Yale University

Quantum Query Algorithms

CPSC 4470/5470

Introduction to Quantum Computing

Instructor: Prof. **Yongshan Ding**

Computer Science, Applied Physics, Yale Quantum Institute

Quantum Query Algorithms

Phase Oracle:

$$O_f^{\pm} = \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle\langle x|$$

$$\sum_x \alpha_x |x\rangle \;\text{—}\boxed{O_f^{\pm}}\text{—}\; \sum_x \alpha_x (-1)^{f(x)} |x\rangle$$

Example (1 qubit):

$$|0\rangle \;\text{—}\boxed{H}\text{—}\boxed{O_f^{\pm}}\text{—}\boxed{H}\text{—}\boxed{Z}$$

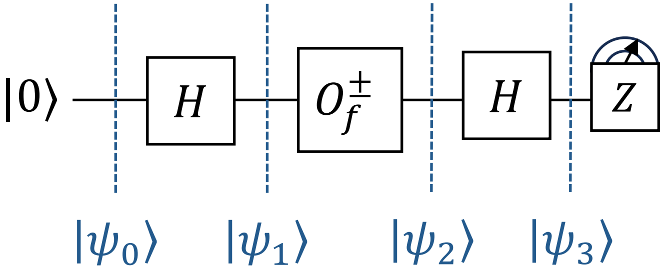
For Boolean function $f: \{0,1\} \to \{0,1\}$:

- What if $f(0) = f(1)$?
- What if $f(0) \neq f(1)$?

Feynman Path Diagram

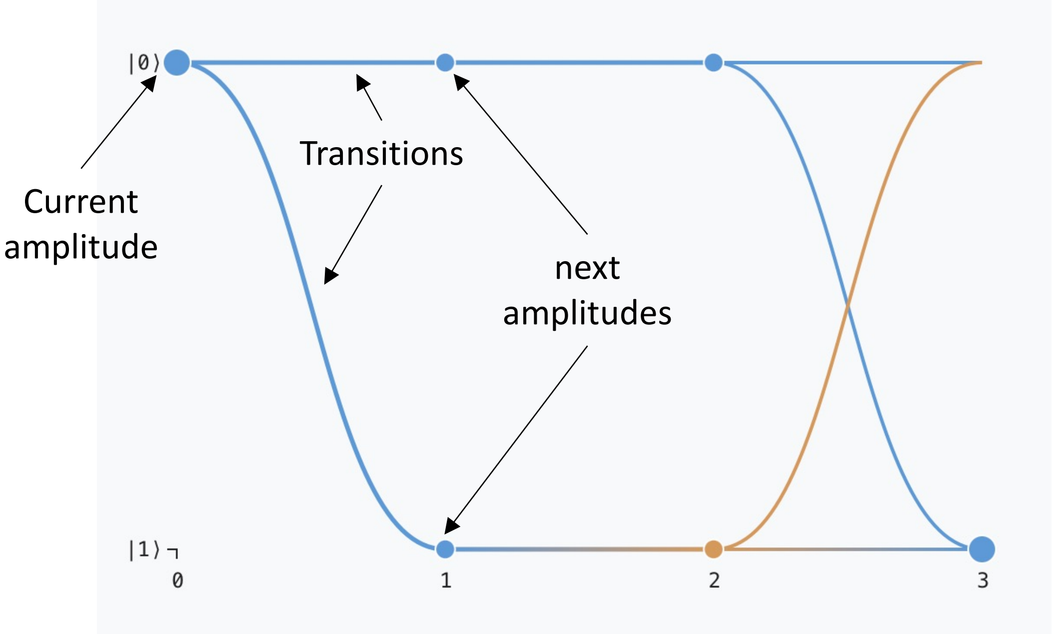
Phase Oracle:

$$O_f^{\pm} = \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle\langle x|$$

