Practical Quantum Computing

Lecture 08

The early algorithms: Bernstein-Vazirani and Simon’s

with slides from Dave Bacon https://homes.cs.washington.edu/~dabacon/teaching/siena/

Week	Tuesday (3h)			Wednesday (3h)			Deadlines	
1. The Basics	Introduction	Gates	Circuit Identities	Qiskit	Cirq/Qualtran	Q&A		
	Programming Assignment 1: The basics of a quantum circuit simulator			**Programming Assignment 1:** The building blocks of a quantum circuit simulator				
2. Entanglement and its Applications	Teleportation	Superdense Coding	Quantum Key Distribution	Qualtran/ Assignment2	Terminology of Projects	Q&A		
	Programming Assignment 2: The basics of a quantum circuit optimizer			**Programming Assignment 2:** The building blocks of a quantum circuit optimizer				
3. Computing	Phase Kickback and Toffoli	Distinguishing quantum states and The First Algorithms	Grover’s Algorithm	Invited TBA	PennyLane	Q&A		11 May 2024
4. Advanced Topics*	Arithmetic Circuits*	Fault-Tolerance*	QML*	Invited TBA	Crumble	Q&A	18 May 2024	

* not evaluated

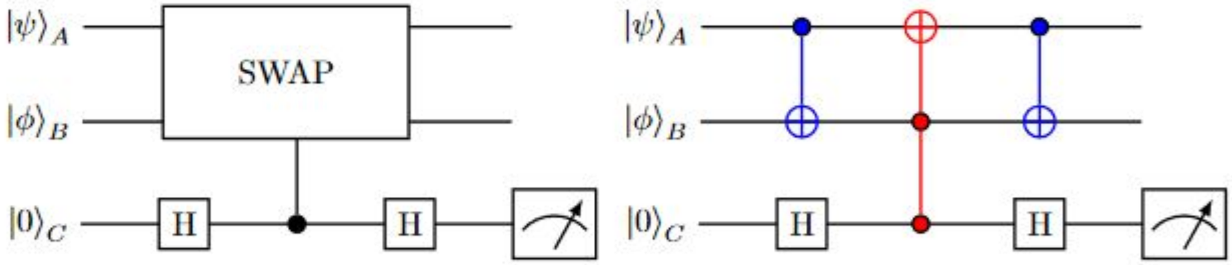
Learning goals - 08 State Discrimination and The First Algorithms (Computing)

1. What you have learned by now
    a. Quantum circuits: mathematics, diagrams and circuit identities
    b. Entanglement: teleportation, superdense coding, quantum games, QKD
    c. Phases, Superpositions and Phase Kickback
2. **Distinguishing between two states**
    a. building a controlled-SWAP from three Toffoli gates
    b. the controlled-SWAP test: circuit and math behind it
3. **Bernstein-Vazirani and Simon's Algorithms**
    a. Problem Statement
    b. Building a superposition to query a Boolean function in parallel
    c. Phase kickback in action - using it for solving the problem
    d. It is a probabilistic algorithm - What is the probability of success?

- Deadline for programming Assignment 1
- 11 May 2024

Distinguishing quantum states

Controlled Swap



(a) (b)

Figure 1: The quantum circuit for an equivalency SWAP test on the two states $|\psi\rangle$ and $|\phi\rangle$. H is a Hadamard gate from equation (7). (a) The SWAP gate swaps all qubits in the test states on the condition that the control qubit is in state $|1\rangle$. (b) shows the SWAP gate broken down into individual gates for the one-qubit test state case. The central gate, shown in red, is a Toffoli gate from equation (9) and the two gates either side in blue are CNOT gates from equation (8), where the crossed circles are controlled on the dots. The final CNOT gate – not necessary for the test outcome – returns the system to its initial state in the case of equivalent states.

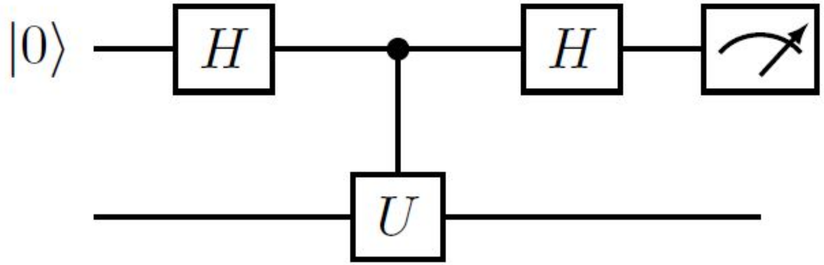
https://arxiv.org/pdf/2009.07613.pdf

$$\mathrm{H}\,|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \quad \text{and} \tag{7}$$
$$\mathrm{H}\,|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right).$$

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{8}$$

$$\mathrm{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{9}$$

Controlled Swap



The CtrlSwap simulates measurement of the SWAP operator as an observable (as opposed to a unitary transformation).

