IAS/Park City Mathematics Series
Volume 3, 1993

Quantum Computation

Ran Raz

LECTURE 3
Quantum circuits and Shor's factoring algorithm

In this lecture, we will present Shor's quantum polynomial time algorithm for factoring integers. The algorithm gets as an input an integer N and outputs all its factors. Since we believe that there is no classical polynomial time algorithm for factoring, Shor's result gives a strong indication that classical computers cannot simulate quantum systems, and that quantum computers, if ever built, will be more powerful than classical ones.

3.1. Quantum Circuit Complexity

Suppose that we have a quantum register of n qubits, which is a part of a quantum computational device. Denote by $H = \mathbb{C}^{2^n}$ the space of quantum states of the register. What are the operations that can be applied on the register ? In general, every unitary transformation $U : H \to H$ may be possible. However, most unitary transformations are hard even to describe and are very unlikely to be implemented in a real computational device. If one could implement every unitary transformation then any computation would have been made trivial (by applying one unitary matrix that performs that computation). Obviously, we need to measure in some way the complexity of a unitary transformation.

Thus, we don't want to assume that we are able to apply an arbitrary unitary transformation. It is usually assumed that one is able to apply transformations from some small set, referred to as, the set of *quantum gates*. The common assumption is

[1]Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel.
E-mail address: `ranraz@wisdom.weizmann.ac.il`.

July 21, 2000

Based on lecture notes taken by Yoad Lustig.

that it may be easier to implement transformations that operate on a small number of qubits (say, 1 or 2 qubits). Such transformations are usually referred to as, *local transformations*, or, *local matrices*. Here, we will assume that we are able to apply any local transformation, that is, any unitary transformation that operates on one or two qubits.

More formally, consider partitions of the n qubits into a register A of one or two qubits and a register B of all the other qubits. Given such a partition, we have

$$H = H_A \otimes H_B,$$

where H_A is the space of states of A and H_B is the space of states of B. We say that a unitary matrix $U : H \to H$ is a *local matrix* if for some A, B (such that A is of length one or two),

$$U = U_A \otimes I_B,$$

where $U_A : H_A \to H_A$ is an arbitrary unitary matrix and $I_B : H_B \to H_B$ is the identity matrix.

We will assume that in a quantum circuit we can use any local matrix. That is, the matrix U_A above can be arbitrary. It can be argued that since there are infinitely many such matrices, it is probably not possible to implement all of them. However, it turned out that there is a finite set of unitary matrices (operating on one or two qubits), such that every local matrix can be approximated (up to any $\epsilon > 0$) by a product of local matrices with U_A taken from that finite set, (and the number of matrices in the product is polynomial in n and poly-logarithmic in ϵ^{-1}).

A *quantum circuit* is just a sequential application of local matrices, as described in Figure 1. We say that the circuit computes U if the product of all these local

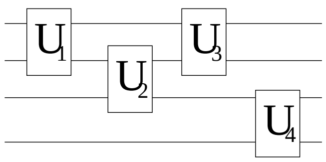

Figure 1. A quantum circuit

matrices is U. We would like to define the quantum circuit complexity of a unitary matrix U as the minimal number k, such that U can be represented as a product of k local matrices. This is analogous to the definition of the classical circuit complexity of a function. There are, however, two minor complications.

The first complication is that for the computation of U we would like to allow the circuit to use a "working space", that is, an additional quantum register that can be used as a memory device. We must demand that at the end of the computation the additional register is not entangled to the original one (otherwise, as we have seen, the interference may be destroyed). For simplicity, we just demand that the additional register is not effected at all by the computation. Formally, we define the quantum circuit complexity of a unitary matrix U as the minimal number k, such that for some register W the transformation $U \otimes I_W$ can be represented as a product of k local matrices. We say that a quantum circuit computes U if for some W the circuit computes $U \otimes I_W$.

