# Lecture 5: Basic quantum algorithms

Rajat Mittal

IIT Kanpur

We will now look at some of the basic and fundamental algorithms in quantum computing. Deutsch-Jozsa and Bernstein-Vazirani were the first few quantum algorithms which showed significant speed-up through quantum computation. These algorithms solved very simple and contrived problems, just to show that quantum computation can do better than the classical computation. We will also study quantum Fourier transform and phase estimation, the basic building blocks of many other quantum algorithms.

Before we begin, let us take a look at the *query model*. Most of the algorithms in the quantum world are query algorithms, or are described as query algorithms. The two best known algorithms, Shor's and Grover's, can both be put in this format.

The query format is different from the standard format because the input is not directly given to the algorithm. Let us assume that we want to compute a function $\{0,1\}^n \to \{0,1\}$. The input $x$ is an $n$ bit string $\{0,1\}^n$, then the algorithm has access to a black-box which on an input $i \in [n]$ outputs the bit $x_i$. This is described by an oracle $O_x$,

$$O_x|i,b\rangle \to |i, b \oplus x_i\rangle.$$

*Exercise 1.* Prove that the oracle $O_x$ is unitary.

*Note 1.* This is in the same spirit as the transformation $(x,y) \to (x, y \oplus f(x))$.

Specifically, to obtain the $i^{th}$ bit $x_i$, we input $|i,0\rangle$ into the oracle and get the output on the second part of the register. The first part of the register is the *address* and the second part is called the *target*. Since our alphabet is $\{0,1\}$, the target will be one qubit.

The dimension of the address part should be enough to specify the index $i$. Most of the time we will choose $n = 2^k$, so the index $i$ will be a $k$ bit string and hence the address part will be $k$ qubits.

Because of the difference between query and standard model, the exponential separation between quantum and classical model do not transfer to exponential separation in the standard model.

Sometimes a different oracle is used for querying the input instead of the one mentioned above. It is called the *phase oracle* (as it puts the phase depending upon the input bit $x_i$),

$$O'_x|i,b\rangle = (-1)^{bx_i}|i,b\rangle.$$

We will show that the query oracle and phase oracle are equivalent and only differ by an application of Hadamard to the target part of the register.

*Exercise 2.* Show that,

$$O_x|i,-\rangle = (-1)^{x_i}|i,-\rangle.$$

*Exercise 3.* Why is this global phase not useless?

*Note 2.* Since these are quantum oracles, we assume that we can query in superposition. In other words, the state of the address part (as well as the target part) can be in superposition.

You can also check that,

$$O_x|i,+\rangle = |i,+\rangle.$$

*Exercise 4.* Using the facts above, show that the oracle $O_x$ and $O'_x$ are equivalent.

There is another possible definition of phase query oracle, instead of $i \in [n]$, consider $i \in \{0,1,2,\cdots,n\}$. Then,

$$O''_x|i\rangle = (-1)^{x_i}|i\rangle,$$

Where we assume $O''_x|0\rangle = |0\rangle$.

*Exercise 5.* Show that if we only take $i \in [n]$ for the above oracle, then we can't distinguish between $x$ and its complement.

We will use either of the three mentioned oracles depending upon the application, remember that all three of them are equivalent.

As mentioned before, most of the time, we will choose $n = 2^k$. Many a times, the input is interpreted as a function $f$ from $\{0,1\}^k$ to $\{0,1\}$. This implies, every index $i \in [n]$ is interpreted as a $k$ bit binary string with $x_i$ being considered as the function value $f(i)$. This interpretation shows another reason why query model is important. Suppose there is a subroutine for $f$ and that subroutine is very expensive. Then, query model translates to the task of computing another function through this subroutine, where we want to minimize the applications of this subroutine.

*Exercise 6.* Convince yourself that this is just the relabelling of the input.

## 1 Deutsch-Jozsa algorithm

The Deutsch-Jozsa problem is, given a string $x \in \{0,1\}^n$ (where $n = 2^k$), find whether

- all $x_i$'s are same,
- or string $x$ is balanced, i.e., exactly half the $x_i$'s are 1 and other half of them are 0.

