Quantum Algorithms: Circuit Model

Course Info

Meetings: Fri 15:00-18:00

Meetings will include lecture and tutorial time.

Instructor: Troy Lee

Contact me: troyjlee@gmail.com

Grading

§ Quizzes: 30% of your grade

- Short 10 min understanding checks
- Given on Canvas, due every week before lecture

Problem Sets

§ Problem Sets: **30% of your grade**

- More challenging problems
- Can work in groups but submit individually

Problem Set 1	12 March 2021
Problem Set 2	2 April 2021
Problem Set 3	30 April 2021

Project

§ Project: **40% of your grade**

- Prepare a writeup and presentation about a quantum algorithms paper/topic not discussed in lecture. 80% Writeup, 20% Presentation.
- Can work individually or in groups of up to 3.
- Suggested topics on course website.

Topic selection	19 March
1 page summary	23 April
Written report due	14 May
Presentations	21 May

Resources

§ Other Courses

- Ronald de Wolf's course notes: arXiv1907.09415

- Andrew Child's course notes: http://www.cs.umd.edu/~amchilds/qa/

- Ryan O'Donnell on YouTube and https://www.cs.cmu.edu/~odonnell/quantum15

- Umesh Vazirani's course http://people.eecs.berkeley.edu/~vazirani/f04quantum/quantum.html

Quantum Basics

States

A quantum state is a unit vector in an N-dimensional space.

$$|\psi\rangle = \sum_{i=1}^{N} \alpha_i |i\rangle$$

Each $\alpha_i \in \mathbb{C}$ and $\sum_{i=1}^{N} |\alpha_i|^2 = 1$.

When we measure $|\psi\rangle$ in the computational basis we see outcome $j \in [N]$ with probability $|\alpha_j|^2$ and in this case $|\psi\rangle$ collapses to state $|j\rangle$.

States

Most typically our basis vectors will be labeled by strings $x \in \{0,1\}^n$.

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

This is a 2^n dimensional vector that we call an n-qubit state.

Projective Measurements

An orthogonal projector P is a Hermitian matrix ($P^* = P$) such that $P^2 = P$.

A projective measurement is specified by a set of matrices $P_1, \ldots, P_m$ where

- Each P_i is an orthogonal projector.
- $\sum_{i=1}^{m} P_i = \mathbb{I}$.

Problem Set: Show that this means $P_i P_j = 0$ for all $i \neq j$.

Projective Measurements

When we measure according to $P_1, \ldots, P_m$ we see outcome j with probability $\|P_j|\psi\rangle\|^2$ and in this case $|\psi\rangle$ collapses to

$$\frac{P_j|\psi\rangle}{\|P_j|\psi\rangle\|}$$

Exercise: Show that $\sum_j \|P_j|\psi\rangle\|^2 = 1$

Answer

$$1 = \| \sum_j P_j |\psi\rangle \|^2$$

$$= \sum_{i,j} \langle \psi | P_i P_j | \psi \rangle$$

$$= \sum_j \langle \psi | P_j^2 | \psi \rangle$$

$$= \sum_j \| P_j |\psi\rangle \|^2$$

$$\sum_j P_j = \mathbb{I}$$

$$P_i P_j = \mathbf{0}$$

$$i \neq j$$

Projective Measurements

Measuring in the computational basis is an example of a projective measurement where $P_i = |i\rangle\langle i|$ for $i = 1, \ldots, N$.

We won't need more general measurements or density matrix formalism.

Operations

Quantum operations are linear transformations and must map unit vectors to unit vectors.

Such transformations are unitary transformations.

An N-by-N matrix U is unitary iff $UU^* = \mathbb{I}$ where the star indicates conjugate transpose.

In other words, the rows of U form an orthonormal set of vectors.

Complexity

In this course we want to talk about the cost of quantum algorithms.

In order to do this, we need a way to measure the cost of implementing a unitary.

We will follow the lead of classical complexity theory and look at implementing unitaries via circuits.