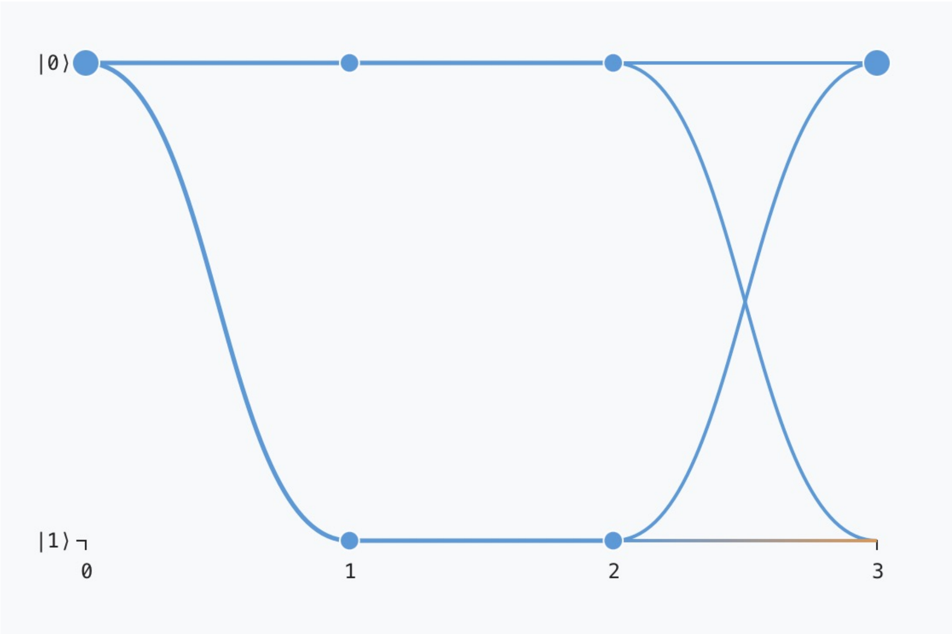


(Derive on board)

Consider function: $f(0) = 0, f(1) = 1$



Consider function: $f(0) = 0, f(1) = 0$



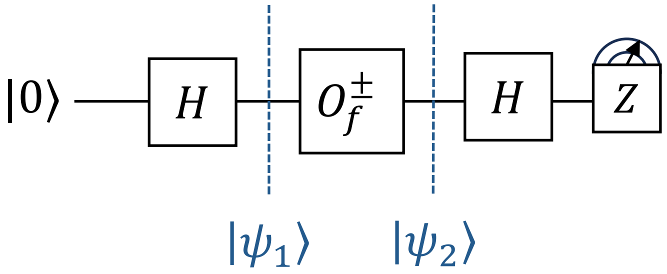
Quantum Query Algorithms

Phase Oracle:

$$O_f^{\pm} = \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle\langle x|$$

$$\sum_x \alpha_x |x\rangle \;—\; \boxed{O_f^{\pm}} \;—\; \sum_x \alpha_x (-1)^{f(x)} |x\rangle$$

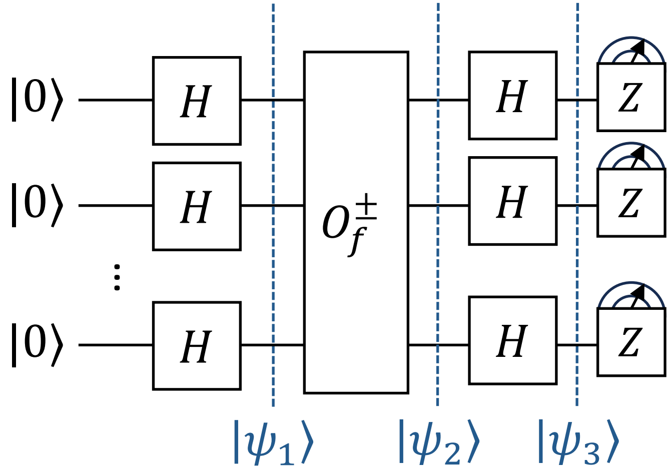
Example (1 qubit):

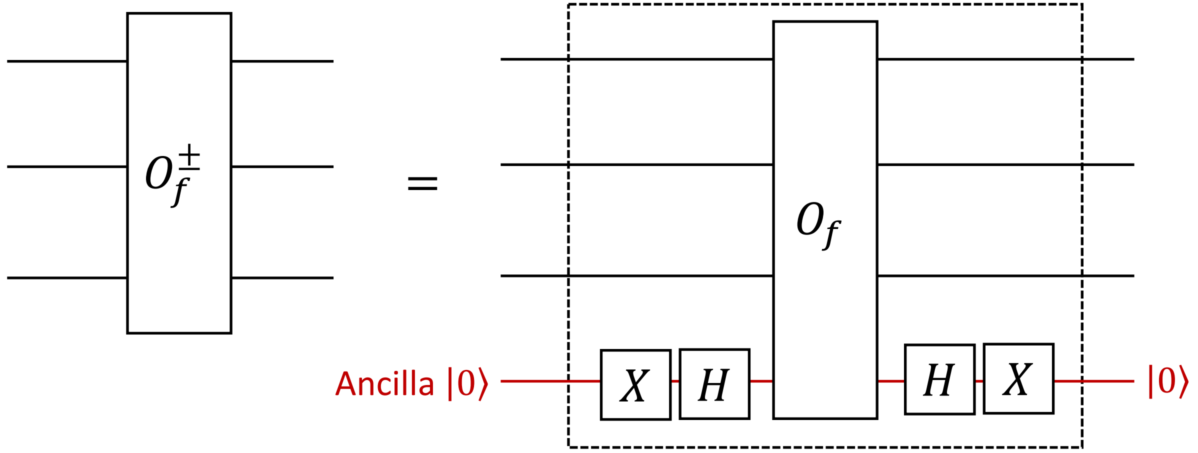


Comparison: To distinguish $f(0) = f(1)$ or $f(0) \neq f(1)$:

- **Quantumly**, we query oracle $O_f^{\pm}$ only **once**.
- **Classically**, we need to query function f at least **twice**.