You are promised that the input falls in one of the cases mentioned above. From the discussion about viewing the input as being the output of a function $f$, this problem is equivalent to determining whether the corresponding function to the input is balanced or constant.

*Note 3.* Deutsch-Jozsa problem is a generalization of Deutsch's problem, covered in the introduction.

We are given the query oracle,

$$O_x|i,b\rangle = (-1)^{bx_i}|i,b\rangle,$$

where $i$ is a $k$ length binary string. So, the state space of the input is $k+1$ qubits.

We can rephrase the Deutsch-Jozsa problem in terms of phases,

- All $x_i$ are same, i.e., $|\sum_i (-1)^{x_i}| = 2^k$.
- String $x$ is balanced, i.e., $|\sum_i (-1)^{x_i}| = 0$.

Before we describe the algorithm, let us take a look at the action of Hadamard on $|i\rangle$. As an assignment, show that,

$$H^{\otimes k}|i\rangle = \frac{1}{\sqrt{2^k}} \sum_j (-1)^{i.j}|j\rangle,$$

where $i.j$ is the bitwise inner product between two binary strings.

*Exercise 7.* What is the inverse of $H^{\otimes n}$?

Abusing the notation $|0\rangle = |00\cdots0\rangle$, then

$$H^{\otimes k}|0\rangle = \frac{1}{\sqrt{2^k}} \sum_j |j\rangle.$$

This also means: if $|\psi\rangle = \sum_j \alpha_j|j\rangle$ then $\langle\psi|H^{\otimes k}|0\rangle = \frac{1}{\sqrt{2^k}} \sum_j \alpha_j$.

In other words, Hadamard spreads the amplitude to every state when applied to $|0\rangle$. It also collects the amplitude to state $|0\rangle$ when applied to a state $|\psi\rangle$.

*Exercise 8.* Before looking at the circuit below, can you guess the Deutsch-Jozsa algorithm?

From the observations about the Hadamard gate, the Deutsch-Jozsa algorithm can be inferred. We start with $|0,1\rangle$, where the first part of the register is a $k$ bit string. Using Hadamard, we can transfer it to equal superposition of all index states,

$$\frac{1}{\sqrt{2^k}}\sum_j |j,1\rangle.$$

Then we apply the phase query oracle,

$$\frac{1}{\sqrt{2^k}}\sum_j (-1)^{x_j}|j,1\rangle.$$

Again, using Hadamard, we can collect the amplitudes on state $|0\rangle$.

*Exercise 9.* Show, if the input is constant then we get state $|0\rangle$, else if the input is balanced then the amplitude on the state $|0\rangle$ is zero.

Measuring in the standard basis, if we get state $|0\rangle$ then algorithm will answer *constant* else it will answer *balanced*.

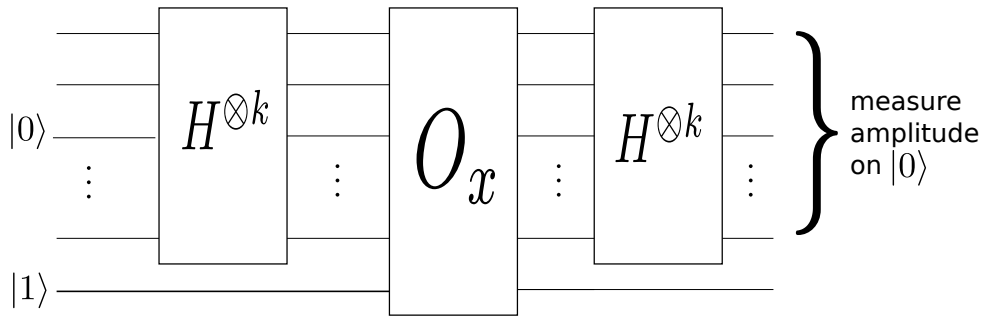


**Fig. 1.** Deutsch's Algorithm

*Exercise 10.* Convince yourself that the circuit given in Fig. 1 works.

Notice that this algorithm is without error. Any deterministic algorithm will require at least $O(n)$ queries of the classical oracle to determine the correct answer, while this algorithm takes only one query to the quantum oracle.