Circuits

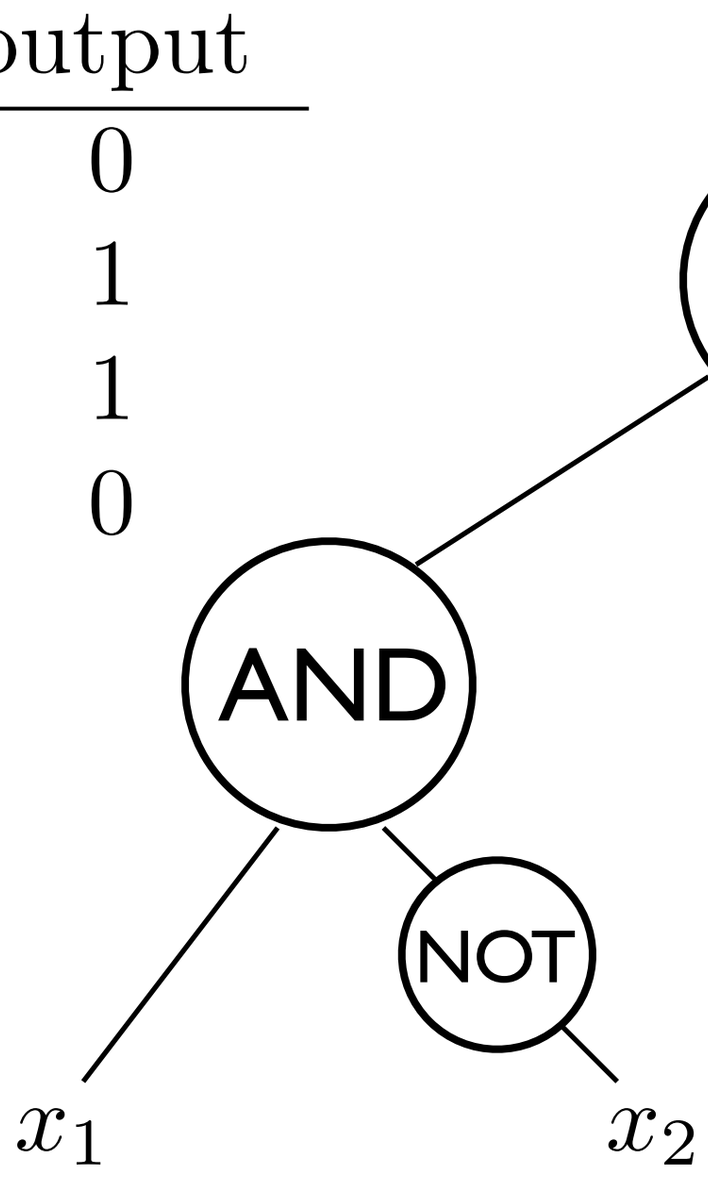
Outline

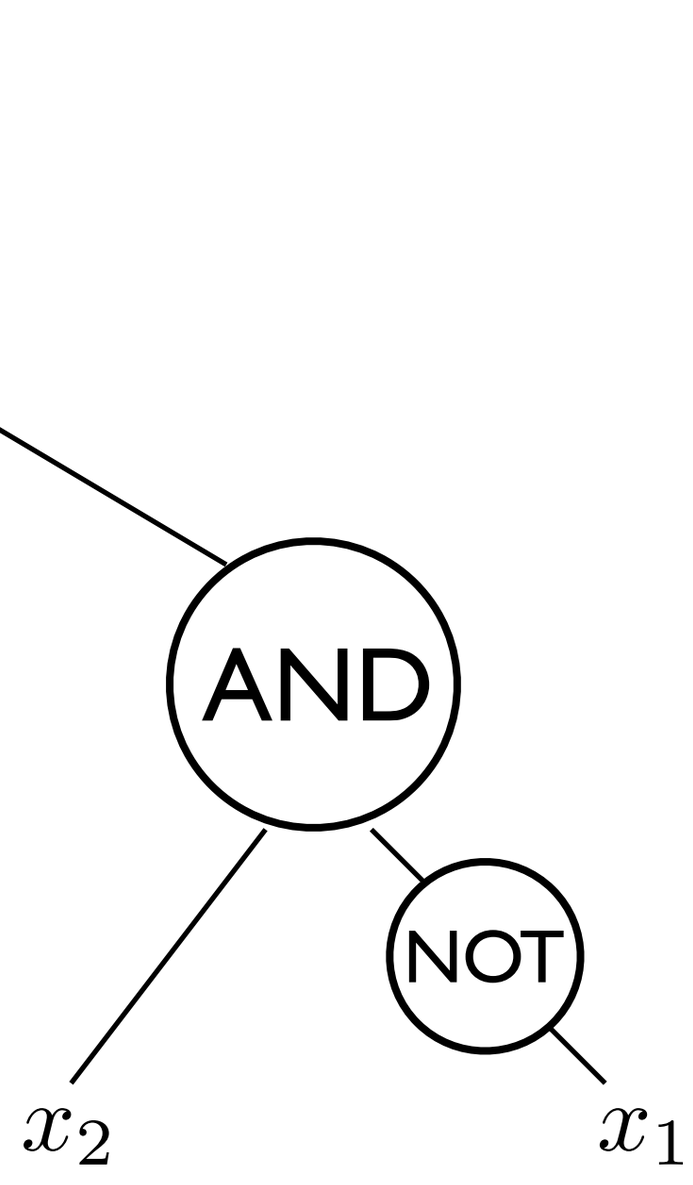
We are going to build up to quantum circuits via a classical detour.

- Classical AND/OR/NOT circuits.
- Classical reversible circuits.
- Classical randomized circuits.
- Quantum circuits.

Classical Circuits

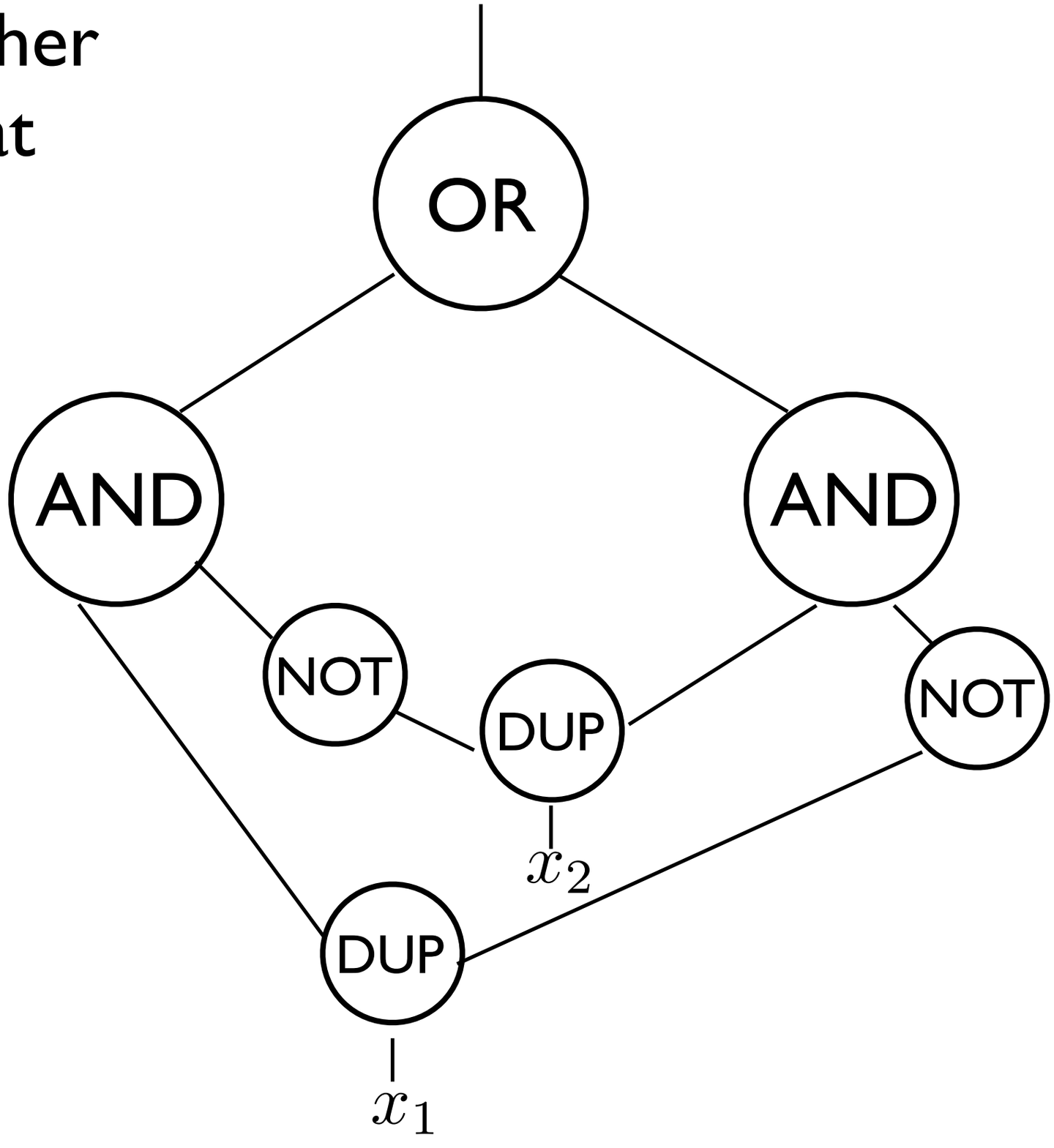
input	output
00	0
01	1
10	1
11	0





Classical Circuits

We add another gate DUP that does fanout.



Disjunctive Normal Form

Any Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be constructed out of AND,OR,NOT and DUP gates.

One way to see this is the Disjunctive Normal Form.

For every $x \in \{0,1\}^n$ there is a clause that evaluates to one on and only on x.

$x_1 \wedge \neg x_2 \wedge x_3$ evaluates to one only on the string 101

We can take the OR of all such conjunctions for the strings on which f evaluates to one.

What is efficient?

The number of gates in a disjunctive normal form is $O(n2^n)$.

Most Boolean functions require at least $\frac{2^n}{3n}$ gates [Shannon].

A function (family) is considered efficiently computable if it can be constructed with n^k gates for constant k.

Reversible Circuits

In a reversible circuit every gate is invertible.

In particular, the number of inputs equals the number of outputs.

AND and OR are not invertible, so we need to find new basic gates.

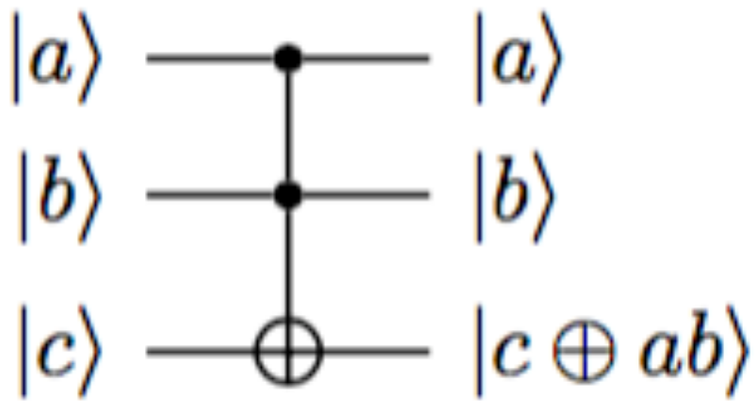
Reversible circuits were studied in the 1960's and 70's as a means to reduce energy dissipation in circuits.

