm can be noted in the following four steps :

- A classical specification of the problem and data.
- Encoding of classical data into quantum states.
- Evolution of the classical data via the quantum evolution.
- Extract the answer from the quantum system.

The advantage of a quantum algorithm lies primarily in the third point-the fact that classical data can be manipulated efficiently by encoding them in a quantum degree of freedom. For instance, the key concept of "quantum parallelism" implies that a lot of classical information can be uploaded in a quantum superposition, and any operation can be implemented in parallel.

So our main target here is to look at a set of algorithms that basically implement the above four steps to solve a problem.

1. Bit strings, states, functions and the 'oracle'

In this section, we carry over our notation and the general notion of what is "classical" and "quantum." For instance, a string of bits, $x = 0010011$ or $x = x_0x_1x_2\ldots x_n$. The quantum version using qubits is given by the tensor product:

$$|x\rangle = |x_0\rangle \otimes |x_1\rangle \otimes \cdots \otimes |x_n\rangle\,, \quad \text{where } |x_i\rangle \in \{|0\rangle\,, |1\rangle\}$$

The set of all n-bit strings is denoted by $\{0,1\}^n$, and for example:

$$\{0,1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

Therefore, any quantum state $|x\rangle$ where $x \in \{0,1\}^3$ would imply any one of the states in the 3-qubit computational basis.

Quantum algorithms often revolve around the analysis of a function $f(x)$, which takes specific values on bit strings and depending on the task in hand outputs a specific bit string. For example, $f : \{0,1\}^2 \to \{0,1\}$ is a function that allocates the values 0 & 1 to a two-bit string.

In general, $f : \{0,1\}^n \to \{0,1\}^m$ sends a bit string of length n to a bit string of length m as output.

A function $f(x)$ in quantum algorithms can often be thought of as some kind of a "black box" or an oracle. It is an abstract subroutine, the cost of which can be ignored for the time being. This allows us to focus on the process of the quantum algorithm rather than how a function is implemented.

For example, consider the search for the factors of a prime number p. To find a divisor of p, we assume a quantum oracle that checks whether x divides p, such that :

$$f(p,x) = \begin{cases} 1 & \text{if } r(p,x) = 0 \\ 0 & \text{if } r(p,x) \neq 0 \end{cases}$$

$r(p,x)$ is a function that returns the remainder when p is divided by x.

In the grander scheme of finding an efficient search algorithm, the implementation of a function $f(p,x)$ is not important and can perhaps also be done very efficiently using classical computing. Such subroutines are often referred to as the oracle.

Importantly, the use of a quantum oracle needs to be defined very specifically within the context of a quantum algorithm. Boolean functions discussed earlier need to be implemented as unitary operators, and so are the oracles acting on bit strings.

For example, $f : \{0,1\}^3 \to \{0,1\}$, where $f(x_1, x_2) = x_1 \oplus x_2$. Simply, writing $O_p |x\rangle = |f(x)\rangle$ does not work, as O_p here is not reversible or unitary. Therefore, the oracle O_p is defined as:

$$O_p \left(|x\rangle \otimes |0\rangle\right) = |x\rangle \otimes |f(x)\rangle$$

2. Computation Costs & the Notion of Complexity

A natural question to understand before further exploring quantum algorithms is the cost associated with solving a problem. One of the key goals of building a well-functioning quantum computer would be to reduce the computational cost associated with certain tasks that cannot be solved using classical computation.

The basic structure of a computational problem is as follows:

- **Input:** (what we have)
- **Promise:** (what assumptions can be made / guarantees)
- **utput** (what we are expected to return)

For instance if we are checking whether $f(|x\rangle)$ implements the function we want:

$$f : \{0,1\}^2, \quad f(x_1x_2) = f(x_2x_1)$$
$$\text{Determine if } f(x_1x_2) = x_1 \oplus x_2$$

An algorithm is simply a recipe to get the right output when we are handed the input. There could be different algorithms for a fixed problem.

The most intuitive algorithm would be to try all possible inputs and check if all the outputs match up. Say we try $f(00), f(01), f(10)$ and $f(11)$ and check if the function returns $0, 1, 1, 0$ and then return YES. So, we need 4 different queries of the function f, and we ask if we can do better.

We have the promise $f(x_1x_2) = f(x_2x_1)$, so we need only one of $f(10)$ and $f(01)$, this was promised to us. So we have only 3 queries. Can we do better? Probably NOT.

Let us generalize the last problem, say we consider n-bits instead of two.

- **Input:** A function $f : \{0,1\}^n \to \{0,1\}$
- **Promise:** $f(x_1x_2x_3 \ldots x_n)$ is independent of $x_1, x_2, \ldots, x_n$
- **Output:** Determine if $f(x_1x_2 \ldots x_n) = x_1 \oplus x_2 \oplus \ldots \oplus x_n$

A straightforward analysis shows that we only need to check strings that have different number of 0s (or 1s) in the sequence. So we have a total of $n + 1$ different Hamming weights:

$$00\ldots0, \quad 00\ldots1, \quad 00\ldots11, \quad \ldots, \quad 111\ldots1$$

Therefore, we need $n+1$ queries to determine the output. Let us label this as $Q = n+1$. If we had simply checked all possible strings then we would need $Q = 2^n$ queries of the function f, which is exponentially larger than the number of bits. Why is 2^n? Because each bit takes 2 values $\{0, 1\}$ and there are n such bits. Think of the computational basis, where we use qubits instead of bits.

Suppose we did not have the "promise"; the upper bound on the number of queries and how it scales with n can be represented by the expression, $Q = O(2^n)$: this implies the queries scale at most as 2^n. Now if there is an algorithm that can get the job done with n^3 queries, then we have $Q = O(n^3)$, and we have an exponential speed up, since Q has been reduced from exponential to polynomial. Now, one can invoke the promise and get the job done in $n + 1$ queries, then $Q = O(n)$, and one has achieved a polynomial speedup over the previous algorithm.

Now, the number of queries required to solve the problem, among others, is an insight into the complexity of the problem. So, one may ask, can someone do better than $n + 1$ queries? Intuitively, it is NOT. But could there exist some physical laws or transformations that allow you to do better - say reduce the complexity to $O(n^{1/3})$ or $O(1)$. This is

exactly where quantum physics clips in - can we find such strange transformations that allow us to come up with quantum algorithms that reduce complexity in certain classical problems?

Another aspect of complexity that we discussed earlier is the number of single qubit universal gates and CNOT gates needed to realize a particular algorithm. While studying universal gates we showed that for a generic n-qubit the number of single qubit and CNOT gates required scales as $O(n^2 4^n)$, which is represented by $T = O(n^2 4^n)$.

3. Complexity of Probabilistic Algorithms

Let us consider a slightly different problem than before:

- **Input:** Again a function $f : \{0,1\}^n \to \{0,1\}$
- **Promise:** Say one of the two cases holds
  - (a) $f(x_1 x_2 \dots x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$ (50% of strings) and $f(x_1 x_2 \dots x_n) = 1 \oplus x_1 \oplus x_2 \oplus \dots \oplus x_n$ (50% of strings)
  - (b) $f(x_1 x_2 \dots x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$ (for all the strings)
- **Output:** Determine if $f(x_1 x_2 \dots x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$ or NOT (Determine which of the two cases holds)

We can again check all the strings to verify the result-we do not really have any permutation symmetry based "promise". So, at first glance we have $Q = O(2^n)$. Can we reduce this?

An obvious reduction without any exponential speed-up is to check half of all possible strings and then one. We need to check $2^{n-1} + 1$ of all the strings. So, $Q = 2^{n-1} + 1 = O(2^{n-1})$.

