Algorithms for quantum computers

Andrew Childs
Department of Combinatorics & Optimization
and Institute for Quantum Computing
University of Waterloo

IQC

What is a computer?

A means for performing calculations by
following a sequence of instructions.

Turing machines

In 1936, Alan Turing formulated what has become the standard mathematical model of computation.

The Turing machine is a mathematical abstraction of a concrete physical process.

The Church-Turing thesis

Church-Turing Thesis

Any calculation that can be performed by mechanical means can be performed by a Turing machine.

Consistent with everything we know about physics.

What about efficiency?

Strong Church-Turing Thesis

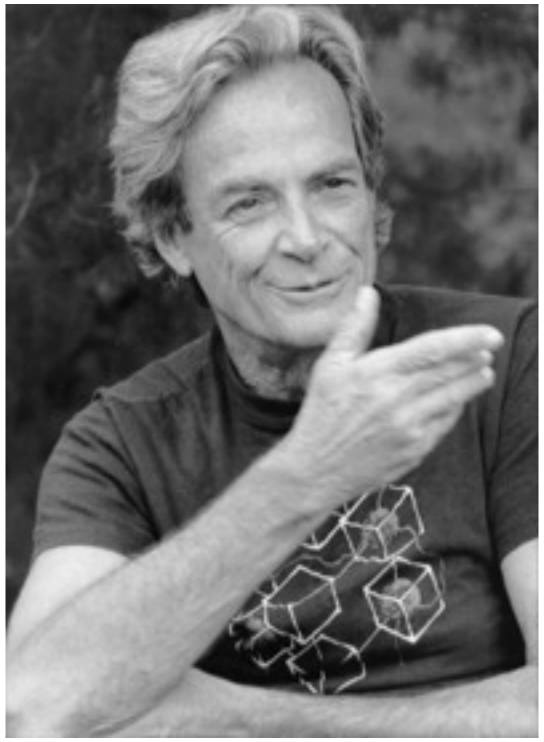
Any calculation that can be performed efficiently by mechanical means can be performed efficiently by a Turing machine.

Challenging the strong Church-Turing thesis

Quantum mechanics seems to be hard for computers to simulate.

A system of n quantum particles is described by 2^n complex numbers. We don't know how to predict the outcomes of experiments using less than an exponential amount of computation.

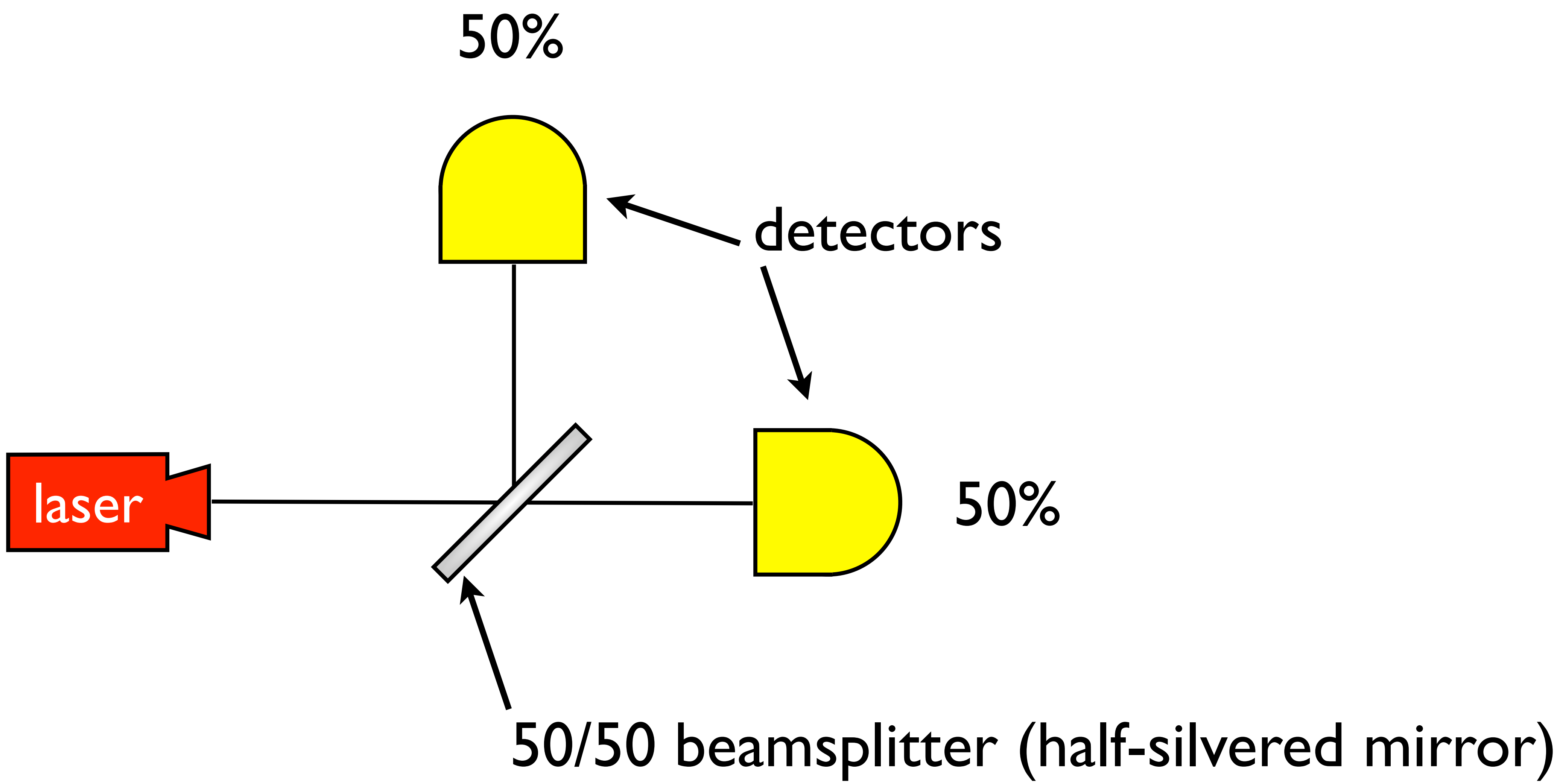
"As far as I can tell, you can simulate this with a quantum system, with quantum computer elements. It's not a Turing machine, but a machine of a different kind."



