CSCI 4140 Quantum Algorithms Part One

Mark Green

Faculty of Science

Ontario Tech

Introduction

- Examine the common quantum algorithms
- There are far fewer quantum algorithms than classical algorithms:
  - Until recently mainly theoretical interest, no way to run them
  - Require a very different way of thinking
  - Few people were investigating them, not of practical interest
- The quantum algorithms zoo lists most of the known quantum algorithms

  https://quantumalgorithmzoo.org/

Introduction

- The algorithms examined here are constructed from the gates we discussed earlier
- Many require a combination of classical and quantum computers
- Part of the algorithm is performed on the classical computer and part on the quantum computer
- Example:
  - Loop body on the quantum computer
  - Loop control on the classical computer

Quantum Teleportation

- This is a somewhat interesting algorithm to get your head around, you may need to read through it several times
- Problem: Alice has a qubit $|\psi\rangle$ that she wants to send to Bob, but she only has a classical communications channel that transmits bits
- We have that $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and both α and β are complex
- There is potentially an infinite amount of information here, so sending it over a classical communications channel seems to be impossible
- Entanglement to the rescue!

Quantum Teleportation

- Assume that sometime in the past Alice and Bob created an entangled pair of qubits:

$$q = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

- Bob takes one qubit and Alice takes the other one
- Alice now has $|\psi\rangle$ and one half of q, Bob has the other half
- The complete state of the system is given by:

$$|\psi\rangle q\rangle = \frac{1}{\sqrt{2}}\big(\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|00\rangle + |11\rangle)\big)$$

Quantum Teleportation

- Now Alice applies a CNOT to her qubits giving:

$$\frac{1}{\sqrt{2}}\left(\alpha|0\rangle(|00\rangle+|11\rangle)+\beta|1\rangle(|10\rangle+|01\rangle)\right)$$

- Next Alice applies a Hadamard gate to obtain:

$$\frac{1}{\sqrt{2}}[\alpha(|0\rangle+|1\rangle)(|00\rangle+|11\rangle)+\beta(|0\rangle-|1\rangle)(|10\rangle+|01\rangle)]$$

- We can rewrite this in the following way:

$$\frac{1}{\sqrt{2}}[|00\rangle(\alpha|0\rangle+\beta|1\rangle)+|01\rangle(\alpha|1\rangle+\beta|0\rangle)+|10\rangle(\alpha|0\rangle-\beta|1\rangle) \\ +|11\rangle(\alpha|1\rangle-\beta|0\rangle)]$$

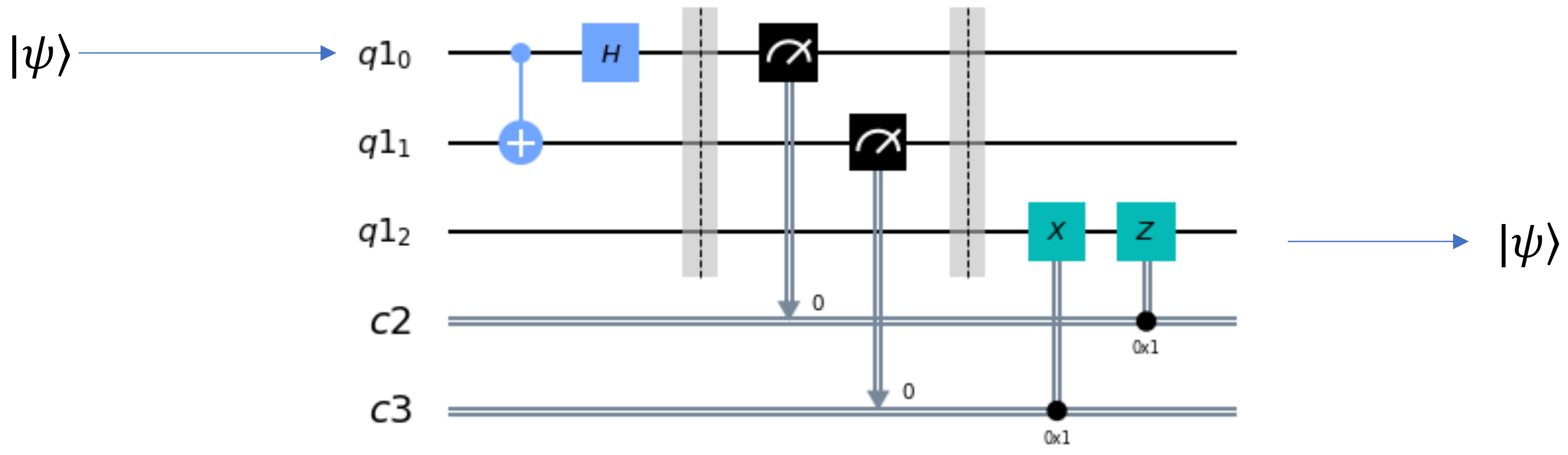
Quantum Teleportation

- We now see our solution, we measure Alice's two qubits to get two classical bits
- These two bits are sent to Bob, who can now reconstruct the original $|\psi\rangle$:
    - If the two bits are 00, Bob's qubit has the correct value
    - If the two bits are 01, Bob applies a X gate to his qubit
    - If the two bits are 10, Bob applies a Z gate to his qubit
    - If the two bits are 11, Bob applies a X gate followed by a Z gate to his qubit

Quantum Teleportation

- This involves two gates at Alice’s end, a CNOT and a Hadamard
- There are also two gates at Bob’s end, a X and a Z gate
- We have essentially used two bits to move an infinite amount of information in a qubit
- But, we need a previously shared pair of entangled qubits, so we just can’t send a qubit anywhere we like, we must preplan for the exchange