Toffoli Gate

The Toffoli gate or controlled controlled NOT (CCNOT) gate acts on 3 bits and has the following behavior.

input	output
000	0
001	1
010	0
011	1
100	0
101	1
110	1
111	0

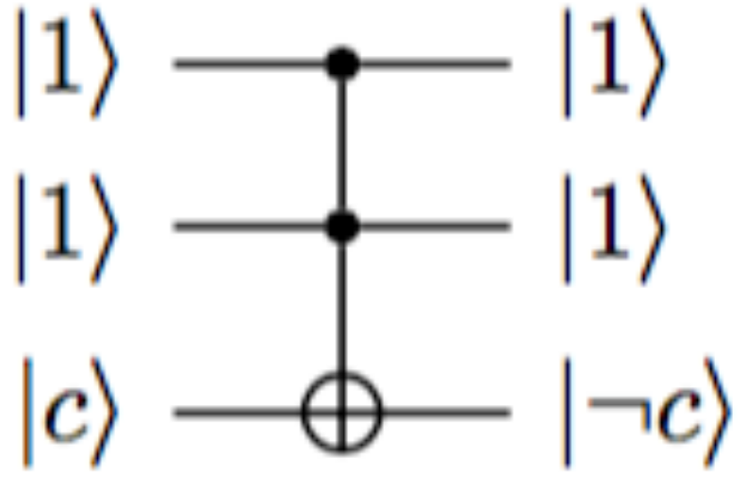
If the first two bits are one then negate the third bit.



Ancilla bits

For reversible circuits we allow "extra" wires that carry hard-coded bits.

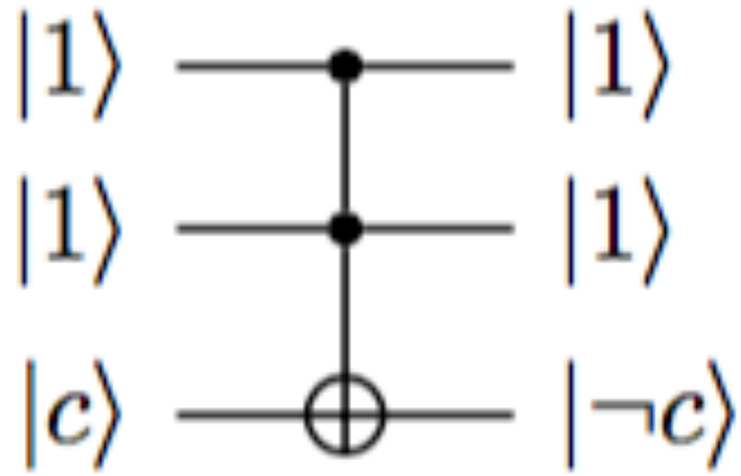
Using ancilla bits we see that a Toffoli gate can compute NOT.



We hard-code the first two inputs to be one.

Garbage bits

For reversible circuits we will also (potentially) only be interested in some of the outputs.

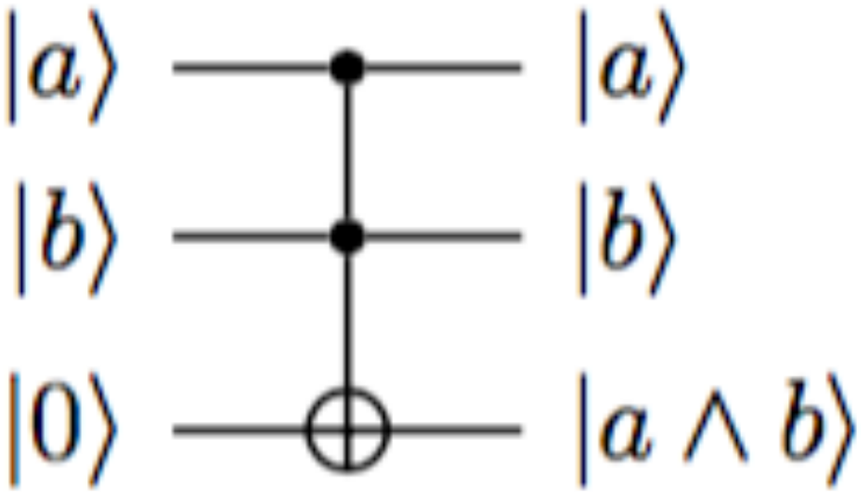


We read some of the output bits for our answer. The rest are "garbage" bits.

For NOT the answer is given in the 3rd bit.

Toffoli Gate

Allowing ancilla bits and garbage bits, there is a reversible circuit using only Toffoli gates that simulates any AND, OR, NOT and DUP circuit.

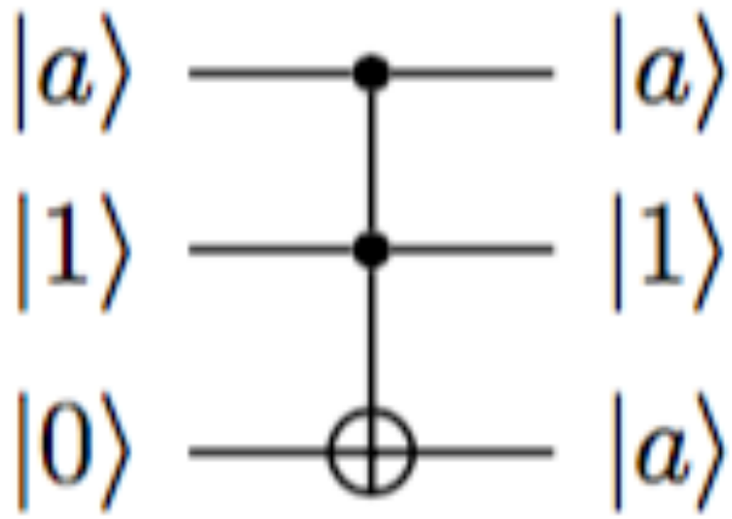


Computes AND(a,b) in the third bit of the output.

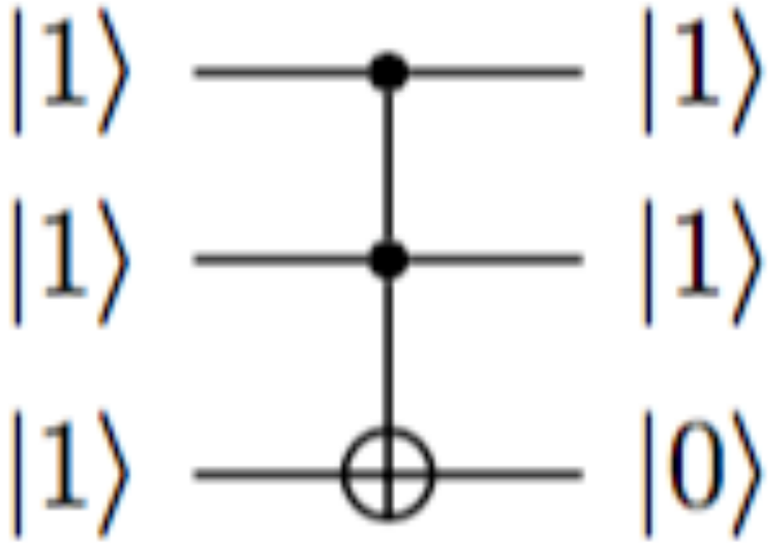
With AND and NOT we can compute OR as well by De Morgan's law:

$$a \vee b = \neg(\neg a \wedge \neg b)$$

Finally we can also simulate DUP gates.



Note: we have used ancilla bits set to $|0\rangle$ and $|1\rangle$.
We can create $|0\rangle$ from $|1\rangle$ ancillas, however, thus we may assume all ancilla bits are set to $|1\rangle$.