Swap is Hermitian

because U^2 = I

eigenvalues are ±1

Measure eigenvalue using the relative phase

$$\frac{1}{2}(|0,\phi,\psi\rangle + |1,\phi,\psi\rangle + |0,\psi,\phi\rangle - |1,\psi,\phi\rangle) = \frac{1}{2}|0\rangle(|\phi,\psi\rangle + |\psi,\phi\rangle) + \frac{1}{2}|1\rangle(|\phi,\psi\rangle - |\psi,\phi\rangle)$$

$$P(\text{First qubit} = 0) = \frac{1}{2}\Big(\langle\phi|\langle\psi| + \langle\psi|\langle\phi|\Big)\frac{1}{2}\Big(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle\Big) = \frac{1}{2} + \frac{1}{2}|\langle\psi|\phi\rangle|^2$$

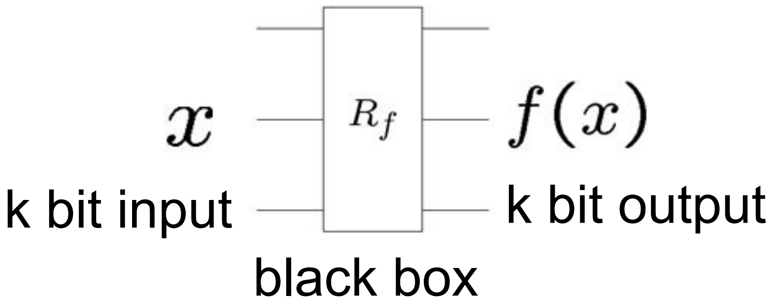
If **states are orthogonal** -> the probability that 0 is measured is **0.5**

If **states are equal** -> the probability that 0 is measured is **1**

Instead of an
Introduction to Complexity Theory

Classical Promise Problem Query Complexity

Given: A black box which computes some function



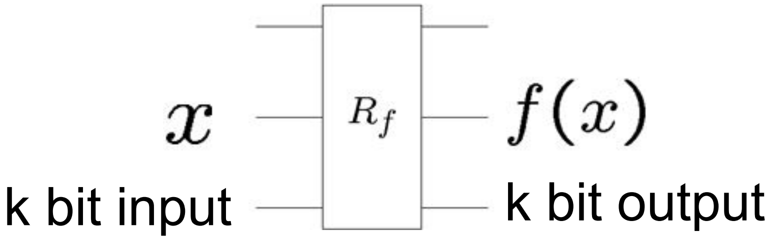
Promise: the function belongs to a set $\mathcal{S}$ which is a subset of all possible functions.

Properties: the set $\mathcal{S}$ can be divided into disjoint subsets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$

Problem: What is the minimal number of times we have to use (query) the black box in order to determine which subset $\mathcal{S}_i$ the function belongs to?

Functions

We can write the unitary



in outer product form as

$$U_f = \sum_{x=0}^{2^n-1} |f(x)\rangle\langle x|$$

so that

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$

$$\begin{aligned} U_f|y\rangle &= \left(\sum_{x=0}^{2^n-1} |f(x)\rangle\langle x|\right)|y\rangle \\ &= \sum_{x=0}^{2^n-1} |f(x)\rangle\langle x|y\rangle = \sum_{x=0}^{2^n-1} |f(x)\rangle\delta_{y,x} = |f(y)\rangle \end{aligned}$$

Functions

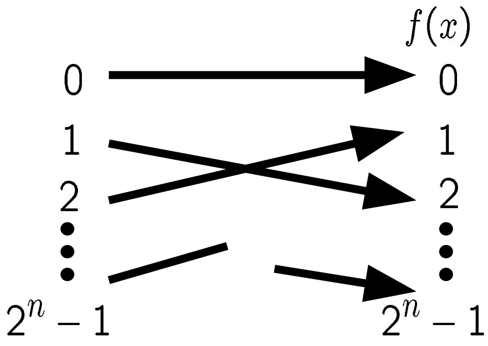
Note that the transform is unitary

$$U_f^\dagger = \left(\sum_{x=0}^{2^n-1} |f(x)\rangle\langle x|\right)^\dagger = \sum_{x=0}^{2^n-1} (|f(x)\rangle\langle x|)^\dagger = \sum_{x=0}^{2^n-1} |x\rangle\langle f(x)|$$

$$U_f U_f^\dagger = \left(\sum_{x=0}^{2^n-1} |f(x)\rangle\langle x|\right)\left(\sum_{y=0}^{2^n-1} |y\rangle\langle f(y)|\right)$$

$$= \sum_{x,y=0}^{2^n-1} |f(x)\rangle\langle x|y\rangle\langle f(y) = \sum_{x,y=0}^{2^n} |f(x)\rangle\langle f(y)|\delta_{x,y}$$

$$= \sum_{x=0}^{2^n-1} |f(x)\rangle\langle f(x)| = I$$



precisely when f(x) is one to one!