Deutsch-Jozsa Algorithm





Consider function $f: \{0,1\}^n \rightarrow \{0,1\}$:

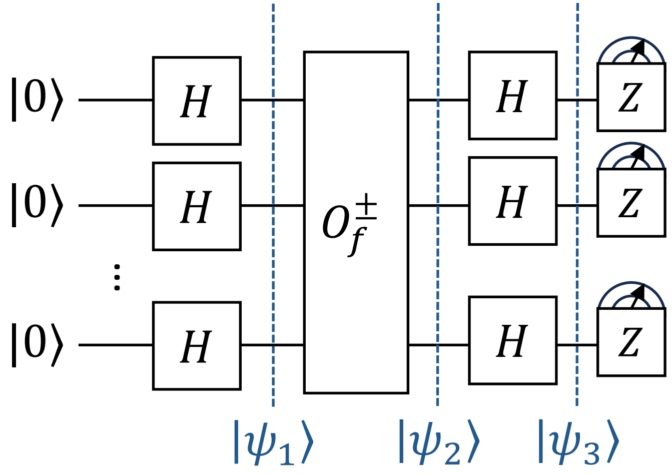
- **Constant** function:
  - $f(x) = f(y), \forall x, y.$
- **Balanced** function:
  - $f(x) = 0$ for half of inputs, and $f(x) = 1$ for the other half.

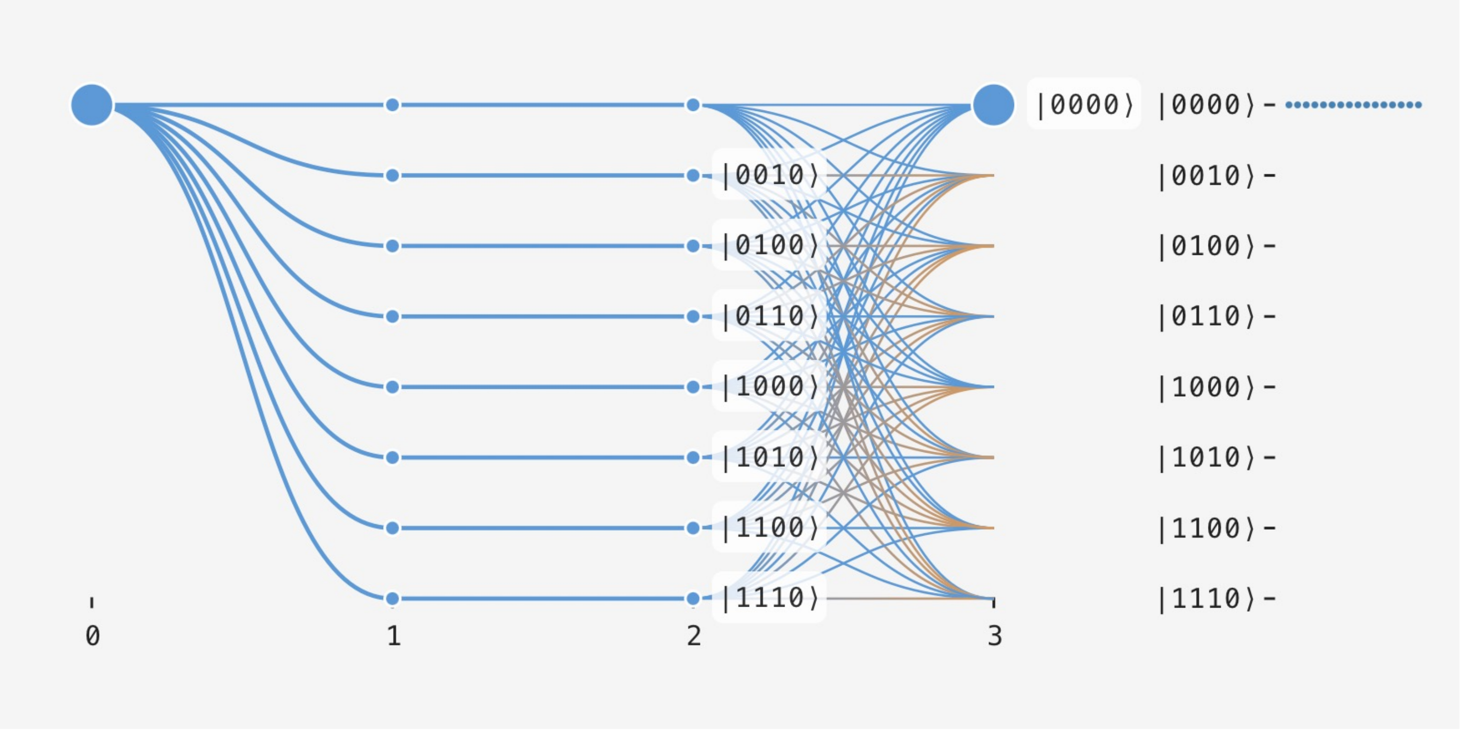
For **constant function**: $O_f^{\pm} = \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle\langle x| = I$