But we can take a radical approach. Let us randomly pick a string $x = x_1 x_2 \dots x_n$, then we have 50% chance that it satisfies $f(x) = \bigoplus_k x_k$ and 50% otherwise. So, the probability that we can check if $f(x) \neq \bigoplus x_k$ is $1 - \frac{1}{2} = \frac{1}{2}$.

Now, if we take another random string, the probability of successfully checking $f(x) = \bigoplus_k x_k$ is now $1 - \frac{1}{2^2} = 1 - \frac{1}{4} = \frac{3}{4}$. So, if we do M such random strings, the probability of successfully detecting is $1 - \frac{1}{2^M}$. Importantly, the queries do not scale with n and therefore $Q = O(1)$ if one allows a small error $\frac{1}{2^M}$.

Hence, certain probabilistic algorithms allow for exponential speedups where the complexity was reduced from $O(2^{n-1})$ to $O(1)$, provided a small error can be tolerated. Note, if we want a deterministic outcome we will need $M = 2^{n-1} + 1$ queries.

Primitives and algorithms

In subsequent sections, we will study a set of quantum primitives and quantum algorithms. The primitives represent a certain action being performed by a function (and not really a specific problem). On the other hand, we can use these primitives to write down interesting quantum algorithms to solve specific problems.

4. Primitives

4.1 Phase kickback

Phase kickback is a method by which we upload classical information (the value of a function) into a nonclassical degree of freedom (the phase of a quantum state).

We start with a function $f : \{0,1\}^n \to \{0,1\}$, for which one can use an oracle that achieves the following transformation:

$$O_p[|x\rangle \otimes |y\rangle] = |x\rangle \otimes |y \oplus f(x)\rangle .$$

Note that the oracle transforms classical data to another set of classical bits but does it coherently (it's an unitary). In terms of a unitary operator, we can write:

$$O_p = \sum_z |z\rangle \langle z| \otimes X^{f(z)}, \quad \text{where } X^0 = I \text{ and } X^1 = X.$$

If $f(x) = 0$, then $O_p[|x\rangle \otimes |y\rangle] = \sum_z |z\rangle \langle z| |x\rangle \otimes X^0 = |x\rangle \otimes |y\rangle$

If $f(x) = 1$, then $O_p[|x\rangle \otimes |y\rangle] = \sum_z |z\rangle \langle z| |x\rangle \otimes X = |x\rangle \otimes |y \oplus 1\rangle$

For phase kickback we need something that is a bit more "non-classical", such that the function can be saved in the phase of a quantum state. We want something like:

$$U_p |x\rangle = (-1)^{f(x)} |x\rangle , \quad \text{where } x \in \{0,1\}^n$$

As discussed earlier, $x = x_0 x_1 x_2 \ldots x_n$, where $x_i \in \{0,1\}$. In terms of qubits:

$$|x\rangle = |x_0\rangle \otimes |x_1\rangle \otimes \cdots \otimes |x_n\rangle , \quad \text{where } x_i \in \{0,1\}$$

How can we achieve phase kickback? Consider the state $|x\rangle \otimes |-\rangle_B$, such that:

$$|-\rangle_B = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \text{ [The ancilla qubit]}$$

The $|-\rangle_B$ state can be obtained by applying the Hadamard gate to the state $|1\rangle_B$.

$$|x\rangle \to |x\rangle \otimes |-\rangle_B$$

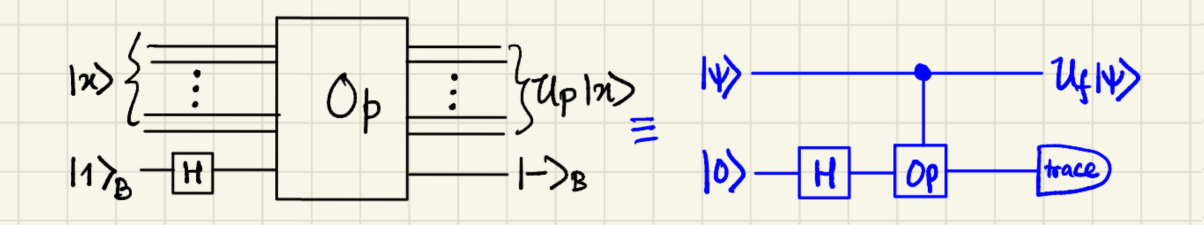
$$
\begin{aligned}
O_p\left[|x\rangle \otimes |-\rangle_B\right] &= \sum_z |z\rangle\langle z| \otimes X^{f(z)}\left[|x\rangle \otimes |-\rangle_B\right] \\
&= |x\rangle \otimes (-1)^{f(x)} |-\rangle_B \\
&= (-1)^{f(x)} |x\rangle \otimes |-\rangle_B
\end{aligned}
$$

We can then trace out the ancilla $|-\rangle_B$, giving:

$$
U_P |x\rangle = (-1)^{f(x)} |x\rangle
$$

This process can be summarized as:

$$
U_f |\psi\rangle = \mathrm{Tr}_B\left[O_f\left(|\psi\rangle \otimes |-\rangle_B\right)\right]
$$



In fact all values of the function can be uploaded using the oracle just a single time. Let's see how that can be done:

Let's start with the input state $|000\ldots0\rangle$ and apply n Hadamards, i.e., $H^{\otimes n}$, such that each qubit is acted upon by a Hadamard gate. ($|z\rangle$ is the n-qubit computational basis.)

$$
H^{\otimes n} |00\ldots0\rangle = \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} |z\rangle
$$

Moreover, $\frac{1}{\sqrt{2^n}}$ is the normalization. Now, we apply the phase kickback mechanism through a single oracle query to O_p, resulting in:

$$
U_P\left(\frac{1}{\sqrt{2^n}} \sum_z |z\rangle\right) = \frac{1}{\sqrt{2^n}} \sum_z (-1)^{f(z)} |z\rangle
$$

Theorem 4.1

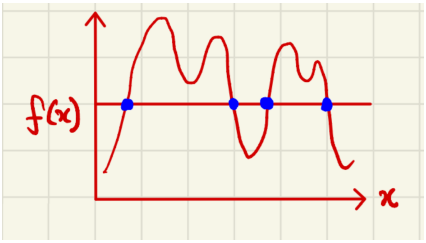
Given a function $f : \{0,1\}^n \to \{0,1\}$ with oracle O_p, the phase kickback unitary is given by:

$$
U_P |x\rangle = (-1)^{f(x)} |x\rangle ,
$$

and can be realized with a single use of the oracle O_p.

4.2 Pre-image states

The second primitive to upload classical information to a quantum degree of freedom is to find out the set of strings x for which one gets the output y, i.e., the pre-image $f^{-1}(y)$ of the output y.



The blue dots are the pre-image of the function $y = f(x)$. The idea is to create a uniform superposition over the pre-image of some output y.

In contrast to phase kickback, the present primitive considers a more general function $f : \{0,1\}^n \to \{0,1\}^m$ that can output a string y of length m. For instance,

$$|f^{-1}(y)\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=y} |x\rangle$$

where M is the cardinality of the set of pre-image states or the number of states x that give output y.