Quantum Algorithms

David Deutsch

Richard Jozsa

1992: Deutsch-Jozsa Algorithm

Exact classical query complexity: $2^{n-1}+1$

Bounded error classical query complexity: $O(1)$

Exact quantum q. complexity: 1

Umesh Vazirani

Ethan Bernstein

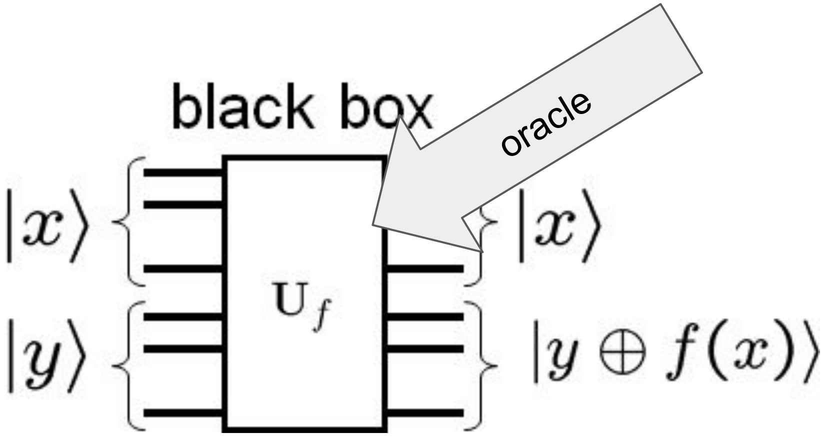
1993: Bernstein-Vazirani Algorithm (non-recursive)

Exact classical query complexity: n

Bounded error classical query complexity: $\Omega(n)$

Exact quantum q. complexity: 1

Query Complexity



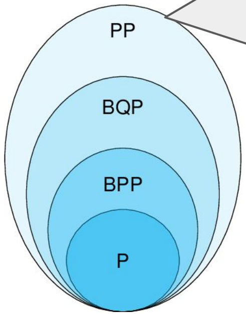
$$f : x \in \{0,1\}^n \to \{0,1\}^k$$

<table>
<tr><td></td><td colspan="2">probability</td></tr>
<tr><td>Exact classical query complexity</td><td>0</td><td rowspan="4">Bounded error algorithms are allowed to fail with a bounded probability of failure.</td></tr>
<tr><td>Bounded error classical query complexity</td><td>1/3</td></tr>
<tr><td>Exact quantum query complexity</td><td>0</td></tr>
<tr><td>Bounded error quantum query complexity</td><td>1/3</td></tr>
</table>

BPP, BQP

Informally, a problem is in **BPP** (bounded-error probabilistic polynomial time) if there is an algorithm for it that has the following properties:

- is allowed to flip coins and make random decisions
- is guaranteed to run in polynomial time
- on any given run of the alg probability of at most 1/3 c wrong answer, whethe or NO.



In complexity theory, PP is the class of decision problems solvable by a probabilistic Turing machine in polynomial time, with an error probability of less than 1/2 for all instances. The abbreviation PP refers to probabilistic polynomial time. A PP algorithm is permitted to have a probability that depends on the input size, whereas BPP does not.

Informally, a decision problem is a member of **BQP** (bounded-error quantum polynomial time) if there exists a quantum algorithm (an algorithm that runs on a quantum computer):

- that solves the decision problem with high probability
- is guaranteed to run in polynomial time
- a run of the algorithm will correctly solve the decision problem with a probability of at least 2/3.

It is the quantum analogue to the complexity class BPP

Learning goals - 08 State Discrimination and The First Algorithms (Computing)

1. What you have learned by now
   a. Quantum circuits: mathematics, diagrams and circuit identities
   b. Entanglement: teleportation, superdense coding, quantum games, QKD
   c. Phases, Superpositions and Phase Kickback
2. **Distinguishing between two states**
   a. building a controlled-SWAP from three Toffoli gates
   b. the controlled-SWAP test: circuit and math behind it
3. **Bernstein-Vazirani and Simon's Algorithms**
   a. Problem Statement
   b. Building a superposition to query a Boolean function in parallel
   c. Phase kickback in action - using it for solving the problem
   d. It is a probabilistic algorithm - What is the probability of success?

- Deadline for programming Assignment 1
- 11 May 2024

The Bernstein-Vazirani Algorithm

Bernstein-Vazirani Problem

Given: A function with n bit strings as input and one bit as output

$$f : x \in \{0,1\}^n \rightarrow \{0,1\}$$

Promise: The function is of the form

$$f(x) = (a \cdot x) \oplus b \qquad a \in \{0,1\}^n \qquad b \in \{0,1\}$$

$$y \cdot x = y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n$$

Problem: Find the n bit string a

Bernstein-Vazirani Problem

Given: A function with n bit strings as input and one bit as output

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

Promise: The function is of the form

$$f(x) = (a \cdot x) \oplus b \qquad a \in \{0,1\}^n \qquad b \in \{0,1\}$$

$$y \cdot x = y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n$$

Problem: Find the n bit string a

Notice that the querying f yields a single bit of information. But we need n bits of information to describe a.

Classical Bernstein-Vazirani

Notice that the querying f yields a single bit of information. But we need n bits of information to describe a.

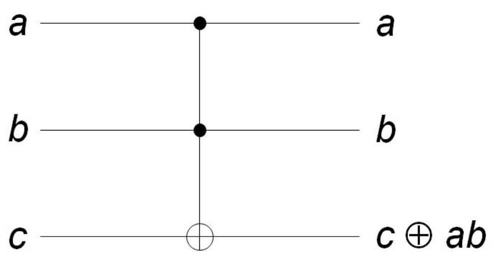
Classically, the most efficient method to find the secret string is by evaluating the function n times with the input values $x = 2^i$ for all $i \in \{0, 1, \ldots, n-1\}$

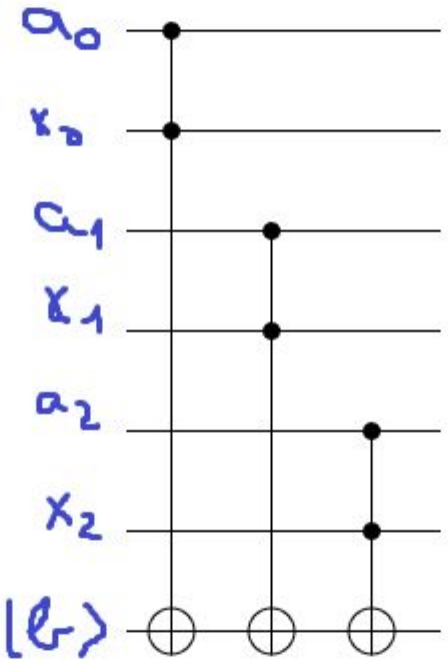
$$
\begin{gathered}
f(1000\cdots0_n) = s_1 \\
f(0100\cdots0_n) = s_2 \\
f(0010\cdots0_n) = s_3 \\
\vdots \\
f(0000\cdots1_n) = s_n
\end{gathered}
$$

Implement the oracle

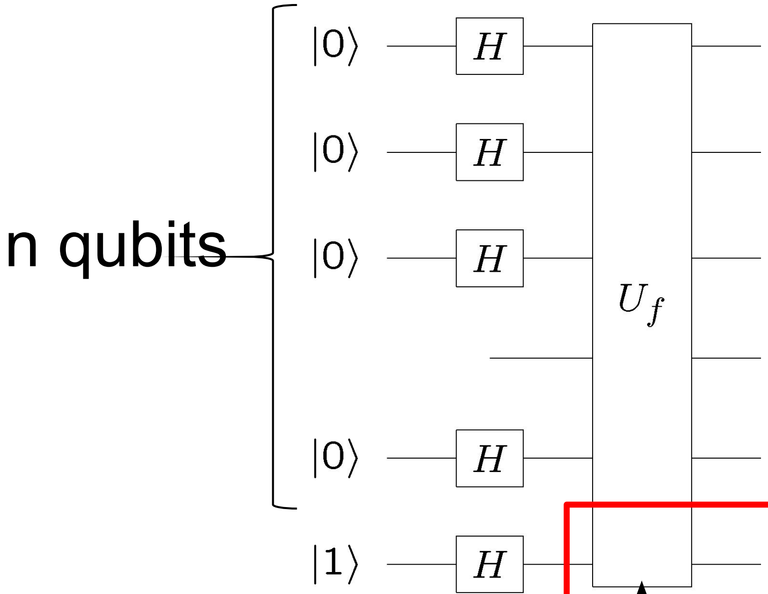
$$f(x) = (a \cdot x) \oplus b$$

$$y \cdot x = y_1 x_1 \oplus y_2 x_2 \oplus \cdots \oplus y_n x_n$$





Quantum Bernstein-Vazirani



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

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

for phase kickback

$$H^{\otimes n}|0\rangle \otimes H|1\rangle = \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}|y\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Quantum Bernstein-Vazirani