The second minor complication (that we usually ignore) is that we allow the circuit to compute an approximation of $U \otimes I_W$. That is, we are interested in the minimal number k, such that for some register W the transformation $U \otimes I_W$ can be *approximated* as a product of k local matrices. We say that a unitary matrix U is efficiently computable (or, polynomial time computable) if its quantum circuit complexity is polynomial in n.

The measurement in a quantum circuit is usually performed only once, at the end of the computation. That is, the output of the circuit is measured. Hence, the measurement is not really a part of the circuit. It can be proved that this is equivalent to allowing any number of partial measurements at any intermediate stage of the computation. That is, at any point in the circuit some of the qubits may be measured. All the measurements are performed according to the standard base.

3.2. Quantum Simulation of Classical Computations

We would like our quantum circuit to compute Boolean functions

$$f : \{0,1\}^n \rightarrow \{0,1\}^m.$$

However, the circuit computes unitary transformations and not Boolean functions. In particular, the unitary transformation is always reversible, whereas the function f is not necessarily reversible. We hence define for every Boolean function f : $\{0,1\}^n \rightarrow \{0,1\}^m$, a corresponding unitary matrix

$$U_f : \mathbb{C}^{2^n} \otimes \mathbb{C}^{2^m} \rightarrow \mathbb{C}^{2^n} \otimes \mathbb{C}^{2^m}.$$

We think of U_f as a transformation that operates on two registers. A register A of length n and a register B of length m. The transformation U_f is defined by

$$U_f(|i\rangle_A \otimes |j\rangle_B) = |i\rangle_A \otimes |f(i) \oplus j\rangle_B$$

where $f(i) \oplus j$ denotes the bitwise xor of $f(i)$ and j (recall that since U_f is linear, it is enough to define its operation for base elements). That is, the matrix U_f xors the value $f(i)$ to register B (see Figure 2). This is a reversible operation, since from

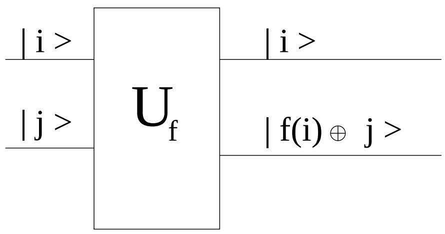

Figure 2. The matrix U_f

the pair $(i, f(i) \oplus j)$ one can obtain the pair (i, j). Hence, it is not hard to verify that U_f is unitary. Note that

$$U_f(|i\rangle_A \otimes |0\rangle_B) = |i\rangle_A \otimes |f(i)\rangle_B.$$

Thus, if we apply U_f on the state $|i\rangle_A|0\rangle_B$ and we measure register B we obtain $f(i)$. Hence, we think of a quantum circuit for U_f as a quantum circuit for the function f.

Quantum operations are unitary and hence they are always reversible. Classical operations and gates are not necessarily reversible. Hence, it is not clear in advance that quantum circuits can simulate classical Boolean circuits. Nevertheless, such a simulation does exist.

Theorem 3.1. *If $f : \{0,1\}^n \to \{0,1\}^m$ is computable by a (classical) Boolean circuit of size s then U_f is computable by a quantum circuit of size poly(s).*

In particular, if f is computable by a polynomial size classical Boolean circuit then U_f is computable by a polynomial size quantum circuit. Hence, quantum circuits are at least as strong as classical ones. We will not give here the proof of the theorem.

3.3. Quantum Parallelization

Suppose that for some function $f : \{0,1\}^n \to \{0,1\}^m$, we have an efficient quantum circuit for the matrix U_f. That is, we have an efficient quantum circuit for the transformation

$$U_f(|i\rangle_A|j\rangle_B) = |i\rangle_A|f(i) \oplus j\rangle_B.$$

Let

$$|x\rangle_A = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle_A.$$

Then

$$U_f(|x\rangle_A|j\rangle_B) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle_A|f(i) \oplus j\rangle_B,$$

and in particular

$$U_f(|x\rangle_A|0\rangle_B) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle_A|f(i)\rangle_B.$$

Thus, we have obtained a quantum state that contains information about all the values of the function f. This is called *quantum parallelization.* As explained in the previous lecture, we cannot extract from that state all the information about f, because when we measure the state most of the information is lost. Nevertheless, in some cases it is possible to extract interesting information about f that is not easy to obtain classically.