Let us begin with a state $|0\rangle^n \otimes |0\rangle^m$. On $n+m$ qubits, we apply a Hadamard gate to the first n qubits $H^{\otimes n}$, such that:

$$(H^{\otimes n} \otimes I_m) |0\rangle^n \otimes |0\rangle^m = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} |z\rangle \otimes |0\rangle^m$$

Now, we apply the oracle $O_p[|x\rangle \otimes |y\rangle] = |x\rangle \otimes |y + f(x)\rangle$ to the above state:

$$O_p \left[\frac{1}{\sqrt{2^n}} \sum_z |z\rangle \otimes |0\rangle^m \right] = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle \otimes |f(z)\rangle$$

Measuring the m qubits to get a specific output $|y\rangle$, which updates the total state to:

$$|f^{-1}(y)\rangle \otimes |y\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=y} |x\rangle \otimes |y\rangle$$

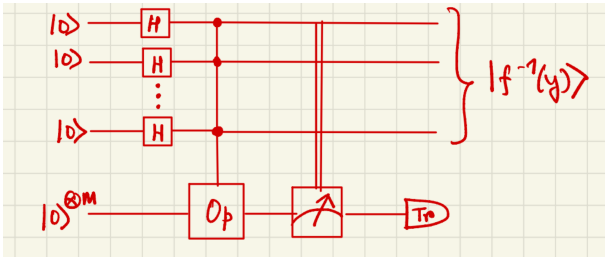
where M is again the number of states x that give $f(x) = y$ with probability $M/2^n$. If we trace out the state $|y\rangle$, we are then left with:

$$|f^{-1}(y)\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=y} |x\rangle$$

Theorem 4.2

Given a function $f : \{0,1\}^n \to \{0,1\}^m$, there is a quantum primitive that randomly outputs the pre-image state $|f^{-1}(y)\rangle$ with probability $M/2^n$, using a single query of the oracle O_p.

The circuit for the pre-image state is something like this:



Here we note that the measurement returns some random $|y\rangle$ and the pre-image state $|f^{-1}(y)\rangle$ associated with y, with a certain probability. **We cannot control which pre-image state we get.**

So phase kickback and pre-image states give us two ways to map classical information to nonclassical states. Next, we look at an algorithm that uses the phase kickback.

4.3 Quantum Fourier Sampling

Fourier transform is ubiquitous not only in physics but science in general. However, it is computationally hard to do and the best classical algorithms, such as the Fast Fourier Transform (FFT), take $O(n2^n)$ computational steps to achieve results. The Quantum Fourier Transform is a non-classical version of the discrete Fourier transform, which allows the sampling of the amplitudes of the transform—something that can't be done efficiently classically.

A **discrete Fourier transform** takes a input, a vector of complex numbers $x_0, x_1, \cdots, x_{N-1}$ and outputs the transformed data, which is a vector of complex numbers $y_0, y_1, \cdots y_{N-1}$, defined by:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2\pi i}{N} jk} x_j = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega_N^{jk} x_j$$

here ω_N^{jk} are the N^{th} roots of unity.

A similar approach is adopted in QFT, where we think of an orthonormal basis $|0\rangle, |1\rangle, \cdots, |N-1\rangle$ and the linear operator on the states:

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} jk} |k\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle$$

This maps any general quantum state in a basis to another, such that

$$\sum_j x_j |j\rangle \to \sum_k y_k |k\rangle$$

where y_k are the Fourier transform of the amplitudes x_j.

4.3.1 The Hadamard Gate—QFT of Qubits

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{and} \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

In a more relatable form to QFT, this can be written as:

$$H|x\rangle = \frac{1}{\sqrt{2}} \sum_{y\in\{0,1\}} (-1)^{xy} |y\rangle$$

We can generalize this to n qubits, where $|x\rangle = |x_1 x_2 ... x_n\rangle$ and we apply $H^{\otimes n}$ to the n qubits. This gives us,

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y} |y\rangle$$

where, $x \cdot y = x_0 y_0 \otimes x_1 y_1 \cdots \otimes x_n y_n = (x_0 y_0 + x_1 y_1 + ... + x_n y_n) mod\, 2$.

We need to make a clear distinction here; the above refers to QFT applied individually to n qubits. This is not the same as the quantum Fourier transform of an $n-$dimensional quantum state, which we will revisit later.

4.3.2 Qubit Fourier Sampling

Theorem 4.3

Given an n qubit state $|\psi\rangle = \sum_x \phi(x)|x\rangle$, we can efficiently do a qubit Fourier sampling of the distribution $p(y) = |\phi(y)|^2$ where $\phi(y) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot y} \phi(x)$.

We note that $\phi(y)$ above is the quantum Fourier transform of $\phi(x)$. To prove the above, we note that we already have the state $|\psi\rangle$, and applying $H^{\otimes n}$ gives us:

$$\begin{aligned} H^{\otimes n}|\psi\rangle &= H^{\otimes n} \sum_x \phi(x)|x\rangle \\ &= \sum_x \phi(x) \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y} |y\rangle \quad = \sum_{y\in\{0,1\}^n} [\sum_x \frac{1}{\sqrt{2^n}} \phi(x)(-1)^{x\cdot y}] |y\rangle \end{aligned}$$

If we make a measurement, we will get the state $|y\rangle$ with probability,

$$\phi(y) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot y} \phi(x)$$

Note that the sampling primitive takes us from quantum to classical, in contrast to phase-kickback and pre-image states, which encode or upload classical data into quantum states and therefore transfer classical data to quantum states. Combining the two, therefore, allows us to come up with a classical-to-quantum algorithm that operates with distinct advantages.

5. Deutsch-Jasza Algorithm

The Deutch-Jasza is a deterministic quantum algorithm that combines phase kickback and quantum Fourier sampling to achieve an exponential speedup over a deterministic classical algorithm. The statement of the problem is as follows:

- **Input:** A function $f : \{0,1\}^n \to \{0,1\}$ and an oracle O_p to implement the function.
- **Promises:** One of the two holds:
  (i) $f(x)$ is constant. This implies that either $f(x) = 0$ or $f(x) = 1$ for all x.
  (ii) f(x) is balanced, i.e., $f(x) = 0$ on half the strings and $f(x) = 1$ on the remaining strings.
- **Output:** Find out whether $f(x)$ is constant or balanced.

Step 1: We use phase kickback to upload the classical data to the phase of the state.

$$H^{\otimes n} |0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$$

$$U_p H^{\otimes n} |0\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle$$

Step 2: Use the Fourier transform on the above superposition

$$H^{\otimes n} \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle \to \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} H^{\otimes n} |x\rangle$$
$$= \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \quad = \frac{1}{2^n} \sum_y [\sum_x (-1)^{f(x)} (-1)^{x \cdot y}] |y\rangle$$

If we measure the output state, the probability of finding the state $|y\rangle = |00\cdots0\rangle$ is equal to

$$p(y = 0^{\otimes n}) = |\frac{1}{2^n} \sum_x (-1)^{f(x)}|^2$$

If f(x) is constant, $p(y = 0^{\otimes n}) = 1$ and therefore the output state has to be $|00\cdots0\rangle$. For any other output, we know that $f(x)$ is not constant and therefore must be balanced. So, the Deutsch-Josza algorithm can solve the problem using a single query of the oracle, i.e., $Q = O(1)$.

Theorem 5.1

Given a function $f : \{0,1\}^n \to \{0,1\}$, with either
(i)$f(x) = 0$ or $f(x) = 1$ for all x, or
(ii) $f(x) = 0$ and $f(x) = 1$ for half of the cases, there exists a quantum algorithm (Deutsch-Josza algorithm) that deterministically finds whether $f(x)$ satisfies (i) or (ii) with $Q = 1$ queries to the oracle.

6. Bernstein-Vazironi Algorithm

This algorithm again uses phase kickback and Fourier sampling to obtain a polynomial speedup over classical computation. However, unlike the Deutsch-Jozsa algorithm, the speed up here also extends to non-deterministic or probabilistic classical algorithms and thus opens a true separation in the complexity between classical and quantum computation.
The problem being addressed in the BV algorithm can be stated as:

- **Input:** A function $f : \{0,1\}^n \to \{0,1\}$ and an oracle.
- **Promise:** $f(x) = a.x \, mod \, 2$ for some $a = a_0 a_1 a_2 \cdots a_{n-1}$.
- **Output:** Find the string a.