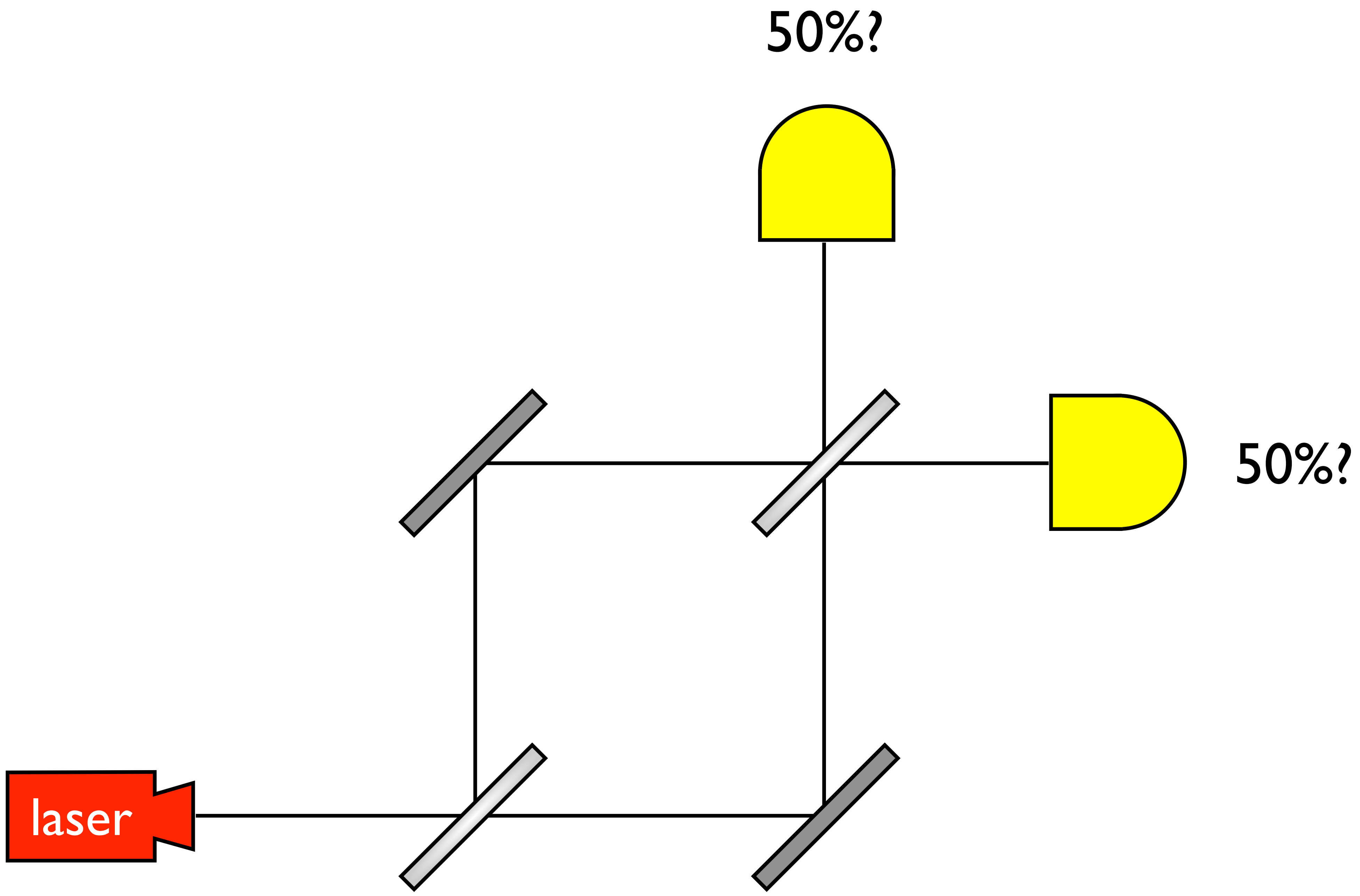
Do quantum systems naturally perform exponentially hard calculations?

Can we harness this power?