Classical Reversible Circuit

A classical reversible circuit thus looks as follows:

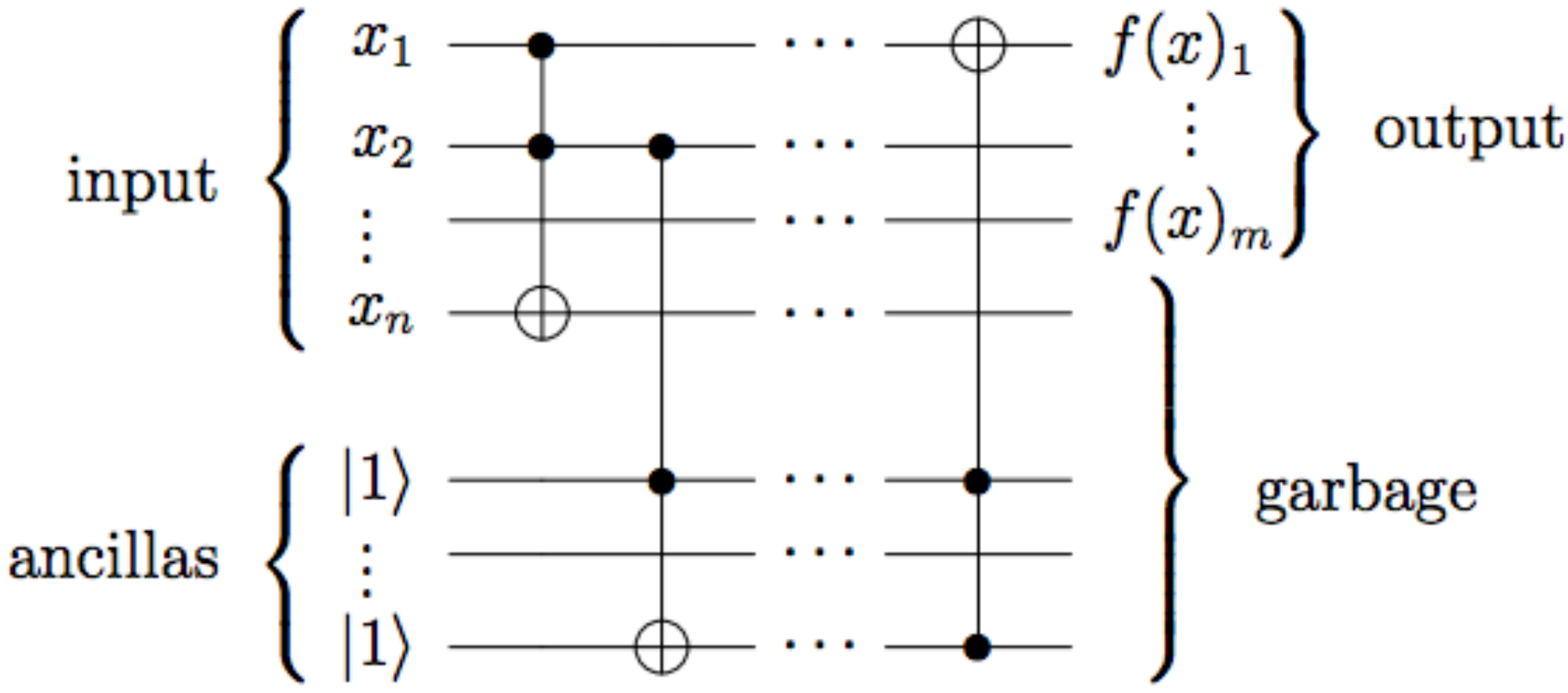
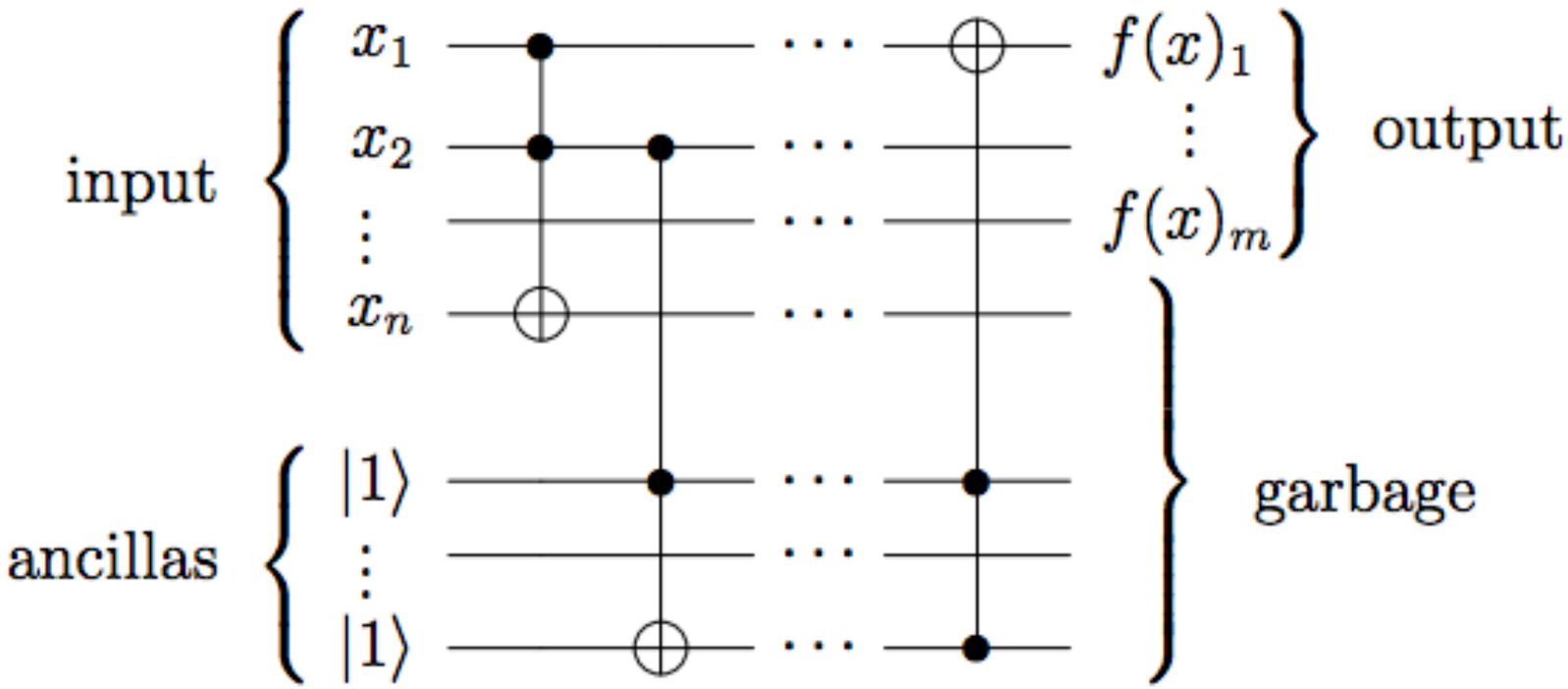