Constant function: $f(x) = 0$

Derive on board: how does the state evolve in case of each scenario?

Deutsch-Jozsa Algorithm





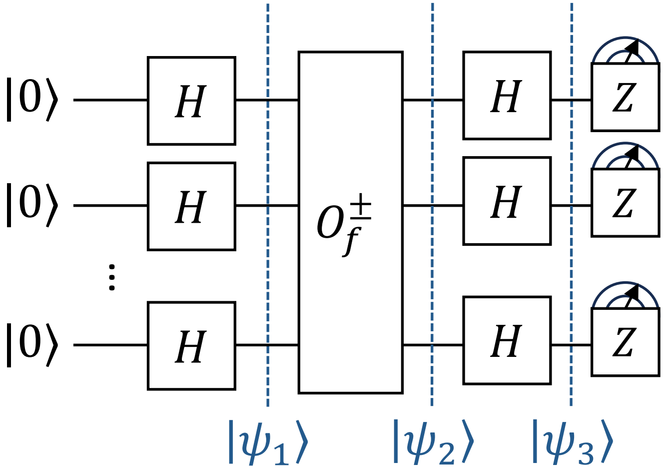
Consider function $f\colon \{0,1\}^n \to \{0,1\}$:

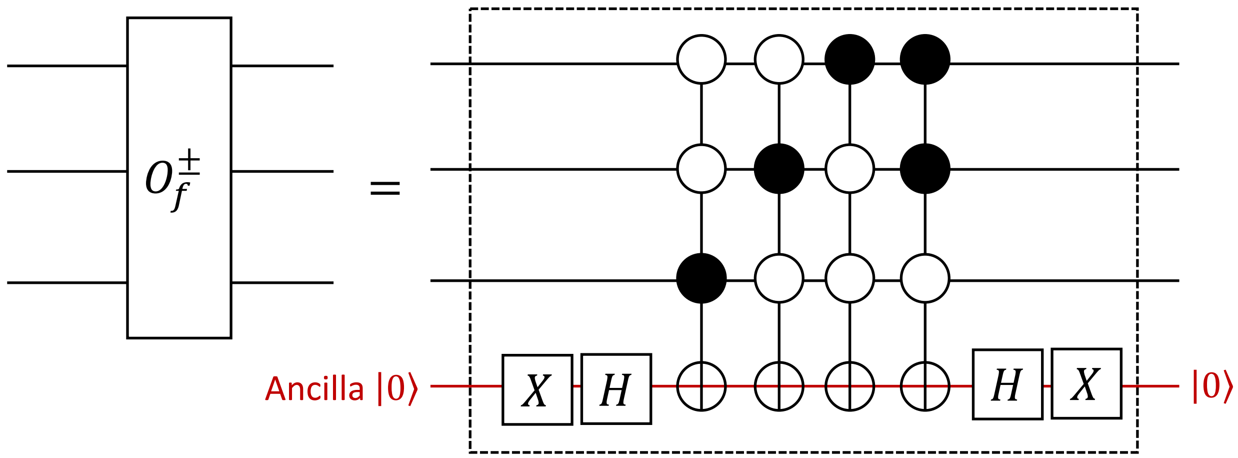
- **Constant** function:
  - $f(x) = f(y), \forall x, y.$
- **Balanced** function:
  - $f(x) = 0$ for half of inputs, and $f(x) = 1$ for the other half.

Constant function: $f(x) = 0$

<u>Derive on board</u>: how does the state evolve in case of each scenario?

Deutsch-Jozsa Algorithm





Consider function $f: \{0,1\}^n \rightarrow \{0,1\}$:

- **Constant** function:
  - $f(x) = f(y), \forall x, y$.
- **Balanced** function:
  - $f(x) = 0$ for half of inputs, and $f(x) = 1$ for the other half.

