INTRO TO QUANTUM COMPUTING

Week 20 Lab

LINEAR SEARCH AND THE DEUTSCH-JOSZA ALGORITHM

<insert TA name>

<insert date>

PROGRAM FOR TODAY

- Canvas attendance quiz
- Pre-lab zoom feedback
- Lab content
- Post-lab zoom feedback

Canvas attendance quiz passcode:

CANVAS ATTENDANCE QUIZ

- Please log into Canvas and answer your lab section's quiz (using the password posted below and in the chat).
  - This is lab number:
  - Passcode:
- **Question:** Please rate your level of comfort with the three quantum protocols we've learned (5 = very comfortable; 1 = not at all comfortable)
  1. Superdense Coding
  2. Quantum Teleportation
  3. Quantum Key Distribution
- **This quiz is not graded, but counts for your lab attendance!**

Canvas attendance quiz passcode:

PRE-LAB ZOOM FEEDBACK

On a scale of 1 to 5, how would you rate your understanding of this week's content?

- 1 –Did not understand anything
- 2 – Understood some parts
- 3 – Understood most of the content
- 4 – Understood all of the content
- 5 – The content was easy for me/I already knew all of the content

In lecture this week, Fran gave us an overview of quantum algorithms

Canvas attendance quiz passcode:

LEARNING OBJECTIVES FOR LAB 20

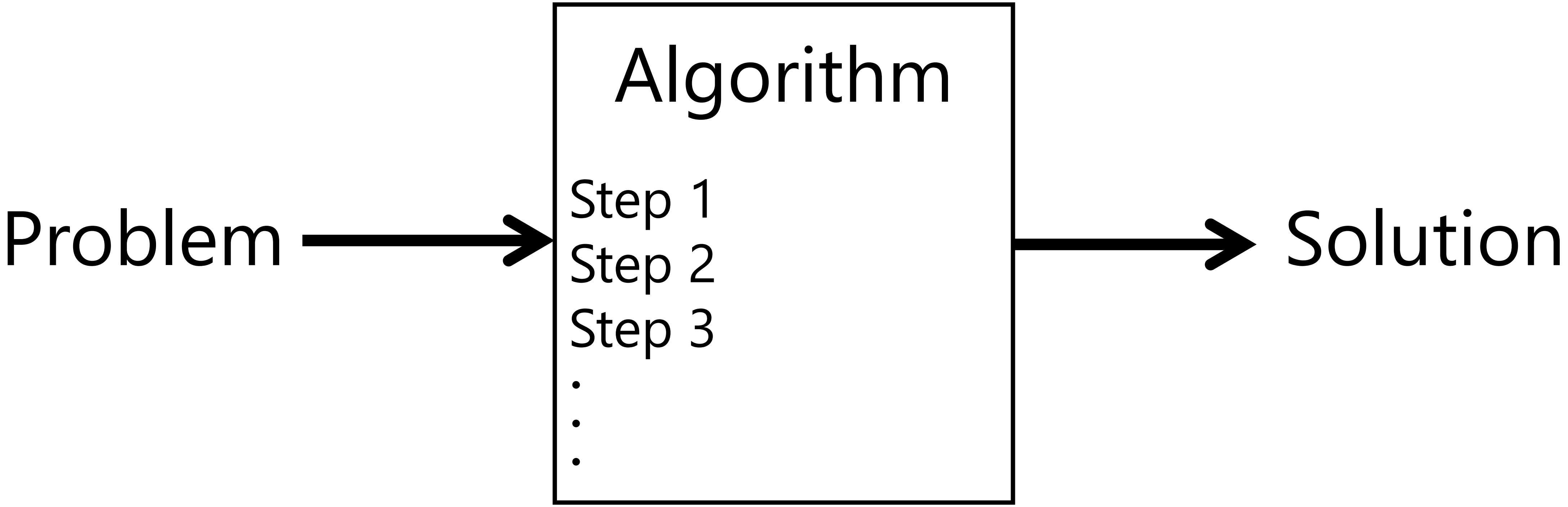
- Linear search algorithms
  - What is an algorithm
  - Searching a list
  - Complexity of linear search
  - Coding a linear search algorithms

- The Deutsch-Josza algorithm
  - The Goldilocks problem
  - Classical solution
  - Oracles and the quantum solution

Canvas attendance quiz passcode:

ALGORITHMS

An algorithm is a set of steps to solve a problem



Problem

Algorithm

Step 1
Step 2
Step 3
.
.
.