image from https://www.cs.cmu.edu/~odonnell/quantum15/lecture01.pdf

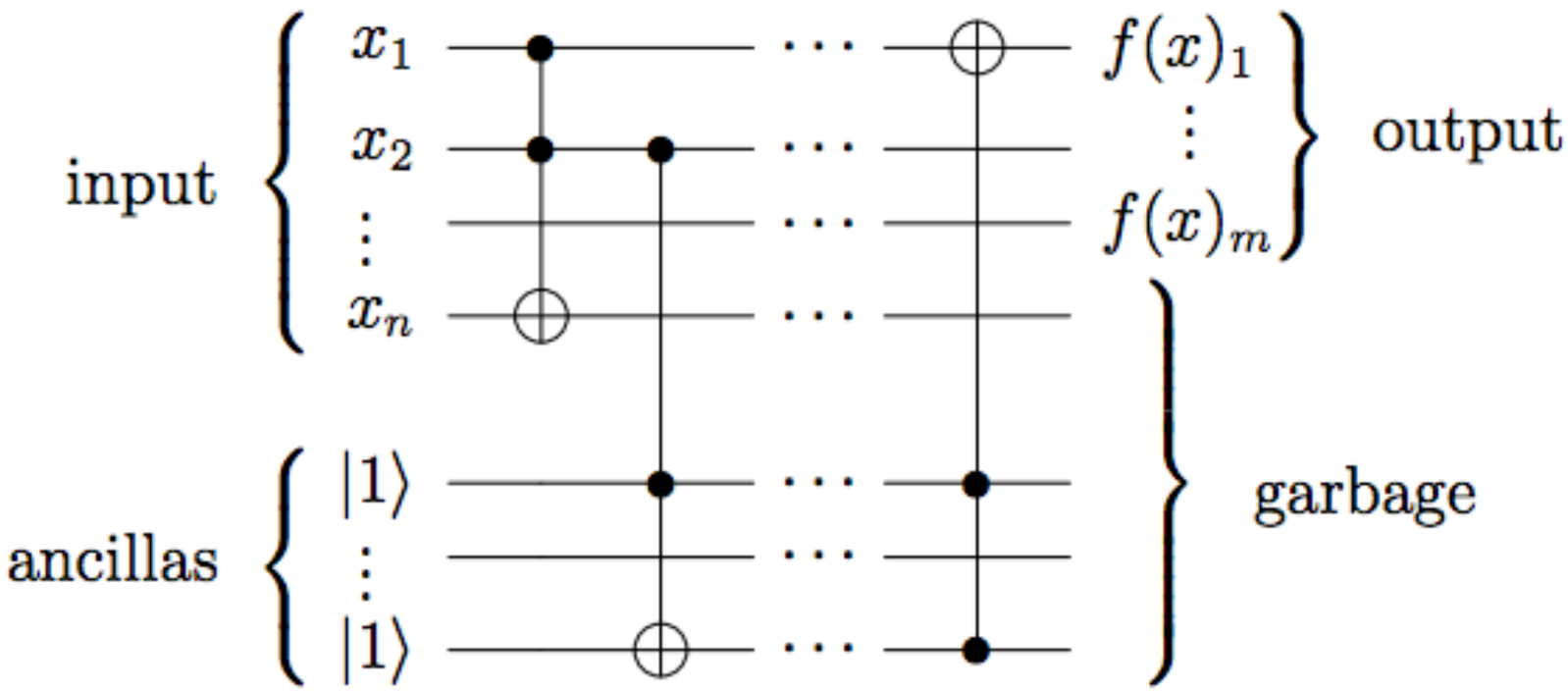
Reading a circuit



What we see from this circuit

- Input is in input registers $1, \ldots, n$
- Ancillas in input registers $n+1, \ldots, n+c$ initialized to $|1\rangle$.
- Read output in output registers $1, \ldots, m$

Reading a circuit



Then reading from left to right we obtain a sequence of operations:

- $x_n \leftarrow \mathrm{CCNOT}(x_1, x_2, x_n)$
- $x_{n+c} \leftarrow \mathrm{CCNOT}(x_2, x_{n+1}, x_{n+c})$

$\ldots$

- $x_1 \leftarrow \mathrm{CCNOT}(x_{n+1}, x_{n+c}, x_1)$

Randomized circuits

Now we discuss adding randomization to our circuits.

We augment the circuit with a "coin toss" gate that outputs zero or one uniformly at random.

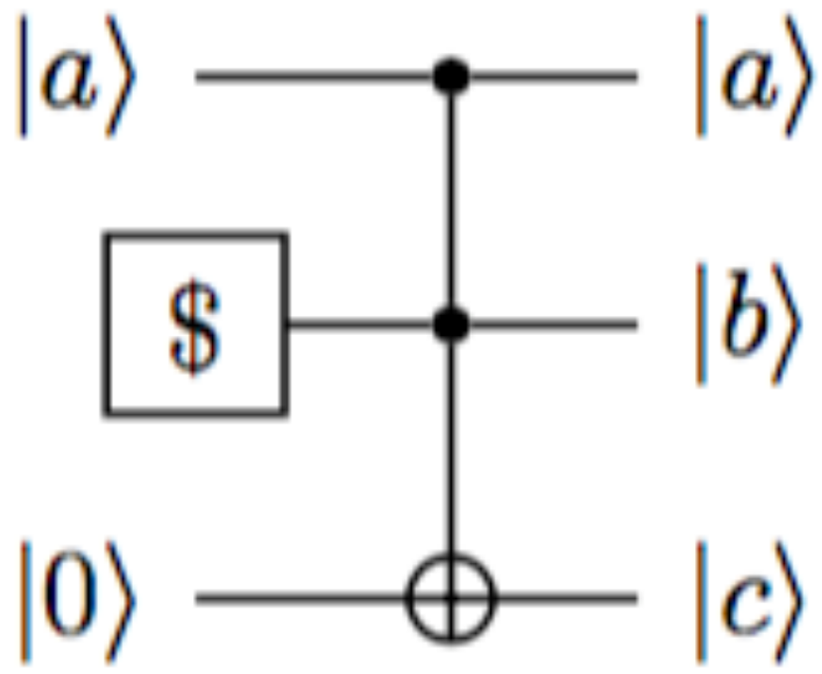
We can push all coin tosses to the very beginning of the algorithm.

We thus imagine a circuit that has n many input wires, c many ancilla wires and r-many coin toss wires.

Randomized circuits

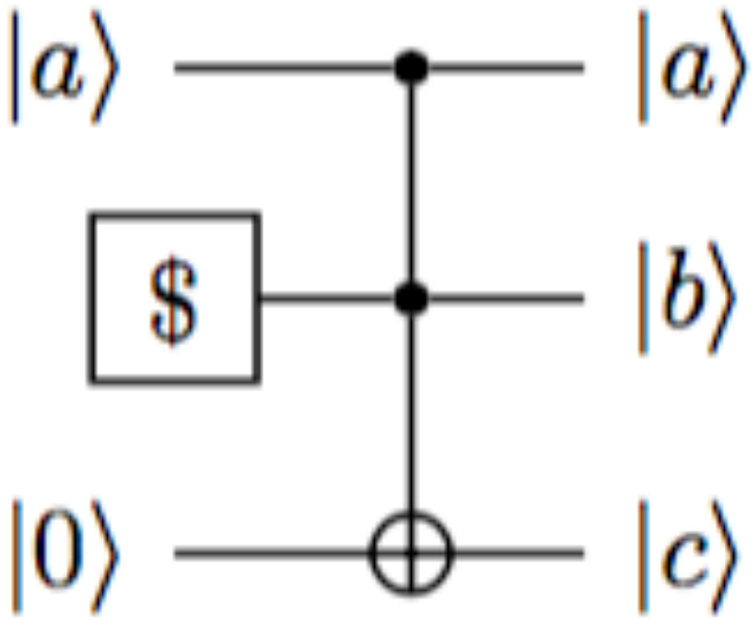
Let's play with randomized circuits to build up to analyzing quantum circuits.

What is the output $|c\rangle$ of this circuit?



Controlled Coin

What is the output $|c\rangle$ of this circuit?



