QUOTATION 1

AUTHOR, *SOURCE*

QUOTATION 2

AUTHOR

QUOTATION 3

AUTHOR

YONGLI CHEN

# QUANTUM COMPUTING: THEORY AND PRACTICE

PUBLISHER NAME

*First printing, April 2018*

# *Contents*

# *List of Figures*

*List of Tables*

*Dedicated to my family and friends.*

# *Preface*

This is Yongli Chen's note on learning quantum computing. The reason the author write this book is simple: Although there are a few books on quantum computing, they are mainly focus on theory rather than application. This book will introduce readers to quantum computing with a focus on applications as well as theory behind those applications.

In addition, this book will be written in a popular science style, which means it will contain A LOT OF explanations. This is because the author is using Feynman technique to study quantum computing, and would like to write down his thought process. Please be aware.

# 1 Chapter 1 Introduction

## 1.1 *What is quantum computing?*

*Quantum computing* sounds pretty intimidating to most people. What if I remove the "quantum" part, leaving only the "computing" part? Much friendlier, huh? The "computing" part is plain as simple: Give a system some input, and it will give you some output. Now comes the jargon, "quantum". The word "quantum" comes from *Quantum Mechanics*, a branch of physics that study tiny things. Therefore, *quantum computing* is essentially a way of using tiny particles to do calculation for us. The quantum computer manipulates on subatomic particles to do calculations, so you can imagine it is EXTREMELY difficult to build a quantum computer. Although there are prototypes of quantum computer as year of 2018, none of them has real application yet. However, quantum computing is not that far away from us. In fact, people believe it will become an critical technology in the following decades.

## 1.2 *Why quantum computing?*

In short, quantum computer (Figure 1.1[1]) can solve problems we can never solve with classical computer. For example, there is an encryption algorithm we use everywhere, everyday, called RSA algorithm. By using RSA algorithm, our passwords are encrypted during transmission and keep our money/data safe. It is safe because it will take centuries for the most powerful super computer to decrypt the message encrypted by RSA. But for quantum computer? This decryption process could be just a matter of seconds. What does that mean? If you have a quantum computer today, you would be able to hack almost all bank accounts on earth. Of course, we don't want to build a quantum computer only good for hacking bank accounts, right? There are a number of other applications like climate simulation, machine learning, material research, etc. You already know how computers change the world. Then just imagine the world will be changed again by quantum computers, and embrace the future!

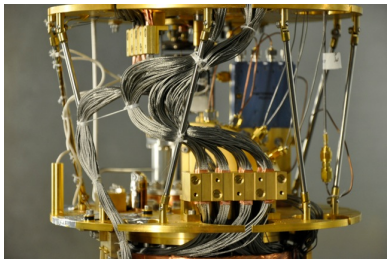

Figure 1.1: Support structure for a D-WAVE quantum computer.

[1] NASA. Picture of quantum computer. https://www.nasa.gov/, 2013. Accessed: 2018-04-02

## 1.3 *How can we try quantum computing?*

As of April 2018, there are a few options for us to write quantum programs on a simulated quantum computer. Why simulated quantum computer? Because the real quantum computer doesn't exist yet! Furthermore, using simulated quantum computer can get our hands dirty already. The Q# language from Microsoft will be used as the main language for quantum programs throughout the book.

## 1.4 *Introduction to quantum mechanics*

Before playing around with quantum computing, we need to have some basic background in quantum mechanics. But don't worry, let's go through the basics together. Since the mathematical foundation of quantum mechanics is linear algebra, we are gonna learn linear algebra first. Then we will use the powerful mathematical tool to explain a few important ideas in quantum mechanics. Finally, we will apply our newly learned knowledge to write our first quantum program! How cool is that! So please bear with me for the theoretical part. Otherwise, you will not understand how we code the quantum program.

### 1.4.1 *Linear algebra for quantum computing*

It could easily take a few months for normal people to learn linear algebra. Therefore, we will only cover essential parts here. There are numerous resources on linear algebra out there, if you want to learn more about it. For example, the MIT open course: Linear Algebra 18.06 from Prof. Strang[2] is extremely good, and you can benefit a lot by completing it.

[2] MIT. Linear algebra by prof. gilbert strang. https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/, 2017. Accessed: 2018-04-11

#### 1.4.1.1 ***What** is linear algebra?*

## 1.5 *Your first quantum program with Q#*

Now let's get down to business. It's time to write our first quantum program! But first things first, let me give you an overview of Q#. Q# is a quantum programming language. This means that Q# is designed to express quantum algorithm or logic instead of classical computation. The Q# language is NOT a program language that we can use for implementing classical algorithms. Therefore, when we want to write a complete quantum program, we might end up using some classical programming language along with quanutm programming language like Q#. For example, let's say you live in the U.S., and you want to visit China. What you will probably do is to take a car ride to the airport, then take the plain to China, right? In the example, the car is the classical language and the plain is the quantum language. They both have their own strength, and only by working together they can get you to your destination.

Now let's do the coding. This program is taken from Microsoft blog[3], but that blog assumes the reader has some quantum computing background. So instead of redirecting you to the article, let's walk through the program together with our beloved rubber duck methodology[4].

For those who doesn't know what rubber duck methodology is, I encourage you to spend a minute and search for rubber duck debugging[5].

[3] Microsoft. The q# programming language. https://docs.microsoft.com/en-us/quantum/quantum-qr-intro?view=qsharp-preview/, 2017a. Accessed: 2018-04-06

[4] Andrew Hunt and David Thomas. *The pragmatic programmer: from journeyman to master*. Addison-Wesley, 2002

### 1.5.1 *Setting up environment*

To setup a Q# development environment, you need to install

1. .NET core 2.0 SDK or later
2. Visual Studio Code (For Windows users, you can use Visual Studio)
3. Microsoft Quantum Development Kit Extension for Visual Studio Code/Visual Studio

For detailed instructions, please refer to Installing and Validating the Q# Development Environment[6].

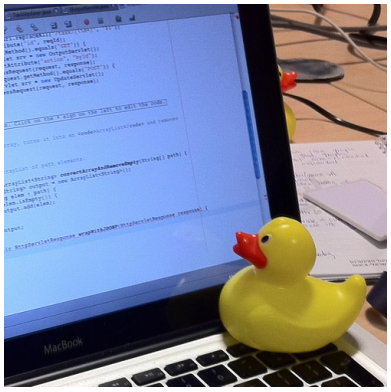

Figure 1.2: Rubber duck debugging

[5] Tom Morris. Rubber ducker debugging. https://en.wikipedia.org/, 2011. Accessed: 2018-04-06

[6] Microsoft. Installing and validating the q# development environment. https://docs.microsoft.com/en-us/quantum/quantum-installconfig?view=qsharp-preview&tabs=tabid-vscode, 2017b. Accessed: 2018-04-11

### 1.5.2 *Creating a Bell state*

# 2 Chapter 2 Title

# *Bibliography*


Andrew Hunt and David Thomas. *The pragmatic programmer: from journeyman to master*. Addison-Wesley, 2002.

Microsoft. The q# programming language. https://docs.microsoft.com/en-us/quantum/quantum-qr-intro?view=qsharp-preview/, 2017a. Accessed: 2018-04-06.

Microsoft. Installing and validating the q# development environment. https://docs.microsoft.com/en-us/quantum/quantum-installconfig?view=qsharp-preview&tabs=tabid-vscode, 2017b. Accessed: 2018-04-11.

MIT. Linear algebra by prof. gilbert strang. https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/, 2017. Accessed: 2018-04-11.

Tom Morris. Rubber ducker debugging. https://en.wikipedia.org/, 2011. Accessed: 2018-04-06.

NASA. Picture of quantum computer. https://www.nasa.gov/, 2013. Accessed: 2018-04-02.