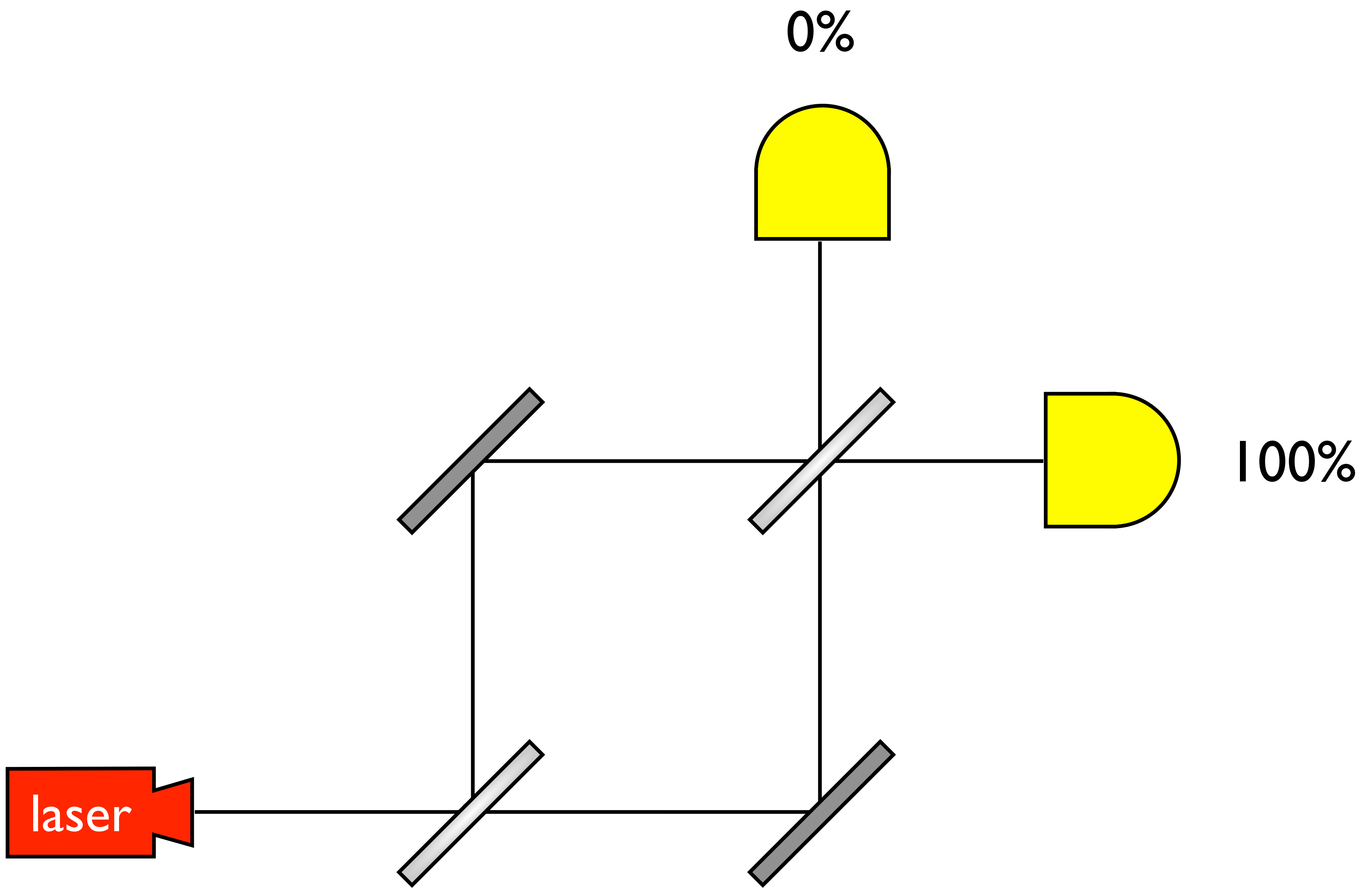
A simple experiment



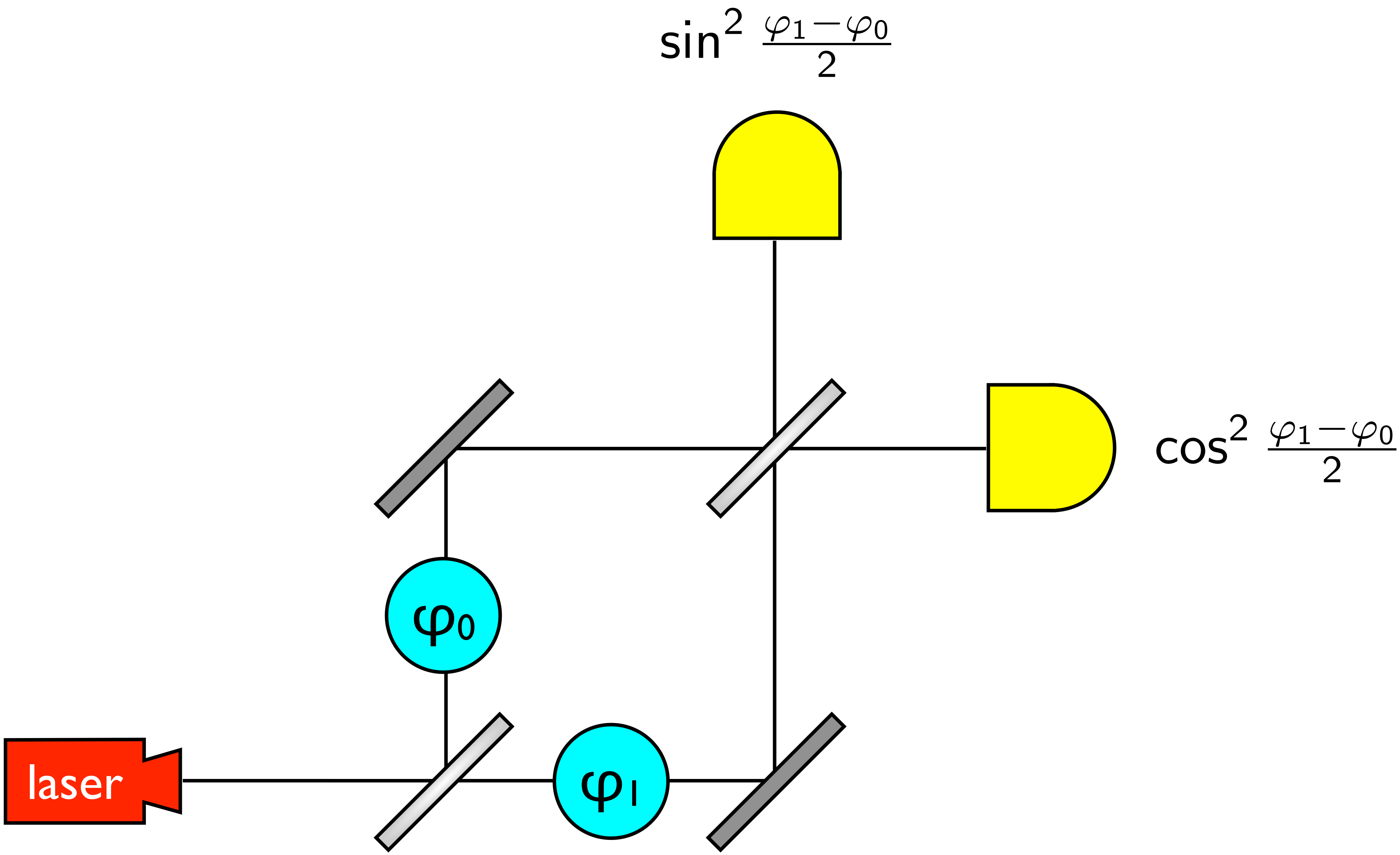
Interferometer



Interferometer



Phase shifts



Deutsch's problem

Given: A function $f: \{0,1\} \rightarrow \{0,1\}$
(As a black box: You can call the function f, but you can't read its source code.)