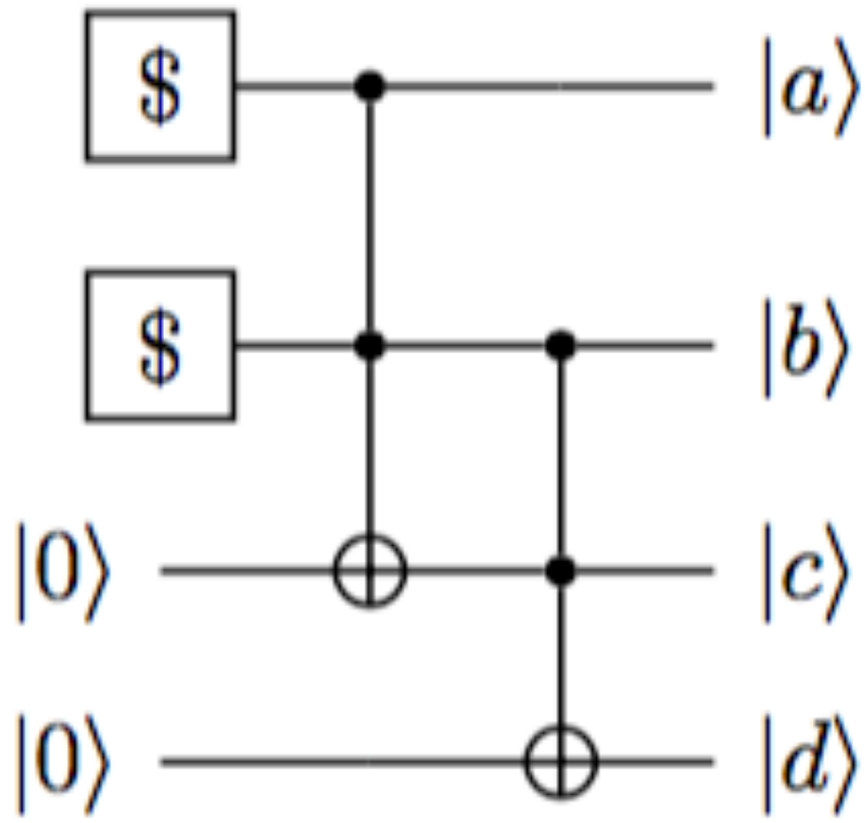
We can think of this as a controlled coin gate.

If $|a\rangle = |0\rangle$ then $|c\rangle = |0\rangle$.

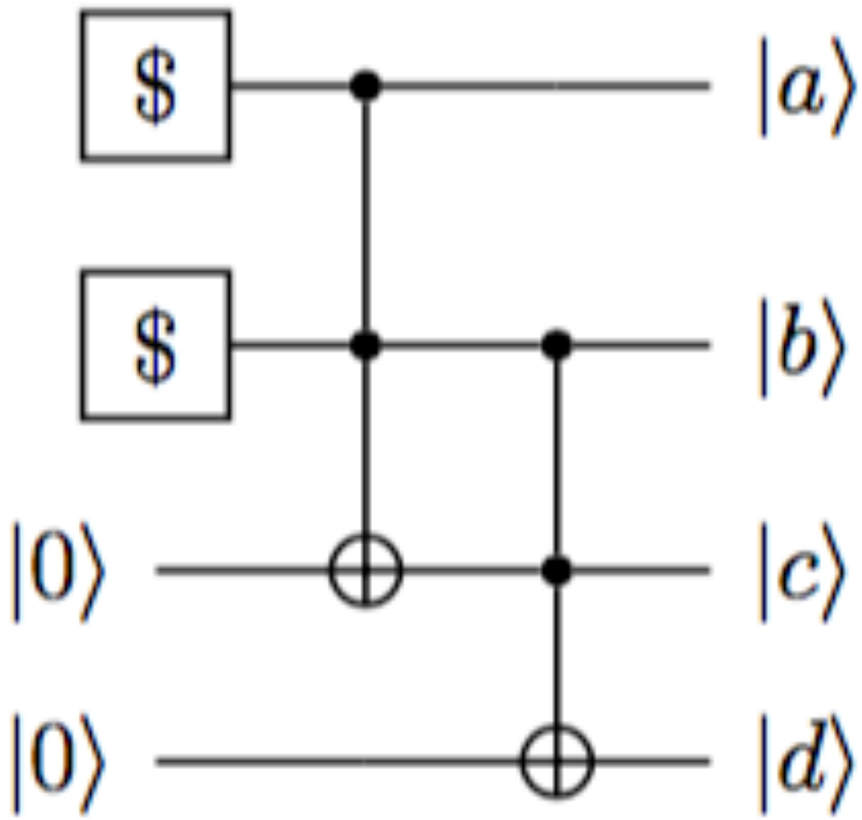
If $|a\rangle = |1\rangle$ then $|c\rangle = |0\rangle$ with prob. 1/2 and
$|c\rangle = |1\rangle$ with prob. 1/2.

Example

What is the probability distribution over outputs in this circuit?



The circuit is linear so it suffices to analyze it separately for each input.



Input	Prob	Output
$\|0000\rangle$	$\frac{1}{4}$	$\|0000\rangle$
$\|0100\rangle$	$\frac{1}{4}$	$\|0100\rangle$
$\|1000\rangle$	$\frac{1}{4}$	$\|1000\rangle$
$\|1100\rangle$	$\frac{1}{4}$	$\|1111\rangle$

We can describe the output as the "superposition"

$$\frac{1}{4} \cdot |0000\rangle + \frac{1}{4} \cdot |0100\rangle + \frac{1}{4} \cdot |1000\rangle + \frac{1}{4} \cdot |1111\rangle$$

Quantum Circuits

What should we choose as elementary gates for quantum circuits?

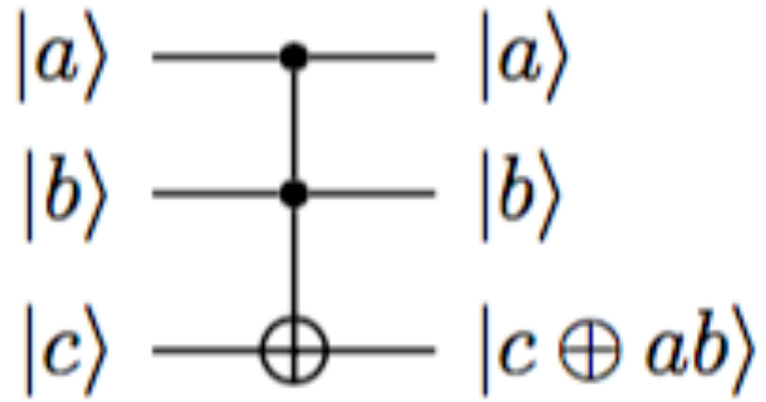
Just for one qubit gates there are already continuously many possibilities.

Desirable properties:

- Finite gate set.
- Generalizes classical reversible and randomized circuits.
- "Universality": any unitary can be approximated by a large enough circuit.

Gate set

To generalize classical reversible computation it is natural to include the CCNOT gate.

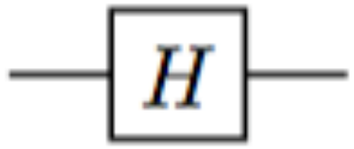


action on basis states

To generalize randomized circuits we also need to simulate a coin toss. How can we do this?

Hadamard

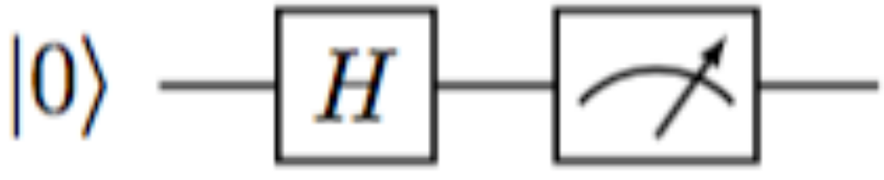
Hadamard gate:



$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$



This returns a uniformly random bit, i.e. simulates a coin toss.

Deferred measurement

This returns a uniformly random bit, i.e. simulates a coin gate.

One drawback to this simulation is that it requires measurement in the middle of the computation.

On the problem set you will show how to simulate randomized circuits even deferring all measurements to the end.

Universality

Hadamard and Toffoli gates can efficiently approximate a quantum circuit on any other gate set.

Gate: Unitary on < 4 qubits.

Efficiently: $O(m \cdot \mathrm{polylog}(n, m, 1/\varepsilon))$ Hadamards and Toffolis to approximate a circuit on n qubits with m gates to error ε.

Approximate: Allowed to add extra ancillas. For any measurement on other circuit there is a measurement on H,T circuit close in variation distance.

Other gates

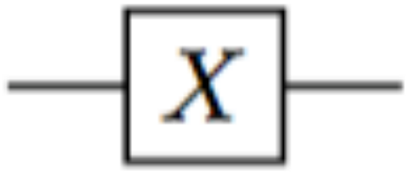
The interest of this to us is that there is a simple finite gate set that is universal.

We will not slavishly translate all algorithms into only using Hadamards and Toffolis.

So let us look at some other useful gates.

One qubit gates

X gate:



$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

NOT

Z gate:

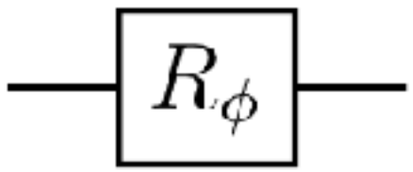
$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

phase flip

This multiplies the phase of the $|1\rangle$ qubit by -1.

One qubit gates

Phase shift:



$$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$

This multiplies the phase of the $|1\rangle$ qubit by $e^{i\phi}$.

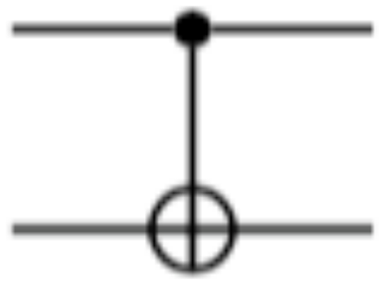
Of particular interest is the $T = R_{\pi/4}$ gate.

The set $\mathcal{G} = \{H, T, \mathrm{CNOT}\}$ is dense in the set of all unitaries.

CNOT gate

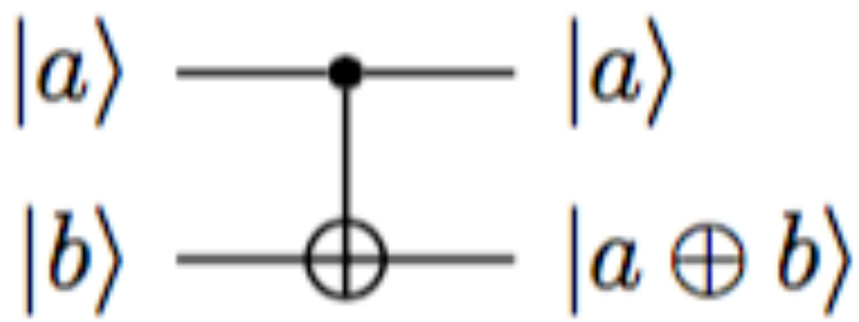
CNOT:

2-qubit gate



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

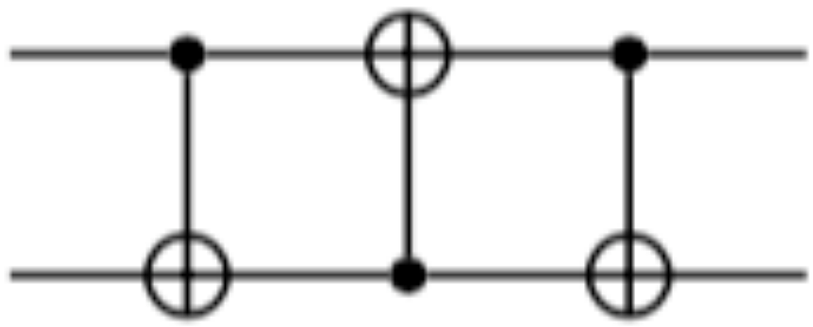
If the first qubit is $|0\rangle$ do nothing. If it is $|1\rangle$ then flip the second qubit.



action on basis states

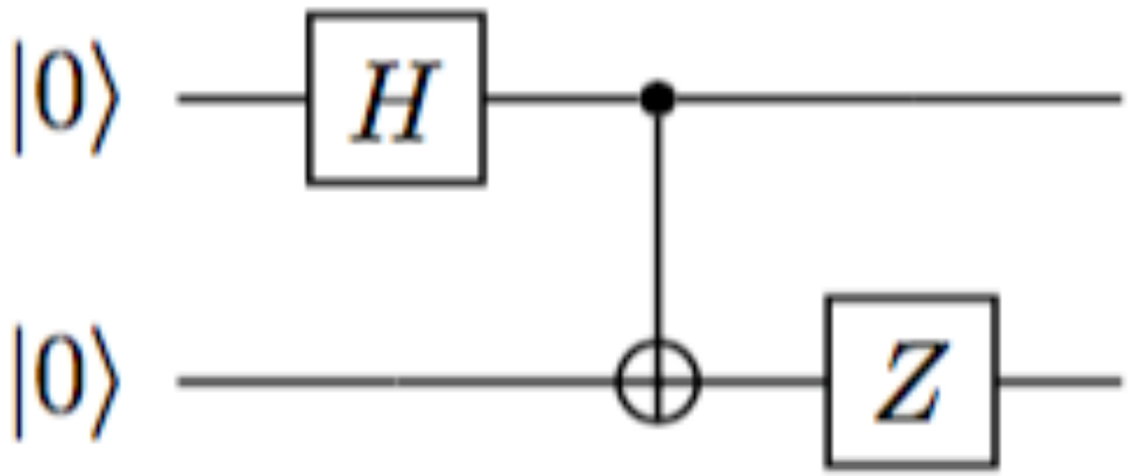
Example Circuit

What does this circuit do?



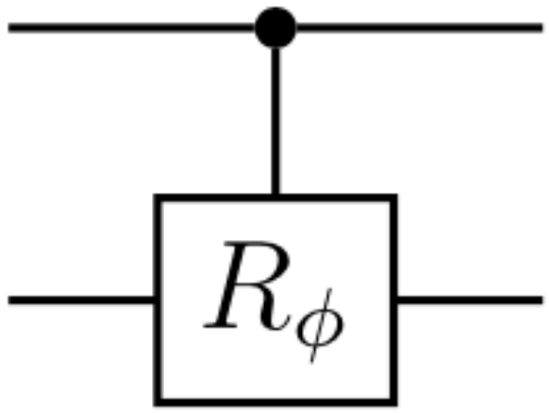
Example Circuit

What is the output of this circuit?



Other controlled gates

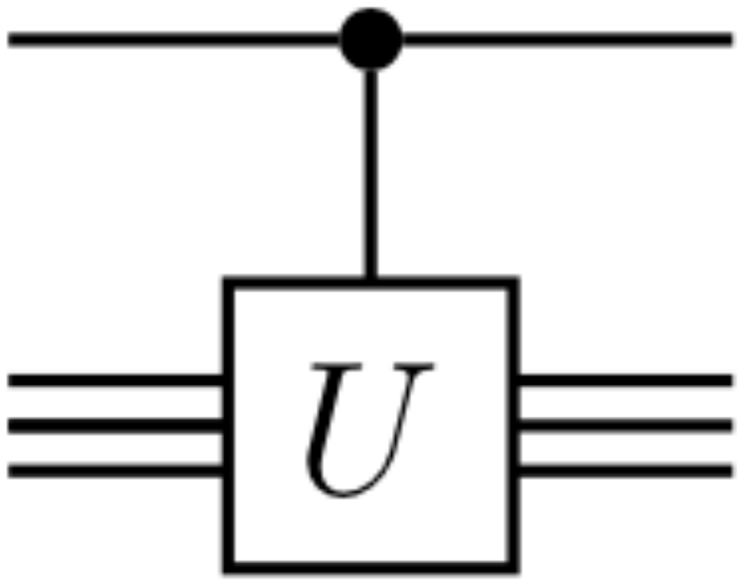
A gate we will use for the Fourier transform is the controlled phase shift



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix}$$

Other controlled gates

For a general N-by-N unitary we can also consider a controlled U gate.