For **balanced function**: $O_f^{\pm} = \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle\langle x|$

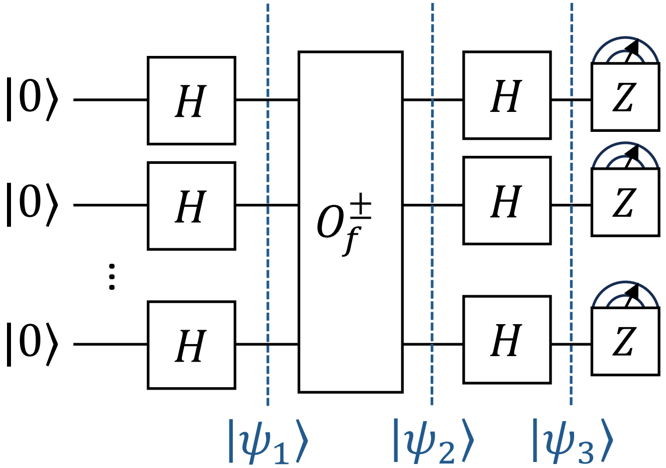
Balanced function:

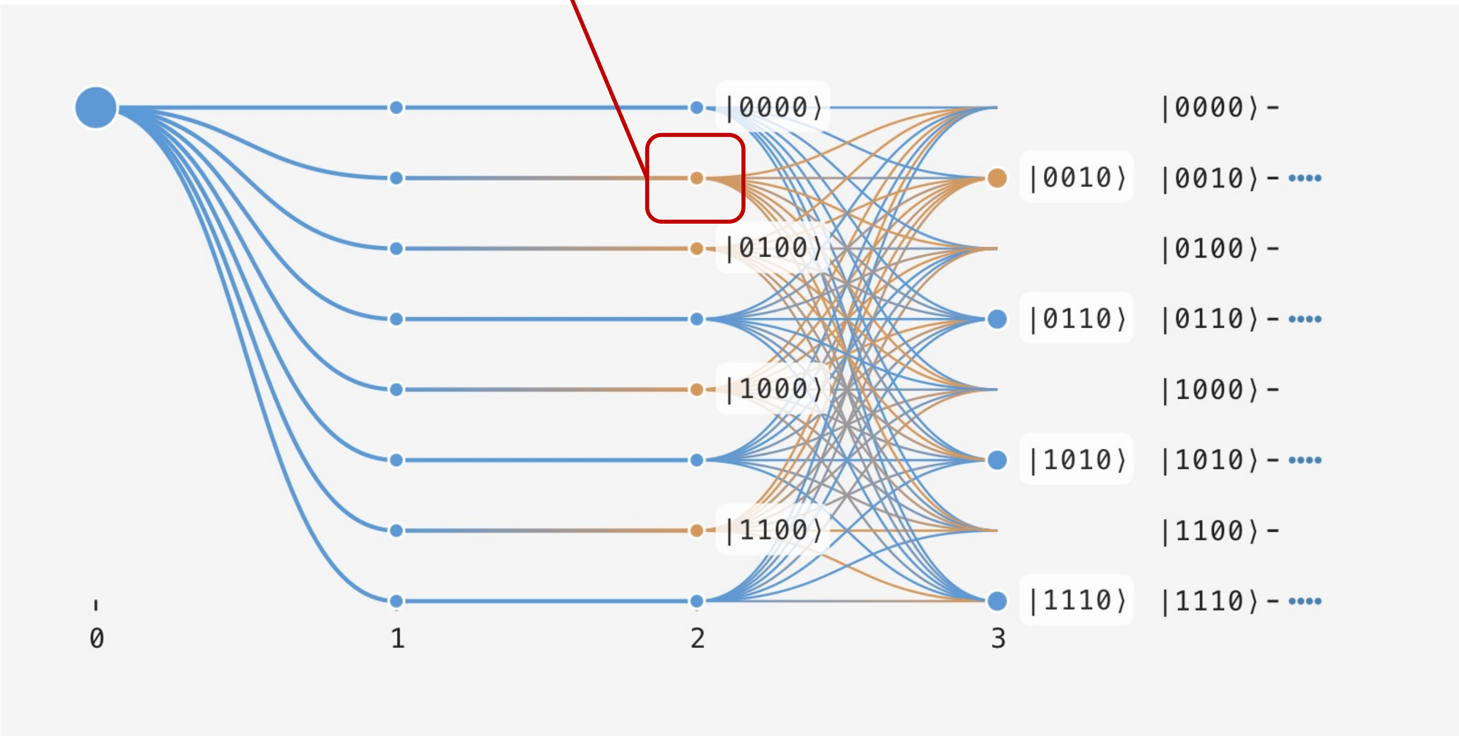
$f(001) = 1$ $\quad$ $f(000) = 0$
$f(010) = 1$ $\quad$ $f(011) = 0$
$f(100) = 1$ $\quad$ $f(101) = 0$
$f(110) = 1$ $\quad$ $f(111) = 0$

Deutsch-Jozsa Algorithm

$$-\frac{1}{\sqrt{8}}|001\rangle \xrightarrow{H^{\otimes 3}} -\frac{1}{\sqrt{8}}|{+}{+}{-}\rangle$$

What's amp on $|000\rangle$?