Now, classically this problem takes $Q = O(n)$ queries to find out the components of a, i.e., $a_0, a_1, a_2, \cdots, a_{n-1}$. For instance,

$$\begin{aligned} f(x = 100\cdots0) &= a_0 \\ f(x = 010\cdots0) &= a_1 \\ &\vdots \\ f(x = 000\cdots1) &= a_{n-1} \end{aligned}$$

This requires n queries to find all the values of a_I. Even probabilistically $Q = O(n)$. However, using the BV algorithm, one can find a with $Q = 1$.

Theorem 6.1

Given $f : \{0,1\}^n \to \{0,1\}$ with $f(x) = a.x$ for some unknown $a = a_0 a_1 ... a_{n-1} \in \{0,1\}^n$, there is a quantum algorithm (BV algorithm) that can find a with certainty with $Q = (1)$ queries to the oracle.

Step 1: We again intialize the state $|0\rangle^{\otimes n}$ and apply $H^{\otimes n}$ to get a uniform superposition state. Again a single call to the oracle we do a phase kickback to get,

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{a.x} |x\rangle$$

Step 2: We now do a quantum Fourier Transform on $|\psi\rangle$,

$$H^{\otimes n} |\psi\rangle = \frac{1}{2^n} \sum_y [\sum_x (-1)^{(a+y).x}] |y\rangle$$

Now, the term $\sum_x (-1)^{(a+y).x} = 2^n \delta_{a,y}$ where $\delta_{a,y} = 1$ if $a = y$ and zero otherwise.

Therefore, the Fourier transformed state is

$$H^{\otimes n} |\psi\rangle = \frac{1}{2^n} \sum_y 2^n \delta_{a,y} |y\rangle = |a\rangle$$

which is encoded in the output register or qubits.

7. Simon's Algorithm

While Deutsch-Jozsa and Bernstein-Vazirani algorithms used phase kickback to encode the classical information, we now look at an algorithm that uses pre-image state, viz, Simon's algorithm. Importantly, Simon's algorithm, similar to BV algorithm, provides a separation between classical and quantum computation. However, Simon's algorithm allows for an exponential speedup, as compared to the polynomial speedup of BV algorithm. Let us outline the problem.

- **Input:** A function $f : \{0,1\}^n \to \{0,1\}^m$ and an oracle.
- **Promise:** $f(x) = f(y)$ iff $x = y$ and $y = x \oplus a$ for some fixed $a = a_0 a_1 ... a_{n-1} \in \{0,1\}^n$
- **Output:** Find the sting a.

Classically, this is hard to do as we need to sift through all possible states to find $f(x) = f(y)$, in fact we need to query the oracle an exponential number of times before we get a collision between x and y such that $f(x) = f(y)$, which will allow us to then calculate $a = x \oplus y$. The classical complexity turns out to be $Q = O(\sqrt{2^n})$ (check the Birthday Problem on Wikipedia). However, Simon's algorithm allows us to find a in $Q = O(n)$, thus providing an exponential speedup over the classical algorithm.

Theorem 7.1

Given a function $f : \{0,1\}^n \to \{0,1\}^m$ for which $f(x) = f(y)$ iff $x = y$ or $x = y \oplus a$, for some string $a = a_0 a_1 ... a_{n-1} \in \{0,1\}^n$, there is an algorithm (Simon's algorithm) that can find a with certainty with $Q = O(n)$ queries to the oracle.

Step 1: We construct the pre-image state $|f^{-1}\rangle$ by starting with $n+m$ qubits $|0\rangle^n \otimes |0\rangle^m$ and applying $H^{\otimes n}$

$$H^{\otimes n} |0\rangle^n \otimes |0\rangle^m = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |0\rangle^m$$

Applying the oracle O_p to the previous state we get

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |f(x)\rangle$$

We then measure the second register and get some outcome x_0, such that

$$|f^{-1}(x_0)\rangle \otimes |x_0\rangle = \frac{1}{\sqrt{M}} \sum_{x : f(x) = x_0} |x\rangle \otimes |x_0\rangle$$

However, from the function we know that $f(x) = f(y)$ for $y = x$ and $y = x \oplus a$. After discarding the second register, we get

$$|f^{-1}(x_0)\rangle = \frac{1}{\sqrt{2}} (|x\rangle + |x \oplus a\rangle)$$

Step 2: We now apply a Fourier transform to the state $|f^{-1}(x_0)\rangle$

$$\begin{aligned} H^{\otimes n} &= \frac{1}{\sqrt{2}}(H^{\otimes n}|x\rangle + H^{\otimes n}|x \oplus a\rangle) \\ &= \frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2^n}} \sum_{y \in \{0,n\}^n} (-1)^{x \cdot y}|y\rangle + \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{(x \oplus a) \cdot y}|y\rangle) \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_y ((-1)^{x \cdot y}(1 + (-1)^{a \cdot y})|y\rangle) \end{aligned}$$

Let us now measure the output registers. We note that for $a \cdot y' = 1$, the amplitude $(1 + (-1)^{a \cdot y'} = 0$ and therefore $y = y'$ will never occur. Therefore, for all measured values of y we must have $a \cdot y = 0$.
So for a set of measurements we have outcomes $y_1, y_2, y_3, ..., y_k$ and each satisfy

$$\begin{aligned} a \cdot y_1 &= 0 \\ a \cdot y_2 &= 0 \\ &\vdots \\ a \cdot y_k &= 0 \end{aligned}$$

This gives us a set of at most n linearly independent equations to find a. Therefore, one needs to make at most n queries to the oracle and $Q = O(n)$.
Therefore, Simon's algorithm provides an exponential speedup over all classical algorithms, even if we allow probabilistic computation.

8. Algorithm: Amplitude Amplification

All the algorithms we have studied so far make use of "quantum parallelism"—essentially by encoding information in the superposition states and then sampling the Fourier space to obtain some speedup. Let us now look at an algorithm that works a bit differently than those we have studied so far.

Suppose you have a few red balls (3 or 4) in a container with a million balls of different colors. How many searches would be needed to find one red ball?
A million searches!
, Let's rephrase: If a jar has N balls and you need to find one specific ball, you will need queries of the order of N, i.e.$Q = O(n)$. The scaling is not bad, but can be massive for large datasets and it is not intuitively clear how this can be done better. However, quantum theory allows for a quadratic speed up! $(Q = O(\sqrt{n})$.

This is an instance of something more general – a method that is only successful with some probability p, where $p \ll 1$. So, in order to achieve the outcome with probability close to unity, we need roughly $1/p$ attempts. This is akin to the number of searches required to find the red ball that occurs with some probability p. So, the question is – can quantum physics improve your odds and allow you to search for a successful outcome with less than $1/p$ attempts? **In other words, can it amplify the amplitudes or odds of a successful outcome? This is the crux of quantum search algorithms such as Grover's algorithm.**

Another way of thinking of the search algorithm is to think of searching by the *index* rather than the contents, which can just be a number between 0 and $2^n - 1$, using an n-qubit system. The correct indices occur with probability p.

These are two key elements to amplitude amplification – (1) an algorithm that produces the correct result with probability p, and (2) a quantum oracle that is a checking function *i.e.* it checks if the solution is correct or not.

8.1 Probabilistic Algorithms as Unitaries

We know that classical probabilities are associated with mixed states but quantum gates and functions are unitaries that act on pure states. However, any probabilistic algorithm can be represented as a unitary using an extended state space or ancilla qubits.
Consider we have an algorithm A_f that succeeds in computing $f(x)$ with probability p. This can be written as a unitary:

$$A_f\left[|x\rangle_{\text{in}} \otimes |0\rangle_{\text{out}} \otimes |0\rangle_{\text{anc}}\right] = \sqrt{p}\,|x\rangle_{\text{in}} \otimes |f(x)\rangle_{\text{out}} \otimes |h(x)\rangle_{\text{anc}} + \sqrt{1-p}\,|\Psi\rangle_{\text{in,out,anc}}$$

This can be compared to a general quantum operator being written as a unitary acting on the system plus an environment.

