CSCI 4140
Overview of Quantum Computing

Mark Green

Faculty of Science

Ontario Tech

Introduction

- There are two key questions that need to be examined:
  - What is quantum computing?
  - What is it good for?
- Basically why should we be interested in studying this? How is it going to help us?
- Even if we don’t end up using quantum computing, it opens our eyes to different ways of computing, what it means to compute

Quantum Computing

- Quantum computing is the use of quantum properties of matter to perform computations
- This is a pretty general definition, since there are many quantum properties and many ways of using them
- Any device that uses quantum properties is called a quantum computer
- The computers that we are using now we will call classical computers
- We will examine a typical set of quantum properties shortly

What is it good for?

- This is the interesting question, since we are in the process of answering it
- In algorithms study the performance of algorithms, use big-O notation to describe performance
- The asymptotic performance of an algorithm given some measure of the size of the input
- We usually use n as the input size: the number of items, the number of bits or bytes, etc.

What is it good for?

- Given n, we show that the algorithm takes O(f(n)) time for some function f()
- We usually talk about time, but we could also be concerned about memory, this is important for some algorithms
- We want f() to be a polynomial, if it is we say the algorithm is efficient, we can solve the problem in reasonable time on a classical computer
- In computer science we almost always restrict ourselves to polynomial time algorithms

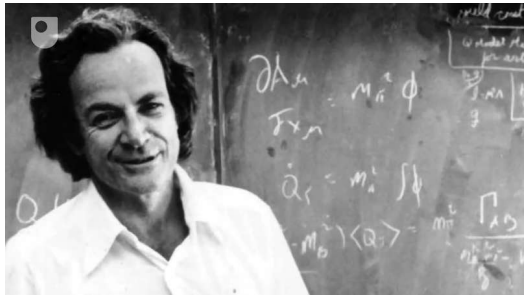
What is it good for?

- But there are problems that are $O(b^n)$ for some $b > 1$
- We cannot solve these problems efficiently on a classical computer
- The time required grows too quickly with the size of the problem, we soon get into years, decades, centuries of computer time
- When we run into these problems in CS we tend to sweep them under the rug, try to find a heuristic or a slightly different problem
- We call these problems exponential, and some of them are quite important

What is it good for?

- We run into exponential problems in computer science, in compilers and software engineering
- They appear in graph theory, which has implications for networks
- They also appear in science:
  - Quantum mechanics
  - Drug design
  - Molecular docking
  - Chemistry

The Beginning



- The first real suggestion of quantum computing was by Feynman in 1981
- People suggested using quantum properties before this, but it was largely a theoretical exercise
- Feynman had a real problem to solve and he was famous, he wanted to simulate quantum systems
- This was an exponential problem, and only very small problems could be simulated on a classical computer, this is still the case today

The Beginning

- Feynman suggested that a quantum system could be used to simulate a quantum system
- That is, build a quantum computer that could be used to simulate quantum systems
- Feynman didn’t know how to build this type of computer, that would take later researchers
- No one knew how it would be programmed, or how algorithms would be developed, that took the next few decades

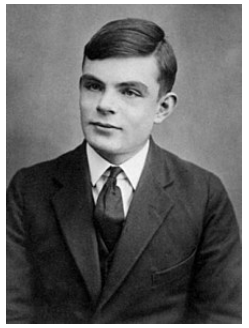
History - Physics

- Around 1900$\pm$5 years physics was going through a revolution
- Classical physics didn’t explain a number of things, it was a good approximation for people sized things at normal speed and normal time scales
- The start of relativity and quantum mechanics, had a major impact on physics, their whole world changed over a short period of time
- We will find that quantum mechanics is very useful for quantum computing