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

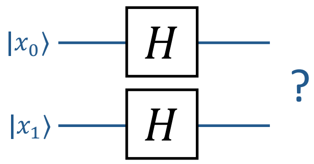
- **Constant** function:
  - $f(x) = f(y), \forall x, y$.
- **Balanced** function:
  - $f(x) = 0$ for half of inputs, and $f(x) = 1$ for the other half.

Comparison: To distinguish constant or balanced:
- **Quantumly**, we query oracle $O_f^\pm$ only **once**.
- **Classically**, we need to query function $2^{n-1} + 1$ times (worst case).

Balanced function:

$f(001) = 1$ $\quad$ $f(000) = 0$
$f(010) = 1$ $\quad$ $f(011) = 0$
$f(100) = 1$ $\quad$ $f(101) = 0$
$f(110) = 1$ $\quad$ $f(111) = 0$

Boolean Fourier Transform ($H^{\otimes n}$)



$$\begin{array}{c} \\ H^{\otimes 2} = \dfrac{1}{\sqrt{2^2}} \end{array} \begin{array}{cccc|c} x{:}\ 00 & 01 & 10 & 11 & y{:} \end{array}$$

$$H^{\otimes 2} = \frac{1}{\sqrt{2^2}}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{array}{l} 00 \\ 01 \\ 10 \\ 11 \end{array}$$

What is $H^{\otimes 2}|x\rangle = \sum_y \alpha_{xy}|y\rangle$

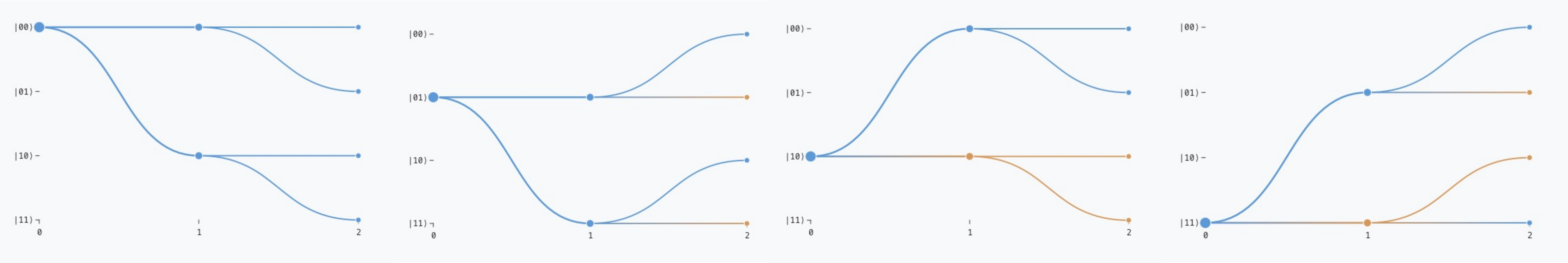
$\alpha_{xy} = \langle y|H^{\otimes 2}|x\rangle$

Write α_{xy} in terms of x, y?

Derive on board: What is $H^{\otimes n}|x\rangle$?

To get the sign correct:

$$\begin{array}{c|cccc|c} & x{:}\ 00 & 01 & 10 & 11 & y{:} \\ \hline & 0 & 0 & 0 & 0 & 00 \\ & 0 & 1 & 0 & 1 & 01 \\ & 0 & 0 & 1 & 1 & 10 \\ & 0 & 1 & 1 & 2 & 11 \end{array}$$

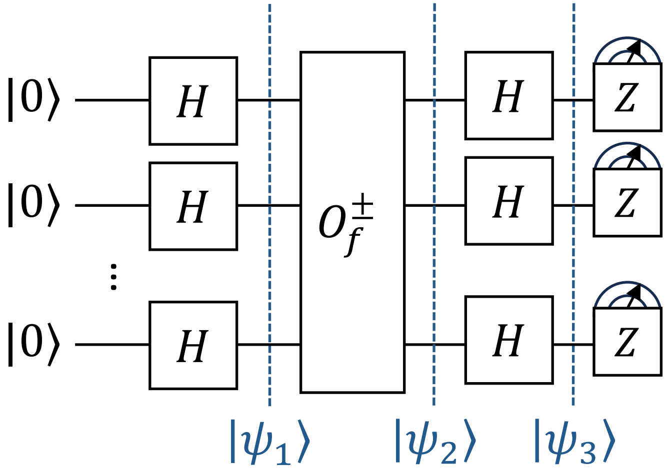


Yale

Boolean Fourier Transform ($H^{\otimes n}$)

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_y (-1)^{\sum_i x_i y_i \pmod 2}|y\rangle$$

What is $H^{\otimes n}\left(\frac{1}{\sqrt{2^n}}\sum_y (-1)^{\sum_i x_i y_i \pmod 2}|y\rangle\right)$?