In fact, there can be several good output states such that:

$$A_f\left[|x\rangle_{\text{in}} \otimes |0\rangle_{\text{out}} \otimes |0\rangle_{\text{anc}}\right] = \sqrt{p}\,|x\rangle_{in} \otimes \sum_y |y\rangle_{\text{out}}|h(x,y)\rangle_{\text{anc}} + \sqrt{1-p}\,|\Psi\rangle_{\text{in,out,anc}}$$

Tracing out the ancilla gives us the "good states" with probability p and the "bad states" with probability $1-p$. To simplify, we can then write:

$$A_f\left[|x\rangle_{\text{in}} \otimes |0\rangle_{\text{out}} \otimes |0\rangle_{\text{anc}}\right] = \sqrt{p}\,|\psi_{\text{good}}\rangle + \sqrt{1-p}\,|\psi_{\text{bad}}\rangle$$

8.2 A Checking Function $c(x, y)$

One can also have an oracle that checks whether a solution is correct; i.e., a checking function. In other words, this function can be defined as follows:

$$c : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$$

for which $c(x, y) = 1$ if y is correct given input x, and $c(x, y) = 0$, otherwise.

There is nothing special or tricky about the oracle here. It appears the oracle already knows the correct answer. But for these kinds of problems, **checking a solution is relatively easier than actually finding the solution**. For example, it is much simpler to check whether a ball is red, rather than find the red ball.

Consider the famous prime factorization problem, where m is a large number, which is a product of two primes p and q. Finding p and q for an arbitrarily large but known m is a difficult task — all your banking details are transferred online depending on their "hardness". However, if you are given p, you can quickly use your laptop to check if p is a factor of m ("simply divide and check").

Let us now look at amplitude amplification and the quadratic speed-up a bit closely. We have the probabilistic algorithm:

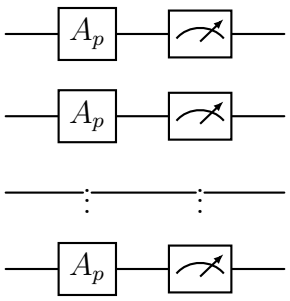
$$A_p\left[|x\rangle_{\text{in}} \otimes |0\rangle\right] = |\psi\rangle = \sqrt{p}\,|\psi_{\text{good}}\rangle + \sqrt{1-p}\,|\psi_{\text{bad}}\rangle$$

$$= \sin\theta\,|\psi_{\text{good}}\rangle + \cos\theta\,|\psi_{\text{bad}}\rangle$$

where $\sin\theta = \sqrt{p}$ is assumed to be small, i.e., $p \ll 1$.

- **Input:** A probabilistic algorithm A_p and a checking function $c(x, y)$
- **Promise:** A_p outputs a correct answer with probability p, and $c(x, y)$ is computable
- **Output:** An algorithm A_q that succeeds with $p \approx 1$

Classically, one can apply $1/p$ copies of A_p in parallel and check all the outputs to get the outcome with $p \approx 1$.



$\sim \frac{1}{p}$ times

In contrast quantum algorithm uses this coherently to provide a speedup.

Theorem 8.1

Given a probabilistic algorithm A_p that gives a correct and efficient answer with a probability p, there exists a quantum algorithm that outputs the correct answer with probability arbitrarily close to 1 using $Q = O(1/\sqrt{p})$ queries to A_p.

The quantum algorithm outputs the state

$$|\psi\rangle = sin(\theta)\,|\psi_{\text{good}}\rangle + cos(\theta)\,|\psi_{\text{bad}}\rangle\,.$$

So the idea is to now rotate $|\psi\rangle$ towards $|\psi_{\text{good}}\rangle$ using rotations. Let us look at the geometric illustration below:

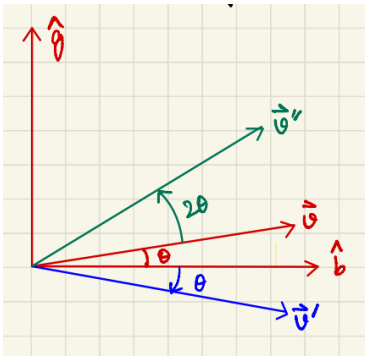
$$\vec{v} = sin(\theta)\hat{g} + cos(\theta)\hat{b}$$

Reflect $\vec{v}$ along the $\hat{b}$ axis

$$\vec{v}' = F_{\hat{b}}\vec{v}$$

$\vec{v}'$ makes an angle 2θ from its initial position. Reflect $\vec{v}'$ along $\vec{v}$:

$$\vec{v}'' = F_{\vec{v}}\vec{v}' = F_{\vec{v}}F_{\hat{b}}\vec{v}$$



$\vec{v}''$ moves away from $\vec{v}$ by an angle 2θ.
So the original $\vec{v}$ is now closer to $\vec{g}$ by an angle 2θ, after two reflections:

$$F_{\vec{v}} \cdot F_{\hat{b}} = R(2\theta)$$

.

Reflection around $\hat{b} = |\psi_{\mathbf{bad}}\rangle$:

To reflect the state or create a flipping about $|\psi_{\text{bad}}\rangle$ we need to know the "good" and "bad" states. This can be done using the checking function $c(x, y)$ and phase-kickback. The oracle $U_c = F_{\hat{b}}$, such that

$$U_c |x\rangle |y\rangle = (-1)^{c(x,y)} |x\rangle |y\rangle$$

It is clear that if x and y are the same then $c(x, y) = 1$ and we get $|x\rangle |y\rangle \to -|x\rangle |y\rangle$ and if $x \neq y$; $c(x, y) = 0$ and $|x\rangle |y\rangle \to |x\rangle |y\rangle$. Therefore, only the "good states" are flipped or reflected along $|\psi_{bad}\rangle$.

$$F_{\hat{b}} = U_c = \sum_{x \neq y} |x\rangle \langle x| - |y\rangle \langle y| = \mathbb{I} - 2 |y\rangle \langle y|$$

Reflection around $\vec{v} = |1_y\rangle$

We know the state $|1_y\rangle = A_p \left[|x\rangle_{\text{in}} \otimes |0\rangle_{\text{out,anc}} \right]$, so we can actually flip the state around $|x\rangle_{\text{in}} \otimes |0\rangle_{\text{out,anc}}$ and then apply the unitary A_p. Again we use a checking function and phase kickback, such that

$$U_{\bar{v}} |\bar{x}\rangle |\bar{y}\rangle = (-1)^{1 \oplus c(\bar{x}, \vec{y})} |\bar{x}\rangle |\bar{y}\rangle \qquad (\text{where, } |\bar{y}\rangle = |x\rangle_{\text{in}} \otimes |0\rangle_{\text{out,anc}})$$

Importantly $U_{\bar{v}}$ maps $|\bar{x}\rangle |\bar{y}\rangle \to |\bar{x}\rangle |\bar{y}\rangle$ if $\bar{x} = \bar{y}$ and flips the state otherwise, where $1 \oplus c(\bar{x}, \bar{y})$ acts like an anti-checking function.
Therefore,

$$U_{\bar{v}} = -\sum_{x \neq \bar{x}} |x\rangle \langle x| + |\bar{x}\rangle \langle \bar{x}| = 2 |\bar{x}\rangle \langle \bar{x}| - \mathbb{I}$$

Now, applying the unitary A_p, we get

$$F_{\vec{v}} = A_p U_v A_p^\dagger = 2 |\psi\rangle \langle \psi| - \mathbb{I}$$

Amplitude Amplification :