Task: Determine whether f is constant.

Four possible functions:

x	$f_1(x)$
0	0
1	0

x	$f_2(x)$
0	1
1	1

constant

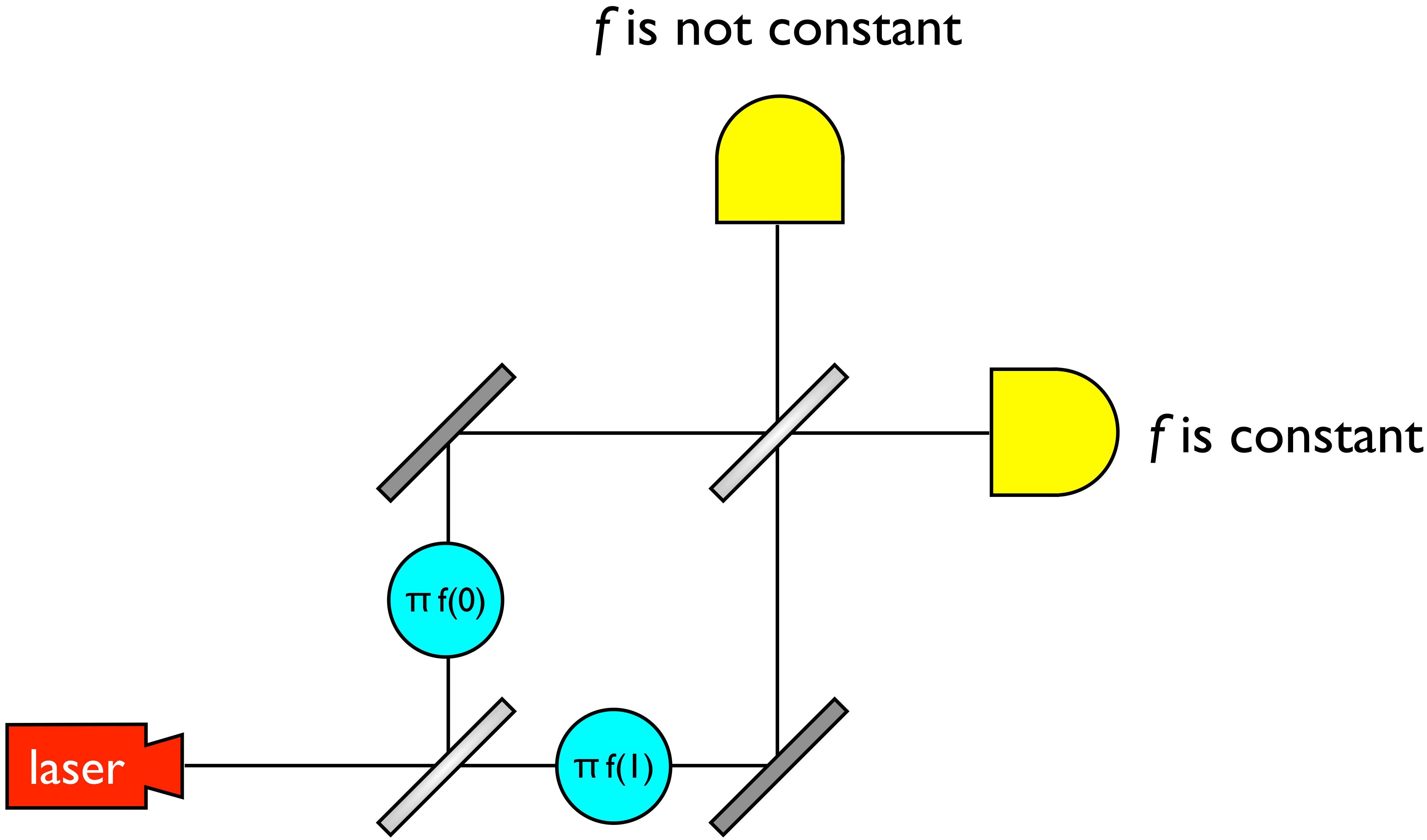
x	$f_3(x)$
0	0
1	1

x	$f_4(x)$
0	1
1	0

not constant

Classically, two function calls are required to solve this problem.