If we set oracle $O_f^{\pm}$ where $f(y) = \sum_i x_i y_i \pmod 2$ for some unknown x.

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}}\sum_y |y\rangle$$

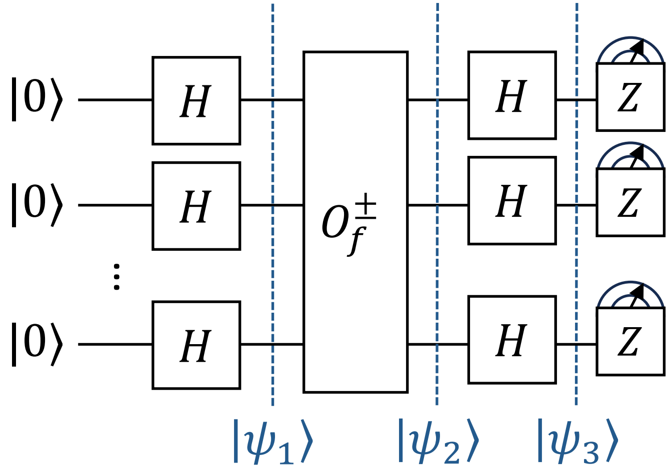
$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}}\sum_y (-1)^{\sum_i x_i y_i \pmod 2}|y\rangle$$

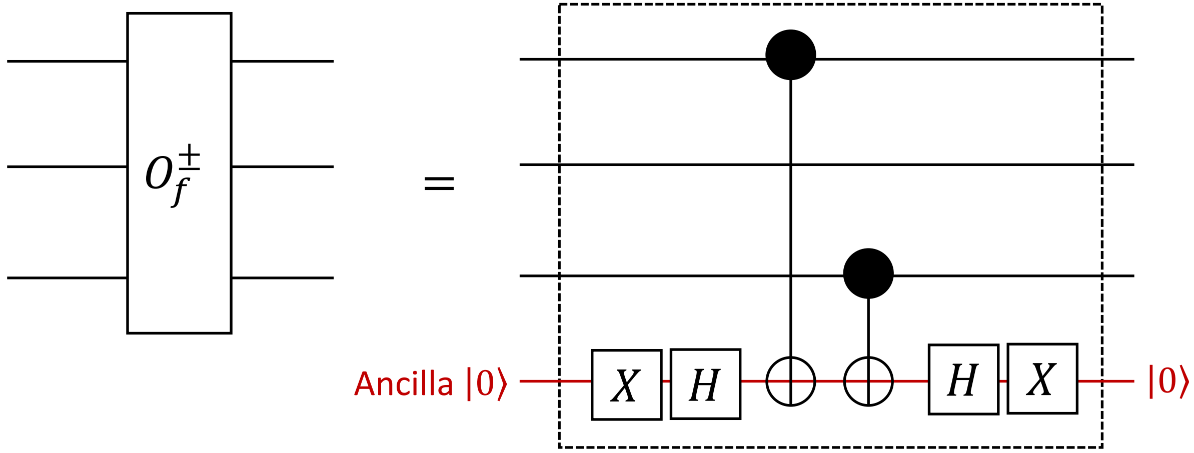
What is $|\psi_3\rangle$?

Bernstein-Vazirani Algorithm

Function with **secret string** s:

$$f(x) = x \cdot s = \sum_i x_i s_i \pmod 2$$





For **secret function**: $O_f^{\pm} = \sum_{x\in\{0,1\}^n} (-1)^{\sum_i x_i s_i \pmod 2} |x\rangle\langle x|$