*Exercise 11.* Why will any deterministic algorithm require $O(n)$ queries?

You will show in the assignment: there exists a one sided randomized algorithm with constant number of queries for this problem. So, this algorithm gives a speed up for quantum vs deterministic algorithm model. In the next section, we will see a problem for which quantum algorithm performs better than the randomized setting.

### 1.1 Bernstein-Vazirani algorithm

The Bernstein-Vazirani problem is very similar to the Deutsch-Jozsa problem. Suppose we have a promise on the input $x$, that $x_i = i.a$ for some $a \in \{0,1\}^k$. The Bernstein-Vazirani problem is to find this $a$.

*Note 4.* This is similar to the Deutsch-Jozsa problem because $a = 0$ case corresponds to constant input and other $a$'s correspond to the balanced input.

*Exercise 12.* Show that Deutsch-Jozsa algorithm (exactly the same algorithm) will work in this case.

*Note 5.* After we measure in the standard basis, the output will give us $a$.

So this problem can also be solved by just one query in the quantum setting. To solve this problem even in the bounded error setting, we need at least $O(k)$ queries. This follows from an information theory argument (one query will give information about at most one bit of $k$), but the exact argument is beyond the scope of this course.

## 2 Fourier transform

We will now look at some basic subroutines, which will be used later for constructing more complex quantum algorithms for natural problems. Our first subroutine/algorithm will be a quantum version of the famous classical algorithm called *discrete Fourier transform*. Let us look at the classical algorithm first, then we will describe the quantum version.

You might have heard of *Fourier transform* as the conversion of a function from time domain to frequency domain. It has applications in analysis of differential equations, spectroscopy, quantum mechanics and signal processing.

Instead, we will introduce Fourier transforms as a general tool to analyze functions over Abelian groups (mostly finite). In general, Fourier transform acts on functions over these Abelian groups (e.g., real or complex numbers) and gives representation of those functions in the *character basis*. Readers are encouraged to look at the definitions of character, group theory and Fourier analysis over an Abelian group. The next few paragraphs will use facts from the basics of character theory and can be found in any standard text on group theory.

Given a function $f : G \to \mathbb{C}$ on an *Abelian group* $G$, there is a standard representation of the function as a $\mathbb{C}^{|G|}$ vector.

*Note 6.* If you are confused about a general group, take $G$ to be $\mathbb{Z}_n$, the group of remainders modulo $n$.

*Exercise 13.* What is this representation?

The representation is simply the value of the function on different elements of $G$, i.e., as a vector in $\mathbb{C}^{|G|}$.

Let us consider some more interesting functions of the kind $\chi : G \to \mathbb{C}$. A *character* $\chi$ of $G$ is the non-zero function $\chi : G \to \mathbb{C}$, s.t.,

$$\chi(g_1)\chi(g_2) = \chi(g_1 g_2) \quad \forall g_1, g_2 \in G$$

*Exercise 14.* Can you think of a simple function which is a character?

The function $\chi_0(g) = 1$, for all $g \in G$, is known as the trivial character and is denoted by $\chi_0$.

*Exercise 15.* What is $\chi(e)$, where $e$ is the identity element?

For any element $g$ in a finite group $G$, there exists an $n$ such that $g^n = e$, where $e$ is the identity element. You can show (as an exercise) that $|\chi(g)| = 1$, for any $g$ in $G$.

*Exercise 16.* What are the characters for $\mathbb{Z}_2$?

*Properties of characters:* These properties of characters are well-known. You are encouraged to read any standard text on group representations for proofs (or try them yourselves). Below, $\chi$ will denote a character and $G$ is a finite Abelian group. Observe that a character $\chi$ can be viewed as a vector over complex numbers with $|G|$ length. With an abuse of notation, let $\chi$ also denote the normalized vector corresponding to function $\chi$.

- For a non-trivial character $\chi$, $\sum_{g\in G}\chi(g) = 0$. If $\chi$ is trivial, then this sum is $|G|$.
- The product of two characters is a character.
- Any two distinct characters of $G$ are orthogonal to each other (as vectors in $\mathbb{C}^{|G|}$).
- There are exactly $|G|$ many distinct characters for $G$ (this is harder to prove).
- With appropriate normalization, the characters of $G$ form an orthonormal basis of the space $\mathbb{C}^{|G|}$.