History - Mathematics

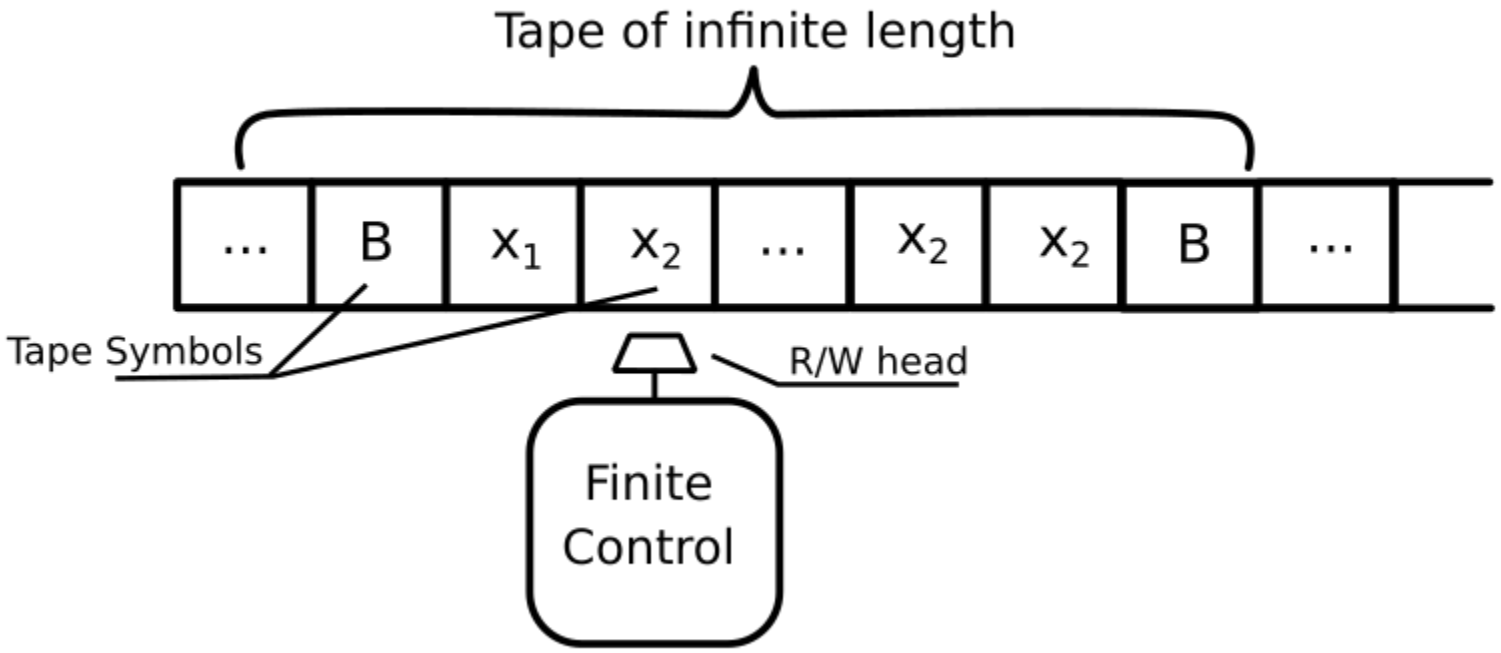
- About a decade later mathematics started going through a similar revolution
- Mathematicians asked: is there a set of basic primitives that all of mathematics can be built from?
- Can we place mathematics on a solid theoretical foundation?
- This revolution saw the development of set theory and logic, as well as number theory
- For us, the most important ideas were around computation

History - Computation



- It was thought that anything could be computed, you just needed enough time and the right formula
- No one really thought very much about this, until the 1930s
- Church and his student Turing started investigating this problem in the 1930s, a decade before the first computers
- Church developed the lambda calculus, which is the basis for functional programming
- Turing developed the Turing machine

Turing Machine



Turing Machine

- Input is written on the tape, the machine runs and the output is written on the tape
- The tape can move left and right and symbols can be read or written on the tape
- The control is basically a table, the current state of the machine and the symbol on tape determines the next action
- This is quite primitive, but since it’s simple it’s easy to do theorems and proofs
- If you work hard enough you can produce programs