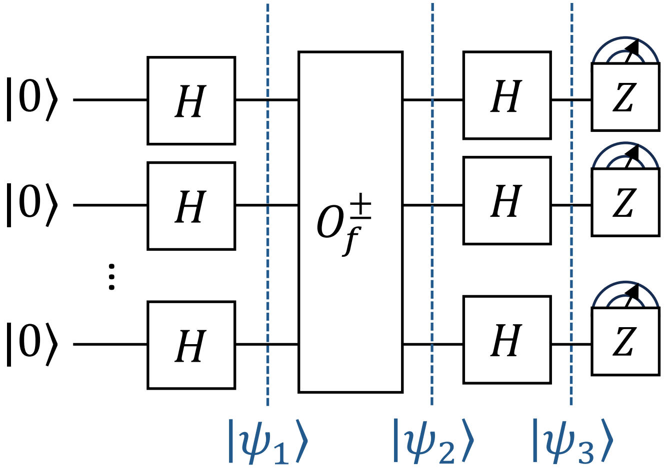
Secret string: $s = 101$

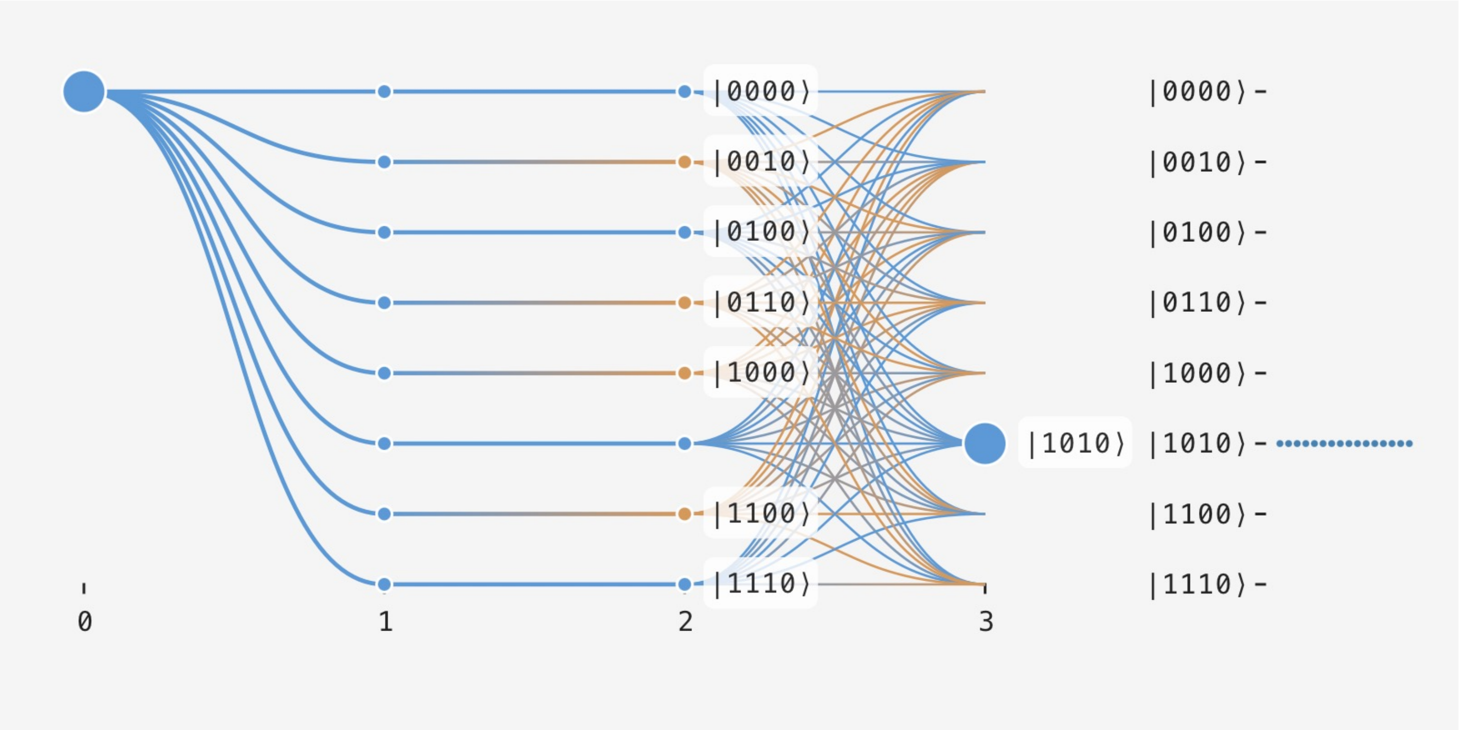
$$f(x) = \sum_i x_i s_i \pmod 2 = x_0 + x_2 \pmod 2 = x_0 \oplus x_2$$

Bernstein-Vazirani Algorithm

Function with secret string s:

$$f(x) = x \cdot s = \sum_i x_i s_i \pmod 2$$





Secret string: $s = 101$

$$f(x) = \sum_i x_i s_i \pmod 2 = x_0 + x_2 \pmod 2 = x_0 \oplus x_2$$

Quantum v.s. Classical

Function with **secret string** s:

$$f(x) = x \cdot s = \sum_i x_i s_i \pmod 2$$

Quantumly, we can reveal the string s by querying $O_f^{\pm}$ only **once.**

Classically, what's a good strategy to learn the string s? At least n queries to function f is necessary!