Quantum Teleportation



The top qubit is the one that is being transmitted, the next two qubits are the entangled pair. The bottom two lines are classical bits

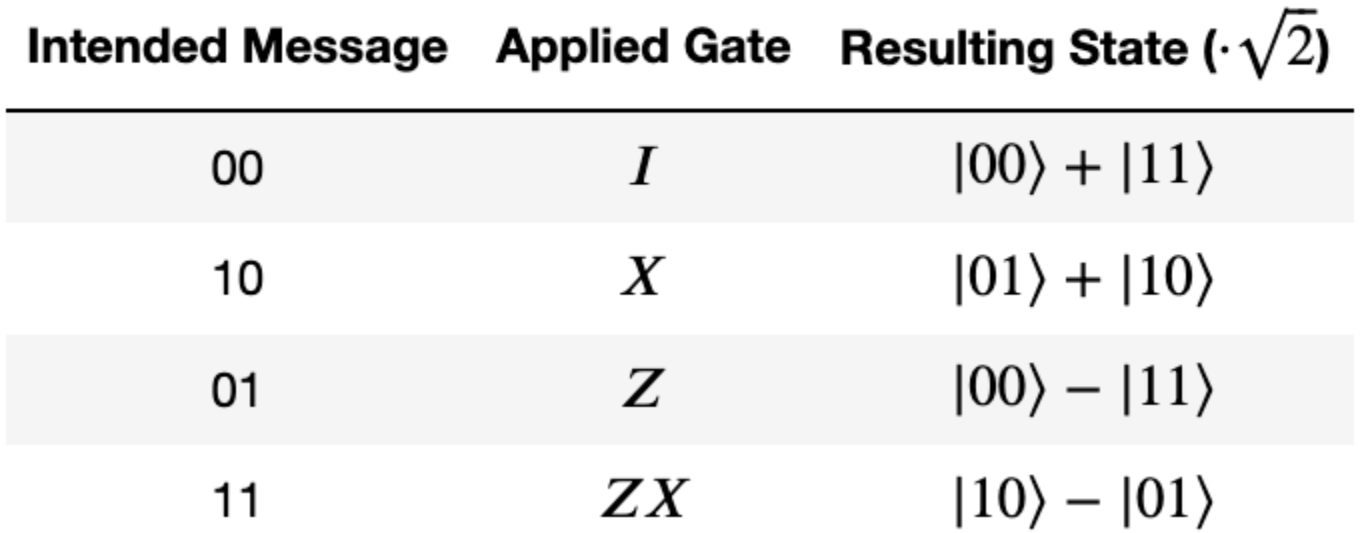
Superdense Coding

- Superdense coding is the opposite of quantum teleportation
- In this case we have two classical bits, and we use one quantum bit to transmit them
- Again we start with an entangled pair of qubits which is shared between Alice and Bob
- Alice has a two bit message that she wants to send to Bob
- For each of the 4 bit combinations she applies a different set of gates and then sends the qubit to Bob

Superdense Coding

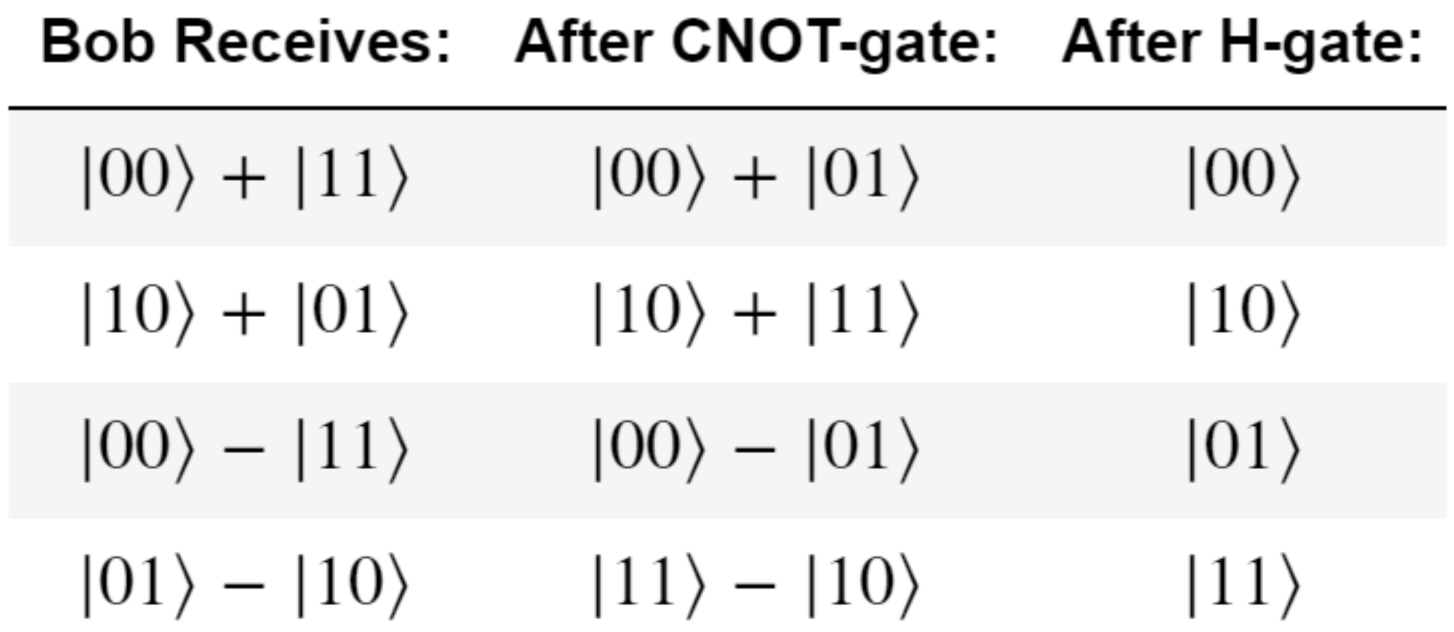
- Bob now has both of the entangled qubits
- He applies a sequence of gates to the two qubits and retrieves the result
- The entangled qubit starts in the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- Alice applies the gates shown on the following slide depending on the message
- Bob applies a CNOT followed by a Hadamard to the pair of qubits he now has