Deutsch's algorithm



Factoring integers

Factoring integers is believed to be computationally difficult.

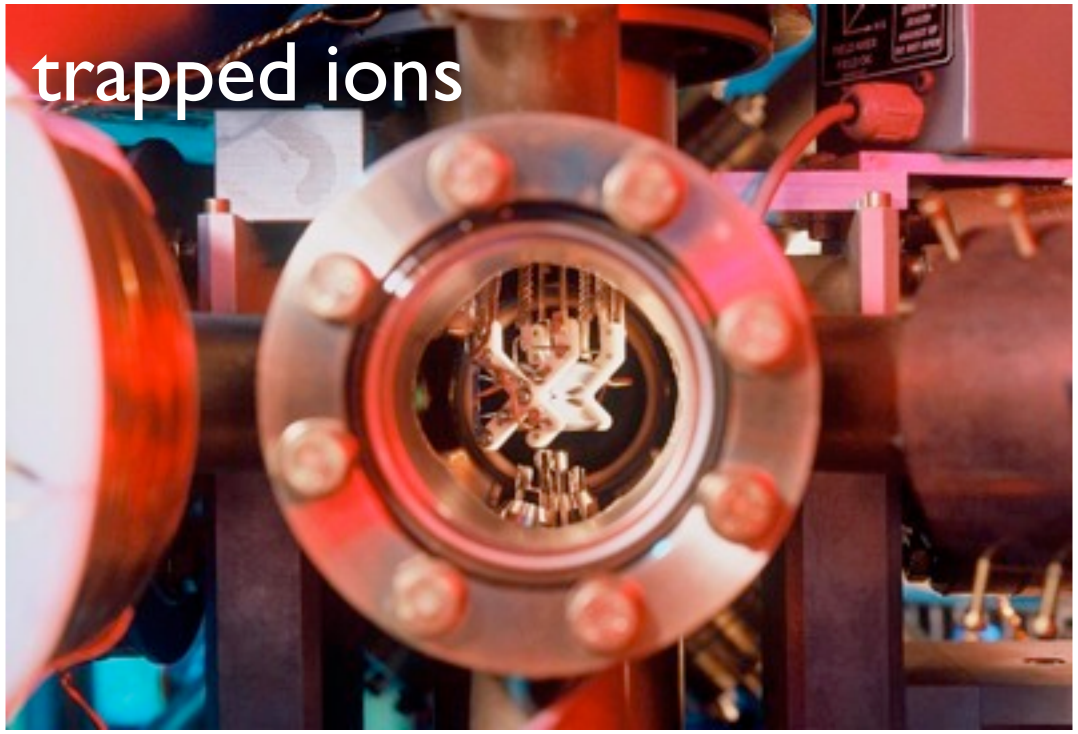
$$\begin{array}{c} 310741824049004372135075003588856793003734602284 2 \\ 7275457201619488232064405180815045563468296717232 \\ 8678243791627283803341547107310850191954852900733 \\ 772482278352574238645401469173660247765234660 9 \end{array} = \begin{array}{c} 16347336458092538484431338838650908598417836700 33 \\ 0923121811108523893331001045081512121181675115 79 \\ \times \\ 19008712816648221131268515739354139754718967899 68 \\ 515493666638539088027103802104498957191261465571 \end{array}$$

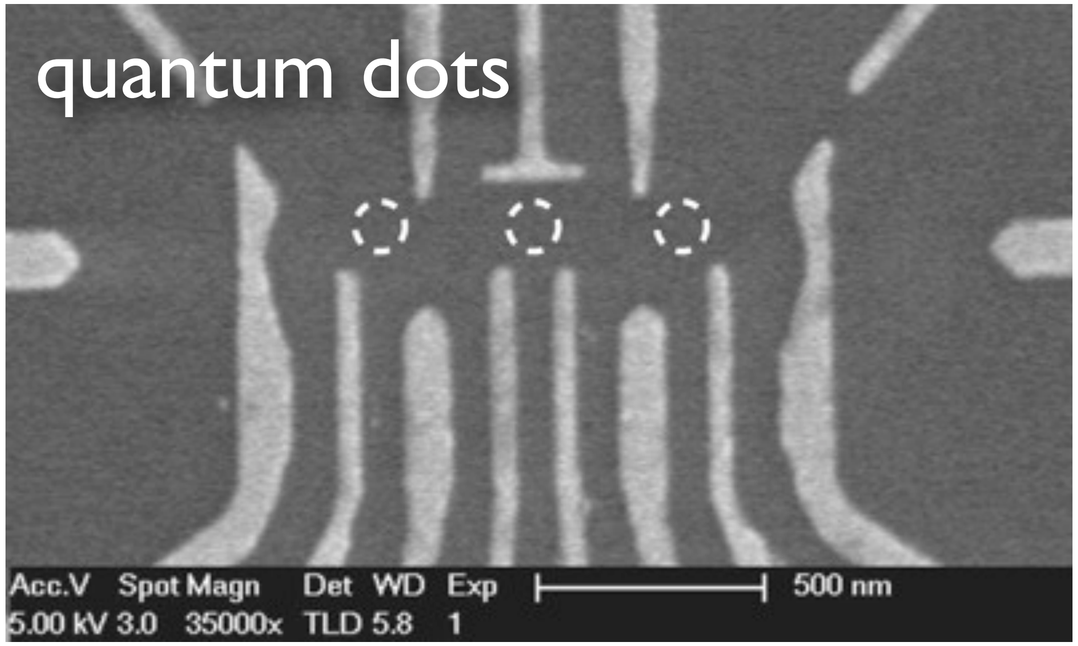
The security of modern electronic commerce relies on this assumption!