Turing Machine

- Important concept: can produce a universal Turing machine that can simulate any other Turing machine
- This can be viewed as the start of programming
- Why is this important?
- Our modern digital computers are equivalent to Turing machines, anything we can do on a modern computer can be done on a Turing machine, no more powerful than a Turing machine
- Lambda calculus and Turing machines are equivalent, they can compute the same thing

Turing Machine

- A Turing machine identifies what can and cannot be computed
- An algorithm that is O(n) on a Turing machine is O(n) on a modern digital computer, we cannot do better
- There are things that can’t be computed on a Turing machine, and things that can’t be efficiently computed
- This is the basis of most of theoretical computer science
- There are a number of models of computation, they are all equivalent to Turing machines

Turing Machine

- There are two aspects of Turing machines that are important to us
- First, they are sequential, the theory of computation is based on sequential processing
- Even our parallel computers are basically sequential, they are no more powerful than a Turing machine
- Second, universality, the notion of a general purpose computer
- Question: can we trade-off universality for more computational power?

Quantum Computing

- Since quantum computing is based on quantum mechanics, we need to know a little bit about it
- Start with some basic concepts and fill in the details later
- Classical physics assumed everything was continuous, in the quantum world objects have discrete states, we are no longer continuous
- We can select states to represent the things that we are computing with
- To make things simple we will use just two states of the quantum system

Quantum Computing

- We call these two states $|0\rangle$ and $|1\rangle$ for obvious reasons
- The problem with quantum systems is we really don't know which state they are really in
- Our system is in a combination of these states, written as:

  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
- This is a superposition of our two basic states, this is our qubit, which is the quantum version of a bit
- In this case both α and β are complex numbers, but that won't concern us for the time being

Quantum Computing

- In 1926 Max Born showed that the probability of observing a qubit in the $|0\rangle$ state is $|\alpha|^2$ and the probability of observing the qubit in the $|1\rangle$ state is $|\beta|^2$
- In our superimposed state $|\psi\rangle$ we must have:

  $|\alpha|^2+|\beta|^2=1$
- This is the Born rule
- While the qubit $|\psi\rangle$ is on its own, it is in the superimposed state, we can operate on this qubit and it remains in a superimposed state, but the probabilities will change

Quantum Computing

- This superimposed state breaks down when we measure the qubit, in that case we will get a 0 or a 1, we will be back to the classical domain
- From then on the qubit will be in the measured state
- We will use a lot of linear algebra for the theory of quantum computing
- The two values |0> and |1> are called the computational basis:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Quantum Computing

- With a bit we only have two potential values: 0 and 1
- With a qubit we have an infinite number of possible values encoded in the two complex coefficients
- Our qubits are vectors in a 2D space
- Now we need some way of manipulating them
- With bits we have the standard logic gates, things like AND, OR, NOT, etc.
- In quantum computing we have quantum gates, which we can represent by matrices

Quantum Computing

- In classical computing there is a limited number of gates, in quantum computing we have an unlimited supply of gate types
- But, there are two conditions that a quantum gate must satisfy
- First, observe that qubits are unit vectors, so our gates must be represented by matrices that transform unit vectors to unit vectors
- This make sense since we want to interpret the qubit coefficients as probabilities
- The second condition requires some explanation

Quantum Computing

- Consider a classical AND gate, in it’s simplest form it has two inputs and one output
- Two bits go into the gate, but only one comes out, what happened to the other bit??
- We need to worry about the conservation of energy, an AND gate is a physical device
- We have two units of energy going in and one going out, that extra unit of energy has to go somewhere
- In digital circuits it’s converted into heat

Quantum Computing

- We can’t do this with a quantum gate, we are using energy levels in a quantum system to represent qubits
- This excess energy will change the value of the qubit in a way we don’t want
- Our gates must conserve energy
- We do this by making our gates reversible:
  - Same number of inputs as outputs
  - Can run the gate backwards to get the original input

Quantum Computing

- Since our qubits are vectors, our gates are modeled as matrices
- In the case of a single qubit gate, it’s a 2x2 matrix
- What happens if we have more than one qubit, say we have 2 qubits
- In this case the total state of the system is all the possible combinations of the two qubits
- We can represent this by a 4D vector and our gates become 4x4 matrices
- If there are two gates in a row, I just multiply the matrices