Superdense Coding



Intended Message	Applied Gate	Resulting State ($\cdot\sqrt{2}$)
00	I	$\|00\rangle + \|11\rangle$
10	X	$\|01\rangle + \|10\rangle$
01	Z	$\|00\rangle - \|11\rangle$
11	ZX	$\|10\rangle - \|01\rangle$

Superdense Coding



Bob Receives:	After CNOT-gate:	After H-gate:
$\|00\rangle + \|11\rangle$	$\|00\rangle + \|01\rangle$	$\|00\rangle$
$\|10\rangle + \|01\rangle$	$\|10\rangle + \|11\rangle$	$\|10\rangle$
$\|00\rangle - \|11\rangle$	$\|00\rangle - \|01\rangle$	$\|01\rangle$
$\|01\rangle - \|10\rangle$	$\|11\rangle - \|10\rangle$	$\|11\rangle$

Superdense Coding

- After applying the two gates a simple measurement retrieves the message that Alice sent
- Both of these examples depend upon entangled qubits, so a source of entangled qubits is required
- These qubits need to be sent over a considerable distance for communications to occur
- Luckily we can entangle photons, and they are easy to transmit

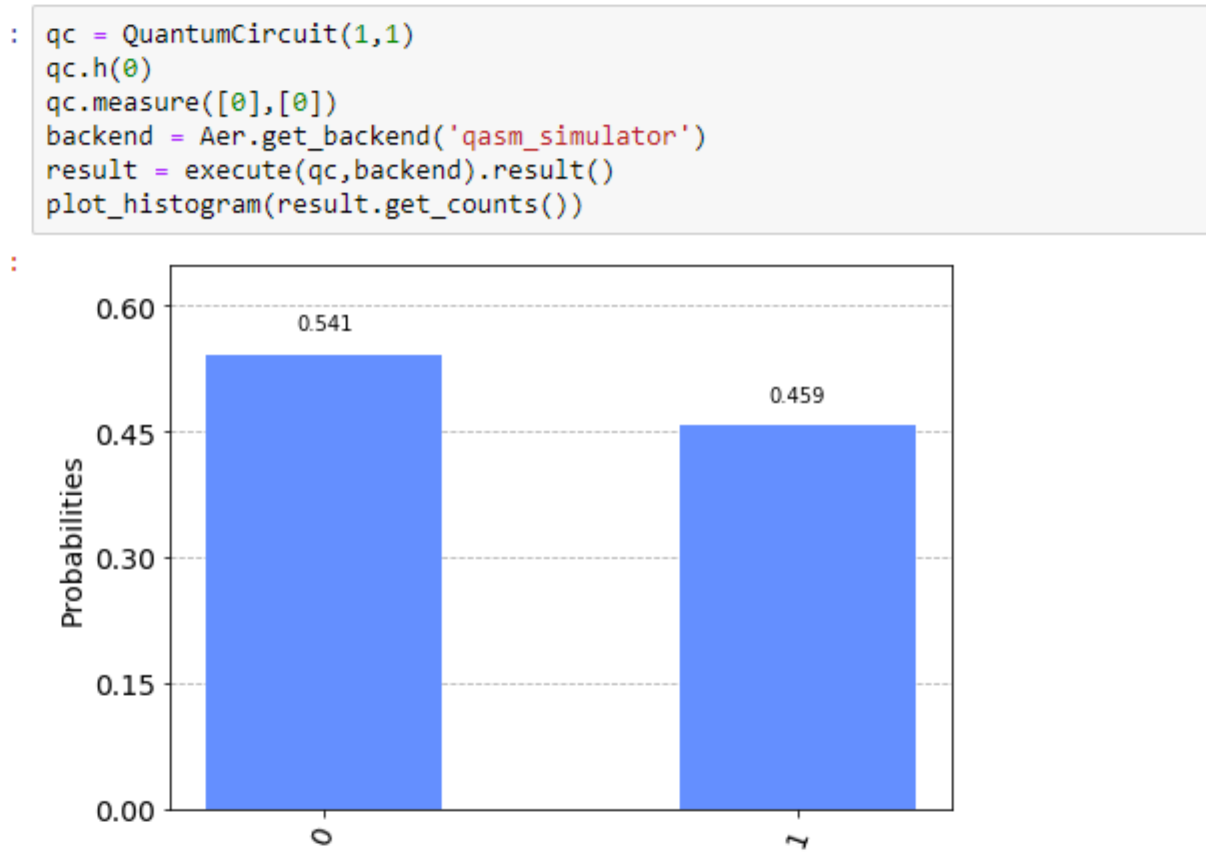
Measurement

- In a real quantum application we need to measure the result, we need classical bits that we can display
- It turns out we can only measure in the Z basis, that is get a 0 or a 1 as classical bits
- Later we will examine the hardware for doing this, for the time being just assume that it is possible
- If we have a qubit in state |0> or |1> everything is okay, we will measure the correct result, maybe with a bit of noise

Measurement

- This is not the case for qubits in other basis or superimposed
- Consider $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, if we measure this we have a 50% chance of getting |0> or |1>
- Once we have done the measurement the qubit stays in the state that we measured, it is no longer a |+>
- If we want to correctly measure |+> we need to first change the basis using the H gate and then perform the measurement