So, now we combine the two reflections $F_{\vec{v}}$ and F_b to get the rotation. This was done using the function $c(x, y)$ twice and two queries to A_p.
Now, we have:

$$G := F_{\vec{v}} \cdot F_b = R(2\theta) = \begin{bmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{bmatrix}$$

This then gives us the rotated state

$$G\left|\psi\right\rangle = sin(\theta + 2\theta)\left|\psi_{good}\right\rangle + cos(\theta + 2\theta)\left|\psi_{bad}\right\rangle .$$

If we apply the rotation k times we get

$$G^k\left|\psi\right\rangle = sin([1 + 2k]\theta)\left|\psi_{good}\right\rangle + cos([1 + 2k]\theta)\left|\psi_{bad}\right\rangle .$$

For the algorithm to be successful, we would like the final state to be $G^k\left|\psi\right\rangle \approx \left|\psi_{good}\right\rangle$. This implies that $(1 + 2k)\theta \approx \pi/2$ or that we should apply $k = \frac{\pi}{4\theta} - \frac{1}{2}$ (or nearest integer) rotations, which is a finite number for a small value of θ (i.e., $p \ll 1$). Now, we were promised $\sqrt{p} = sin\theta$ from the algorithm. So, we know θ. For $p \ll 1$, which is the area of interest, we have $\sqrt{p} = sin\theta \approx \theta \ll 1$.

$$\boxed{\therefore k \approx \frac{\pi}{4\sqrt{p}}}$$

Therefore with $Q = O(1/\sqrt{p})$ we get a close estimate of $\left|\psi_{good}\right\rangle$.

So after k rotations, measurement of the output register or qubits gives us a state close to the desired output. Since each rotation queries A_p twice, so the query is actually $\frac{\pi}{2\sqrt{p}}$. In essence, we have achieved a quadratic speed, with $Q = O(1/\sqrt{p})$, up over any classical algorithm by means of amplitude amplification.

An important point here is that the search algorithm works perfectly for $p \ll 1$, where $\theta \ll \frac{\pi}{2}$. If p is large (say $p = 1/2$), then a couple of searches should be enough anyway. More quantitatively, if $\theta > \pi/6$, then $k = \frac{\pi}{4\theta} - \frac{1}{2} < 1$ and the integer number of required searches vanishes. So, amplitude amplification is not well-defined at higher amplitudes. If the probability p (i.e., the number of correct solutions) is a priori not known, then an interesting approach is to double the search space and update the checking function of the oracle.

9. Quantum Search Algorithm

Grovers' search algorithm seeks with high probability a unique state from an unstructured set of data.

Suppose we have $N = 2^n$ basis states and we are seeking the state $|q\rangle$. Applying Hadamard to $|0\rangle^{\otimes n}$:

$$H^{\otimes n}|0\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_x |x\rangle$$
$$|S\rangle = \frac{1}{\sqrt{N}}|q\rangle + \frac{1}{\sqrt{N}}\sum_{x\neq q}|x\rangle$$

So the rotations we can apply are given by:

$$F_b = \mathbb{I} - 2|q\times q| \text{ and } F_v = 2|S\times S| - \mathbb{I}$$

Since $|S\rangle = \frac{1}{\sqrt{N}}|q\rangle + \frac{\sqrt{N-1}}{\sqrt{N}}|\bar{q}\rangle$ where $\langle q|\bar{q}\rangle = 0$, putting $\frac{1}{N} = p$ and defining the quantities V and U gives -

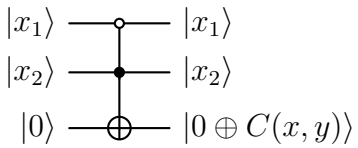
$$V = 2|S\times S| - \mathbb{I}$$
$$V = (1-2p)|\bar{q}\times\bar{q}| + (2p-1)|q\times q| + 2\sqrt{p(1-p)}(|q\times\bar{q}| + |\bar{q}\times q|)$$
$$U = \mathbb{I} - 2|q\times q| = |\bar{q}\times\bar{q}| - |q\times q|$$
$$VU = (1-2p)(|q\times q| + |\bar{q}\times\bar{q}|) + 2\sqrt{p(1-p)}(|q\times\bar{q}| - |\bar{q}\times q|)$$

If we take $\sqrt{p} = \sin\theta$, we get

$$VU = R(2\theta) = \begin{pmatrix}\cos 2\theta & -\sin 2\theta \\ \sin 2\theta & \cos 2\theta\end{pmatrix}$$

QSA and circuit for two qubits

Consider search space N(= 4). The checking function gives C(x,y)=1 for the correct search, otherwise it returns 0. So space is spanned by $x = \{00, 01, 10, 11\}$ and say the correct response was y = 01. Then



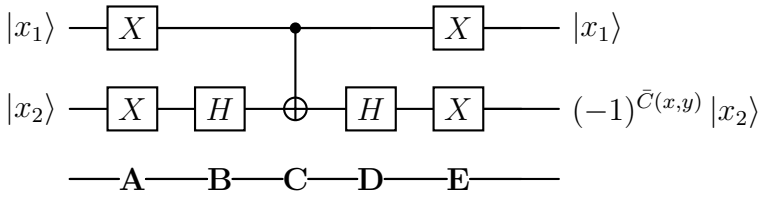
Consider the orcale is $O_p = \sum_x |x\rangle\langle x| \otimes X^{C(x,y)}$

- First step is to use Hadamard gates to create the search space and then apply a phase kickback using O_p. This gives $(-1)^{C(x,y)} = -1$ only for $|y\rangle\langle y|$ or the correct search. As such the circuit implements

$$\sum_x |x\rangle\langle y| - |y\rangle\langle y| = \mathbb{I} - 2|y\rangle\langle y|$$

which was the reflection around the bad state.

- Rotate around the initial state $2\left|\psi\right\rangle\left\langle\psi\right| - \mathbb{I}$, where $\left|\psi\right\rangle = H^{\otimes 2}\left|00\right\rangle = 1/2\sum_x \left|x\right\rangle$. This can be written as $H^{\otimes 2}(2\left|00\right\rangle\left\langle 00\right| - \mathbb{I})H^{\otimes 2}$, so next step would be to implement the circuit.
- The circuit (upto a global phase) can be implemented as-



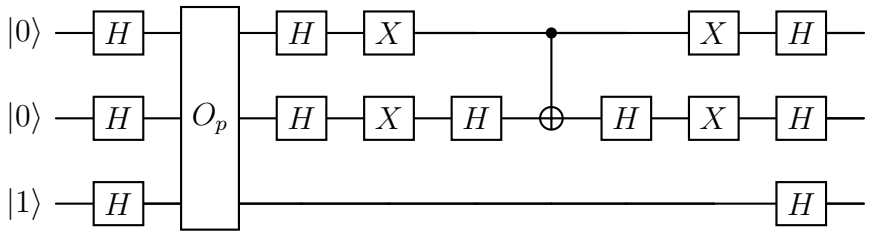
  This circuit remains unchanged for $\left|x_1\right\rangle$ and only $\left|x_2\right\rangle$ is affected.