Solution

Canvas attendance quiz passcode:

ALGORITHMS AND BIG-O NOTATION

Problem: Find the student whose age is 19 years from the following list:

Student name	Age
Sarah	17
Rahul	16
Elina	20
Aziza	23
Aman	18
Phil	14
Corbin	15

●
●
●

Brian	19
Fran	20

●
●
●

LINEAR SEARCH

Problem: Find the student whose age is 19 years from the given list

Linear search algorithm:

1. Start from the first element of the list and one by one compare *x* with each element of the data set
2. If *x* matches with current element, return the index
3. If *x* does not match the current element, move on to the next one
4. If *x* doesn't match with any of elements in the array, return -1 (search failed, or element not found)

Canvas attendance quiz passcode:

BIG-O NOTATION FOR LINEAR SEARCH

- If the list has N students, we will perform N comparisons in the worst-case
- **Big-O complexity**: The worst-case number of comparisons or computations in an algorithm

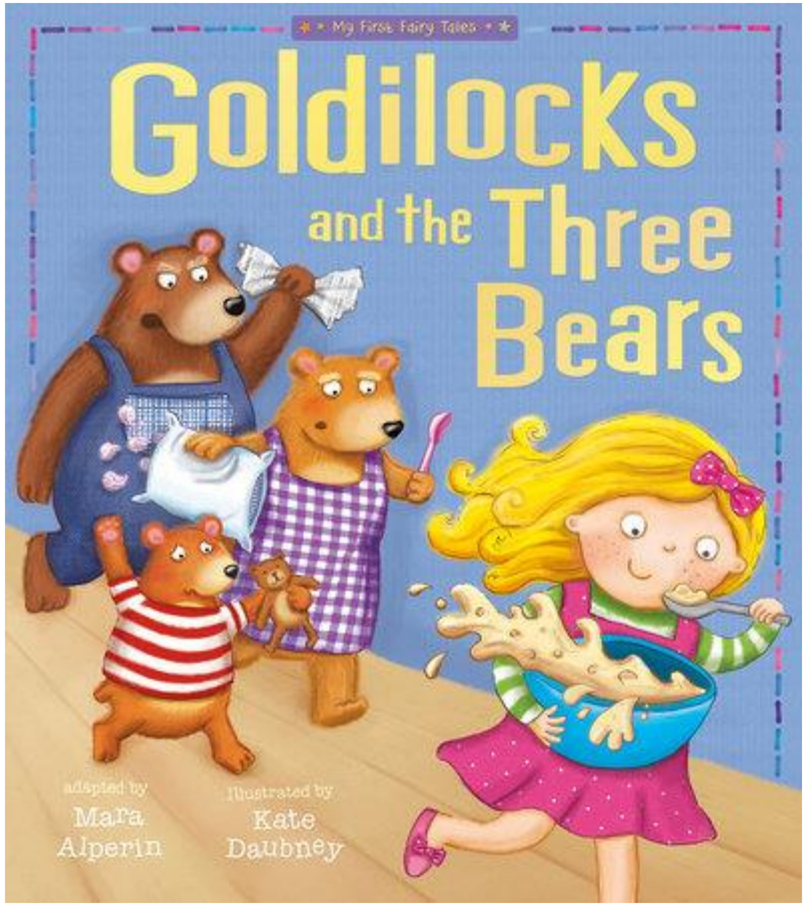
- The complexity of the linear search algorithm is **O(N)**

TIME TO CODE!

Canvas attendance quiz passcode:

THE DEUTSCH-JOSZA ALGORITHM

- Remember the story of Goldilocks and the Three Bears?
- In one part of the story, Goldilocks wants to figure out if a soup is too hot, too cold, or just right
- The D-J algorithm solves a very similar problem – it tells us if a function is too hold or too cold (**constant**) or just right (**balanced**)
- This is a **toy** problem, designed to show that there are problems for which quantum algorithms are better than classical



*This analogy is taken from a project developed by the Unitary Fund

PROBLEM STATEMENT

- We have a function $f(x)$, which takes as input a binary variable x. The function also returns a binary number for each input value of x
- We have four possible forms of $f(x)$:

x	$f(x)$
0	0
1	0

Constant

x	$f(x)$
0	1
1	1

Constant

x	$f(x)$
1	1
0	0

Balanced

x	$f(x)$
1	0
0	1

Balanced

- The problem is – can you figure out if a given function $f(x)$ is constant or balanced? How many times do you need to test the function to find out the answer?

Canvas attendance quiz passcode:

THE CLASSICAL SOLUTION

x	$f(x)$
0	0
1	0

Constant

x	$f(x)$
0	1
1	1

Constant

x	$f(x)$
1	1
0	0

Balanced

x	$f(x)$
1	0
0	1

Balanced

- How would you solve this problem classically?