Quantum Computing

- If we have 3 qubits, they are represented by an 8D vector and we use 8x8 matrices for our gates
- If we have n qubits we have a 2^nD vectors and 2^n complex coefficients
- The number of coefficients grows exponentially with the number of qubits

This is the promise of quantum computing

Quantum Computing

- Recall: exponential problems can’t be solved efficiently on digital computers
- We have an exponential number of complex coefficient in a quantum computer
- We can process all of these coefficients in parallel, each step of the computation could produce an exponential number of results
- It appears that we can efficiently solve exponential problems on a quantum computer, why quantum computers are so interesting

Quantum Computing

- Classical computers simulating quantum systems:
  - Can do 20 qubit systems on a desktop of laptop computer
  - Can do 50 qubits on largest supercomputer, given a few years to complete the computation
- We have 53 qubit quantum computers, larger ones on the way
- Sounds like we've solved the problem, we can now solve exponential problems
- Well, not so fast, we've got a couple of problems

Quantum Computing

- While we have 2^n complex coefficients we can't measure all of them
- We can only measure n qubits, when this happens the 2^n complex coefficient collapse to an n bit value
- While we can compute exponentially we cannot retrieve the result!
- This is the challenge for quantum algorithms, the last step of the algorithm must produce an n bit result
- There are algorithms that do this
- Oh, and all of this is statistical, remember we are dealing with probabilities

Quantum Computing

- The complex coefficients are probabilities, so the result has a probability associated with it
- Our algorithms must produce the correct result with a high probability
- We can run the computation multiple times, the most common result is probably the correct one
- Does this matter? Maybe yes, maybe no
- We'll come back to this later