Show the phase kickback

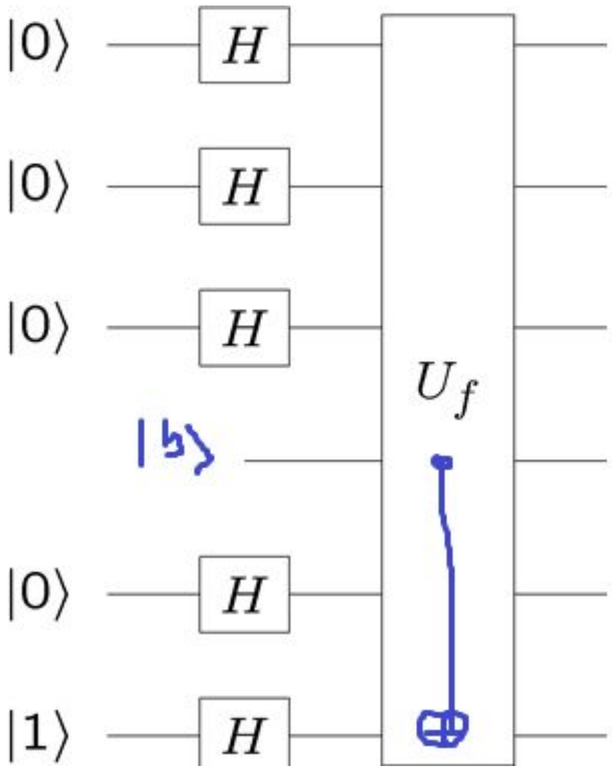
|Register>|b>|->

if |b> == |0> (when f(x) == 0)

+|Register>|b>**|->**

elif |b> == |1> (when f(x) == 1)

-|Register>|b>**|->**



Hadamard it! (Interference)

$$H^{\otimes n}\left[\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{a\cdot y+b}|y\rangle\right] = \frac{1}{2^n}\sum_{x,y=0}^{2^n-1}(-1)^{a\cdot y+b}(-1)^{x\cdot y}|x\rangle$$

$$\left.\begin{aligned} |0\rangle &\xrightarrow{H} |0\rangle + |1\rangle \\ |1\rangle &\xrightarrow{H} |0\rangle - |1\rangle \end{aligned}\right\} \quad |0\rangle + (-1)^{xb}|b\rangle$$

$$\otimes|x| \qquad |x\rangle \xrightarrow{H} \sum_{0}^{2^{|x|}-1}(-1)^{xy}|y\rangle$$

$$|1\rangle \longrightarrow \underbrace{(-1)^{1\cdot 0}|0\rangle + (-1)^{1\cdot 1}|1\rangle}$$

$$= |0\rangle - |1\rangle \qquad \sum(-1)^{xb}|b\rangle$$

$$= \frac{(-1)^b}{2^n}\sum_{x=0}^{2^n-1}\left(\sum_{y=0}^{2^n-1}(-1)^{a\cdot y+x\cdot y}\right)|x\rangle$$

Hadamard it! (Interference)

$$H^{\otimes n}\left[\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{a\cdot y+b}|y\rangle\right] = \frac{1}{2^n}\sum_{x,y=0}^{2^n-1}(-1)^{a\cdot y+b}(-1)^{x\cdot y}|x\rangle$$

$$= \frac{(-1)^b}{2^n}\sum_{x=0}^{2^n-1}\left(\sum_{y=0}^{2^n-1}(-1)^{a\cdot y+x\cdot y}\right)|x\rangle$$

$$\sum_{y=0}^{2^n-1}(-1)^{a\cdot y+x\cdot y} = \sum_{y_1,\ldots,y_n=0}^{1}(-1)^{a_1y_1+\cdots+a_ny_n+x_1y_1+\cdots+x_ny_n}$$

$$= \sum_{y_1=0,1}\cdots\sum_{y_n=0,1}(-1)^{a_1y_1+\cdots+a_ny_n+x_1y_1+\cdots+x_ny_n}$$

$$= ((-1)^0+(-1)^{a_1+x_1})\sum_{y_2=0,1}\cdots\sum_{y_n=0,1}(-1)^{a_2y_2+\cdots+a_ny_n+x_2y_2+\cdots+x_ny_n}$$

$$= 2^n\delta_{a_1,x_1}\delta_{a_2,x_2}\cdots\delta_{a_n,x_n} = 2^n\delta_{a,x}$$

Hadamard it! (Interference)

$$H^{\otimes n}\left[\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{a\cdot y+b}|y\rangle\right] = \frac{1}{2^n}\sum_{x,y=0}^{2^n-1}(-1)^{a\cdot y+b}(-1)^{x\cdot y}|x\rangle$$

$$= \frac{(-1)^b}{2^n}\sum_{x=0}^{2^n-1}\left(\sum_{y=0}^{2^n-1}(-1)^{a\cdot y+x\cdot y}\right)|x\rangle$$

$$\sum_{y=0}^{2^n-1}(-1)^{a\cdot y+x\cdot y} = \sum_{y_1,\ldots,y_n=0}^{1}(-1)^{a_1y_1+\cdots+a_ny_n+x_1y_1+\cdots+x_ny_n}$$