- You would have to test out the function with inputs (x values) of 0 and 1
- You need **two** steps!

PREPARATION FOR THE QUANTUM SOLUTION

x	$f(x)$
0	0
1	0

Constant

x	$f(x)$
0	1
1	1

Constant

x	$f(x)$
1	1
0	0

Balanced

x	$f(x)$
1	0
0	1

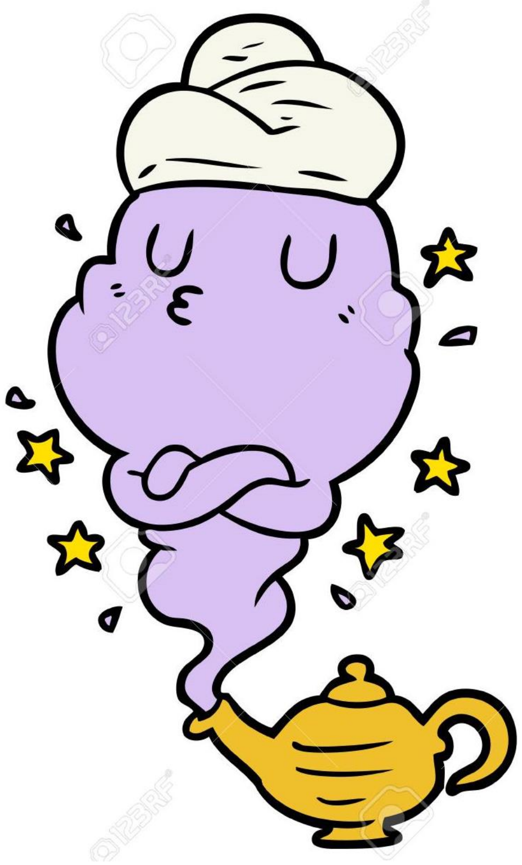
Balanced

- We want to make a quantum circuit to solve this problem
- We'll want a way to implement $f(x)$ in a quantum circuit
- All quantum gates have to be reversible. Is $f(x)$ reversible?

THE QUANTUM SOLUTION

- **Problem:** Quantum gates need to be reversible. $f(x)$ is not reversible

- **Solution:** Oracles! The function is reframed as a 'genie' that will answer questions based on the inputs you give it.



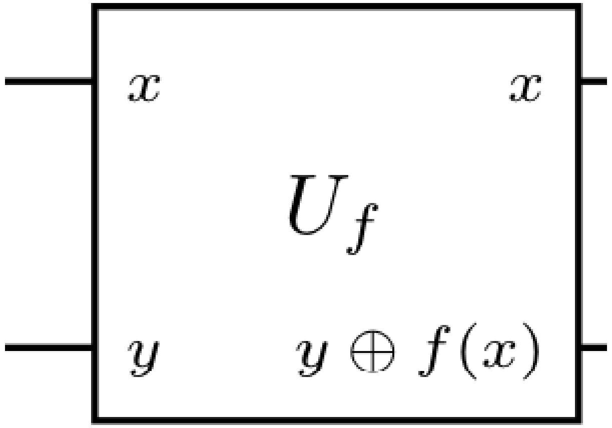
ORACLES, WHAT EVEN ARE THEY?!

- Oracles are like “black-boxes”, we often don’t worry too much about how they are made
- They give an answer to queries we make
- Ultimately, an oracle is just a fancy gate, and can be written as a matrix
- Oracles are used in the D-J algorithm, and Grover’s search algorithm (next week)

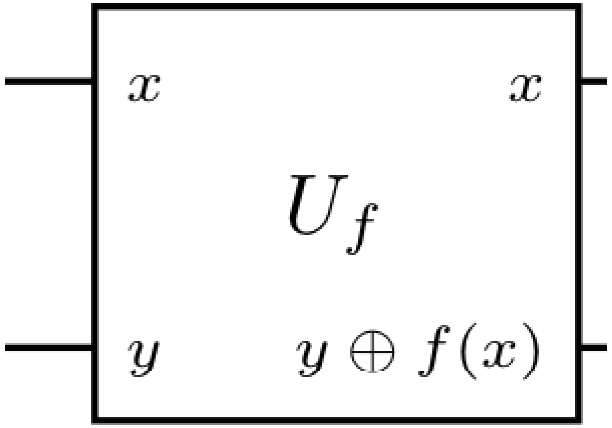
THE QUANTUM SOLUTION

- Since $f(x)$ is not reversible in general, we need to make a reversible "version" of it for our quantum algorithm