*Note 7.* It is known that the set of characters of $G$ form a group under multiplication, called $\hat{G}$. We already saw that the size of $\hat{G}$ and $G$ is same, not just that, they are actually isomorphic. In other words, we can index characters of $G$ with elements of $G$ (though this indexing is not unique).

Using these properties, we can define *the Fourier transform* over an Abelian group $G$. The Fourier transform is basically the *basis change operator* from the standard basis $e_1, e_2, \cdots, e_{|G|}$ to the *character basis* $\chi_1, \chi_2, \cdots, \chi_{|G|}$.

Any function $f$ in $\mathbb{C}^{|G|}$ can be represented in Fourier basis as,

$$f = \sum_{i=1}^{|G|} \hat{f}(i)\chi_i.$$

The coefficients $\hat{f}(i)$ is called the $i$-th Fourier coefficient of $f$. Taking inner product with $\chi_i$,

$$\hat{f}(i) = \langle \chi_i | f \rangle.$$

*Exercise 17.* Why is the character basis different from any other orthonormal basis?

Hint: Look at the entry-wise multiplication of character basis vectors.

These Fourier coefficients capture properties of our function which might not be evident in the standard representation. For instance, let us look at the first Fourier coefficient, one corresponding to the trivial representation. By definition,

$$\hat{f}(1) = \langle \chi_1 | f \rangle = \frac{1}{\sqrt{|G|}} \sum_x f(x). \tag{1}$$

In other words, first Fourier coefficient tells us about the summation of function value at all points.

*Deutsch Jozsa algorithm as a Fourier transform:* Remember Deutsch's problem, we want to find whether a function $f : \{0,1\}^n \to \{0,1\}$ is balanced or constant. Clearly, sum of all function values when $f$ is constant is either 0 or $2^n$, it is $2^{n-1}$ for the balanced case.

A much better separation can be attained by looking at the function $(-1)^{f(x)}$ (this is similar to moving to range $\{-1,1\}$). If $f$ is balanced then $\sum_x (-1)^{f(x)} = 0$; if $f$ is constant then $\left|\sum_x (-1)^{f(x)}\right| = 2^n$.

So, to solve Deutsch's problem, we do Fourier transform of $(-1)^{f(x)}$ and look at the Fourier coefficient of the trivial representation.

*Exercise 18.* Which Abelian group should we take Fourier transform over?

Given the definition of the function, it is clear that we need Fourier transform over group $\mathbb{Z}_2^n$. To find it, we first find the Fourier transform over $\mathbb{Z}_2$.

Notice that there are two characters of the group $\mathbb{Z}_2$.

$$\chi_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \text{ and } \chi_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}.$$

*Exercise 19.* What is the Fourier transform for $\mathbb{Z}_2$?

The change of basis matrix from standard basis to $\chi_1, \chi_2$ is our old friend, Hadamard matrix. So, the Fourier transform over $\mathbb{Z}_2^n$ is just $H^{\otimes n}$. A rigorous proof of the previous line requires some work, readers are encouraged to do it.

*Exercise 20.* Convince yourself that Deutsch-Jozsa algorithm is basically a Fourier transform over $\mathbb{Z}_2^n$.

### 2.1 Discrete Fourier transform

The *discrete Fourier transform* (DFT) is the Fourier transform over the group $\mathbb{Z}_n$. It is a linear transformation on functions of the type $f : \mathbb{Z}_n \to \mathbb{C}$.

*Exercise 21.* Can you find all characters of the group $\mathbb{Z}_n$?

Remember, this function $f$ can be represented by a vector $x \in \mathbb{C}^n$. The DFT is given by the DFT matrix $F$ ($n \times n$ matrix),

$$F_{jk} = \omega^{jk}.$$

Here $\omega = e^{2\pi i/n}$ is the $n^{th}$ root of unity and $j, k$ range from 0 to $n-1$. So the matrix looks like,

$$F_n = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^n \\ 1 & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix}$$

If the co-ordinates of $x$ are $x_0, x_1, \cdots, x_{n-1}$ where $x_i = f(i)$. The DFT outputs $n$ complex numbers $y_0, y_1, \cdots, y_{n-1}$, s.t.,

$$y_k = \frac{1}{\sqrt{n}} \sum_{0 \le j \le n-1} \omega^{jk} x_j.$$

*Exercise 22.* Show that $F_n$ is a unitary matrix. What is the inverse of $F_n$?

Another important operation associated with Fourier transform is called *convolution*. Given two vectors $a, b$ of dimension $n$, their convolution is a vector of dimension $n$,

$$(a * b)_l = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} a_j b_{(l-j) \bmod (n)}, \ 0 \le l \le n-1.$$

*Exercise 23.* Suppose the Fourier transform of vector $a$ is denoted by $\hat{a}$. Prove,

$$\widehat{(a * b)}_l = \hat{a}_l \hat{b}_l.$$

### 2.2 Extra reading: fast Fourier transform

*Exercise 24.* How many additions and multiplications are required for DFT?

There are $n$ entries in the resulting vector. To compute every entry, we need $O(n)$ additions and multiplications. So the obvious computation will require $O(n^2)$ steps. The fast Fourier transform (FFT) accomplishes the task in $O(n \log n)$ steps.

Suppose $n = 2^k$, the main idea of FFT is,

$$
\begin{aligned}
y_k &= \frac{1}{\sqrt{n}} \sum_{0 \le j \le n-1} \omega^{jk} x_j \\
&= \frac{1}{\sqrt{n}} \left( \sum_{\text{even} j} \omega^{jk} x_j + \sum_{\text{odd} j} \omega^{jk} x_j \right) \\
&= \frac{1}{\sqrt{n}} \left( \sum_{\text{even} j} \omega_{n/2}^{kj/2} x_j + \omega^k \sum_{\text{odd} j} \omega_{n/2}^{k(j-1)/2} x_j \right).
\end{aligned} \tag{2}
$$

Here $\omega_{n/2}$ is the $(n/2)$-th root of unity. Notice that the terms in the parenthesis are the Fourier transform $\mathbb{Z}_{n/2}$.

The algorithm for FFT will compute the Fourier transform of even entries of $x$, say $y_e$. We can similarly define $y_o$, the second term in the summation. Then we need to copy the entries twice, so that, $y_e$ and $y_o$ become vectors of dimension $n$. That is done by setting $y_{e,k+n/2} = y_{e,k}$ and similarly setting $y_{o,k+n/2} = -y_{o,k}$.

*Exercise 25.* Show that $y$, the Fourier transform of $x$ is equal to $y = y_e + y_o$.

To analyze the complexity of this algorithm. We need to compute two Fourier transforms on $n/2$ (one each for $y_e$ and $y_o$) and then $O(n)$ more operations to add them up. Let $A(n)$ be the complexity of FFT on $\mathbb{Z}_n$, the recursion for the complexity becomes, $A(n) = 2A(n/2) + O(n)$.

*Exercise 26.* Show that the FFT algorithm works in time $O(n \log n)$.

### 2.3 Quantum discrete Fourier transform

We will show a circuit for quantum discrete Fourier transform using 1 qubit and controlled unitaries on 1 qubit. Since any 1 qubit or controlled unitaries on 1 qubit can be performed efficiently using any universal gate set (Solovay-Kitaev theorem), this circuit is efficient in the usual sense.

The classical discrete Fourier transform takes a vector $x$ to its image $y$. Suppose we consider $x$ as a quantum state $|x\rangle$ (co-ordinates being the amplitudes), then the *quantum Fourier transform (QFT)* takes this state to $|y\rangle$ (similar to quantum Fourier transform over $\mathbb{Z}_2^n$).

To specify its action in terms of the basis $|0\rangle, |1\rangle, \cdots, |n-1\rangle$,

$$
|j\rangle \to \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \omega^{jk} |k\rangle,
$$

where $\omega = e^{2\pi i/n}$ is the $n$-th root of unity.

*Exercise 27.* Show that it is a valid quantum operation.

*Exercise 28.* Will implementing QFT imply that we can do DFT?

It turns out that QFT can be done in polylog$(n)$ operations, much better than even FFT. Though, in this case, we don't have direct access to the amplitudes of the state. So QFT directly does not imply that we can do DFT. They are definitely related and we will show some surprising results using QFT in later lectures (period finding, Shor's algorithm).

*Exercise 29.* Is the Fourier transform matrix Hermitian? What is its inverse?

Again, we assume $n = 2^k$ for simplicity. So, the basis states are $k$ bit strings $|j\rangle = |j_1 j_2 \cdots j_k\rangle$. QFT has a very useful *product* representation, it allows us to implement it.

$$
\begin{aligned}
F_{2^k}|j_1 j_2 \cdots j_k\rangle &= F_{2^k}|j\rangle \\
&= \frac{1}{2^{k/2}} \sum_{l=0}^{2^k-1} \omega^{jl}|l\rangle \\
&= \frac{1}{2^{k/2}} \sum_{l_1} \sum_{l_2} \cdots \sum_{l_k} \omega^{j(\sum_{h=1}^{k} l_h 2^{k-h})}|l_1 l_2 \cdots l_k\rangle \\
&= \frac{1}{2^{k/2}} \sum_{l_1} \sum_{l_2} \cdots \sum_{l_k} \bigotimes_{h=1}^{k} e^{2\pi i j l_h 2^{-h}}|l_h\rangle \\
&= \frac{1}{2^{k/2}} \bigotimes_{h=1}^{k} \left( \sum_{l_h} e^{2\pi i j l_h 2^{-h}}|l_h\rangle \right) \\
&= \frac{1}{2^{k/2}} \bigotimes_{h=1}^{k} \left( |0\rangle + e^{2\pi i j 2^{-h}}|1\rangle \right)
\end{aligned}
\tag{3}
$$

Hence we can say,

$$
|j_1 j_2 \cdots j_k\rangle \xrightarrow{F_{2^k}} \frac{1}{2^{k/2}}(|0\rangle + e^{2\pi i 0.j_k}|1\rangle)(|0\rangle + e^{2\pi i 0.j_{k-1}j_k}|1\rangle) \cdots (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_k}|1\rangle).
$$

*Note 8.* $0.j_1 j_2 \cdots j_k$ means the expression $\sum_{l=1}^{k} j_l 2^{-l}$, as in the usual binary notation.

*Exercise 30.* Convince yourself that the above product representation is correct.

Using this product representation and the elementary gates,

$$
R_l = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^l} \end{pmatrix},
$$

we can derive the circuit for quantum Fourier transform.

*Exercise 31.* What are the gates $R_1, R_2, R_3$?

Consider the first qubit of the transformed state, $(|0\rangle + e^{2\pi i 0.j_k}|1\rangle)$. If $j_k$ is 0 then we get $|+\rangle$ state, else $|-\rangle$ state. That means, just apply a Hadamard to $j_k$ to get this qubit.

*Exercise 32.* Show that a Hadamard on $j_k$ will produce the desired state.

Similarly to produce $(|0\rangle + e^{2\pi i 0.j_{k-1}j_k}|1\rangle)$, we can apply a Hadamard to $j_{k-1}$ and then a controlled $R_2$ on $j_{k-1}$ with control bit $j_k$. Continuing this process will give us the required circuit. The only thing is that the order of qubits are flipped. We can use the SWAP operation (swap the two qubits) to get the correct order.

*Exercise 33.* Show that the circuit 2 gives QFT for $k = 3$.

*Note 9.* The operations on $(k-1)$-th qubit needs to be applied before the operations on $k$-th qubit.

It is easy to swap qubits using the CNOT operation. You will show in the assignment that swap can be performed using CNOT.

*Exercise 34.* Give the QFT circuit for general $k$.

*Exercise 35.* What is the number of operations in our implementation of QFT in terms of $n$?

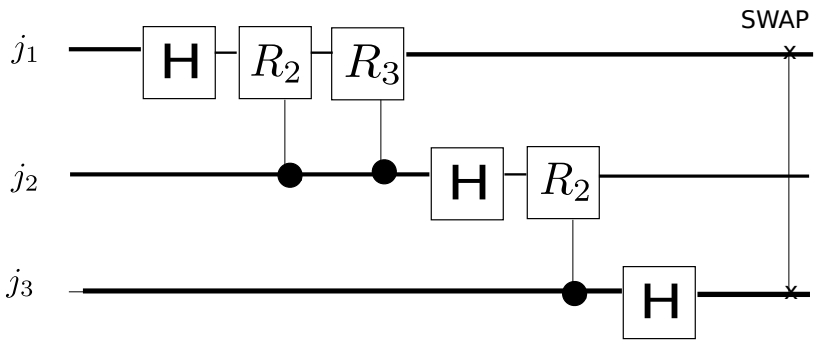


**Fig. 2.** QFT for $k = 3$

## 3 Phase estimation

Another subroutine, very useful in quantum computing, is known as *phase estimation*. Given a unitary $U$, we know that all its eigenvalues have norm 1. In other words, all the eigenvalues are of the form $e^{2\pi i\theta}$. To determine the eigenvalue, it is enough to find $\theta$, called *the phase of the eigenvalue*.

The phase estimation subroutine, given a unitary $U$ and its eigenvector $|u\rangle$, finds the phase of the eigenvalue corresponding to the eigenvector $|u\rangle$.

To be precise, we need the eigenvector $|u\rangle$ and the ability to perform controlled $U^{2^k}$ operations to determine the phase.

For simplicity, let us assume that $\theta = 0.j_1j_2\cdots j_k$.

*Exercise 36.* Why can we assume that $\theta \leq 1$?

Also, assume that we have the ability to perform $U^l$ for all $l \leq 2^k = n$ (instead of just controlled $U^{2^k}$).

We will start with the state $|0, u\rangle$, where the first part of the register holds $k$ qubits and second register holds the eigenvector $|u\rangle$. Then we will apply Hadamard on the first part and obtain,

$$\frac{1}{2^{k/2}} \sum_{l=1}^{2^k} |l, u\rangle.$$

*Exercise 37.* What other operation can we use instead of applying Hadamard?

Now we can perform the operation $|l, u\rangle \to |l\rangle U^l|u\rangle$. Notice that this can be done classically on the basis states and hence can be done quantumly.

This gives us the state,

$$\frac{1}{2^{k/2}} \sum_{l=1}^{2^k} |l\rangle U^l|u\rangle = \frac{1}{2^{k/2}} \sum_{l=1}^{2^k} e^{2\pi i\theta l}|l, u\rangle.$$

*Exercise 38.* What can be done to recover $\theta$ now?

Some thought shows that the first part of the register is the Fourier transform of $2^k\theta$. Hence applying inverse Fourier transform, we get the state $|2^k\theta\rangle$.

If we are only given the controlled versions of $U^{2^l}$ where $l \leq k$, then how can we achieve the same phase estimation? Notice that $l$ now varies only up to $k$. Essentially, we are given the power to apply $U, U^2, U^4 ..., U^{2^k}$.

The simple idea is to break any integer $0 \leq h \leq 2^k$ as powers of 2. Then using the controlled version, we can apply $U^h$. The following circuit has been taken from [1].

*Exercise 39.* Show that the following circuit (Fig. 3) works.

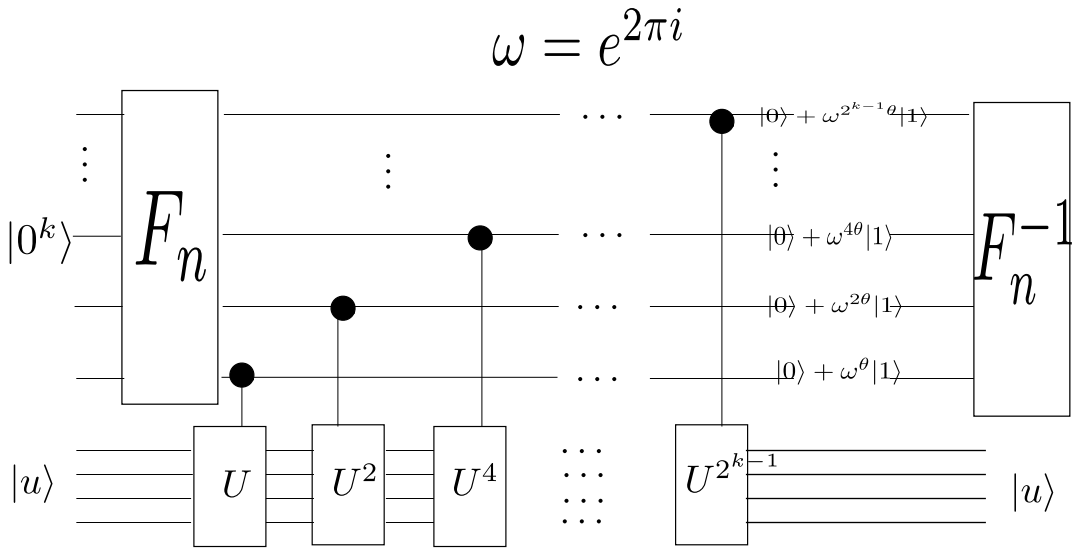


**Fig. 3.** Phase estimation

Most of the time, it is not possible to know the number of digits in the binary expansion of $\theta$ beforehand. What can be done in this case?

If we want to approximate $\theta$ up to $k$ bits of accuracy, using the same circuit with $k + f(\epsilon)$ qubits instead of $k$ qubits will give us the answer with probability $1 - \epsilon$. Here, $\epsilon$ should be treated as a parameter and $f(\epsilon)$ is some function of $\epsilon$. The details can be found in Nielsen and Chuang [1].

Suppose we don't have the eigenvector $|u\rangle$. If the same procedure is done over $|\psi\rangle = \sum_i \alpha_i |u_i\rangle$, we will get the phase corresponding to $|u_i\rangle$ with probability $|\alpha_i|^2$.

*Exercise 40.* Prove the above assertion.

## 4 Assignment

*Exercise 41.* Show that the action of Hadamard is,

$$H^{\otimes k}|i\rangle = \frac{1}{\sqrt{2^k}} \sum_j (-1)^{i.j} |j\rangle,$$

*Exercise 42.* Give a constant query randomized one sided error algorithm for Deutsch-Jozsa problem.

*Exercise 43.* Read about characters and groups.

*Exercise 44.* For any element $g$ in a finite group $G$, there exists an $n$ such that $g^n = e$, where $e$ is the identity element. Show that $|\chi(g)| = 1$ for any $g$ in $G$.

*Exercise 45.* What are the characters of $\mathbb{Z}_n$.

*Exercise 46.* Show that polynomial multiplication $p(x)q(x)$, where both are polynomials of degree $d$, can be done in $O(d \log d)$ operations.

Hint: Use FFT and convolution

*Exercise 47.* Give a circuit to perform inverse Fourier transform.

*Exercise 48.* Show that SWAP can be performed using CNOT.

*Exercise 49.* Read about the approximate phase estimation from [1].

*Exercise 50.* Is there a difference between $H^{\otimes k}$ and QFT on $k$ qubits ($2^k$ dimensional vector)?

## References


1. M. A. Nielsen and I. L. Chuang. Quantum computation and quantum information. *Cambridge*, 2010.
2. S. Arora and B. Barak. Computational Complexity: A modern approach. *Cambridge*, 2009.
3. R. Lidl and H. Niederreiter. Finite Fields. *Cambridge University Press*, 1997.
4. B. Kleinberg. Course notes: Introduction to algorithms. *http://www.cs.cornell.edu/courses/cs4820/2010sp/handouts/MillerRabin.pd* 2010.
5. D. R. Simon. On the power of quantum computation. *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on: 116123*, 1994.
6. A. Childs. Course notes: Quantum algorithms. *https://www.cs.umd.edu/ amchilds/qa/qa.pdf*, 2013.
7. P. Kurur. Survey: Quantum error correcting codes: An introduction. *https://www.cse.iitk.ac.in/users/ppk/research/publication/Surv 02-15-Quantum-codes.pdf*, 2005.