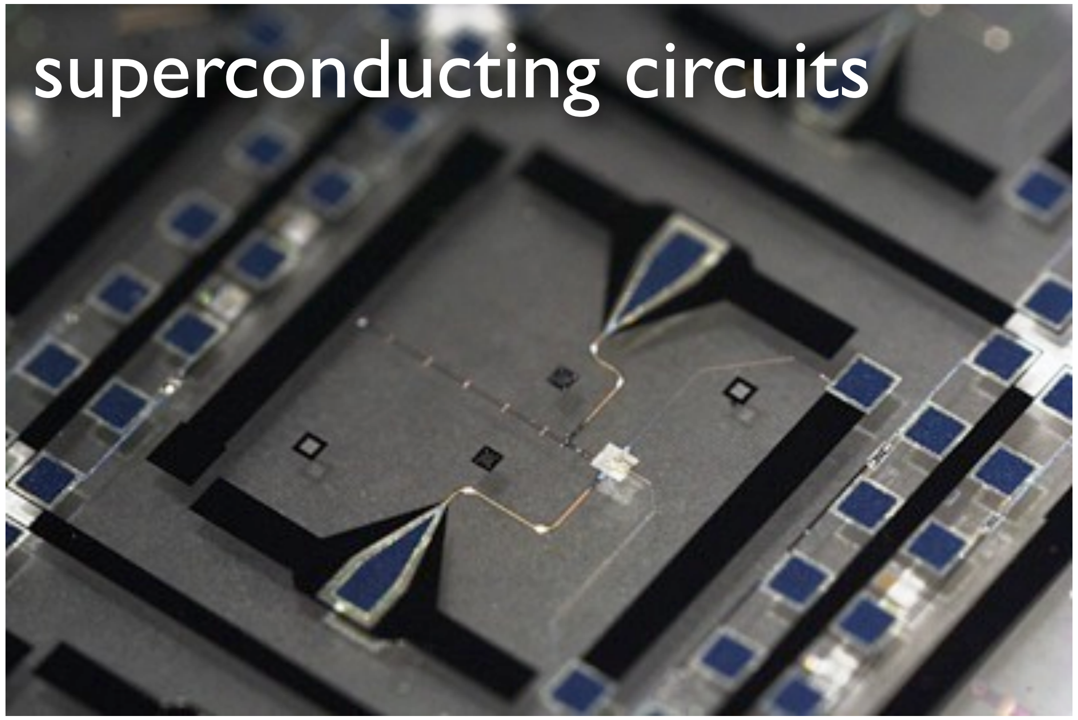
In 1994, Peter Shor showed that quantum computers can efficiently factor integers.

In a nutshell: Quantum computers can efficiently detect periodicity. The periodicity of the powers of a number modulo N is closely related to the prime factorization of N.