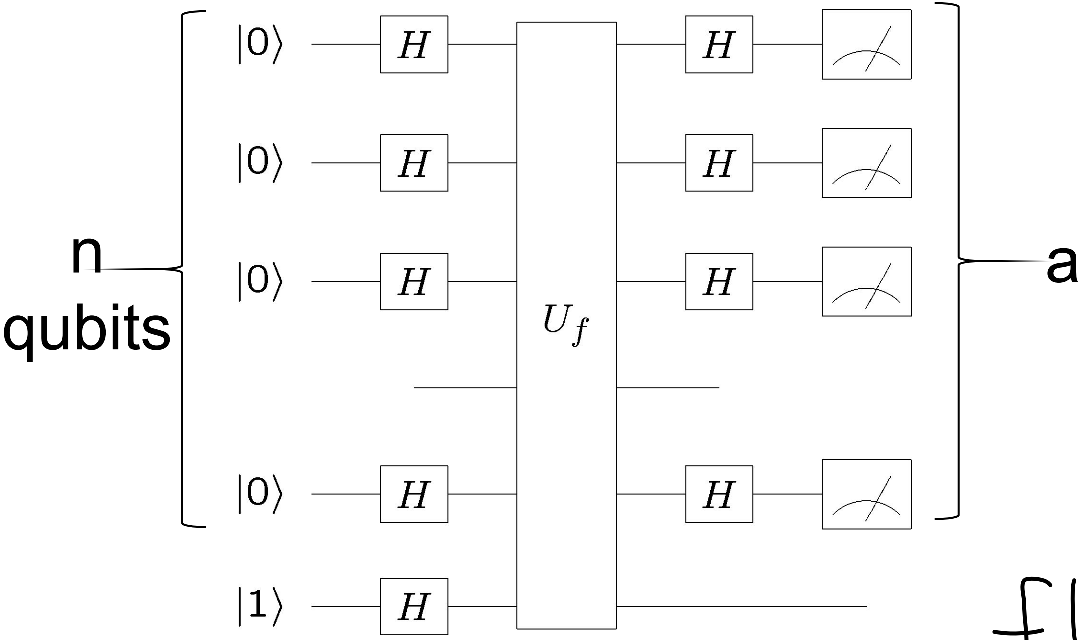
$$= \sum_{y_1=0,1}\cdots\sum_{y_n=0,1}(-1)^{a_1y_1+\cdots+a_ny_n+x_1y_1+\cdots+x_ny_n}$$

$$= \boxed{((-1)^0+(-1)^{a_1+x_1})}\sum_{y_2=0,1}\cdots\sum_{y_n=0,1}(-1)^{a_2y_2+\cdots+a_ny_n+x_2y_2+\cdots+x_ny_n}$$

$$= \boxed{2^n}\delta_{a_1,x_1}\delta_{a_2,x_2}\cdots\delta_{a_n,x_n} = 2^n\delta_{a,x}$$

$$H^{\otimes n}\left[\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{a\cdot y+b}|y\rangle\right] = (-1)^b\sum_{x=0}^{2^n}\delta_{a,x}|x\rangle = (-1)^b|a\rangle$$

Quantum Bernstein-Vazirani



$f(x) = a \cdot x \oplus b$

We can determine a using only a single quantum query!

Learning goals - 08 State Discrimination and The First Algorithms (Computing)

1. What you have learned by now
   a. Quantum circuits: mathematics, diagrams and circuit identities
   b. Entanglement: teleportation, superdense coding, quantum games, QKD
   c. Phases, Superpositions and Phase Kickback
2. **Distinguishing between two states**
   a. building a controlled-SWAP from three Toffoli gates
   b. the controlled-SWAP test: circuit and math behind it
3. **Bernstein-Vazirani and Simon's Algorithms**
   a. Problem Statement
   b. Building a superposition to query a Boolean function in parallel
   c. Phase kickback in action - using it for solving the problem
   d. It is a probabilistic algorithm - What is the probability of success?

- Deadline for programming Assignment 1
- 11 May 2024

Simon’s Algorithm

Context

The Deutsch-Jozsa problem showed an exponential quantum improvement over the *best deterministic* classical algorithms.

The Bernstein-Vazirani problem shows a polynomial improvement over the *best randomized* classical algorithms that have error probability ≤ 1/3.

Combine these two features and see a problem where quantum computers are exponentially more efficient than bounded-error randomized algorithms.

https://arxiv.org/pdf/1907.09415.pdf

Simon's Problem

Given: A function with n bit strings as input and one bit as output

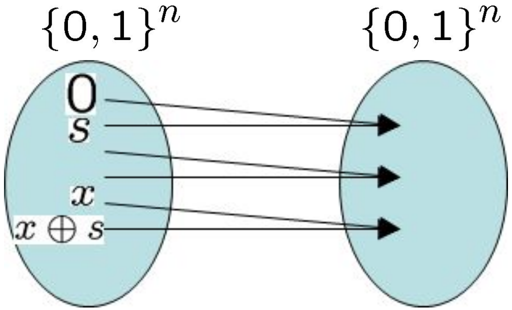
$$f : x \in \{0,1\}^n \rightarrow \{0,1\}^n$$

