Introduction to the Theory of Quantum Computing
Algorithmics

Ugo Dal Lago

PhD Program in Computer Science and Engineering, Bologna, July 2020

Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

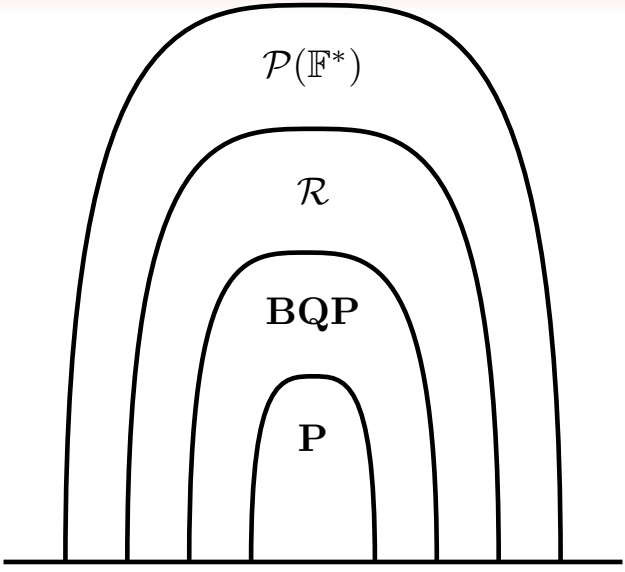
Abstract

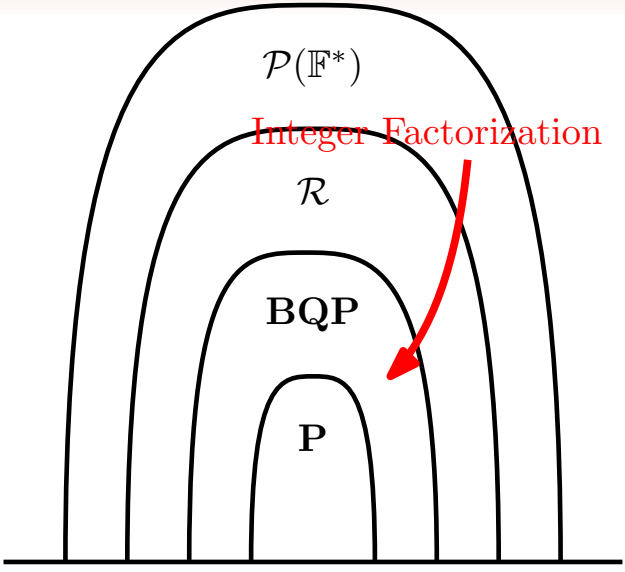
A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in computation time of at most a polynomial factor. It is not clear whether this is still true when quantum mechanics is taken into consideration. Several researchers, starting with David Deutsch, have developed models for quantum mechanical computers and have investigated their computational properties. This paper gives Las Vegas algorithms for finding discrete logarithms and factoring integers on a quantum computer that take a number of steps which is polynomial in the input size, e.g., the number of digits of the integer to be factored. These two problems are generally considered hard on a classical computer and have been used as the basis of several proposed cryptosystems. (We thus give the first examples of quantum cryptanalysis.)

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits and investigated some of their properties.

The next part of this paper discusses how quantum computation relates to classical complexity classes. We will thus first give a brief intuitive discussion of complexity classes for those readers who do not have this background. There are generally two resources which limit the ability of computers to solve large problems: time and space (i.e., memory). The field of analysis of algorithms considers the asymptotic demands that algorithms make for these resources as a function of the problem size. Theoretical computer scientists generally classify algorithms as efficient when the number of steps of the algorithms grows as

PETER SHOR


$\mathcal{P}(\mathbb{F}^*)$
$\mathcal{R}$
BQP
P


$\mathcal{P}(\mathbb{F}^*)$
Integer Factorization
$\mathcal{R}$
BQP
P

A fast quantum mechanical algorithm for database search

Lov K. Grover
3C-404A, Bell Labs
600 Mountain Avenue
Murray Hill NJ 07974
lkgrover@bell-labs.com

Summary

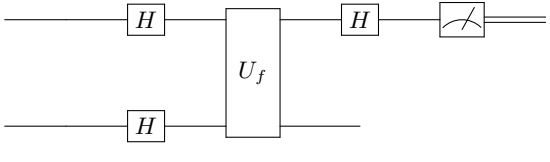
Imagine a phone directory containing N names arranged in completely random order. In order to find someone's phone number with a probability of $\frac{1}{2}$, any classical algorithm (whether deterministic or probabilistic) will need to look at a minimum of $\frac{N}{2}$ names. Quantum mechanical systems can be in a superposition of states and simultaneously examine multiple names. By properly adjusting the phases of various operations, successful computations reinforce each other while others interfere randomly. As a result, the desired phone number can be obtained in only $O(\sqrt{N})$ steps. The algorithm is within a small constant factor of the fastest possible quantum mechanical algorithm.