Measurement



```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.measure([0],[0])
backend = Aer.get_backend('qasm_simulator')
result = execute(qc,backend).result()
plot_histogram(result.get_counts())
```

Quantum Key Distribution

- Problem: secure communications over an insecure communications channel, want to have a private conversation without someone listening in
- Standard approach is cryptography, most secure version uses a private key, but we need some way of transmitting the key
- Our communications channel is insecure, so how can we send the key?
- The solution is quantum key distribution or QKD

Quantum Key Distribution

- We can send qubits over a standard fibre optics channel using photons, so this is practical
- If all the qubits are sent in the same basis it's easy to measure them and not be detected
- On the other hand if the sender Alice mixes up the basis, the listener will measure at least some of the qubits in the wrong basis
- Bob, the receiver will then get the wrong value 50% of the time when he measures the qubit
- The idea is to make the message long enough that the probability of detecting the listener is close to 100%

Quantum Key Distribution

- Clearly, Alice can’t send the basis sequence she is using, otherwise the listener can decode the message
- At the beginning both Alice and Bob will generate a random sequence of basis, either X or Z, these sequences will not be identical
- But, since there are only 2 choices the two sequences will agree in a significant number of places
- There are basically 5 steps in the algorithm

Quantum Key Distribution

- Step One: Alice chooses a random sequence of (classical) bits, and a random sequence of basis (either X or Z), she doesn’t share this information
- Step Two: Alice encodes the classical bits into qubits based on the random sequence of basis, bit k is encoded using basis k, the sequence of qubits is now sent to Bob
- Step Three: Bob generates a random sequence of basis and uses this to measure the qubits sent by Alice, the results in a string of classical bits

Quantum Key Distribution

- Step Four: Alice and Bob now exchange their random basis sequences, they examine the sequences basis by basis, if the two basis are the same they keep the corresponding bit, otherwise they discard the bit
- Step Five: They now have the secret key, now they need to check if someone is listening, they share a random sample of the key, if all the bits match the key is likely to be safe

Quantum Key Distribution

- Step five is the important part, if someone was listening there is a 50% chance they will measure in the wrong basis, there is then a 50% chance of Bob measuring the wrong value
- For a single bit there is a 25% chance of detecting a listener, or 0.25 as a probability
- For two bits the probability is $0.25^2=0.0625$, with n bits the probability goes to 0.25^n which rapidly becomes a small number
- Therefore, the number of bits in the random sample determines the confidence in the result

Quantum Key Distribution

- This illustrates an important property of some quantum algorithms, they are probabilistic
- We don’t have an absolute answer, but we can increase our confidence by increasing the number of bits in this case
- We aim for good enough for our application
- There are many algorithms for quantum key distribution, an important practical problem
- Can even buy hardware to do it

Road Map

- Next examine three algorithms that really aren’t very useful, no one would want to use them
- Why are we interested?
- The first examples of quantum algorithms that perform better than classical ones, from 1990s
- Showed the potential advantage of quantum computing over classical computing
- Provide a good illustration of how quantum algorithms can be superior

Deutsch-Josza Algorithm

- This algorithm solves the following problem
- We have an unknown function f, this function takes n binary parameters (either 0 or 1) and produce a binary result, either 0 or 1
- We want to determine if f is constant or balanced:
  - Constant – always returns the same value
  - Balanced – returns an equal number of 0s and 1s
- The classical algorithm is brute force trial and error
- Start by checking all the possible inputs and examine the result

Deutsch-Josza Algorithm

- The smallest number of trials is 2:
  - First trial set all the parameters to zero
  - Second trial change the first parameter to 1
- If the results are different it’s a balanced function, if they are the same we don’t know
- Now move to the second parameter and repeat, with n parameters that are 2^n possible parameter value combinations
- We don’t need to test all the possible combinations, just $2^{n-1}+1$ in the worst case

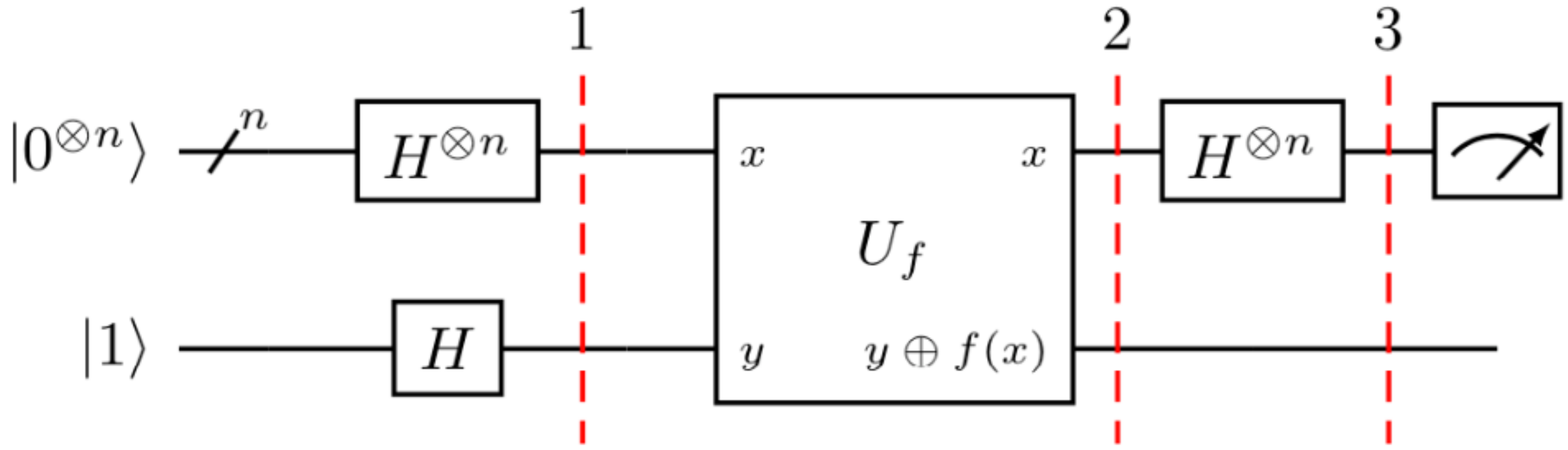
Deutsch-Josza Algorithm

- On a classical computer this is an exponential problem
- Assume that f is implemented on a quantum computer, we don’t know the implementation
- There are n+1 parameters to this implementation:
  - The first n parameters are the parameters to f, call it x
  - The last parameter is used to collect the value of f, call it y
- The output of the implement is:
  - The input x, passed unchanged
  - The value $y \oplus f(x)$ where $\oplus$ is addition mod 2

Deutsch-Josza Algorithm

- We can solve this problem with just one evaluation of f, see the next slide
- The trick is the superposition of all the possible input values, so we can essentially evaluate all of them in parallel
- So how does this work, note that we are applying Hadamard gates to all of the inputs
- If f is constant it will have no effect on its input (except possibly a global phase we can’t measure), we are applying a Hadamard to the output, so we will just measure our original input, which is 0

Deutsch-Josza Algorithm



Deutsch-Josza Algorithm

- On the other hand if the function is balanced phase kickback will add a negative phase to half the inputs, producing an output that is orthogonal to the input
- When we apply the Hadamard gate there will be at least one 1 in the output vector
- The mathematical explanation is a bit hard to follow, but will be useful in the future
- Our input state is $|0\rangle^{\otimes n}|1\rangle$

Deutsch-Josza Algorithm

- After applying the Hadamard gate we have:

$$\sum_{x\in\{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

- The sum is x, the second part is y
- Now we apply the function $|x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$

$$\sum_{x} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

- Now we need to apply the Hadamard gate to the first n results of the above

Deutsch-Josza Algorithm

- This gets a bit complicated, take it one step at a time
- Start by applying H to just one qubit in the sum:

$$H|x\rangle = \sum_z \frac{(-1)^{xz}|z\rangle}{\sqrt{2}}$$

- Expanding this to the entire vector we get:

$$H^{\otimes n}|x_1, \dots, x_n\rangle = \frac{\sum_{z_1,\dots,z_n}(-1)^{x_1 z_1 + \cdots + x_n z_n}|z_1, \dots, z_n\rangle}{\sqrt{2^n}}$$

- Or

$$H^{\otimes n}|x\rangle = \frac{\sum_z (-1)^{x \cdot z}|z\rangle}{\sqrt{2^n}}$$

Deutsch-Josza Algorithm

- Putting this all together our final result is:

$$\sum_z \sum_x \frac{(-1)^{x\cdot z+f(x)}|z\rangle}{2^n}\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

- Now we are going to measure this, recall the probability of measuring a particular value is given by the modulus of the coefficient
- Thus, the probability of measuring $|0\rangle^{\otimes n}$ is:

$$\left|\frac{1}{2^n}\sum_x (-1)^{f(x)}\right|^2$$

- This evaluates to 1 if f is constant and 0 if f is balanced, thus our result

Deutsch-Josza Algorithm

- So what have we done?
  - Started with n+1 inputs
  - Evaluate f 2^n times in parallel
  - Reduced this to n qubits
  - Measured the qubits to get a 1 bit result
- We produced a way of reducing the 2^n results to a single bit
- We need to do something similar in all of our algorithms, come up with a way of reducing this large amount of data to a simple result

Bernstein-Vazirani Algorithm

- This is similar to the previous algorithm, but maybe a bit more useful
- Again, we have a function f from n binary parameters to a single binary result
- In this case we know what the function f is:

$$f(x) = s \cdot x \; mod \; 2$$

- The only problem is we don't know the value of s, that is our problem find the value of s
- The classical solution is again a brute force one

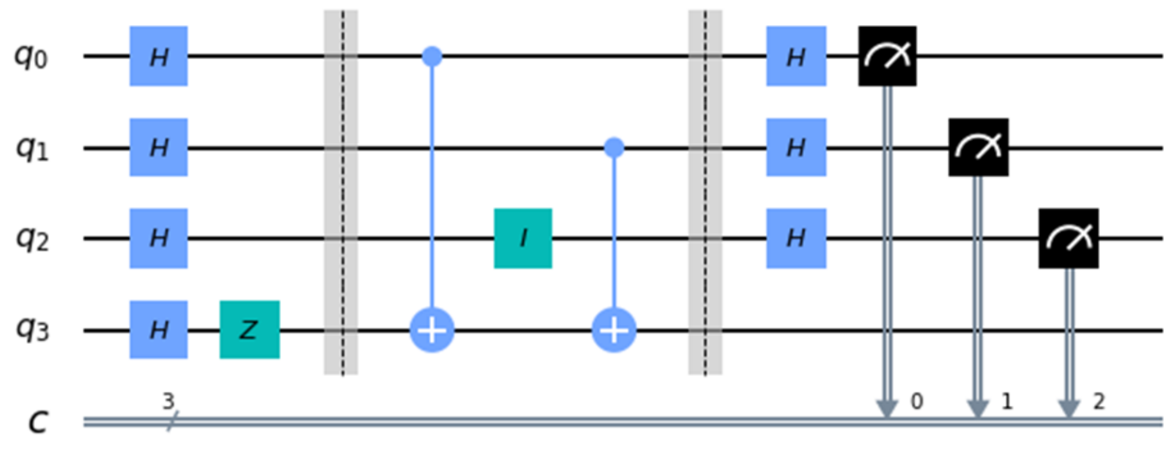
Bernstein-Vazirani Algorithm

- Classical algorithm:
    - Set all input bits to 0
    - Set first input bit to 1, evaluate f(), result tells us the first bit of s
    - Set first input bit to 0, second bit to 1, evaluate f(), etc.
    - …
    - Set last bit to 1, evaluate f(), result tells us the last bit of s
- This is an O(n) algorithm, linear, which really isn’t all that bad
- But, we can do much better with a quantum algorithm

Bernstein-Vazirani Algorithm

- The quantum algorithm starts with an input register x of n quantum bits set to |0> and one auxiliary register, y, set to |->
- The Hadamard gate is applied to x to get it in superposition
- Then f() applied to x with the output going into y
- The Hadamard gate is applied to x
- The result is measured giving the bits in s
- This requires just one execution of f(), so its an O(1) algorithm

Bernstein-Vazirani Algorithm



Bernstein-Vazirani Algorithm

- How does this work? Again phase kickback is the trick
- In general applying a Hadamard to n qubits |a> gives:

$$|a\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{a\cdot x}|x\rangle$$

- In our case the input qubits are |0>, so the -1 factor drops out giving

$$|0,\dots,0\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$

Bernstein-Vazirani Algorithm

- Using the phase kickback with f() and |-> we have the following

$$|x\rangle \xrightarrow{f} (-1)^{s \cdot x}|x\rangle$$

- Applying this to the output of the first Hadamard gates give:

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{s \cdot x}|x\rangle$$

- Recall that the Hadamard gate is its own inverse so when we apply the second one the result is |s>
- We just need to measure this to get the hidden value s

Simon’s Algorithm

- Again this looks like an abstract problem, but it leads to one of the most important quantum algorithms, which we will discuss next
- Again we have an unknown function f() with an input of n bits, and an output of n bits (it can be generalized to m bits, m >n)
- We want to know if f() is one-to-one or two-to-one
- A function is one-to-one if each input produces a unique output
- A function is two-to-one if two different inputs can produce the same output, for example $f(x) = x^2$

Simon’s Algorithm

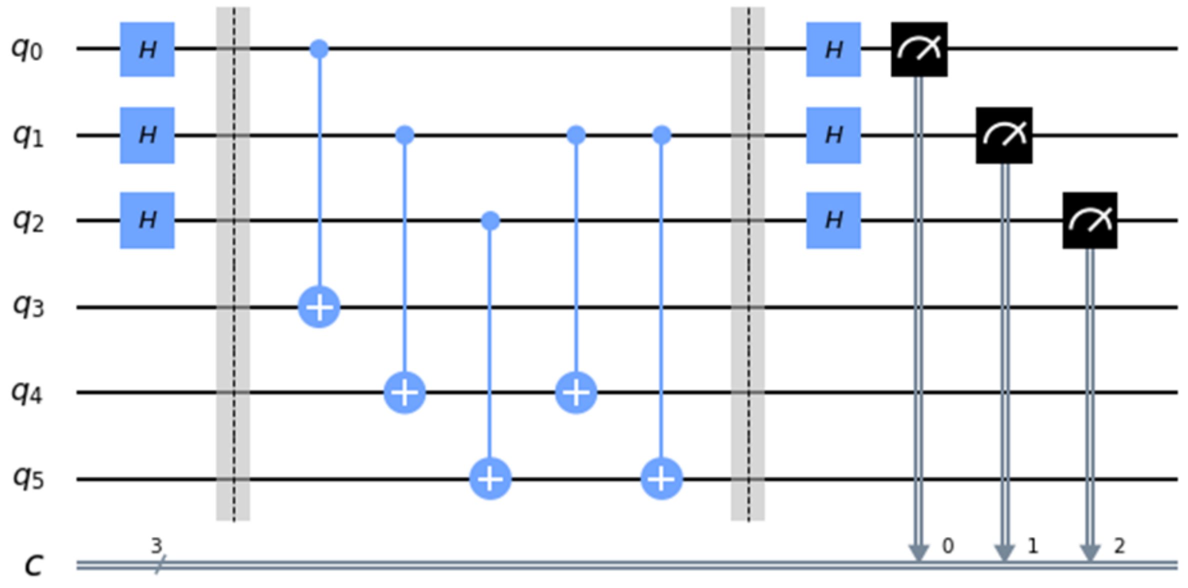
- In this case we have a special type of two-to-one function with the following property:
- $f(x_1) = f(x_2)$ if and only if $x_1 \oplus x_2 = b$, where b is called the period of the function
- The condition can also be expressed as $x_2 = x_1 \oplus b$
- The problem was originally stated as determining if f() is one-to-one or two-to-one, but the real problem is determining the value of b
- If $b = |0, \ldots, 0\rangle$ the function is one-to-one

Simon’s Algorithm

- The classical algorithm is again brute force, if the input is n bits long we need to test the first $2^{n-1}+1$ input values, since the period could be as much as 2^{n-1}
- this is the worst case, we could find it after just two or three function evaluations
- This is clearly an exponential algorithm
- The quantum algorithm is based on two n bit input registers
- Our function performs the following operation:

$$|x>|a> \rightarrow |x>|a \oplus f(x)>$$

Simon’s Algorithm



Simon's Algorithm

- In our case we initialize both registers to |0, …, 0> so we are really evaluating:

$$|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$$

- We apply the Hadamard gate to the first register, which converts our input to:

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle^{\otimes n}$$

- Now we apply f() to get

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle$$

Simon’s Algorithm

- Now we measure the second output to obtain f(x), which could be produced by two possible inputs x and y=x⊕b, with b unknown
- Therefore our first register is now

$$\frac{1}{\sqrt{2}}(|x\rangle + |y\rangle)$$

- Next we apply a Hadamard gate to this output to get

$$\frac{1}{\sqrt{2^{n+1}}}\sum_{z\in\{0,1\}^n}[(-1)^{x\cdot z} + (-1))^{y\cdot z}]|z\rangle$$

Simon’s Algorithm

- Next we do a measurement, we will get a non-zero result if

$$(-1)^{x \cdot z} = (-1)^{y \cdot z}$$

- This gives the following line of reasoning:

$$x \cdot z = y \cdot z$$
$$x \cdot z = (x \oplus b) \cdot z$$
$$x \cdot z = x \cdot z \oplus b \cdot z$$
$$b \cdot z = 0$$

- Recall this is all done mod 2
- z is the result of the measurement

Simon’s Algorithm

- If we perform the measurement n times we will get the following system of equations:

$$
\begin{gathered}
b \cdot z_1 = 0 \\
b \cdot z_2 = 0 \\
\dots \\
b \cdot z_n = 0
\end{gathered}
$$

- This is a simple linear system of equations that we can solve to get the value of b
- This gives us an exponential speedup

Quantum Fourier Transform

- The Fourier transform and its algorithms are widely used in computer science
- You use them every day, unless you live in a cave off the beaten path
- Many of the compression algorithms used in digital media are based on the Fourier transform
- Given a function f(x), its Fourier transform is:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx$$

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i x \xi} d\xi$$

Quantum Fourier Transform

- There needs to be a factor of $^1/_{2\pi}$ somewhere in these integrals
- This is not good for a computer so we have a discrete version of the Fourier transform
- Start with a vector on n complex numbers $x = (x_1, \ldots, x_n)$ and produce the transformed vector $y = (y_1, \ldots, y_n)$ in the following way:

$$y_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j e^{2\pi i \frac{jk}{n}}$$

- We get the inverse by using $-2\pi i$
- Note: the quantum computing community switch the forward and inverse transformation

Quantum Fourier Transform

- In the quantum Fourier transform the coefficients are basically the same, but we are now acting on state vectors:

$$|x\rangle \mapsto \frac{1}{\sqrt{n}} \sum_{y=0}^{n-1} e^{2\pi i \frac{xy}{n}} |y\rangle$$

- Taking this a step further we get the unitary matrix U_{QFT}

$$U_{QFT} = \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} e^{2\pi i \frac{xy}{n}} |y\rangle\langle x|$$

- So what's going on here?
- We can view this as a transformation from the Z basis (|0> and |1> to the X basis (|+> and |->)

Quantum Fourier Transform

- In the Z basis the qubits switch between |0> and |1> in a counting pattern
- But, in the X basis the qubits are rotating about the Z axis, with each qubit having a different frequency
- Qubit 0 has the lowest frequency and each qubit after that doubles in frequency
- The QisKit textbook has a nice interactive demonstration of this:

https://qiskit.org/textbook/ch-algorithms/quantum-fourier-transform.html

Quantum Fourier Transform

- We quite often view the classical Fourier transform as going from the time or space domain to the frequency domain
- We can see that the quantum Fourier transform is doing basically the same thing
- Now let’s see how we can implement this on a quantum computer
- Start with a single qubit with the input $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
- Next we apply our formulas to compute the transform

Quantum Fourier Transform

- For y_0 we have:

$$y_0 = \frac{1}{\sqrt{2}}\left(\alpha e^{2\pi i\frac{0\times 0}{2}} + \beta e^{2\pi i\frac{0\times 1}{2}}\right) = \frac{1}{\sqrt{2}}(\alpha + \beta)$$

- For y_1 we have:

$$y_1 = \frac{1}{\sqrt{2}}\left(\alpha e^{2\pi i\frac{0\times 1}{2}} + \beta e^{2\pi i\frac{1\times 1}{2}}\right) = \frac{1}{\sqrt{2}}(\alpha - \beta)$$

- So the final state is:

$$U_{QFT}|\psi\rangle = \frac{1}{\sqrt{1}}(\alpha + \beta)|0\rangle + \frac{1}{\sqrt{2}}(\alpha - \beta)|1\rangle$$

- This is the same as the Hadamard gate

Quantum Fourier Transform

- So for n=2 we know we just need a Hadamard gate
- Now what happens if we have $N=2^n$ values?
- Note: if we don’t have a power of 2 we can always pad with zeros
- The derivation from the QisKit text book is shown on the next slide
- Now we need to think about how we are going to implement this
- We clearly need a Hadamard gate, but the last line of the derivation looks like we will need to do some rotations as well

Quantum Fourier Transform

$$
\begin{aligned}
QFT_N|x\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega_N^{xy} |y\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i xy/2^n} |y\rangle \text{ since } \omega_N^{xy} = e^{2\pi i \frac{xy}{N}} \text{ and } N = 2^n \\
&= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i (\sum_{k=1}^{n} y_k/2^k) x} |y_1 \dots y_n\rangle \text{ rewriting in fractional binary notation } y = y_1 \dots y_n, y/2^n = \sum_{k=1}^{n} y_k/2^k \\
&= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{k=1}^{n} e^{2\pi i x y_k/2^k} |y_1 \dots y_n\rangle \text{ after expanding the exponential of a sum to a product of exponentials} \\
&= \frac{1}{\sqrt{N}} \bigotimes_{k=1}^{n} \left(|0\rangle + e^{2\pi i x/2^k} |1\rangle\right) \text{ after rearranging the sum and products, and expanding } \sum_{y=0}^{N-1} = \sum_{y_1=0}^{1} \sum_{y_2=0}^{1} \dots \sum_{y_n=0}^{1} \\
&= \frac{1}{\sqrt{N}} \left(|0\rangle + e^{\frac{2\pi i}{2} x} |1\rangle\right) \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^2} x} |1\rangle\right) \otimes \dots \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^{n-1}} x} |1\rangle\right) \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^n} x} |1\rangle\right)
\end{aligned}
$$

Quantum Fourier Transform

- We are going to need a controlled rotation gate CROT$_k$ defined in the following way:

$$CROT_k = \begin{bmatrix} I & 0 \\ 0 & UROT_k \end{bmatrix}$$

$$UROT_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{bmatrix}$$

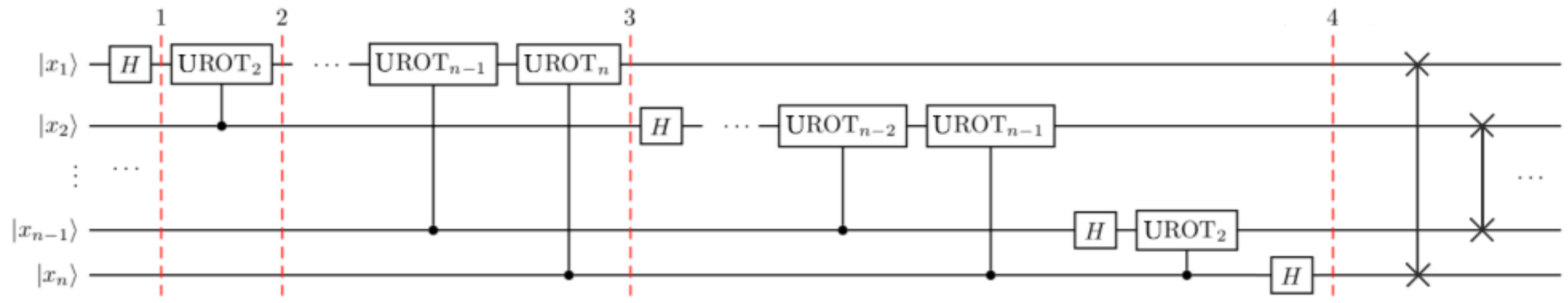
- This has the following effect on two qubits, where the left one is the control:

$$CROT_k|0x_j\rangle = |0x_j\rangle$$

$$CROT_k|1x_j\rangle = e^{\frac{2\pi i x_j}{2^k}}|1x_j\rangle$$

Quantum Fourier Transform

- The circuit for the quantum Fourier transform is:



- How does this thing work?
- We need to work through its impact on each of the bits

Quantum Fourier Transform

- Start with the first bit, the one on the top line
- After the Hadamard gate we have:

$$H_1|x_1x_2\ldots x_n\rangle = \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2}x_1\right)|1\rangle\right] \otimes |x_2x_3\ldots x_n\rangle$$

- Next we apply the UROT$_2$ gate to get the following:

$$\frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^2}x_2 + \frac{2\pi i}{2}x_1\right)|1\rangle\right] \otimes |x_2x_3\ldots x_n\rangle$$

Quantum Fourier Transform

- Finally after all the UROT gates are applied to the first bit we have:

$$\frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^n}x_n + \frac{2\pi i}{2^{n-1}}x_{n-1} + \ldots + \frac{2\pi i}{2^2}x_2 + \frac{2\pi i}{2}x_1\right)|1\rangle\right] \otimes |x_2x_3 \ldots x_n\rangle$$

- Since we have $x = 2^{n-1}x_1 + 2^{n-2}x_2 + \ldots + 2^1x_{n-1} + 2^0x_n$
- This simplifies to:

$$\frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^n}x\right)|1\rangle\right] \otimes |x_2x_3 \ldots x_n\rangle$$

Quantum Fourier Transform

- After going through all of the bits we have:

$$\frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^n}x\right)|1\rangle\right] \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^{n-1}}x\right)|1\rangle\right] \otimes \ldots$$
$$\otimes \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^2}x\right)|1\rangle\right] \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^1}x\right)|1\rangle\right]$$

- The final expression we had for the QFT was

$$= \frac{1}{\sqrt{N}}\left(|0\rangle + e^{\frac{2\pi i}{2}x}|1\rangle\right) \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^2}x}|1\rangle\right) \otimes \ldots \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^{n-1}}x}|1\rangle\right)$$
$$\otimes \left(|0\rangle + e^{\frac{2\pi i}{2^n}x}|1\rangle\right)$$

- This is the same as what we had before, but the bits are reversed

Quantum Fourier Transform

- This explains the very last part of the circuit which swaps the bits so they are in the correct order
- The quantum Fourier transform is faster than the best known classical algorithm, but there is a problem with it, the IO problem
- We are not interested in the Fourier transform of individual bits, but real and complex numbers
- We could encode our input as the coefficients of the qubits, but this would involve some very tricky circuits

Quantum Fourier Transform

- Our output will then be encoded in the coefficients of the qubits at the end of the circuit
- But, we have no way of retrieving these coefficients, they are the probabilities of get a |0> or |1> when measuring the bits
- If we repeated the computation many times we could estimate these probabilities and then use that to retrieve the result
- But, this only works for the real version of the transform
- But this could end up being slower than the classical algorithms, so really not much of a solution

Quantum Fourier Transform

- On its own the quantum Fourier transform isn’t that useful
- But, we will see that its an important building block for useful algorithms
- In particular the inverse transform is very useful, which is closely related and can be programmed in the same way

Summary

- Examined four fundamental quantum algorithms
- None of these algorithms are particularly useful on their own, but they illustrate how quantum algorithms are developed
- Several of them are of theoretical importance since they show that quantum computers could be faster than classical computers
- With this background we can examine the key quantum algorithms in the next part of the course