Promise: The function is guaranteed to satisfy

$$f(x) = f(y) \Leftrightarrow y = x \oplus s, s \neq 0$$

$$x \oplus s = (x_1 \oplus s_1, x_2 \oplus s_2, \ldots, x_n \oplus s_n)$$

Problem: Find the n bit string $s \neq 0$



Classical Simon's Problem

Promise: The function is guaranteed to satisfy

Suppose we start querying the function and *build up a list of the pairs* $(x_\alpha, f(x_\alpha))$

If we find $x_\alpha \neq x_\beta$ such that $f(x_\alpha) = f(x_\beta)$ then we solve the problem

$$f(x_\alpha) = f(x_\beta) \Rightarrow x_\alpha = s \oplus x_\beta$$
$$s = x_\alpha \oplus x_\beta$$

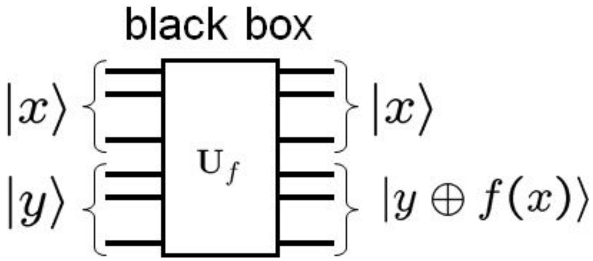
But suppose we start querying the function m times

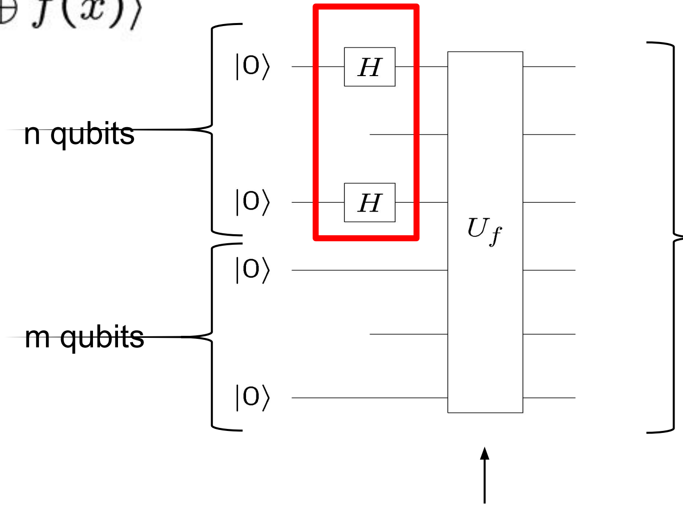
Probability of getting a matching pair: $\approx \frac{\binom{m}{2}}{2^n} = \frac{m(m-1)}{2^{n+1}}$

Bounded error query complexity: $m = O\left(2^{\frac{n}{2}}\right)$

Quantum Simon's Problem

Unlike previous problems, we can't use the phase kickback trick because there is no structure in the function.





$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |f(x)\rangle$$

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

Quantum Simon's Problem

Measure the second register

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |f(x)\rangle$$

Using the promise on the function

$$f(x) = f(y) \Leftrightarrow y = x \oplus s, s \neq 0$$

This implies that after we measure, we have the state

$$\frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle) \otimes |f(x)\rangle$$

For random uniformly distributed $x \in \{0,1\}^n$

uniformly distributed = all strings equally probable.

measuring this state at this time does us no good ...

Quantum Simon's Problem

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

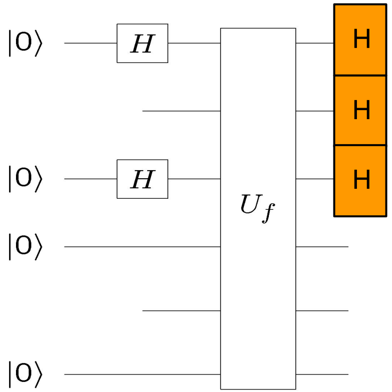
Measuring this state at this time in the **computational basis** does us no good….