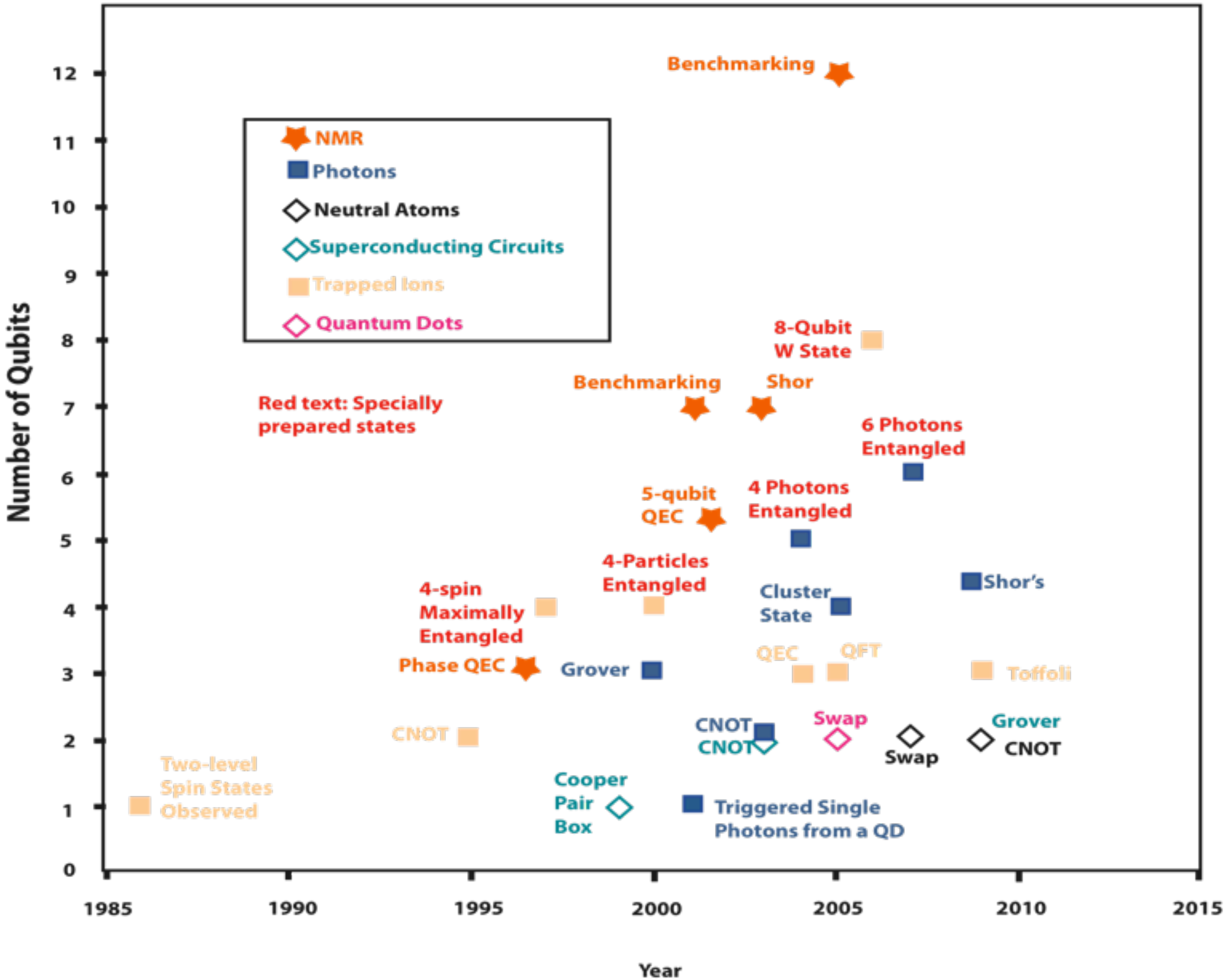
Building a quantum computer







Experimental progress



What are quantum computers good for?

Factoring integers

Other hard number theory problems
(Pell's equation, counting points on curves, ...)

Simulating quantum mechanics

Search

Approximating topological invariants

⋮

→

A two-player game

Consider a two-player game between Andrea and Orlando where

- Andrea goes first
- players alternate moves
- each player has two possible moves during their turn ("left" or "right")
- there are a total of k turns
- the winner after any given sequence of moves ($n = 2^k$ possibilities) is given by a function $f\colon \{L, R\}^k \to \{0, 1\}$

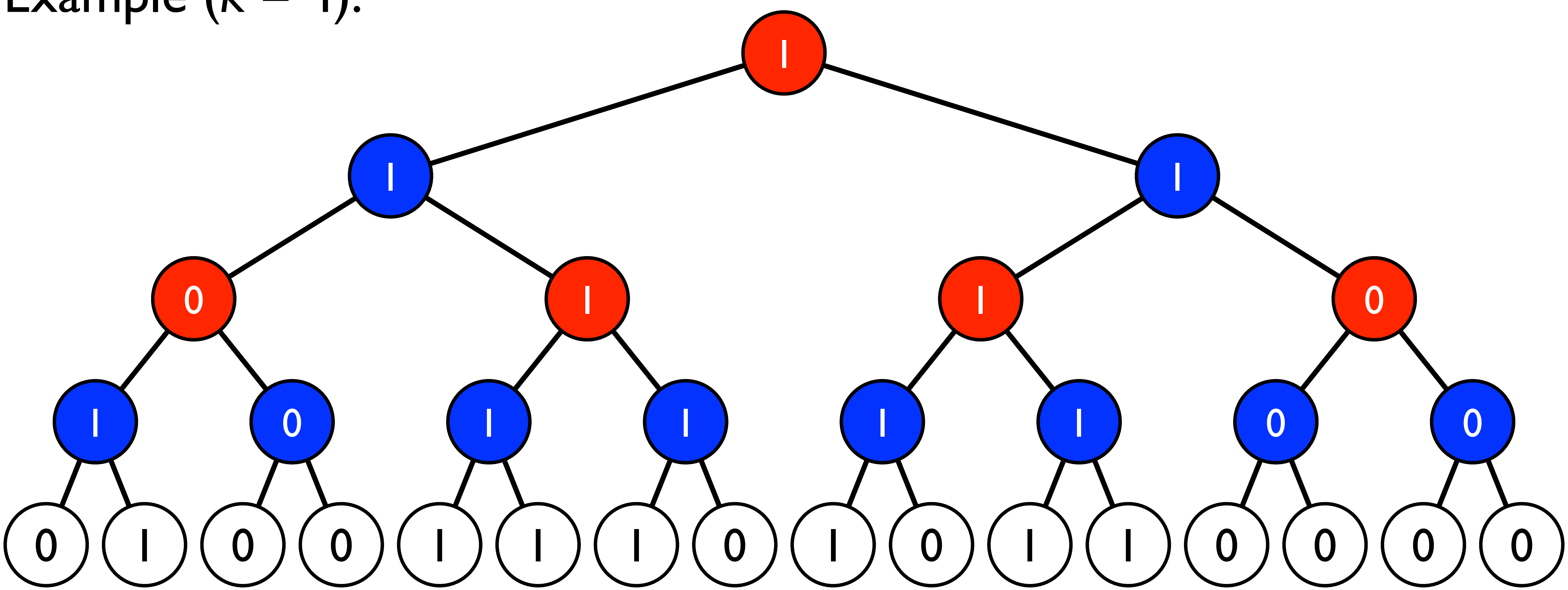
Problem: How many times must we call the function f to decide whether Andrea has a winning move?