3.4. Quantum Fourier Transform

We have already seen unitary matrices that are efficiently computable by quantum circuits (the matrices U_f for efficiently computable Boolean functions f). However, this computation was just a simulation of classical computation. We would like to show the additional power of quantum circuits. In order to design quantum algorithms that are more efficient than classical ones, one has to show other examples for efficiently computable unitary matrices. The most important unitary matrices that were used for quantum algorithms are the Fourier Transforms over different groups. Here, we will describe the Fourier Transform over the group $GF(2^n)$ which is the one used in Shor's factoring algorithm. This will be another example for a polynomial time computable unitary matrix.

Let

$$\omega = e^{\frac{2\pi i}{2^n}}.$$

ω is a unit root of order 2^n, that is,

$$\omega^{2^n} = 1.$$

The matrix FT_n of size $2^n \times 2^n$ is defined by

$$(FT_n)_{a,b} = \frac{1}{\sqrt{2^n}} \cdot \omega^{a \cdot b} = \frac{1}{\sqrt{2^n}} \cdot e^{\frac{2\pi i \cdot a \cdot b}{2^n}}.$$

For example, for $n = 2$,

$$FT_n = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{pmatrix}$$

(where $\omega = e^{2\pi i/4} = i$).

It turns out that the unitary matrix FT_n is efficiently computable by a quantum circuit. The proof is based on the classical Fast Fourier Transform (FFT) algorithm. It turns out that the FFT algorithm can be viewed as an efficient quantum circuit for FT_n, and the proof is not hard. This is called Quantum Fourier Transform (QFT). Note that the FFT algorithm and the QFT algorithm do different things. The (classical) FFT algorithm gets as an input a vector of 2^n elements and outputs the product of that vector with the matrix FT_n (i.e., the output is also of 2^n elements). The running time of this algorithm is $2^n \cdot poly(n)$. In contrast, the (quantum) QFT algorithm gets as an input a quantum state $|x\rangle$ on n qubits and outputs the quantum state $FT_n|x\rangle$. The running time of this algorithm is $poly(n)$.

3.5. Finding the Period of a Vector

We will now show one application of Quantum Fourier Transform: finding the period of a periodical state vector. This is used in Shor's factoring algorithm. Suppose that we have the following quantum state on n qubits:

$$|x\rangle = \sum_{j=0}^{M-1} |j \cdot r + s\rangle,$$

where

$$M = \left\lfloor \frac{2^n}{r} \right\rfloor$$

and r, s are two integers, such that $s < r$ (r is called the period of the vector and s is called the shift of the vector). That is, $|x\rangle$ is the transpose of

$$(\underbrace{0, \ldots, 0}_{s}, \underbrace{1, 0, \ldots, 0}_{r}, \underbrace{1, 0, \ldots, 0}_{r}, 1, 0, \ldots)$$

For example, for $s = 3, r = 4$,

$$|x\rangle = |3\rangle + |7\rangle + |11\rangle + |15\rangle + \cdots$$

Assume that r is much smaller than 2^n (say, $r^2 < 2^n$) and suppose that somehow we are able to produce many such vectors $|x\rangle$, each one of them with the same period r (but not necessarily with the same shift s). Suppose that we don't know r and we would like to find it.