This paper applies quantum computing to a mundane problem in information processing and presents an algorithm that is significantly faster than any classical algorithm can be. The problem is this: there is an unsorted database containing N items out of which just one item satisfies a given condition - that one item has to be retrieved. Once an item is examined, it is possible to tell whether or not it satisfies the condition in one step. However, there does not exist any sorting on the database that would aid its selection. The most efficient classical algorithm for this is to examine the items in the database one by one. If an item satisfies the required condition stop; if it does not, keep track of this item so that it is not examined again. It is easily seen that this algorithm will need to look at an average of $\frac{N}{2}$ items before finding the desired item.

LOV GROVER

Deutsch Algorithm

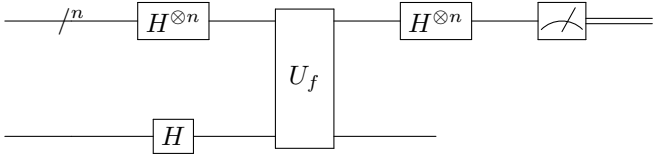
- Consider the following computational problem:
  - *Input*: a boolean circuit (whose structure is not accessible) computing an unknown function $f : \{0, 1\} \to \{0, 1\}$
  - *Output*: the value of $f(0) \oplus f(1)$ (where $\oplus$ is the XOR boolean operator).
- **Classically**, this requires computing the value of f on *both* 0 and 1.
- **Quantum parallelism** can be exploited in such as way as to superimpose the two evaluations:
  - First of all, from f one can efficiently build a quantum circuit U_f working on two qubits such that $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$.
  - Then, consider the following quantum circuit:



  - The circuit above, when fed with the input $|0\rangle|0\rangle$ will return a classical value (on the first qubit) equal to $f(0) \oplus f(1)$ with probability 1.

Deutsch-Jozsa Algorithm

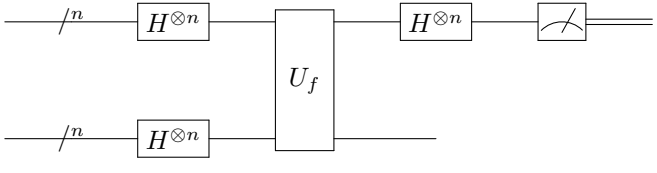
- The advantage provided by the Deutsch Algorithm over classical algorithms is debatable.
  - Perhaps the overhead due to the presence of quantum operation exceeds the advantage induced by quantum parallelism.
- Consider the following slight variation on the computational problem we considered:
  - *Input*: a boolean circuit (whose structure is not accessible) computing an unknown function $f : \{0,1\}^n \to \{0,1\}$ which is either balanced or constant.
  - *Output*: a boolean value capturing the fact that f is either constant of balanced.
- The following circuit family, when fed $|0\rangle^{\otimes n}|0\rangle$, returns in the first n qubits, the string 0^n iff the function f is constant:



Here, the circuit U_f is such that $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$, like in Deutsch Algorithm. Observe, however that $x \in \{0,1\}^*$ here.

Simon's Algorithm

- Another somehow related problem is the so-called Simon's problem:
  - *Input*: a boolean circuit (whose structure is not accessible) computing an unknown function $f : \{0,1\}^n \to \{0,1\}^n$ which is known to have the following property: there exists $s \in \{0,1\}^n$ such that $f(x) = f(y)$ iff $x = y$ or $x = s \oplus y$.
  - *Output*: the string $s \in \{0,1\}^n$ above.
- Simon's algorithm is based on the following circuit family, which looks quite familiar



  Here, the circuit U_f is again such that $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. Observe, however that $x, y \in \{0,1\}^*$ here.
- One run of the circuit S_n above is not sufficient to get the desired output. It needs to be run multiple times.
  - In other words, Simon's algorithm does not consist in $\{S_n\}_{n\in\mathbb{N}}$ itself, but in a classical algorithm using S_n as a subroutine.

Simon's Algorithm

- The actual Simon's Algorithm has the following structure:
  1. $i \leftarrow 1$.
  2. Apply the circuit S_n to $|0^n\rangle|0^n\rangle$, and obtain in the first n qubits the value $|\mathbf{w}_i\rangle$
  3. Check whether the subspace of $\{0,1\}^n$ generated by the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_i$ has dimension $n-1$. If so, go to step 4, otherwise, increment i by 1 and go back to 2.
  4. Solve the linear system of equations

$$\begin{cases} \mathbf{x} \cdot \mathbf{w}_1 = 0 \\ \vdots \\ \mathbf{x} \cdot \mathbf{w}_1 = 0 \end{cases}$$

  and return the unique nonnull solution $\mathbf{x}$.

Simon's Algorithm

- The actual Simon's Algorithm has the following structure:
  1. $i \leftarrow 1$.
  2. Apply the circuit S_n to $|0^n\rangle|0^n\rangle$, and obtain in the first n qubits the value $|\mathbf{w}_i\rangle$
  3. Check whether the subspace of $\{0,1\}^n$ generated by the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_i$ has dimension $n-1$. If so, go to step 4, otherwise, increment i by 1 and go back to 2.
  4. Solve the linear system of equations

$$\begin{cases} \mathbf{x} \cdot \mathbf{w}_1 = 0 \\ \vdots \\ \mathbf{x} \cdot \mathbf{w}_1 = 0 \end{cases}$$

  and return the unique nonnull solution $\mathbf{x}$.

Theorem

Simon's Algorithm correctly finds the hidden string s. The expected number of evaluations of f in the execution of the algorithm is less than n, and the expected number of other elementary operations is $O(n^3)$.

Simon's Algorithm

- The actual Simon's Algorithm has the following structure:
  1. $i \leftarrow 1$.
  2. Apply the circuit S_n to $|0^n\rangle|0^n\rangle$, and obtain in the first n qubits the value $|\mathbf{w}_i\rangle$
  3. Check whether the subspace of $\{0,1\}^n$ generated by the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_i$ has dimension $n-1$. If so, go to step 4, otherwise, increment i by 1 and go back to 2.
  4. Solve the linear system of equations

$$\begin{cases} \mathbf{x} \cdot \mathbf{w}_1 = 0 \\ \vdots \\ \mathbf{x} \cdot \mathbf{w}_1 = 0 \end{cases}$$

  and return the unique nonnull solution $\mathbf{x}$.

Theorem

Simon's Algorithm correctly finds the hidden string s. The expected number of evaluations of f in the execution of the algorithm is less than n, and the expected number of other elementary operations is $O(n^3)$.

Theorem

Any classical algorithm that solves Simon's Problem with probability at least $\frac{2}{3}$ for any f must evaluate f at least $\Omega(2^{n/3})$ times.

Shor's Algorithm

- Shor's Algorithm does not solve the factoring problem *directly*, but goes through a series of reductions:

Factoring any Integer

Shor's Algorithm

- Shor's Algorithm does not solve the factoring problem *directly*, but goes through a series of reductions:

Factoring any Integer

$$\Downarrow$$

Splitting any Odd Non-Prime-Power N

Shor's Algorithm

- Shor's Algorithm does not solve the factoring problem *directly*, but goes through a series of reductions:

Factoring any Integer

$\Downarrow$

Splitting any Odd Non-Prime-Power N

$\Downarrow$

Finding the Orders of Integers Modulo N

Shor's Algorithm

- Shor's Algorithm does not solve the factoring problem *directly*, but goes through a series of reductions:

Factoring any Integer

$\Downarrow$

Splitting any Odd Non-Prime-Power N

$\Downarrow$

Finding the Orders of Integers Modulo N

$\Downarrow$

Sampling Estimates to a Random Integer Multiple of $\frac{1}{r}$
(where r is the order of some integer a modulo N)

Grover's Algorithm

- Grover's algorithm solves the following computational problem:
  - *Input*: a boolean circuit (whose structure is not accessible) computing an unknown function $f : \{0,1\}^n \to \{0,1\}$
  - *Output*: a string $s \in \{0,1\}^n$ such that $f(s) = 1$.

Grover's Algorithm

- Grover's algorithm solves the following computational problem:
  - *Input*: a boolean circuit (whose structure is not accessible) computing an unknown function $f : \{0,1\}^n \to \{0,1\}$
  - *Output*: a string $s \in \{0,1\}^n$ such that $f(s) = 1$.
- This is a *search problem*. In the worst case, any classic algorithm must evaluate f on all the 2^n coordinates.

Grover's Algorithm

- Grover's algorithm solves the following computational problem:
  - *Input*: a boolean circuit (whose structure is not accessible) computing an unknown function $f : \{0,1\}^n \to \{0,1\}$
  - *Output*: a string $s \in \{0,1\}^n$ such that $f(s) = 1$.
- This is a *search problem*. In the worst case, any classic algorithm must evaluate f on all the 2^n coordinates.
- Grover's Algorithm, by way of a technique called **amplitude amplification**, manages to solve the same problem in time at most $O(\sqrt{2^n})$.

Thank You!

Questions?