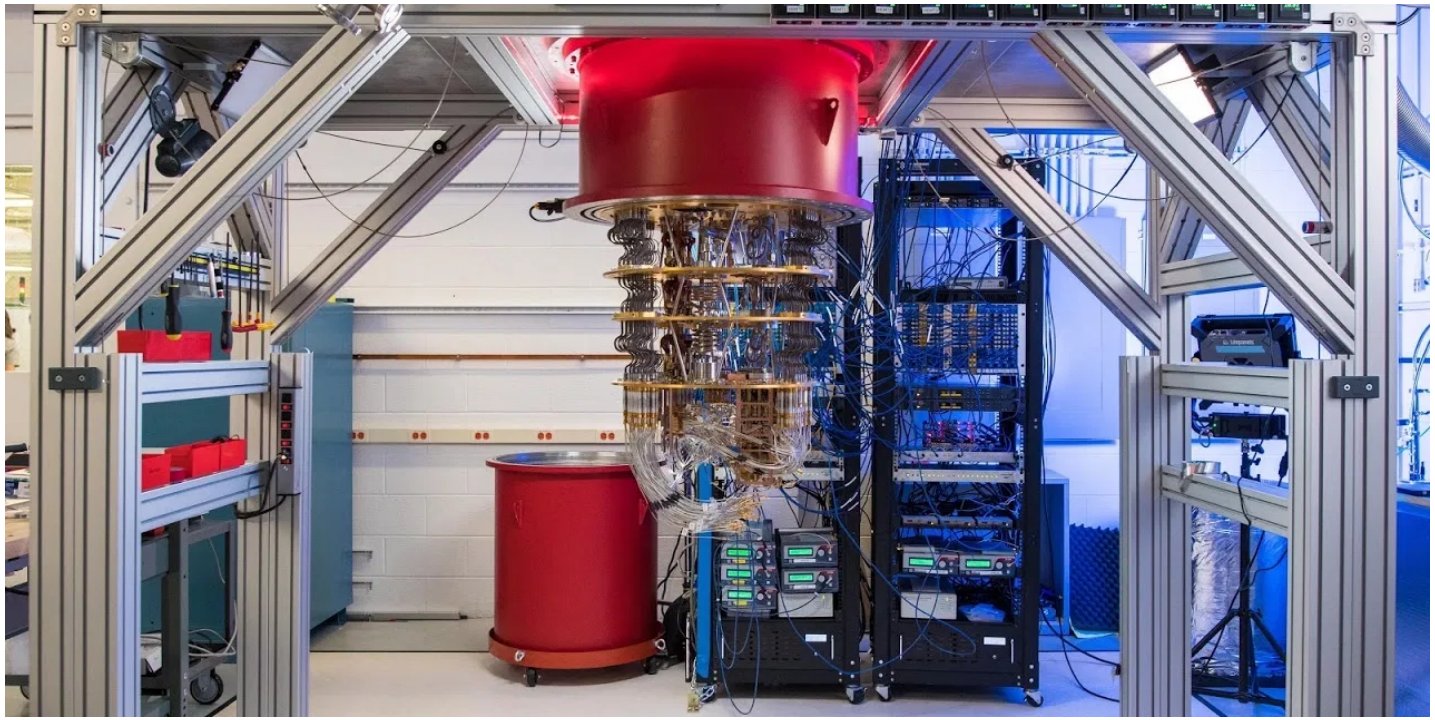
Quantum Computing

- The other problem comes from the hardware we are using
- Qubits are quantum systems, we have one atom per qubit
- This is pretty delicate
- Any stray energy, like heat, could change the value of the qubit
- We need to isolate the qubits from the outside world:
  - Hold them in a vacuum
  - Cool them close to 0K, the coolest temperature possible

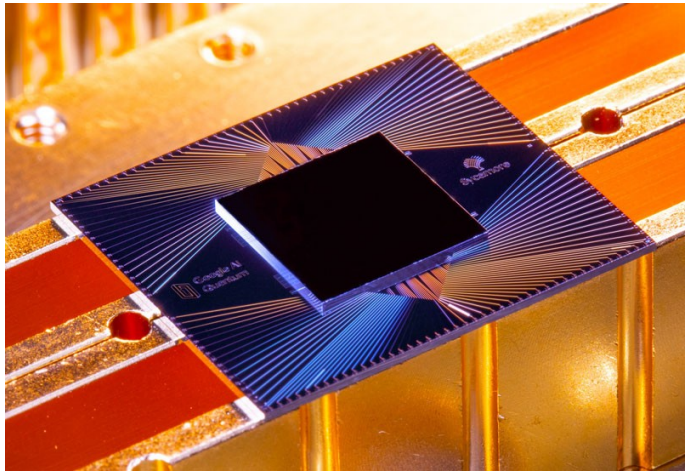
Quantum Computing

- While the quantum computer is the size of a chip, everything around it takes up a small room
- Still doesn’t give us perfect qubits, called physical qubits, they have noise associated with them
- Noise level could be in the 5% range
- Our algorithms use perfect qubits, called logical qubits, they have no noise associated with them
- This is a bit of a mismatch

Google Quantum Computer - Sycamore



Whole System



Quantum Chip

Quantum Computing

- There is a solution to this problem, error correcting codes
- Basic idea: use a collection of physical qubits to implement one logical qubit
- There are many ways of doing this, but even the most efficient strategies require on the order of 5 to 10 physical qubits per logical qubit
- If we had a large number of qubits this would be feasible, but right now we are limited to around 50

Entanglement

- This is just weird
- The states of two qubits can be combined in a way that they can’t be separated, this is called entanglement
- There are gates that can do this for us, take two separate qubits and convert them into a pair of qubits
- These two qubits can be separated, even by a large distance, and they still act as one system
- If one qubit is measured, the other qubit will instantaneously take on the same value

Entanglement

- We have a pair of entangled qubits, we measure one of them, and say the result is 1, in the other qubit the quantum state will collapse and it will also have the value 1
- It doesn’t matter how far the qubits are apart, this will happen instantaneously
- Einstein believed this contradicted relativity, information traveling faster than the speed of light
- But, when the qubits separated they traveled less than the speed of light, this is the actual information transfer

Quantum Security

- Quantum mechanics has a significant impact on security and communications
- One of the first quantum algorithms was Shor's algorithm
- This algorithm produces prime factors of very large integers, which is viewed as a very hard problem
- The common encryption algorithms are based on the fact that this is a hard problem and it would take more than a thousand years on a classical computer to solve it

Quantum Security

- Since the problem is so hard to solve the data is safe
- Shor's algorithm showed that a quantum computer could solve the problem relatively quickly, maybe several days
- This means the data will no longer be secure, it will be relatively easy to crack the code
- Current quantum computers are too small, but it's estimated that some time in the next 10 to 15 years they will be large enough
- This seems like a long time, but it is not

Quantum Security

- Need to develop a new encryption algorithm that is quantum secure
- This process has already started
- Then need to select one of them as the standard
- After that it needs to be deployed on all the computers in the world
- The last time a security hole like this was discovered it took a decade to update all the software
- In addition, there is all the encrypted data that needs to re-encrypted so it stays secure

Quantum Security

- But, quantum communications is very secure
- Two important properties of qubits:
  - Qubits can’t be copied
  - If a qubit is read (measured) it is no longer in the superimposed state
- If someone intercepts a message the receiver will know immediately
- There is no way to intercept a message without it being detected

Quantum Security

- Where this becomes important is quantum key distribution (QKD)
- Quantum communications isn’t used for the message, just the key that is used to decode the message
- If the key is intercepted, a new key can be generated and sent before the actual communications starts
- There are commercial QKD systems
- There are experimental quantum networks, the quantum Internet

Quantum Hardware

- There are few variations on classical computer hardware, all the technology is basically the same
- Can build any digital computer with just NAND gates, it is the universal gate
- In quantum computing it’s not that simple
- There isn’t a single universal gate, there is an infinite number of universal gate sets
- Can implement one set in terms of another, hardware designers can choose which ever one is easier to implement

Quantum Hardware

- There are many ways of doing quantum systems, just need to select a property that has two or more levels
- Examples: energy states, spin, phase, photon polarization
- Each of these leads to multiple technologies
- With digital computers we could run with one technology and develop it quickly
- Can’t do this in quantum, which one do we choose?
- But, if one turns out to be a dead end, we have the others to work on

Quantum Hardware

- What do we need in quantum hardware:
  - Scalable to thousands, if not hundreds of thousands of qubits
  - Low error rates
  - Operate in a normal room environment
  - Low cost, etc.
- We can’t do this yet, we max out at around 50 qubits, error rates in 0.01% range
- Era of NISQ – noisy intermediate scale quantum computers
- No logical qubits, only physical ones

Quantum Hardware

- But, the last 5 years have seen incredible progress
- Have gone from 1 or 2 qubit demos, to 50 qubit real systems
- Multiple small quantum computers on the Internet, you can send jobs to them
- Even have commercial quantum computers, D-Wave in BC, not a universal quantum computer, but does solve a range of important problems

Quantum Software

- This is in the very early stages
- No one worried about software since there was no hardware to run it on
- There are no real quantum programming languages, but there are groups working on it
- The state of the art is programming at the gate level, this is the level at which quantum algorithms are expressed
- We haven’t developed high level notations

Quantum Software

- Most programming systems are based on Python
- Take the form of libraries, procedures for constructing quantum circuits
- There are a lot of them, the number seems to be growing
- We will use Qiskit from IBM:
  - Very well documented
  - Has been around for several years, becoming stable
  - Access to real quantum hardware

Quantum Software

- There are a number of additional tools:
  - Simulators – simulate program running on real quantum hardware
  - Emulators – higher level simulations that can be used for larger programs
  - Resource estimators – determine the hardware resources required to run the program
  - Noise models – simulators and emulators that include realistic error rates

Summary

- A high level view of the quantum computing landscape
- Know what the major pieces are, how they fit together, but none of the details
- Can now start examining the details
- Start developing programs for quantum computers