To find the period r, we will use the Fourier Transform matrix FT_n. A well known property of FT_n is that the vector $FT_n|x\rangle$ has high coefficients in coordinates b such that

$$r \cdot b \approx 0 \mod 2^n,$$

and coefficient close to 0 in other coordinates. Formally, it can be proved that if we measure the state vector $FT_n|x\rangle$ we obtain with high probability (say, with probability larger than 0.1) a result b, such that for some m co-prime to r,

$$-\frac{r}{2} \;\leq\; r \cdot b - m \cdot 2^n \;\leq\; \frac{r}{2},$$

and hence,

$$\left|\frac{m}{r} - \frac{b}{2^n}\right| \;\leq\; \frac{1}{2 \cdot 2^n}.$$

Thus, the number $\frac{m}{r}$ is a very good (rational) approximation for the number $\frac{b}{2^n}$, with a much smaller denominator (recall that r is much smaller than 2^n). The numbers m, r are not known, but the numbers b and 2^n are both known (n is the number of qubits and b is the result of the measurement). There is a (classical) well known and very efficient algorithm for finding all the very good rational approximations of a given number. The number $\frac{m}{r}$ can be found by applying that algorithm on the number $\frac{b}{2^n}$. More formally, all the very good rational approximations of $\frac{b}{2^n}$ can be found by presenting the number $\frac{b}{2^n}$ as a continuous fraction

$$\frac{b}{2^n} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots}}}$$

If we cut the continuous fraction after k steps we obtain a (rational) number q_k that approximates the number $\frac{b}{2^n}$. It is well known that *all* the very good rational approximations of $\frac{b}{2^n}$ are obtained as one of these numbers q_k. In particular, the number $\frac{m}{r}$ will appear as one of these numbers q_k, and it can be proved that it will appear after time at most polynomial in n (i.e., k is at most polynomial in n). Thus, we didn't obtain the number $\frac{m}{r}$, but we did obtain a polynomial number of candidates for this number, and we know that with high probability $\frac{m}{r}$ is one of these candidates. Hence, we can also obtain a polynomial number of candidates for the period r (recall that with high probability m, r are co-primes and hence if we know $\frac{m}{r}$ we can deduce m and r). We will see that for Shor's factoring algorithm this will be enough. Note that r is indeed one of the obtained candidates with high probability (say, with probability larger than 0.1). To get to probability of success close to 1, we have to repeat the entire process several times.

3.6. Finding Order modulo N

It is well known that the problem of factoring integers is equivalent to the problem of finding the order of an integer y modulo an integer N. The input for the order finding problem are two co-prime integers y, N, and the output should be the minimal integer $r > 0$, such that,

$$y^r = 1 \mod N$$

(there exists such an r, because y, N are co-primes). For example, if $N = 10$ and $y = 3$ then

$$y^1 = 3, \quad y^2 = 9, \quad y^3 = 7, \quad y^4 = 1$$

(where all the equalities are taken modulo 10), and hence $r = 4$ is the right answer.

We will not show here the equivalence between the two problem. The first direction (that an algorithm for factoring gives an algorithm for order finding) is easy and is based on the fact that if the factors of N are known then it is easy to find the order of any element modulo N. The other direction (that an algorithm for order finding gives an algorithm for factoring) is based on the fact that unless $N = p^k$ (for some prime p and integer k), if y is chosen randomly in the range $\{1, ..., N\}$ then with high probability y, N are co-primes and r is even and both $y^{r/2} - 1$ and $y^{r/2} + 1$ have a common factor with N. Both directions of the equivalence are classical and very efficient.

We will present here a polynomial time quantum algorithm for finding the order of y modulo N. We assume w.l.o.g. that $y < N$. We will use as an example the numbers $N = 10$ and $y = 3$. Define

$$n = 10 \log N.$$

We will work with numbers of n bits, and with quantum registers of n qubits. For simplicity, we will work with vectors that are not normalized, that is, we will omit the normalization constants of quantum states.