Game trees

Evaluate a tree of AND and OR gates:

- Andrea wins if she can make any move that gives 0
  - i.e., she only loses if all of her moves give 1 (AND)
- Orlando wins if he can make any move that gives 1 (OR)

Example ($k = 4$):



Classical algorithms

Deterministic strategy: May have to query all $n = 2^k$ leaves

Randomized strategy:

- Pick a random move (L or R)
- Recursively evaluate the corresponding subtree
- If the result determines the evaluation ($\mathrm{AND}(0, x) = 0$, $\mathrm{OR}(1, y) = 1$) then return the appropriate value
- Otherwise, evaluate the other subtree to determine the value

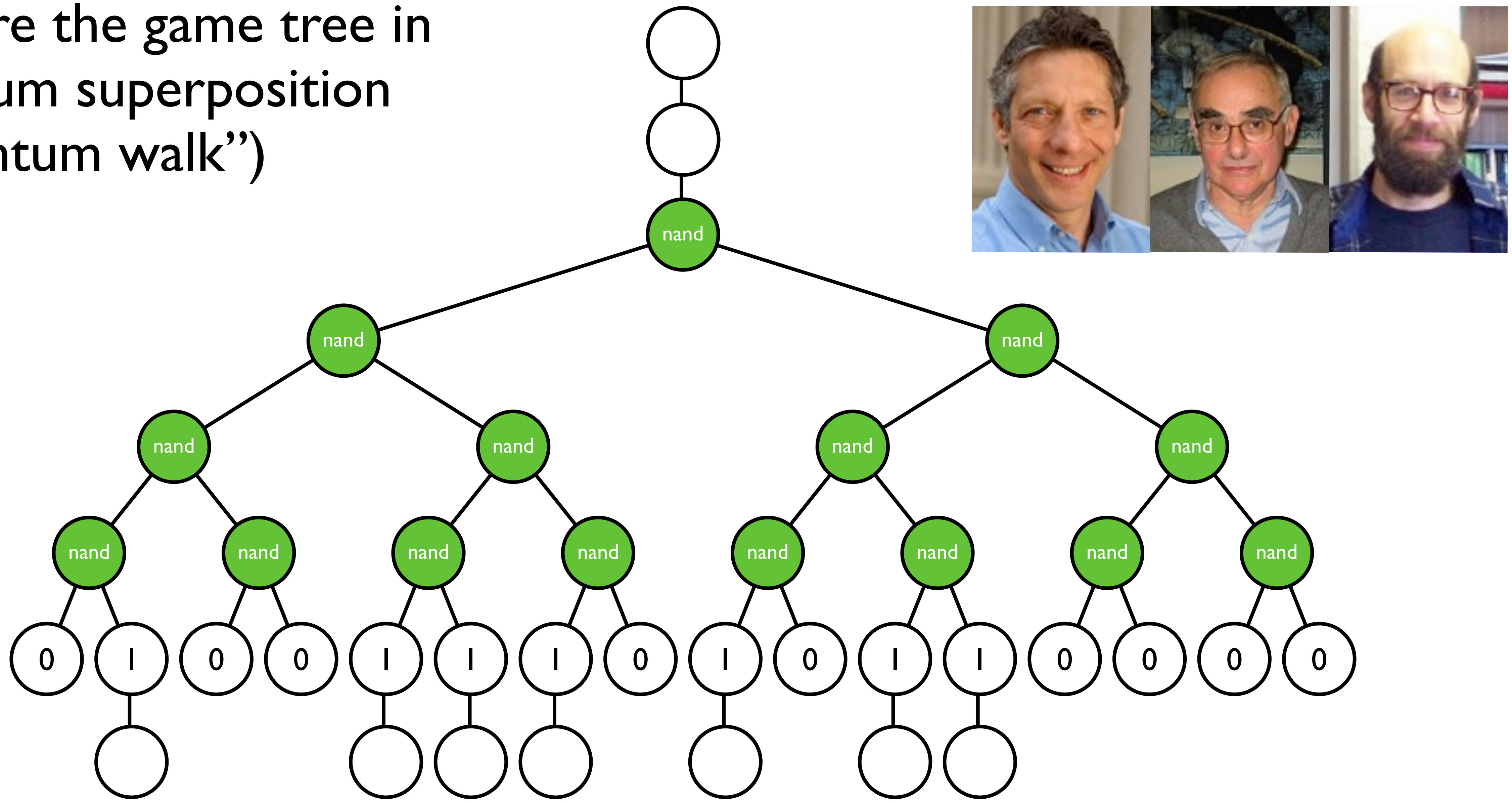
Exercise: The expected number of leaves queried is about

$$\left(\frac{1+\sqrt{33}}{4}\right)^k = n^{0.753\ldots}$$

In fact, this is optimal!

A quantum algorithm

Explore the game tree in quantum superposition ("quantum walk")



Phase depends on the value of the game

This algorithm evaluates the game with arbitrarily good success probability in only about $n^{0.5}$ queries!

The future of quantum computing

Experimental challenge: robust control of quantum systems

How can we build a scalable quantum computer?

Theoretical challenge: programming quantum computers

How can we discover new fast quantum algorithms?