Quantum Factorization

Miguel Sozinho Ramalho

November, 2018

Table of contents

Introduction

This week is dedicated to studying the quantum algorithm insofar that, arguably, has had the most **astonishing impact** on the world as it helped increase interest in Quantum Computing and reach nowadays **quantum's critical mass**!

Unlike other quantum algorithms that we have studied, this one is named after its creator: Peter Shor (**Shor's algorithm**). It brings an **exponential speed-up** on a very important problem nowadays: **prime factorization**.

Introduction

So, we will understand **which** problem Shor's algorithm solves and **how** it does so. However, we do not want an *exponential speed-up* in the complexity of this course and so, some of the mathematical concepts will be concealed and you will have a chance to study it both in the week's exercise as well as in the recommended links at the end.

Prime Factorization problem

Given a number N we would like to find a pair of prime numbers (p, q) whose product equals N: $p \times q = N$. The existence of this pair is **guaranteed** by the fundamental theorem of arithmetic.

Algorithms for achieving this already exist, however this problem fits into the **NP** family and is computationally possible only for small values of N, as nowadays **classical** computers are unable to find it in feasible time.

Cryptography considerations

The opposite process, finding large primes is relatively easy (primality test or Sieve of Eratosthenes) and multiplying them as well.

As such, many cryptography algorithms rely on that easiness and on the factorization hardness to provide a secure means of communication. If this still seems odd, lets just say that the trustworthiness of most requests made from your browser to online servers rely on it, not to mention innumerable encryption mechanisms out there.

Shor's algorithm Overview

It would be a true hindrance if prime factorization were to become a **feasible problem**. The world would have to be redefined from scratch and countless consequences on data privacy and safety would arise, of **unimaginable proportions**.

As a matter of fact, that is what Peter Schor has done (and his name definitely belongs on the algorithm) for the only thing preventing this catastrophic event is that real quantum computers are still a few steps short of allowing for such complex computations. One might say: *thankfully we found the* **theory before the technology** *arrived.*

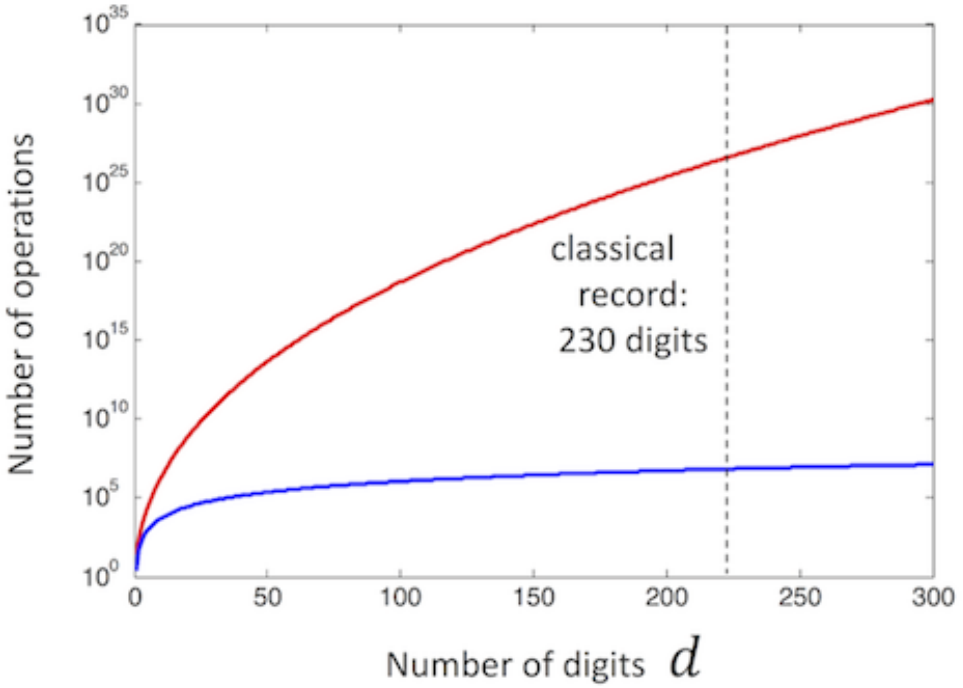
Shor's algorithm Overview

Technically speaking, Shor's algorithm solves prime factorization through the following steps:

1. Formulation of prime factorization as a period finding problem
2. Quantum algorithm to solve period finding
3. Conversion of period back into prime factors

We will go over each of these, but the heart of the speed-up lies in the quantum algorithm in question: **Quantum Fourier Transform**

Shor's algorithm Overview



$\exp(\text{const} \times d^{1/3})$

best classical
algorithm
(number field sieve)

$\text{const} \times d^3$

Shor's algorithm

From Factorization to Period Finding

The first trick of this endeavour was to discover that if we want to factorize N, we can start by the fact that:

$$f(a) = x^a \mod N^*$$

is a **periodic** function, where x is coprime with N and $a \geqslant 0$

*If necessary, review the mod operator.

From Factorization to Period Finding

Since $f(a)$ is periodic, it follows that there is a period **r**:

$$(x^0 \mod N = 1) \Rightarrow (x^r \mod N = 1)$$

Since $x^0 = 1$, and considering the periodic nature of f, which grows from $a = 0$ to $a = r$.

At this stage, let us focus on how the discovery of **r** can be achieved *quantumly*. We will go back to this formulation once we know the period.

Quantum Fourier Transform

The secret behind the exponential speed-up lies at this stage of the algorithm - Quantum Fourier Transform. This is an operation analog to that of the Discrete Fourier Transform. Unlike the classical version, the Quantum Fourier Transform can be implemented on $O(n^2)$ (polynomial) gates, whereas the classical version would require $> O(2^n)$ (exponential).

The Quantum Fourier Transform can be constructed using only Hadamard (H) and phase shift (R_ϕ) gates!

Quantum Fourier Transform

We will get to implement it on this week's exercises. Suffice to say, for now, that it achieves the task of uncovering **r** **exponentially faster** than in classical systems, by making use of quantum **superposition** to test multiple values at once.

It should be noted that the underlying math is advanced and would make little sense to include it here, due to its level of specificity. You are, however, expected to understand the high-level mechanism behind the algorithm and, with time, you may feel the need to study it thoroughly.

From Period to Factors

After **r** is found, we now need to go back to our modular arithmetic and perform the following manipulation:

$$\begin{aligned} x^r &\equiv 1 \mod N \Leftrightarrow \\ (x^{\frac{r}{2}})^2 &\equiv 1 \mod N \Leftrightarrow \\ (x^{\frac{r}{2}})^2 - 1 &\equiv 0 \mod N \Leftrightarrow \\ (x^{\frac{r}{2}} + 1)(x^{\frac{r}{2}} - 1) &\equiv 0 \mod N(\text{even } r) \end{aligned}$$

Notice that $(x^{\frac{r}{2}} + 1)(x^{\frac{r}{2}} - 1)$ is a multiple of N. So long as at least one of those factors is not a multiple of N then it has a common divisor with N.

From Period to Factors

Thus, we need only compute the greatest common divisor (gcd) between N and each of $(x^{\frac{r}{2}} + 1)$ and $(x^{\frac{r}{2}} - 1)$:

$$gcd(N, x^{\frac{r}{2}} + 1) = PossibleFactor1$$

$$gcd(N, x^{\frac{r}{2}} - 1) = PossibleFactor2$$

The *gcd* can be calculated in **polynomial time** using an algorithm that was described more than 2300 years ago by Euclides, the euclidean algorithm.

Shor's algorithm's considerations

Shor's algorithm's power is still limited by today's quantum computers (which can have in the order of 50 to 70 *good quality* qubits) and it can only be used to factor very small numbers (which you could do by hand).

Shor's algorithm's considerations

To make Shor's algorithm a security threat on today's world, one would need at least a couple thousand *good quality* qubits, which is still a few years away (one can only hope). There is time and plenty of ideas like that of Quantum Cryptography (the quantum equivalent of typical cryptography) which will prevent many catastrophes to happen, hopefully!

Hands-on

In this lesson, you have been given the logic behind Shor's algorithm and you should have a good grasp of its impact. This week, there will be two practical exercises: one on Quantum Fourier Transform and another on Shor's algorithm (which also mentions Quantum Fourier Transform). Both provide a more in-depth description of the underlying math (don't worry if it is to overwhelming, you can build on that knowledge) and also a walkthrough using QISKit of the implementation of Shor's algorithm.

Where to learn more?

- Original Paper by Peter Shor
- Peter Shor a very short video of the man, the myth, the legend on his algorithm
- Tutorial from IBM Q Experience on Shor's algorithm
- Shor's poetry on Game of Thrones (for when you feel tired)