First, define a function f by

$$f(i) = y^i \mod N.$$

In our example,

$$\begin{array}{rcccccccccccccc} i & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & \dots \\ f(i) & = & 1 & 3 & 9 & 7 & 1 & 3 & 9 & 7 & 1 & 3 & 9 & 7 & \dots \end{array}$$

Since there is a very efficient classical Boolean circuit for f, there is a very efficient quantum circuit for the unitary matrix U_f. Recall that the matrix U_f is defined by

$$U_f(|i\rangle_A |j\rangle_B) = |i\rangle_A |f(i) \oplus j\rangle_B$$

(as described in Figure 2).

Let

$$|x\rangle_A = \sum_{i=0}^{2^n-1} |i\rangle_A$$

(where we omitted the normalization constant $1/\sqrt{2^n}$). The state $|x\rangle_A$ can be obtained by applying the matrix FT_n on the state $|0\rangle_A$ (or by several other methods). Apply the matrix U_f on the state $|x\rangle_A|0\rangle_B$ to obtain

$$U_f(|x\rangle_A|0\rangle_B) = \sum_{i=0}^{2^n-1} |i\rangle_A |f(i)\rangle_B.$$

In our example, we obtain the state

$$\begin{array}{lclclclc} |0\rangle_A|1\rangle_B & + & |1\rangle_A|3\rangle_B & + & |2\rangle_A|9\rangle_B & + & |3\rangle_A|7\rangle_B & + \\ |4\rangle_A|1\rangle_B & + & |5\rangle_A|3\rangle_B & + & |6\rangle_A|9\rangle_B & + & |7\rangle_A|7\rangle_B & + \\ |8\rangle_A|1\rangle_B & + & |9\rangle_A|3\rangle_B & + & |10\rangle_A|9\rangle_B & + & |11\rangle_A|7\rangle_B & \dots \end{array}$$

The next step is to measure the second register (i.e., register B). Assume for example that the obtained result is $|7\rangle_B$, then the state collapses to

$$(|3\rangle_A + |7\rangle_A + |11\rangle_A + |15\rangle_A + \ldots) \otimes |7\rangle_B.$$

In general, the state collapses to

$$\sum_{j=0}^{M-1} |j \cdot r + s\rangle_A \otimes |f(s)\rangle_B,$$

where r is the order of y modulo N (i.e., the answer that we are looking for), and $|f(s)\rangle_B$ is the result obtained by the measurement, and

$$M = \left\lfloor \frac{2^n}{r} \right\rfloor.$$

The rest of the algorithm will work with register A only (and will not use register B at all). Since the quantum state (after the collapse) is a tensor product of a state in register A and a state in register B, we can (from now on) for simplicity ignore the register B altogether.

The next step is to apply the unitary matrix FT_n on register A and to measure register A. With high probability (say, with probability larger than 0.1), we obtain a result b, such that for some m (co-prime to r),

$$\left| \frac{m}{r} - \frac{b}{2^n} \right| \leq \frac{1}{2 \cdot 2^n}.$$

We now present $\frac{b}{2^n}$ as a continuous fraction and we obtain a polynomial number of rational numbers q_k. Each one of these rational numbers q_k is an approximation for $\frac{b}{2^n}$ and hence is a candidate for $\frac{m}{r}$. Since (with high probability) m, r are co-primes, we can obtain a polynomial number of candidates for r. For each one of these candidates z, we can check whether or not $y^z = 1 \mod N$ and we take the minimal z that satisfies this equality. With high probability (say, with probability larger than 0.1), this will be the order r that we are looking for. To get to probability of success close to 1, we have to repeat the entire process several times. The output will be the minimal z that was found that satisfies the equation $y^z = 1 \mod N$. With probability close to 1, the output will be the right order r.

Note that in the algorithm we have measured register B but we never used the obtained result. The measurement was performed just in order to collapse the quantum state. In fact, this measurement is not really needed. The measurement makes the algorithm easier to present and to understand, but if we just skipped that step and didn't measure register B and instead we just ignored register B from that point on, the algorithm would do just as well. Moreover, as mentioned before, a general result states that measurements in intermediate stages of a quantum algorithm are not needed, and all measurements can be postponed to the end of the computation.