For random uniformly distributed $x \in \{0,1\}^n$

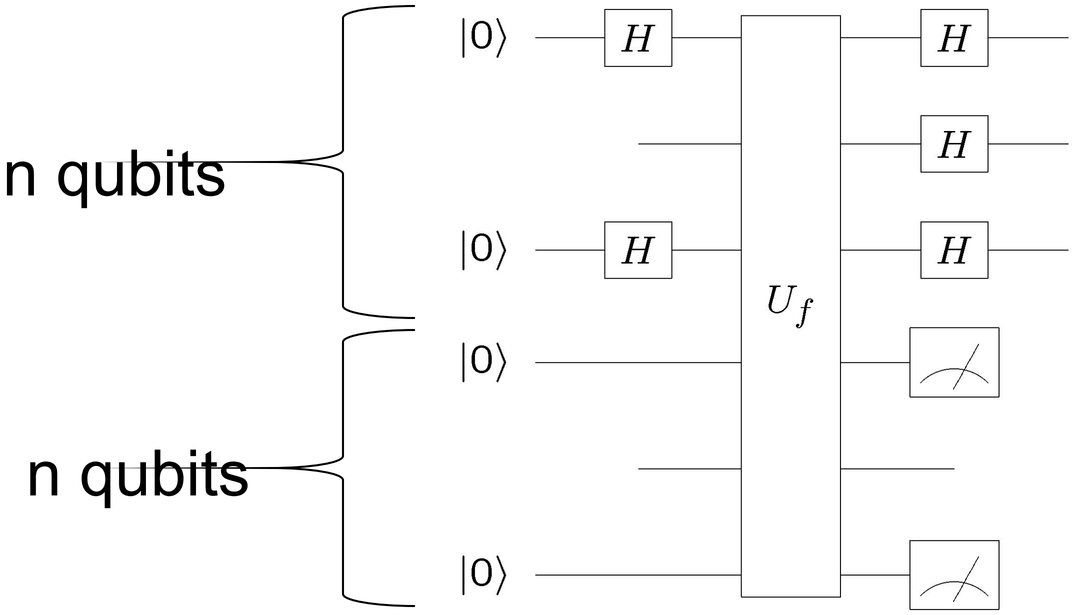
Measurement yields either $|x\rangle$ or $|x \oplus s\rangle$

But we don't know x, so we can't use this to find s.

Add Hadamard gates to the end register



Quantum Simon's Problem



$$H^{\otimes n}\frac{1}{\sqrt{2}}(|x\rangle+|x\oplus s\rangle) = \frac{1}{\sqrt{2^{n+1}}}\sum_{y=0}^{2^n-1}((-1)^{y\cdot x}+(-1)^{y\cdot(x\oplus s)})|y\rangle$$

Quantum Simon's Problem

$$H^{\otimes n}\frac{1}{\sqrt{2}}(|x\rangle+|x\oplus s\rangle) = \frac{1}{\sqrt{2^{n+1}}}\sum_{y=0}^{2^n-1}((-1)^{y\cdot x}+(-1)^{y\cdot(x\oplus s)})|y\rangle$$

$$y\cdot x = y_1x_1 \oplus y_2x_2 \oplus \cdots \oplus y_nx_n$$

$$y\cdot(x\oplus s) = y_1(x_1\oplus s_1)\oplus y_2(x_2\oplus s_2)\oplus\cdots\oplus y_n(x_n\oplus s_n) = (y\cdot x)\oplus(y\cdot s)$$

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

Measuring this state, we obtain uniformly distributed random values of y s.t.

$$y\cdot s = 0 \text{ mod } 2$$

If $y \neq 0$ we have eliminated the possible values of s by half

Quantum Simon's Problem

$$y \cdot s = 0 \bmod 2$$

$$y \cdot s = y_1 s_1 \oplus y_2 s_2 \oplus \cdots \oplus y_n s_n = 0$$

On values of y_i which are 0, this doesn't restrict s_i

On values of y_i which are 1, the corresponding s_i must XOR to 0.

This restricts the set of possible $\boldsymbol{s}$'s by half.

Example: $n = 3$

$y = 3_{10} = 011_2$

$$(011) \cdot s = 0s_1 \oplus 1s_2 \oplus 1s_3 = 0$$

$$s_2 \oplus s_3 = 0$$

possible s's: 000 011 100 111

Quantum Simon's Problem

Think about the bit strings s as vectors in $\mathbb{Z}_2^n$

$$y_0 \cdot s = 0$$
$$y_1 \cdot s = 0$$
$$\vdots$$
$$y_k \cdot s = 0$$

- If we obtain **n** lin. indep. equations of this form, we win
- (Gaussian elimination)

Suppose we have k linearly independent y_i 's. What is the probability that y_{k+1} is linearly independent of previous y_i 's?

$$Pr = \frac{2^n - 2^k}{2^n} = 1 - 2^{k-n}$$

Note that if the j's you have generated at some point span a space of size 2^k, for some $k < n-1$, then the probability that your next run of the algorithm produces a j that is linearly independent of the earlier ones, is $(2^{n-1} - 2^k)/2^{n-1} \geq 1/2$. Hence an expected number of $O(n)$ runs of the algorithm suffices to find $n-1$ linearly independent j's. Simon's algorithm thus finds s using an expected number of $O(n)$ x_i-queries and polynomially many other operations.

https://arxiv.org/pdf/1907.09415.pdf

Quantum Simon's Problem

What is the probability that our n-1 equations are linearly independent?

$$Pr(succ) = \left(1 - \frac{1}{2^n}\right)\left(1 - \frac{1}{2^{n-1}}\right)\cdots\left(1 - \frac{1}{4}\right)$$

$$= \prod_{k=1}^{n}\left(1 - \frac{1}{2^k}\right) > \prod_{k=2}^{\infty}\left(1 - \frac{1}{2^k}\right) \approx 0.28879$$

With constant probability:

- we obtain linearly independence -> Gaussian elimination O(n^3)
- solve Simon's problem