$\left\|x_1\right\rangle\left\|x_2\right\rangle$	A	B	C	D	E
$\left\|1\right\rangle\left\|x_2\right\rangle$	$\left\|0\right\rangle\left\|x_2\right\rangle$	$\left\|0\right\rangle\left\|\pm\right\rangle$	$\left\|0\right\rangle\left\|\pm\right\rangle$	$\left\|0\right\rangle\left\|x_2\right\rangle$	$\left\|1\right\rangle\left\|x_2\right\rangle$
$\left\|0\right\rangle\left\|1\right\rangle$	$\left\|1\right\rangle\left\|0\right\rangle$	$\left\|1\right\rangle\left\|+\right\rangle$	$\left\|1\right\rangle\left\|+\right\rangle$	$\left\|1\right\rangle\left\|0\right\rangle$	$\left\|0\right\rangle\left\|1\right\rangle$
$\left\|0\right\rangle\left\|0\right\rangle$	$\left\|1\right\rangle\left\|1\right\rangle$	$\left\|1\right\rangle\left\|-\right\rangle$	$-\left\|1\right\rangle\left\|-\right\rangle$	$-\left\|1\right\rangle\left\|1\right\rangle$	$\left\|0\right\rangle\left\|0\right\rangle$

- The above circuit gives the transformation

$$-\left|00\right\rangle\left\langle 00\right| + \left|01\right\rangle\left\langle 01\right| + \left|10\right\rangle\left\langle 10\right| + \left|11\right\rangle\left\langle 11\right| = -(2\left|00\right\rangle\left\langle 00\right| - \mathbb{I})$$

  Here, the - sign denotes the global phase.
- Putting all the different elements together, we get the following circuit, which results in a single rotation $R(2\theta)$ or an iteration.



Exercise

For the above example, what would be the θ in terms of "good" and "bad" vectors, and how many rotations would you need?

10. Quantum Fourier Transform in N Dimensions

We will reformulate the QFT discussed earlier for an N-level system, where N can either be a collection of n qubits such that $N = 2^n$ or simply an N-dimensional quantum state. The state can be written in the basis$\{|i\rangle\}$ where $i = 0, 1, \cdots, N-1$ and each classical x is given by the number i. Again, we consider x to be defined modulo N such that $x \equiv N + k \mod N = k$. The periodicity here is consistent with the definition of the Fourier Transform, which deals with functions that oscillate.

Theorem 10.1

Quantum fourier transform (F_N) of a N-level system over $\mathbb{Z}_N = \{0, 1, \cdots, N-1\}$ as given by

$$F_N |x\rangle = \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}_N} \omega_N^{xy} |y\rangle = \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}_N} e^{2\pi \iota xy/N} |y\rangle$$

where $\omega_N = \frac{2\pi\iota}{N}$ is the N^{th} root of unity.

10.1 Circuit for Quantum Fourier Transform

For $N = 2^n$, F_N can be released with a quantum circuit using $O(n)$ gates, which is efficient compared to classical algorithms such as fast fourier transform that reqiuire $O(n2^n)$ gates.

$$|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi \iota xy/2^n} |y\rangle$$

where $x = x_1 x_2 \cdots x_n = x_1 2^{n-1} + \cdots + x_n 2^0$ and $y = y_1 y_2 \cdots y_n = y_1 2^{n-1} + \cdots + y_n 2^0$ Simplifying it further by putting values, one can write-

$$|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y_1} \sum_{y_2} \cdots \sum_{y_n} \otimes_{l=1}^{l=n} e^{2\pi \iota x y_l 2^{-l}} |y_l\rangle$$

$$|x\rangle = \frac{1}{\sqrt{2^n}} \otimes_{l=1}^{l=n} (|0\rangle + e^{2\pi x 2^{-l}} |1\rangle), \text{ since } y_l \text{ can take value 0 and 1}$$

$$|x\rangle = \frac{1}{\sqrt{2^n}} \left[\left(|0\rangle + e^{2\pi \iota [0.x_n]} |1\rangle \right) \otimes \left(|0\rangle + e^{2\pi \iota [0.x_{n-1} x_n]} |1\rangle \right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi \iota [0.x_1 \cdots x_n]} |1\rangle \right) \right]$$

Here $[0.x_m x_{m-1} \cdots x_n]$ represent binary fractions and is equal to

$$[0.x_m x_{m-1} \cdots x_n] = \frac{x_m}{2} + \frac{x_{m+1}}{2^2} + \cdots + \frac{x_n}{2^{n-m+1}}$$

Now,

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi \iota [0 \cdot x_1]} |1\rangle \right) = \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi \iota x_1/2} |1\rangle \right) = H |x_1\rangle$$

Let us consider,

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}, \quad \text{with control to the } k^{\text{th}} \text{ qubit (say)}$$

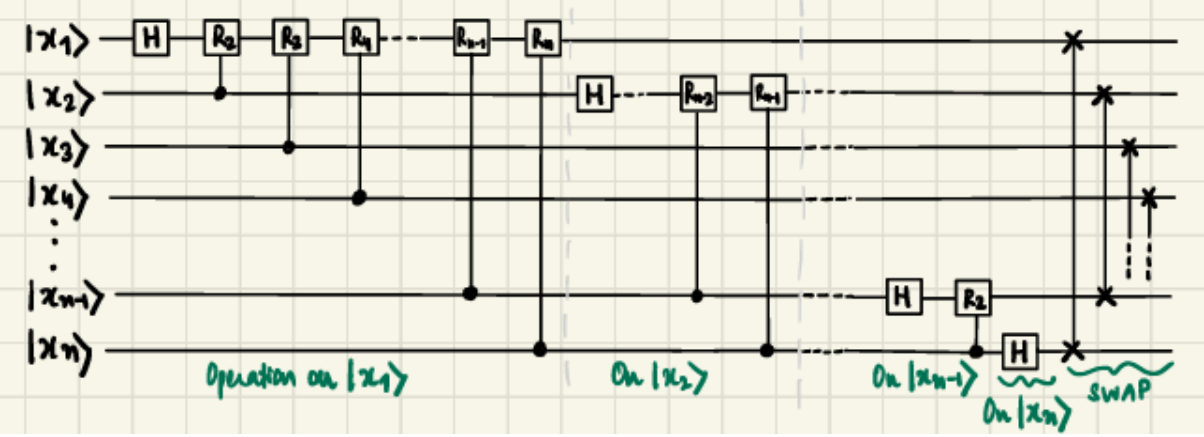
$$R_2 H |x_1\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi\iota(x_1/2 + x_2/2^2)} |1\rangle\right)$$ [1]

$$R_3 R_2 H |x_1\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi\iota(x_1/2 + x_2/2^2 + x_3/2^3)} |1\rangle\right)$$

$$\vdots$$

$$R_n R_{n-1} \cdots R_2 H |x_1\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi\iota[0.x_1 x_2 \ldots x_n]} |1\rangle\right)$$ [2]

This can then naturally be extended to the other qubits. The circuit for a quantum fourier transform using n-qubits is given as-



Theorem 10.2

Given a quantum state $|\psi\rangle = \sum_{x \in \mathbb{Z}_N} \phi(x) |x\rangle$, one can efficiently do a fourier sampling of the probability distribution $p(y) = |\phi(y)|^2$, where

$$\phi(y) = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_N} \omega^{xy} \phi(x)$$

is the quantum fourier transform of $\phi(x)$.

10.2 Period Finding

This uses a pre-image state to upload classical data into the quantum state followed by quantum Fourier sampling.

- **Input:** A function $f : \{0, 1, \ldots, N-1\} \to \{0, 1, \ldots, N-1\}$
- **Promise:** $f(x) = f(y)$ iff $y = x + ka \mod N$ for some $0 \leq a \leq N-1$ and $k \in \mathbb{Z}$. This ensures that f is unique in each period.
- **Output:** Find the period a.

[1] R_2 is applied with a control to x_2 such that it is applied only when $x_2 = 1$

[2] This should have been applied to x_n and not x_1; we swap x_1 and x_2 at the end.

Since everything is done modulo N, this implies the problem can be rephrased as $f(x) = f(y)$ iff $x = y \mod a$ for $a \in \mathbb{Z}$.