- This reversible version is the Deutsch-Josza **oracle** $\boldsymbol{U_f}$

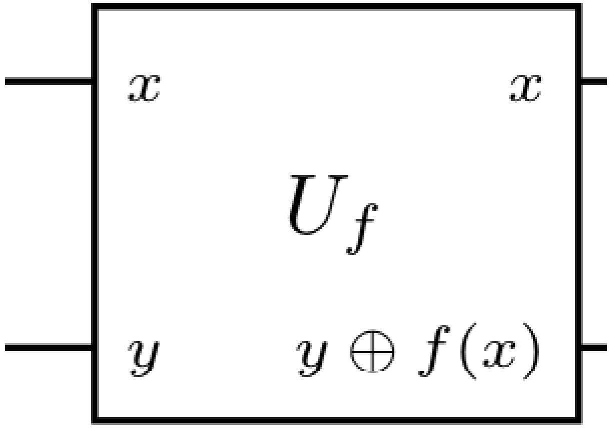


REVERSIBILITY



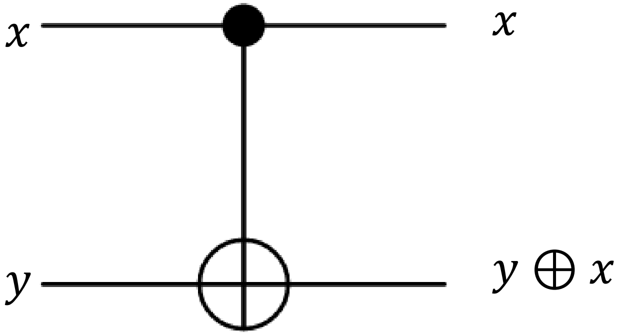
- **Check**: This oracle is reversible
- $y \oplus f(x) \oplus f(x) = y$!

IMPLEMENTING XOR



- The oracle uses an **XOR** operation. How can we implement XOR using quantum gates?

- We can use CNOT!



ONE STEP – USING SUPERPOSITION

- Using the oracle, the Deutsch-Josza algorithm lets you solve the problem in **one** step!
- The algorithm uses **superposition** and **interference** to solve the problem in one step

The D-J algorithm:

1. Initiate a quantum circuit of two qubits, *x* and *y*
2. Apply an X gate to *y*
3. Apply H gates to both *x* and *y* - **Superposition**
4. Apply the oracle to *x* and *y*
5. Apply H gates to both *x* and *y* - **Interference**
6. Measure *x*. If the result is 0, the oracle is constant. If it is 1, the oracle is balanced.

QUANTUM ADVANTAGE

- For this simple 1-bit input case, the classical algorithm uses 2 steps, and the D-J algorithm uses 1
- For an n-bit input, the classical algorithm is $O(2^n)$, whereas the D-J algorithm is still $O(1)$!
- **Check:** For 10 bits, how many steps does the classical algorithm take? How many steps does D-J take?

KEY TAKEAWAYS

- Big-O notation is a way to characterize how many computations an algorithm makes in the **worst-case.**
- Linear search has a complexity of $O(N)$
- The Deutsch-Josza algorithm uses superposition and interference to show an exponential speedup over the classical algorithm to solve the toy "Goldilocks" problem

Canvas attendance quiz passcode:

FURTHER READING AND RESOURCES

- Qiskit textbook page on Deutsch-Josza algo
- The Quantum Talk
- Panel discussion on near-term and long-term QC prospects
- Beyond quantum supremacy: the search for useful QCs
- Shor's algorithm explained
- Arthur Eckert's lectures on quantum information science

Canvas attendance quiz passcode:

POST-LAB ZOOM FEEDBACK

After this lab, on a scale of 1 to 5, how would you rate your understanding of this week's content?

- 1 –Did not understand anything
- 2 – Understood some parts
- 3 – Understood most of the content
- 4 – Understood all of the content
- 5 – The content was easy for me/I already knew all of the content

Canvas attendance quiz passcode:

EXTRA CONTENT

Canvas attendance quiz passcode:

