Quantum Information and Computing
Lecture- 27 : Shor's Factorization Algorithm

Dipan Kumar Ghosh
Physics Department,
Indian Institute of Technology Powai, Mumbai 400076

August 14, 2016

1 Introduction

In the last lecture we discussed the implementation of QFT for the case of one, two and three qubits and provided a way to generalize to the case of n qubits. This was an essential component in understanding Shor's algorithm for integer factorization of a large composite number. However, before we discuss the factorization algorithm, it will be appropriate to make a few comments about the problem of factorization in classical computation.

Consider a number $N = pq$ where p and q are large prime numbers, though for the purpose of illustration in this lecture, we will take these numbers to be small so that a back of the envelop calculation can be done. There are several classical algorithms to do this job though they are not fast enough. The most elementary algorithm is the one due to Euclid which requires of the order of $\sqrt{N}$ operations, as if there exists a factor, one of them has to be less than or equal to $\sqrt{N}$. Euclid algorithm is inefficient for handling large numbers. There are faster classical algorithms, the best among them requiring $\exp\left((\log N)^{1/3}(\log\log N)^{2/3}\right)$ steps, which is still slow. A point which needs to be appreciated is that multiplication of two numbers can be done in polynomial time though the factorization cannot. To get an idea of the difficulty involved consider factorization of a reasonably sized number such as 29803. To factorize this we may use, for instance, Euclid algorithm. If you are manually doing this factorization, you may take a couple of hours doing this. However, if we are told that this number is 229×127, we can check it by doing a multiplication in under a minute. Thus multiplication is easy but factorizaton is hard. It is good to recollect Euclid algorithm, as is taught to us in schools.

Suppose we take two numbers a and b whose greatest common divisor is c. By definition, c divides both a and b, where $a > b$. Let $a = mc$ and $b = nc$, where m and n are integers. When we divide a by b, unless b is a factor of a, a long division of a by b will leave a

remainder. Let $r = a - bq$ be the remainder of such a division. Clearly, since c divides both a and b, it also divides r. Euclid algorithm works like this. We do a long division of a by b. Let $q_1 = \left[\frac{a}{b}\right]$ be the quotient where [ ] is the greatest integer function and let $r_1 = a - bq_1$ be the remainder. We now divide b by this remainder r_1, getting a quotient q_2 and a remainder r_2. We carry on like this till we find a zero remainder at the n-th stage of the algorithm. The last divisor r_n then is the greatest common divisor that we are trying to find. The problem in this method is while this is a reasonably good algorithm to find gcd of two numbers, it is not particularly useful in finding factors of a single number as there is no suitable starting point and we must check numbers from 2 upward up to $\sqrt{N}$. In this lecture we discuss an algorithm due to Peter Shor, which could be implemented using a quantum computer to provide a fast factorization. This is done by solving an equivalent problem of finding a period of a function.

2 Shor's Algorithm

Shor's algorithm for factorizing N has the following steps:

1. Take a random number $m < N$. Calculate G.C.D. of m, N by some standard algorithm, such as Euclid algorithm. If $GCD(m, N) \neq 1$, we have found a factor!. Very unlikely scenario. The number m that we choose is obviously co-prime with N, i.e. m and n have no common factor. We will illustrate by choosing $N = 799$ whose factors are 17 and 47. Choose $m = 7$ whose GCD with 799 is 1.

2. Define a function $f_N : \mathbb{N} \to \mathbb{N}$ such that $f_N(a) = m^a \mod N$. We need to find the smallest $P \in \mathbb{N}$ such that $m^P = 1 \mod N$. This is called the period of f_N This step (period finding) requires a quantum computer. It turns out that $7^{368} = 1 \mod 799$, i.e. $P = 368$..

3. If P is odd, the method fails and we must return to step 1 to choose a different m and start all over. (In the lecture a small number $N = 21$ is used to illustrate, which can be worked out easily. We can choose m to be any number which is co-prime with 21. Thus $m \in \{2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$. Choosing $m = 2$, various powers of 2 are$2^2 = 4, 2^3 = 8, 12^4 = 6, 2^5 = 32, 2^6 = 64$of which the last number 64 is 1 mod 21. Thus in this case P=6).
   Let us consider some results from linear algebra which we will only illustrate here but not prove them (these can be found in any text book on discrete mathematics at college level). Consider a quadratic equation, e.g. $x^2 = 1 \mod N$. Now if N is an odd prime, one can show that this equation only has the trivial solutions, viz., $x = \pm 1$. On the other hand, if N is a composite number, there are non-trivial solutions of the type $x = \pm a$. (Remember we are doing modular arithmetic here which implies that to a we could add kN.)To illustrate consider an example. Consider the equation $x^2 = 1 \mod 41$. This equation only has trivial solutions ± 1.

However consider $N = 55$, in this case $x^2 = 1 \bmod 55$, in addition to having the trivial pair has non trivial solutions $x = \pm 21$ as $x^2 = 441 = 1 \bmod 55$ because $441 = 55 \times 8 + 1$. Since we have, by definition of a period, $m^P = 1 \bmod N$, if we choose $x = m^{P/2}$, this equation would become equivalent to the quadratic equation $x^2 = 1$. In order that we may do it P should be even and we should then choose a different m and repeat the algorithm.

4. if P is even, then, we can factorize $m^P - 1$

$$m^P - 1 = (m^{P/2} + 1)(m^{P/2} - 1)$$

Since by definition $m^P = 1 \bmod N$, $m^P - 1 = 0 \bmod N$. If Now, $(m^{P/2} - 1) \neq 0 \bmod N$ because P is the smallest integer which satisfies $m^P - 1 = 0$. If $m^{P/2} + 1 = kN$ for some integer k, then again the problem is not solved and we need to go back to step 1 and select a different m. If, however, $m^{P/2} + 1$ is a not a multiple of N then, $m^{P/2} \pm 1$ must contain factors of N. One can find the factor by finding the GCD of these two numbers. For the example given, $P = 368$ so that $P/2 = 184$. We then have

$$(7^{184} + 1)(7^{184} - 1) = 799k$$

One can check that the factors are 17 and 47.
As an example which you can work out, let $N = 21$. choose $m = 2$ for which we have seen that $P = 6$ Check that Thus

$$(2^3 + 1)(2^3 - 1) = 21k$$

Thus factors of 21 are contained in 9 and 7. (the factors are 3 and 7).
As yet another example consider $N = 35$. Choose $m = 13$ for which various even powers (mod 35) are $13, 13^2 = 169 \equiv 29, 13^4 = 28561 = 13 \times 816 + 1$ so that in this case $P = 4$. So we get $(13^2 + 1)(13^2 - 1) = 170 \times 168$, the former contains the factor 5 and trhe latter by 7.

We assume that N is not power of some prime for Shor's algorithm fails in this case. (It has been shown that the probabilities of these two things happening is greater than 1/2). it is this order finding part which needs to be done by a quantum computer because such a computer can calculate various powers of m simultaneously.