Step:1 Create the pre-image state for f as before.

$$F_N |0\rangle \otimes |0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_n} |x\rangle \otimes |0\rangle$$

Applying the orcale O_f and then measuring the second register to obtain some outcome y_0 gives us the collapsed state:

$$|f^{-1}(y_0)\rangle \otimes |y_0\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=y_0} |x\rangle \otimes |y_0\rangle$$

which gives pre-image upon discarding the second register. The size M is given by N/a, holds because function is periodic and will divide N exactly.

$$|f^{-1}(y_0)\rangle = \frac{1}{\sqrt{m}} \left(|x_0\rangle + |x_0 + a\rangle + \cdots + |x_0 + (m-1)a\rangle\right)$$

Step 2: Follow the above by Fourier sampling -

$$F_N |f^{-1}(y_0)\rangle = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} F_N |x_0 + ka\rangle = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} \frac{1}{\sqrt{N}} \sum_{y \in Z_N} \omega^{x_0 + ka} |y\rangle$$

$$F_N |f^{-1}(y_0)\rangle = \frac{1}{\sqrt{mN}} \sum_{y \in Z_N} \omega^{x_0 y} \left(\sum_{k=0}^{m-1} \omega^{kay} \right) |y\rangle$$

Computing the geometric sum and simplifying gives -

$$F_N |f^{-1}(y_0)\rangle = \frac{1}{\sqrt{mN}} \sum_{y \in Z_N} \omega^{x_0 y} \frac{1 - \omega^{May}}{1 - \omega^{ay}} |y\rangle$$

Now $\omega^{May} = \omega^{Ny} = 1$, therefore $S(y) = 0$ (given $\omega^{ay} \neq 1$). Therefore only for those values where $\omega^{ay} = 1$, we get $S(y) \neq 0$ and $|y\rangle$ is observed. This will only occur when $ay = rN$ for some multiple $r = 0, 1, 2, \ldots$
Hence when we fourier sample from the pre-image state, we get outcome $y_1 = r_1 N/a = r_1 M$ for a single query of the oracle. Here N is known, but M is unknown. We repeat above steps a few times to get $y_2, y_3, \ldots, y_k$ such that $y_j = r_j M$. Therefore we need to find the common divisor of all these numbers which gives M(and eventually, a).

$$\gcd(r_1 M, \ldots, r_k M) = \gcd(r_1, r_2, \ldots, r_k) M$$

If we now show $\gcd(r_1, r_2, \ldots, r_k) = 1$ for random numbers $r_k \in \mathbb{Z}_N$, then we would get

$$\gcd(y_1, y_2, \ldots, y_k) = M$$

only for $\mathcal{O}(\log N)$ queries!

Theorem 10.3

Given a function $f : \mathbb{Z}_N \to \mathbb{Z}_N$ such that $f(x) = f(y)$ iff $x = y \mod \tau$ for some unknown τ, there exists a quantum algorithm called period finding algorithm that finds τ with only $Q = \mathcal{O}(\log N)$ queries to the function f.

10.3 Algorithm: Phase Estimation

Consider the quantum state: $|\psi\rangle = 1/\sqrt{N} \sum_{y \in \mathbb{Z}_N} e^{\iota\theta y} |y\rangle$ where $\theta \in \mathbb{R}$ is some "unknown real phase" to be estimated. Define x such that $e^{\iota\theta y} \to e^{2\pi\iota xy/N}$ and state can be written as -

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}_N} \omega^{xy} |y\rangle$$

Applying inverse fourier transform to the state:

$$F_N^{-1} |\psi\rangle = \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}_N} \omega^{xy} F_N^{-1} |y\rangle = \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}_N} \sum_{z \in \mathbb{Z}_N} \omega^{(x-z)y} |z\rangle$$

Again applying the geometric sum in the state, we get

$$\mathcal{S} = \frac{\omega^{(x-z)N} - 1}{\omega^{(x-z)} - 1}$$

Assuming $x - z$ is integer, $\mathcal{S}$ vanishes only when $\omega^{x-z} = 1$ or $x = z$. However, in general, this need not be true.

If x is an integer, then we have $\mathcal{S} = \delta(x - z)$ and $F_N^{-1} |\psi\rangle = |x\rangle$, so value of the unknown θ can be directly read off the register.

When x is not an integer, then we do not perfectly encode x in the registers - but if x is very close to an integer, then the algorithm encodes an integer approximation to x.
Let x_0 be the closest integer to x such that $x = x_0 + \varepsilon$, where ε is some small number. The output state is then given by

$$F_N^{-1} |\psi\rangle = \sum_{z \in \mathbb{Z}_N} e^{i\phi(z)} \sqrt{p(z)} |z\rangle ,$$

for some phase $\phi(z)$ and probability distribution $p(z) = \frac{1}{N^2} |\mathcal{S}(x - z)|^2$.
We claim that for $\varepsilon \ll 1$, $p(z)$ is peaked around x_0 and we obtain θ_0, for θ given by $\theta_0 = \frac{x_0}{N} \cdot 2\pi$.
So, the phase estimation algorithm gives us an approximation to θ with the accuracy given by how far the angle θ is from being of the form $\frac{k}{N} \cdot 2\pi$ for some $k \in \mathbb{Z}_N$.

Phase Estimation for a Unitary

Usage of phase estimation algorithm to estimate eigenvalue of a unitary matrix $\mathcal{U}$.

- **Input:** Access to unitary $\mathcal{U}$ and and one of the eigenstate $|\Phi\rangle$
- **Promise:** The state $|\Phi\rangle$ is an eigenstate of $\mathcal{U}$ with eigenvalue λ
- **Output:** Estimate the eigenvalue λ

We introduce a register which stores information about λ in the phase of the register state, and then perform fourier sampling and read off the estimate.
Since λ is an eigenvalue of unitary $\mathcal{U}$, it is of the form $e^{\iota\Phi}$. Choose a large N, whichh will govern the precision of the estimate. We write $\lambda = \omega^x$ for some real number x between 0 and $N-1$. If we estimate x, we estimate λ.

Apply $H^{\otimes n}$ to $|0\rangle^{\otimes n}$:

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}} |y\rangle$$

Next, apply a controlled-U^{2^j} gate to the j^{th} qubit, with $|\Phi\rangle$ being the target qubit. For instance, for a single qubit:

$$\left(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U^{2^j}\right) \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \otimes |\Phi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \lambda^{2^j}|1\rangle\right) \otimes |\Phi\rangle$$

$$= \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{2^j x}|1\rangle\right) \otimes |Phi\rangle$$

$$= \frac{1}{\sqrt{2}} \sum_{y\in\{0,1\}} \omega^{2^j x y_j} |y_j\rangle \otimes |\Phi\rangle$$

The above can be applied to the state $|\psi\rangle$ using all qubits:

$$|\psi\rangle \otimes |\Phi\rangle \to \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} \omega^{\sum_{j=0}^{N-1} 2^j y_j x} |y\rangle \otimes |\Phi\rangle$$

The term $\sum 2^j y_j$ converts qubit basis to N-level basis for the state. Therefore,

$$|\psi\rangle \otimes |\Phi\rangle \to \frac{1}{\sqrt{2^n}} \sum_{y\in\mathbb{Z}_N} \omega^{xy} |y\rangle \otimes |\Phi\rangle$$

where we have successfully uploaded the eigenvalue λ into the phases of the first register and one can then run the phase estimation algorithm to estimate x and the eigenvalue λ.

Generally, we may not have any specific eigenstate of the unitary $\mathcal{U}$. In such a case, we can start with a target state that can be written on the eigenbasis of U, that is,

$$|\psi\rangle \otimes |\chi\rangle \to |\psi\rangle \otimes \sum_\alpha a_\alpha |\Phi_\alpha\rangle$$

The phase estimation then coherently encodes the estimates of the eigenvalues ($\lambda_\alpha = \omega^{x_\alpha}$) in the first register in the form of $|\psi_\alpha\rangle \otimes |\chi_\alpha\rangle$. So, measurement of the register will give you the estimate of any one of the eigenvalues.