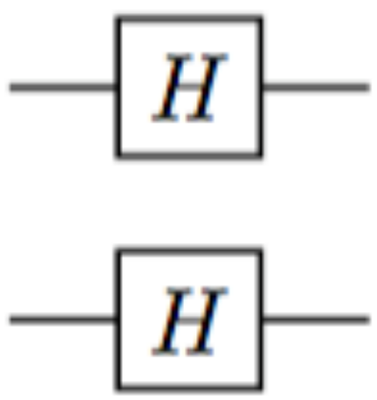


$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & I_N & \mathbf{0}_N \\ 0 & 0 & \mathbf{0}_N & U \end{bmatrix}$$

Circuit Notation

A circuit can equivalently be translated into a list of unitaries to apply to the initial state.

Gates in the same vertical slice correspond to applying the tensor product of the operations. Wires with no gates we take the identity.



$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Efficiency

As in the classical case, when the input is n bits we consider a quantum circuit efficient if the number of gates is polynomial in n .

Basic Algorithms

Early Quantum Algorithms

1992: Deutsch-Josza algorithm

1993: Bernstein-Vazirani algorithm

1993: Simon's algorithm

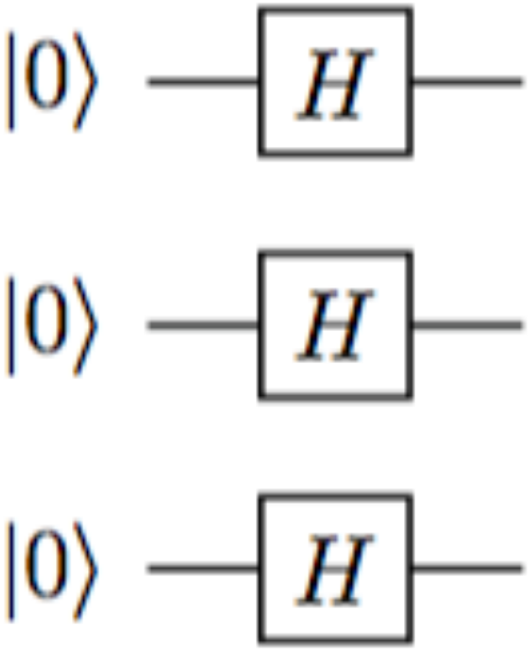
1994: Shor's factoring algorithm

1996: Grover's search algorithm

We will cover all these algorithms in this order!

Hadamards

Let us look at an extremely common starting point of quantum algorithms.



The resulting state is

$$
\begin{aligned}
H^{\otimes 3}|000\rangle &= (H|0\rangle)^{\otimes 3} \\
&= \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes 3} \\
&= \frac{1}{\sqrt{8}} \sum_{x \in \{0,1\}^3} |x\rangle.
\end{aligned}
$$

This gives us the uniform superposition.

Hadamards

By the same argument

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

And for $x \in \{0,1\}^n$

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \bigotimes_{j \in [n]} (|0\rangle + (-1)^{x_j}|1\rangle)$$

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

Oracle model

Quantum algorithms are often studied in a black-box or oracle model.

We want to determine some property of a function

$$f : \{0,1\}^n \to \{0,1\}$$

And we can use the unitary map O_f where

$$O_f|x\rangle|b\rangle = |x\rangle|b \oplus f(x)\rangle$$

for $x \in \{0,1\}^n$ and $b \in \{0,1\}$.

Querying the oracle

We can see the potential of quantum algorithms by creating the uniform superposition and querying the oracle.

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

However, we still have to be clever to figure out how to use this state.

By measuring we just get a random value of f .

Phase kickback trick

We now cover another useful trick you often see in quantum algorithms.

Sometimes it is more convenient to compute the function f in the phase.

$$O_{f,\pm}|x\rangle|b\rangle = (-1)^{b\cdot f(x)}|x\rangle|b\rangle$$

We can easily implement this using O_f.

Phase kickback trick

Step 1: $$(I \otimes H)|x\rangle|b\rangle = \frac{1}{\sqrt{2}}|x\rangle(|0\rangle + (-1)^b|1\rangle)$$

Step 2: $$O_f \frac{1}{\sqrt{2}}|x\rangle(|0\rangle + (-1)^b|1\rangle)$$

$$= \frac{1}{\sqrt{2}}|x\rangle(|f(x)\rangle + (-1)^b|1 - f(x)\rangle$$

$$= (-1)^{b\cdot f(x)} \frac{1}{\sqrt{2}}|x\rangle(|0\rangle + (-1)^b|1\rangle)$$

Step 3: $$(I \otimes H)(-1)^{b\cdot f(x)} \frac{1}{\sqrt{2}}|x\rangle(|0\rangle + (-1)^b|1\rangle)$$

$$= (-1)^{b\cdot f(x)}|x\rangle|b\rangle$$

Deutsch-Josza Algorithm

This is one of the earliest quantum algorithms, and showed a separation between quantum and deterministic query complexity.

We are given access to an oracle O_f for a function $f : \{0,1\}^n \to \{0,1\}$ and are promised that either f is:

1) A constant function (all zero or all one).

2) Balanced, i.e. takes the value zero as often as one.

The task is determine which is the case.

Deutsch-Josza Algorithm

Question: How could you solve this problem with a randomized algorithm?

Question: How many times does a deterministic algorithm need to query the oracle to solve this problem?

Deutsch-Josza Algorithm

Step 1: $H^{\otimes(n+1)}|0^n\rangle|1\rangle$

$$= \frac{1}{\sqrt{2^{n+1}}} \left(\sum_{x \in \{0,1\}^n} |x\rangle \right) (|0\rangle - |1\rangle)$$

Step 2: Apply O_f

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

Step 2: Apply $H^{\otimes(n+1)}$

Step 2: Apply O_f

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

Step 3: Apply $H^{\otimes(n+1)}$

If f is the constant zero function then we get

$$|0^n\rangle|1\rangle$$

If f is the constant one function then we get

$$-|0^n\rangle|1\rangle$$

In both cases measuring gives $|0^n\rangle|1\rangle$ with certainty.

Step 2: Apply O_f

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

Step 3: Apply $H^{\otimes(n+1)}$

Now suppose f is balanced.

Recall: $H^{\otimes n}|x\rangle=\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle$

Thus we get:

$$\frac{1}{2^n}\left(\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle\right)|1\rangle$$

Thus we get:

$$\frac{1}{2^n}\left(\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle\right)|1\rangle$$

Look at the amplitude on the state $|0^n\rangle|1\rangle$.

It is zero when f is balanced!

Thus we never see $|0^n\rangle|1\rangle$ upon measuring in this case and decide with certainty whether f is constant or balanced with one application of O_f .

Bernstein-Vazirani

Let's consider a different promise on $f : \{0,1\}^n \to \{0,1\}$.

Suppose that there is an $s \in \{0,1\}^n$ such that

$$f(x) = x \cdot s \bmod 2$$

We are given access to O_f for some such f, and now the goal is to determine s.

The algorithm is exactly the same!

$$H^{\otimes(n+1)} O_f H^{\otimes(n+1)} |0^n\rangle|1\rangle$$

Bernstein-Vazirani

After $O_f H^{\otimes(n+1)}|0^n\rangle|1\rangle$ we are in the state

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

Next we apply $H^{\otimes n+1}$. For exposition, let's first do $I^{\otimes n}\otimes H$ to get the state

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

Bernstein-Vazirani

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

Think about $H^{\otimes n} \frac{1}{\sqrt{2^n}} \left(\sum_{x \in \{0,1\}^n} (-1)^{x \cdot s} |x\rangle \right)$

This is exactly $H^{\otimes n}(H^{\otimes n}|s\rangle) = |s\rangle$.

Thus after measuring $H^{\otimes (n+1)} O_f H^{\otimes (n+1)} |0^n\rangle|1\rangle$
we get $|s\rangle|1\rangle$ with certainty and hence learn s .