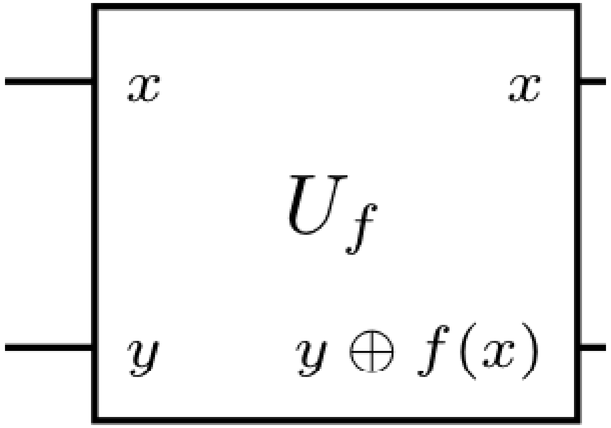
TESTING THE ORACLE

- Let’s see what outputs the oracle produces

- Case 1:

x	$f(x)$
0	0
1	0

Input			Output	
x	y	$f(x)$	x	$y \oplus f(x)$
0	0	0	0	0
0	1	0	0	1
1	0	0	1	0
1	1	0	1	1



- How would you implement this oracle experimentally?

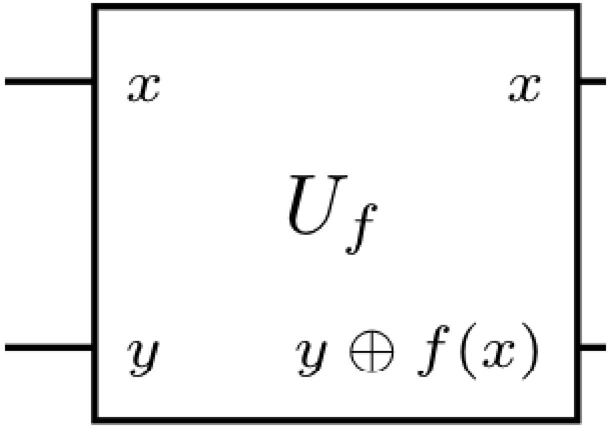
TESTING THE ORACLE

- Let’s see what outputs the oracle produces

- Case 2:

x	$f(x)$
0	1
1	1

Input			Output	
x	y	$f(x)$	x	$y \oplus f(x)$
0	0	1	0	1
0	1	1	0	0
1	0	1	1	1
1	1	1	1	0



- How would you implement this oracle experimentally?

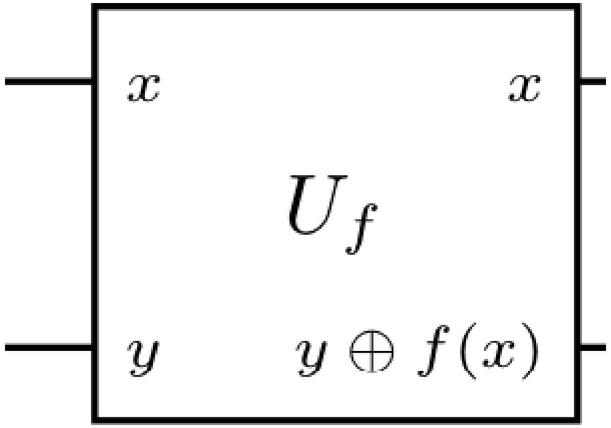
TESTING THE ORACLE

- Let's see what outputs the oracle produces

- Case 3:

x	$f(x)$
0	0
1	1

Input			Output	
x	y	$f(x)$	x	$y \oplus f(x)$
0	0	0	0	0
0	1	0	0	1
1	0	1	1	1
1	1	1	1	0



- How would you implement this oracle experimentally?

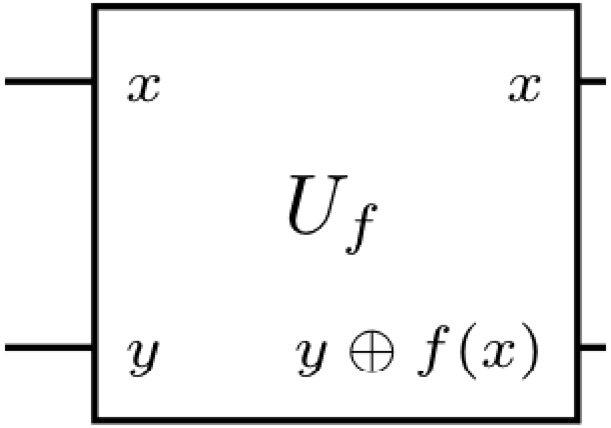
TESTING THE ORACLE

- Let's see what outputs the oracle produces

- Case 4:

x	$f(x)$
0	1
1	0

Input			Output	
x	y	$f(x)$	x	$y \oplus f(x)$
0	0	1	0	1
0	1	1	0	0
1	0	0	1	0
1	1	0	1	1



- How would you implement this oracle experimentally?

Canvas attendance quiz passcode: