Quantum query complexity

Lecture 1

Introduction & The hybrid method

Materials: https://yassine-hamoudi.github.io/pcmi2023/

Focus of this course

Proving that quantum algorithms cannot be too fast

A lower bound statement:

"Any quantum algorithm that solves problem X must run in time at least T"

Challenges:

1/ Formalizing the model of computation ⟵ Quantum query model

2/ Finding methods for proving lower bounds

Focus of this course

Why should we care about lower bounds?

- Understand the limits of quantum algorithms

  ➡ For which problems is it hopeless to find efficient algorithms?

- Design security proofs in cryptography

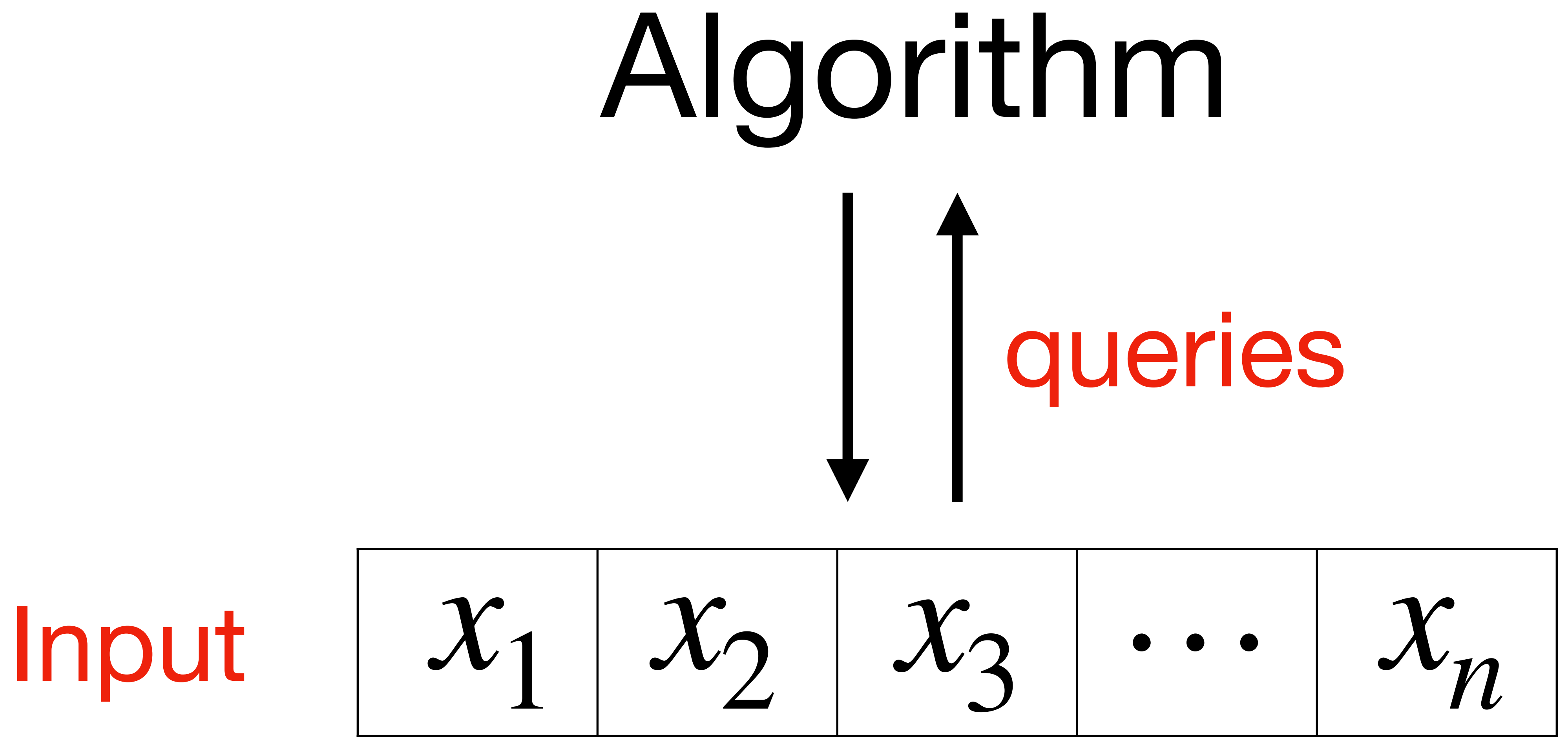
  ➡ Which protocols take a long time to break, even by quantum adversaries?

- Can give insights into new algorithms **(cf Lecture 5)**

The quantum query model

Classical query complexity

(a.k.a. decision tree complexity)

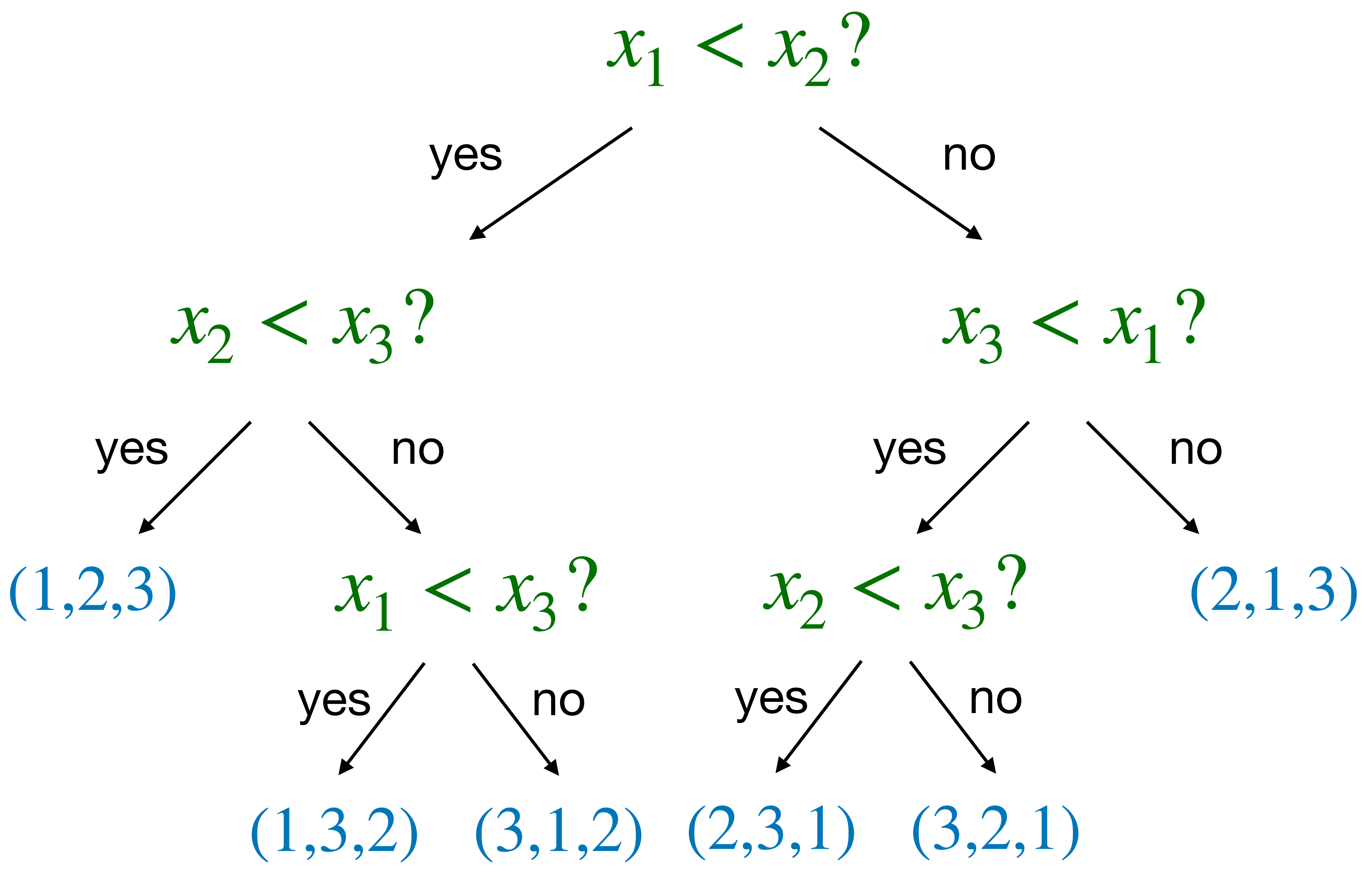


We only count the number of queries to the input (the internal computation of the algorithm is “for free”).

Classical query complexity

(a.k.a. decision tree complexity)

We can model the computation by a decision tree:



Height = query complexity of that algorithm

Sorting 3 numbers using comparison queries

Classical query complexity

(a.k.a. decision tree complexity)

- This is often the "right model" to capture the difficulty of a problem

Example: Any classical sorting algorithm must do $\Omega(n \log n)$ comparison queries.

- The queries give us a grasp on what the algorithm has learnt about the input

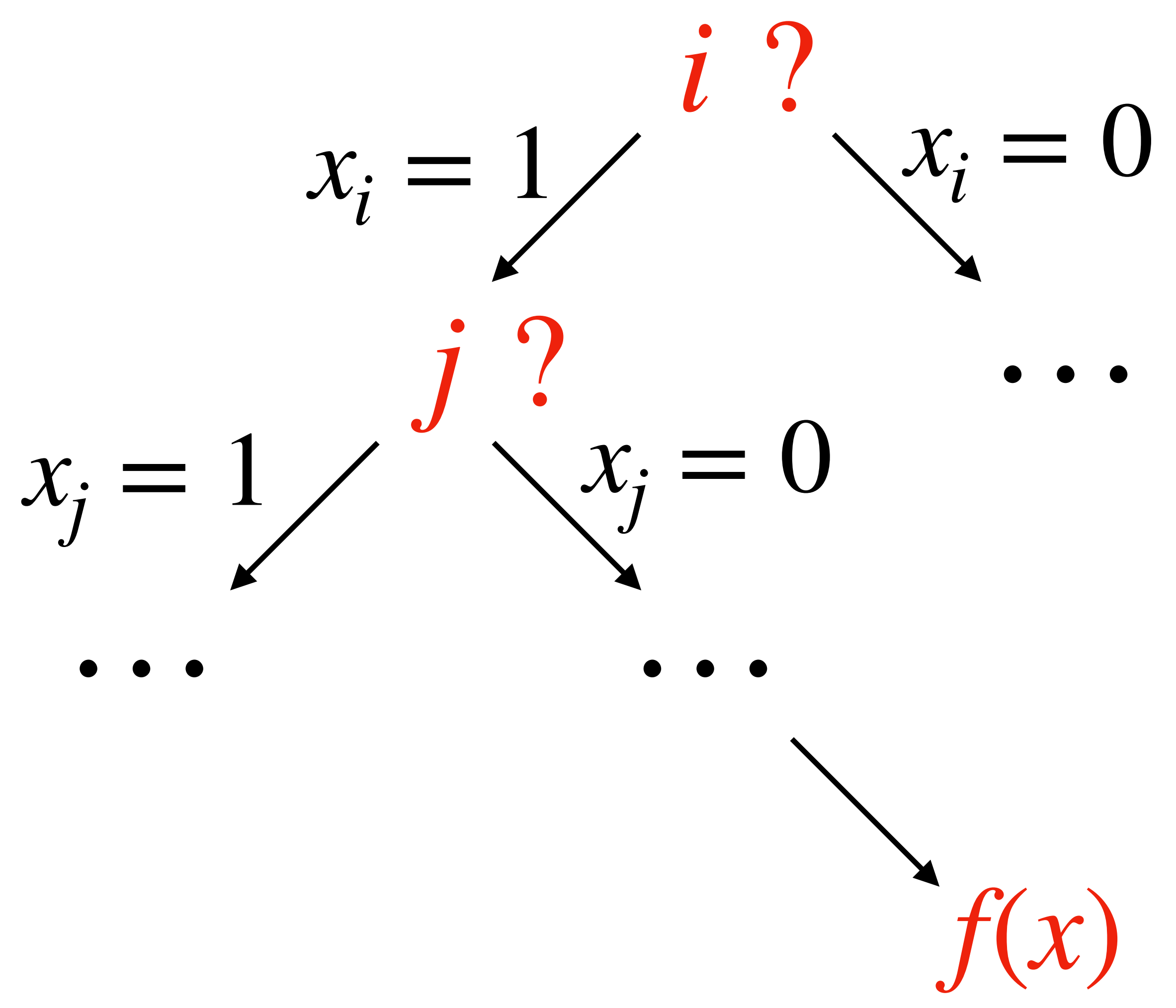
For (most of) the course we will focus on computing boolean functions

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

with Boolean evaluation queries

$$i \mapsto x_i$$

on n-bit inputs $x \in \{0,1\}^n$.



Examples:

OR: $f(x) = 0$ if and only if $x = (0,0,\ldots,0)$

PARITY: $f(x) = x_1 \oplus \ldots \oplus x_n$

MAJORITY: $f(x) = 1$ if and only if $x_1 + \ldots + x_n \geq n/2$

Classical query complexity

(a.k.a. decision tree complexity)

Deterministic query complexity

- We say that a decision tree computes $f : \{0,1\}^n \to \{0,1\}$ if for all $x \in \{0,1\}^n$ its evaluation path ends at a leaf labeled by $f(x)$.

- The deterministic query complexity $D(f)$ of $f : \{0,1\}^n \to \{0,1\}$ is the smallest height over the decision trees computing f.

<u>Fact:</u> $D(f) \leq n$

Classical query complexity

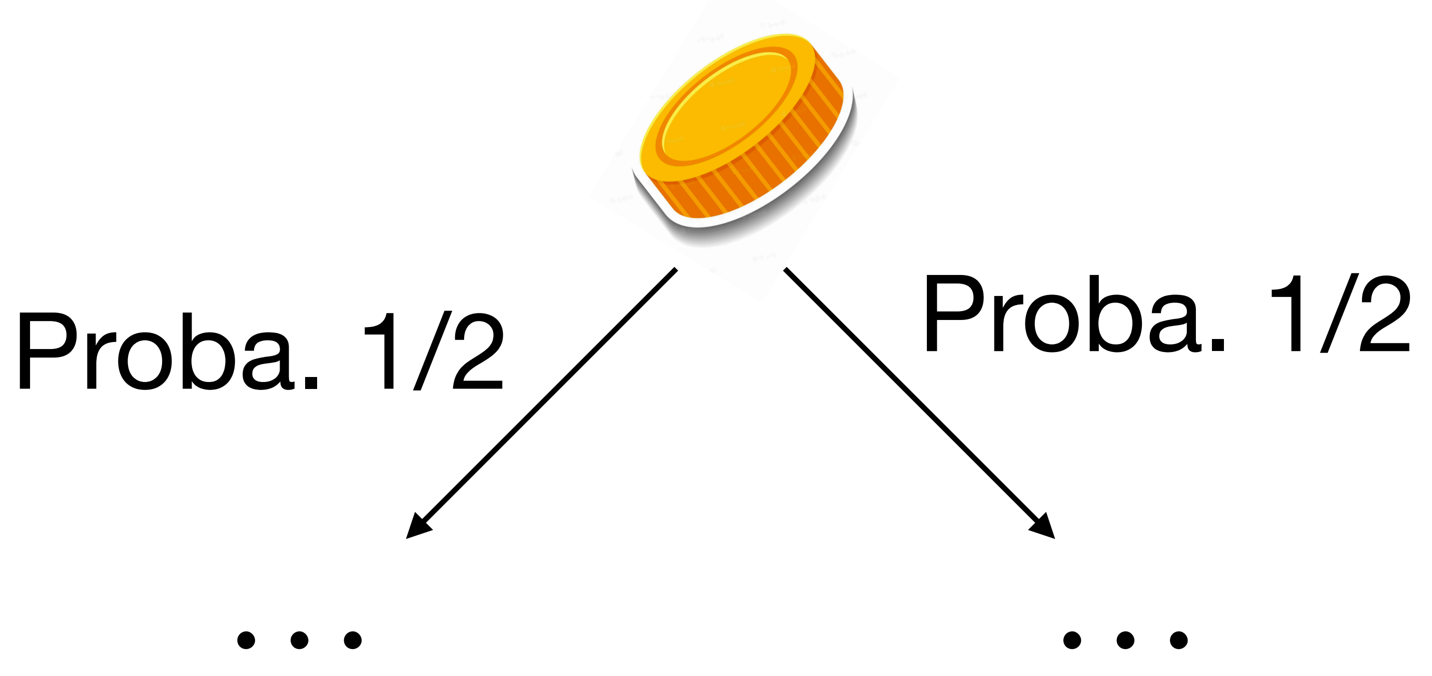
(a.k.a. decision tree complexity)

Randomized query complexity

The algorithm has access to randomness and is allowed for a small error probability.

Definition 1

Decision tree + coin nodes



=

Definition 2

Fix all the randomness "in advance"

1/ Fix a (deterministic) decision tree D_r for each random seed $r \in \{0,1\}^*$

2/ On input x, sample $r \sim \{0,1\}^*$ and run the corresponding decision tree D_r

Classical query complexity

(a.k.a. decision tree complexity)

Randomized query complexity

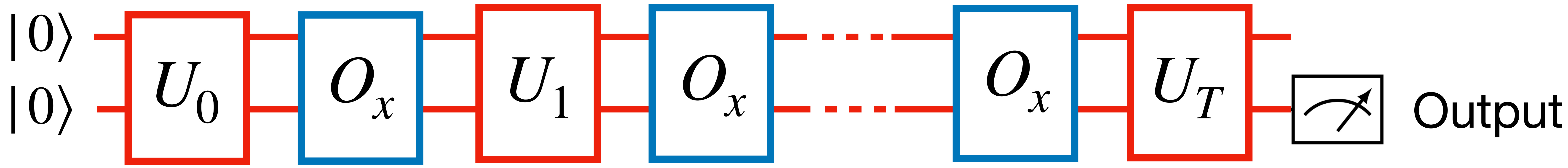
- We say that a randomized decision tree computes f if for all x its evaluation path ends at a leaf labeled by $f(x)$ with probability at least $2/3$.

- The randomized query complexity $R(f)$ of f is the smallest height over the randomized decision trees computing f.

Fact: $R(f) \leq D(f) \leq n$

Quantum query complexity

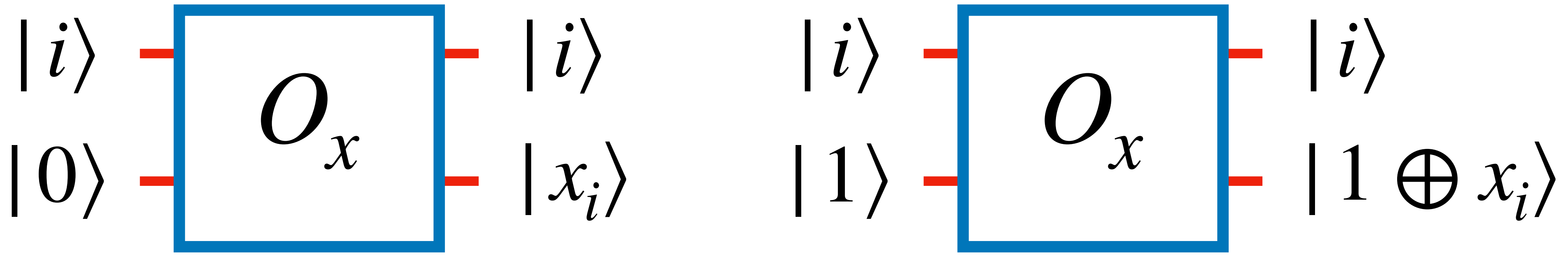
We use the circuit model instead of the decision tree formalism.



$U_0, \ldots, U_T$ are arbitrary unitary operators that don't depend on the input x

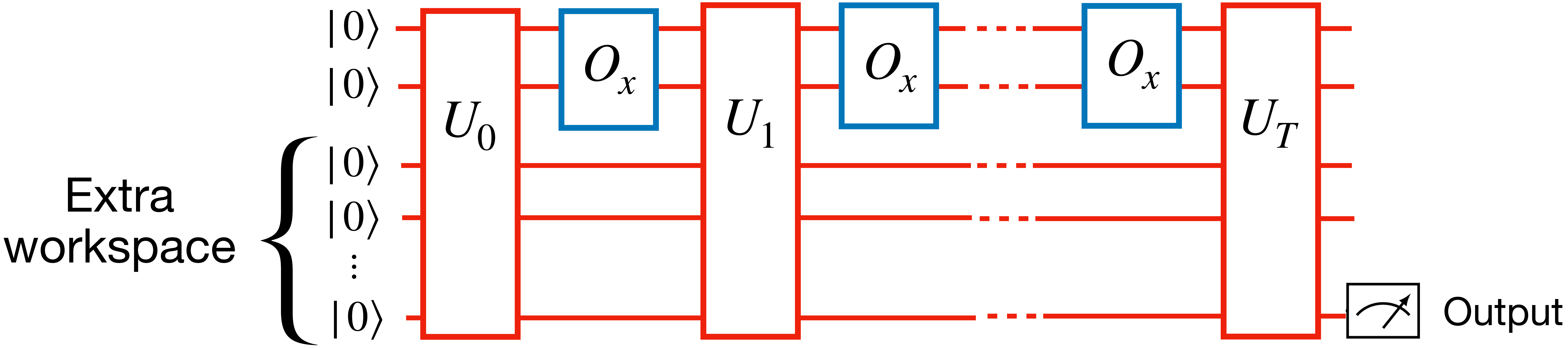
O_x is the oracle gate:

$$O_x |i, b\rangle = |i, b \oplus x_i\rangle$$



Quantum query complexity

Technically, $U_0, \ldots, U_T$ could act on a larger Hilbert space:



We omit this aspect of the model in the lectures (easy to handle)

(In fact, finding lower bounds that are sensitive to the workspace size is a major research problem)

Quantum query complexity

- We say that a quantum circuit computes f if for all x it outputs $f(x)$ with probability at least $2/3$.

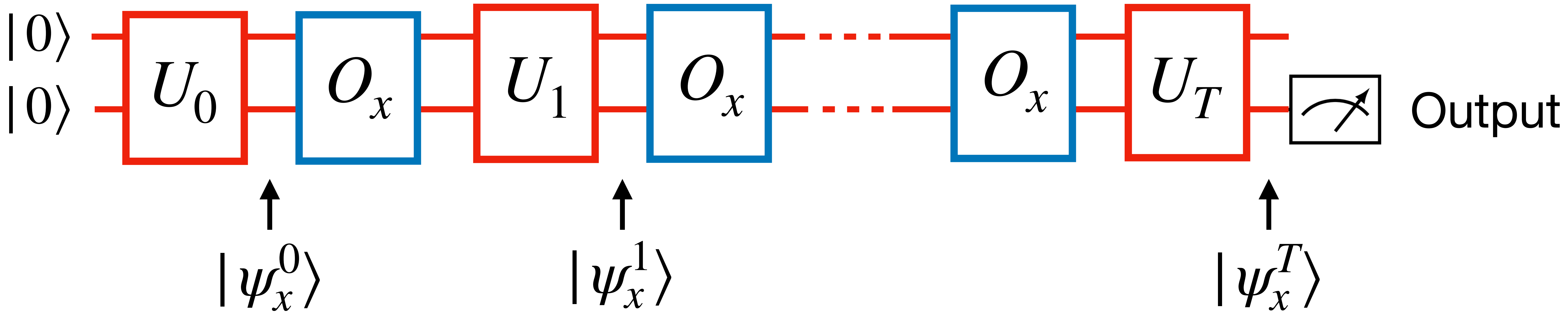
- The quantum query complexity $Q(f)$ of f is the smallest number of oracle gates over the quantum circuits computing f.

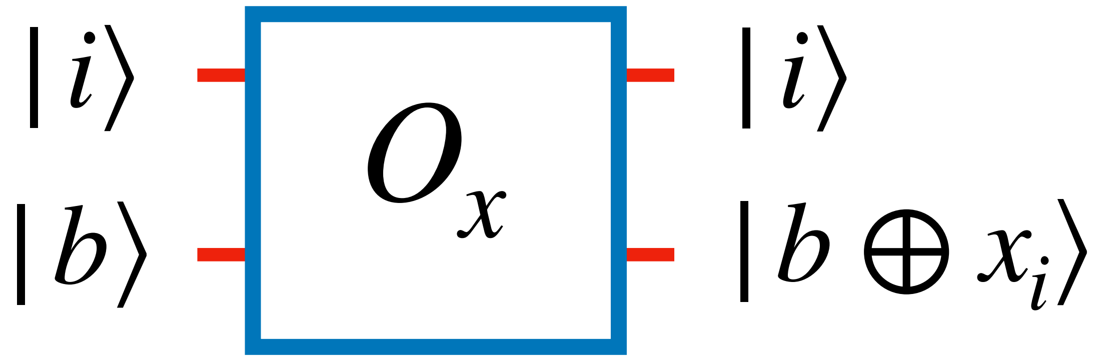
Fact: $Q(f) \leq R(f) \leq D(f) \leq n$

Two main families of quantum lower bounds

Polynomial methods		Adversary methods	
Symmetrization	Lecture 2	Hybrid	Lecture 1
Dual polynomials	Lecture 2	Positive	Lectures 4-5
Completely bounded forms		Spectral	Lectures 4-5
Laurent polynomials		Negative	Lectures 4-5
…		Multiplicative	
		Recording	Lecture 3
		…	

The hybrid method



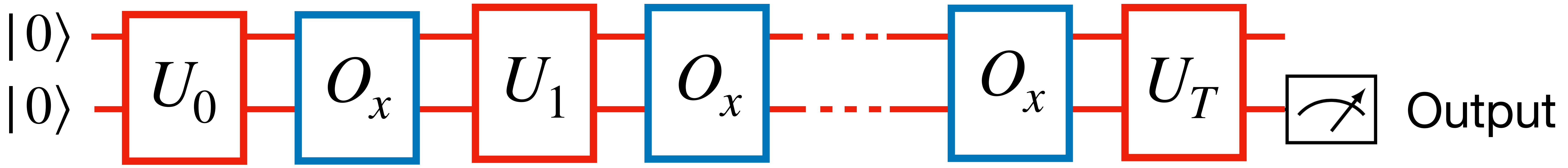


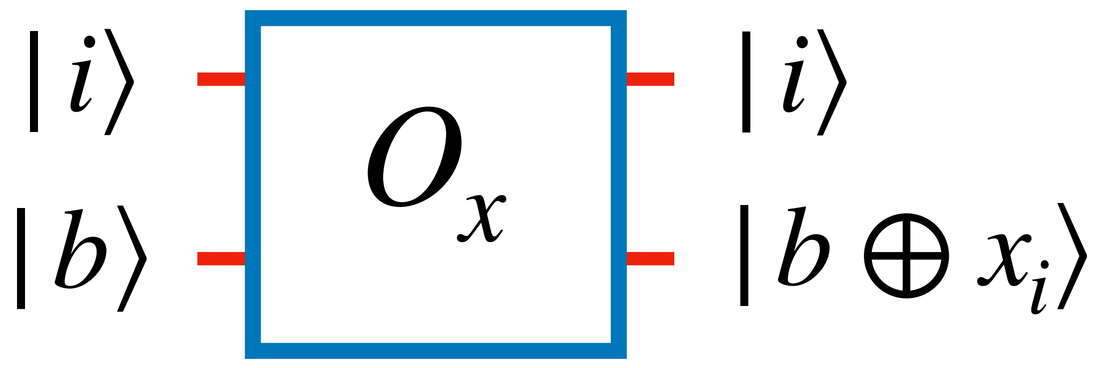
Fix $x \in \{0,1\}^n$ and denote the state of the algorithm after t queries to x :

$$|\psi_x^t\rangle = U_t O_x U_{t-1} O_x \ldots U_0 |0,0\rangle$$

Intuition 1: if $f(x) \neq f(y)$ then $|\psi_x^T\rangle$ should be far from $|\psi_y^T\rangle$

Intuition 2: distinguishing x from y requires querying some indices where $x_i \neq y_i$





$$|\psi_x^t\rangle = U_t O_x U_{t-1} O_x \dots U_0 |0,0\rangle$$

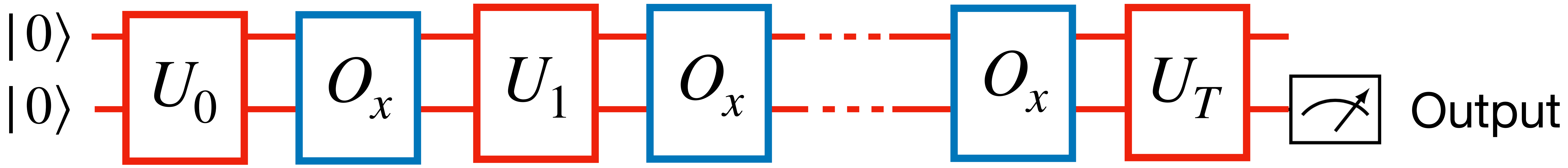
<u>OR function:</u> $f(x) = 0$ if and only if $x = (0,0,\dots,0)$

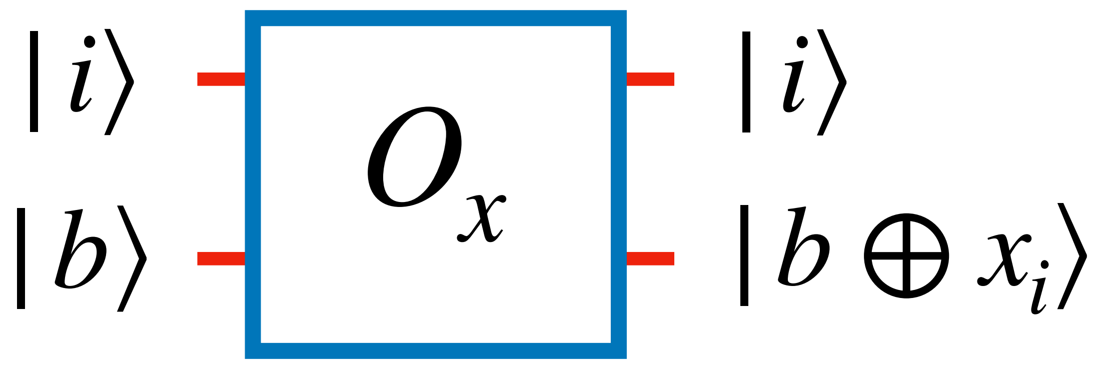
In the proof, we only focus on the $n+1$ "hardest" inputs denoted by:

$\vec{0} = (0,0,\dots,0)$ $\quad \vec{1} = (1,0,\dots,0)$ $\quad \vec{2} = (0,1,0,\dots,0)$ $\quad \dots$ $\quad \vec{n} = (0,0,\dots,1)$

We define the query weight on index i at time t to be:

$$q_i^t = \| (|i\rangle\langle i| \otimes \mathrm{Id}) |\psi_{\vec{0}}^t\rangle \|^2$$





$$|\psi_x^t\rangle = U_t O_x U_{t-1} O_x \dots U_0 |0,0\rangle \qquad q_i^t = \| (|i\rangle\langle i| \otimes \mathrm{Id}) |\psi_{\vec{0}}^t\rangle \|^2$$

For all $i \neq 0$:

Lemma 1: $\| |\psi_{\vec{0}}^0\rangle - |\psi_{\vec{i}}^0\rangle \| = 0$ *(initial condition)*

Lemma 2: $\| |\psi_{\vec{0}}^T\rangle - |\psi_{\vec{i}}^T\rangle \| \geq 1/3$ if the algorithm succeeds wp $\geq 2/3$ *(final condition)*

Lemma 3: $\| |\psi_{\vec{0}}^{t+1}\rangle - |\psi_{\vec{i}}^{t+1}\rangle \| \leq \| |\psi_{\vec{0}}^t\rangle - |\psi_{\vec{i}}^t\rangle \| + \sqrt{q_i^t}$ *(evolution)*

Theorem: $Q(\mathrm{OR}) \geq \sqrt{n}/3$