Lecture 8: Basic quantum algorithms

Rajat Mittal

IIT Kanpur

We will now look at some of the basic and fundamental algorithms in quantum computing. We have already seen Deutsch-Jozsa, a small modification gave us Bernstein-Vazirani; they 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.

In last 30 years, there has been multiple quantum algorithms which provide a speed up for natural computational problems (like Grover's and Shor's algorithm). To understand these landmark results, we first need to study quantum Fourier transform and phase estimation, the basic building blocks of many quantum algorithms.

1 Fourier transform

Our first subroutine/algorithm will be a quantum version of the famous classical algorithm called *discrete Fourier transform*. To see the quantum version, we need to first learn: what is Fourier transform, and then see the classical algorithm for it.

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).

Note 1. Remember that an Abelian group is a commutative group, i.e., $\forall g_1, g_2 \in G : g_1 g_2 = g_2 g_1$.

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.

Characters:

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 (as a vector of complex numbers with length $|G|$).

Note 2. If you are confused about a general group, take G to be $\mathbb{Z}_n$, the group of remainders modulo n.

Exercise 1. 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 special functions of this kind. A *character* χ of G is a function $\chi : G \to \mathbb{C} - \{0\}$, s.t.,

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

Note 3. We remove 0 from $\mathbb{C}$ to make it a group under multiplication.

Exercise 2. 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 χ_0.

Exercise 3. Show that $\chi(e) = 1$, where e is the identity element?

For any element g in a finite group G, there exists an $n \leq |G|$ 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 4. 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, χ will denote a character and G is a finite Abelian group.

Observe that a character χ can be viewed as a vector over complex numbers with length $|G|$.

- For a non-trivial character χ, $\sum_{g \in G} \chi(g) = 0$. If χ 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.
- With appropriate normalization, the characters of G form an orthonormal basis of the space $\mathbb{C}^{|G|}$.

Note 4. 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 normalized *character basis* $\chi_1, \chi_2, \cdots, \chi_{|G|}$. In other words, Fourier transform matrix is the unitary matrix which takes the standard basis $e_1, e_2, \cdots, e_{|G|}$ to the normalized *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 χ_i,

$$\hat{f}(i) = \frac{1}{n} \sum_{g} \chi_i * (g) f(g) := \langle \chi_i | f \rangle.$$

Note 5. We have defined $\langle f | \chi_i \rangle$ slightly differently, with a normalization. This makes χ_i's orthonormal.

Exercise 5. 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,

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

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 condition can be obtained 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 6. 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 $\mathbb{Z}_2$.

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

To get the Fourier transform over $\mathbb{Z}_2$, we put χ_i's in columns and multiply by an appropriate constant to normalize.

Exercise 7. What is the Fourier transform for $\mathbb{Z}_2$?

The change of basis matrix from standard basis to χ_1, χ_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 8. Convince yourself that Deutsch-Jozsa algorithm is basically a Fourier transform over $\mathbb{Z}_2^n$.

1.1 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 9. Can you find all characters of the group $\mathbb{Z}_n$? Hint: use the fact that $\mathbb{Z}_n$ is cyclic.

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,

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

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 10. Show that F_n is a unitary matrix. What is the inverse of F_n?

1.2 Extra reading: fast Fourier transform

Exercise 11. 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 12. Show that y, the Fourier transform of x is equal to $y_e + y_o$.

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

1.3 Quantum discrete Fourier transform

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 $\mathbb{Z}_2^n$).

We will show a circuit for quantum discrete Fourier transform using 1 qubit gates 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 [2]), this circuit is efficient in the usual sense.

To specify the action of the Fourier transform 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 14. Show that it is a valid quantum operation.

Exercise 15. Will implementing QFT imply that we can do DFT?

It turns out that QFT can be done in poly-log(n) operations, much better than even FFT. Though, in this case, we don't have direct access to the amplitudes of the state. So, having an algorithm for QFT does not imply that we have a fast algorithm for DFT. The output required in the two problems is different. They are definitely related and we will show some surprising results using QFT in later lectures (phase estimation, Shor's algorithm).

Exercise 16. Is the Fourier transform matrix Hermitian?

Again, we assume $n = 2^k$ for simplicity (for other n's, the algorithm is slightly different, refer [1]). 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 (the representation gives us the intuition to find the circuit for QFT).

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

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

Exercise 17. 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 18. 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 19. Show that a Hadamard on j_k will produce the desired state.

Similarly the second qubit, $(|0\rangle + e^{2\pi i 0.j_{k-1}j_k}|1\rangle)$, has a relative phase of $(-1)^{j_{k-1}}$ multiplied by i^{j_k}. 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} (to get a relative phase of $(-1)^{j_{k-1}}$), and then a controlled R_2 on j_{k-1} with control bit j_k (to get a relative phase of i^{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 20. Show that the circuit 1 gives QFT for $k = 3$.

Note 7. 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. Have you seen it earlier? Try to construct a circuit for swapping qubits.

Exercise 21. Give the QFT circuit for general k.

Exercise 22. What is the number of operations in our implementation of QFT in terms of n?

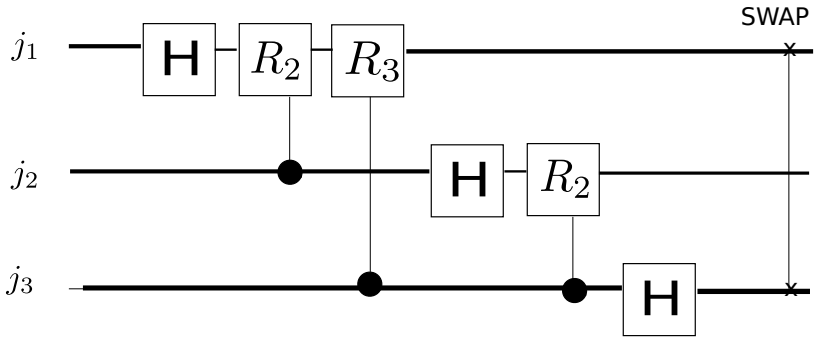

Fig. 1. QFT for $k = 3$

2 Phase estimation

Another very useful subroutine, in quantum computing, is known as *phase estimation*. Given a unitary U ($UU^* = I$), we know that all its eigenvalues have norm 1. Since any complex number can be written as $re^{2\pi i\theta}$, all eigenvalues of U should be of the form $e^{2\pi i\theta}$ for some θ. To determine the eigenvalue, it is enough to find this θ, called *the phase of the eigenvalue* or *eigenphase*.

We will start with a few basic assumptions so that the idea of phase estimation is clear; later we will see how to take care of them.

- Assume that $\theta = 0.j_1j_2\cdots j_k$ in the binary representation.

  Exercise 23. Why can we assume that $\theta \leq 1$?

- We are given the eigenvector as a quantum state $|u\rangle$.

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, the algorithm will take the eigenvector $|u\rangle$ as input, and it needs the ability to perform controlled $U^{2^i}(i \leq k)$ operations; using those, it determines the corresponding eigenphase.

To start with, we will 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}). Later we will show that controlled $U^{2^i}(i \leq k)$ operators can be used to perform U^l for all $l \leq 2^k$.

Exercise 24. Can you think of a way to do it?

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 25. What other operation can we use instead of applying Hadamard?

Now we can perform the operation $|l, u\rangle \rightarrow |l\rangle U^l |u\rangle$. 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 26. What can be done to recover θ 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 [2].

Exercise 27. Show that the following circuit (Fig. 2) works.

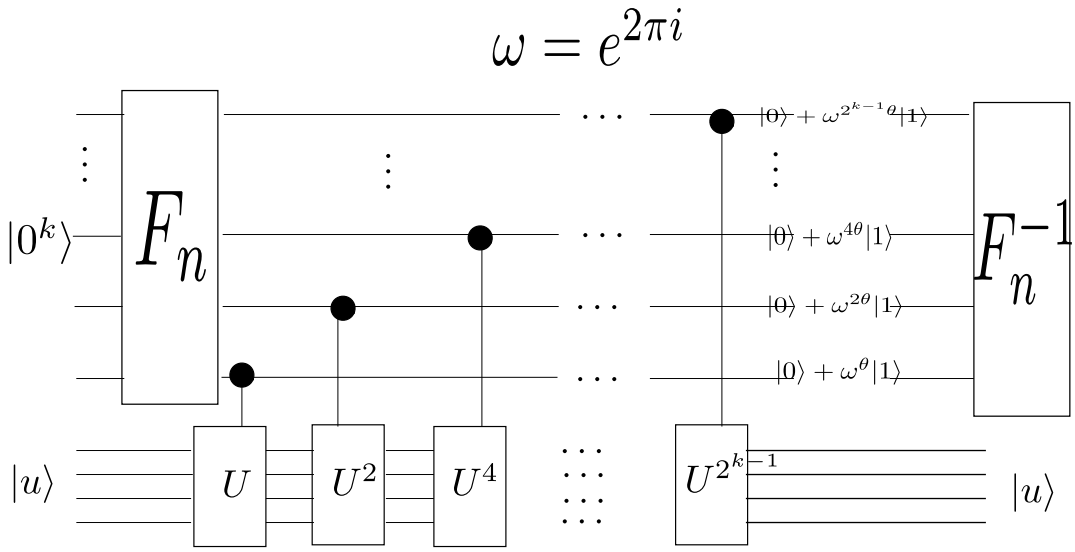

Fig. 2. Phase estimation

Let us see how to take care of the assumptions we made, there are only k bits in the expansion of θ and we have the eigenvector as a quantum state $|u\rangle$.

Most of the time, it is not possible to know the number of digits in the binary expansion of θ beforehand. What can be done in this case? If we want to approximate θ 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, ϵ should be treated as a parameter and $f(\epsilon)$ is some function of ϵ. The details can be found in Nielsen and Chuang [2].

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 28. Prove the above assertion.

Exercise 29. What will be the output of phase estimation on Z gate with $|u\rangle = |+\rangle$ state?

3 Assignment

Exercise 30. 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 31. Read about characters and groups.

Exercise 32. 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 33. What are the characters of $\mathbb{Z}_n$.

Exercise 34. Give a circuit to perform inverse Fourier transform.

Exercise 35. Read about the approximate phase estimation from [2].

Exercise 36. Is there a difference between $H^{\otimes k}$ and QFT on k qubits (2^k dimensional vector)?

Exercise 37. Read about the complexity class BQP.

References

1. A. Childs. Quantum algorithms, 2013.
2. M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge, 2010.