Quantum & Theoretical Computer Science:

a brief background for Nai-Hui & Han-Hsuan's talk

Kai-Min Chung

Academia Sinica, Taiwan

Theoretical Computer Science

A mathematical field attempts to study everything through computational lens.

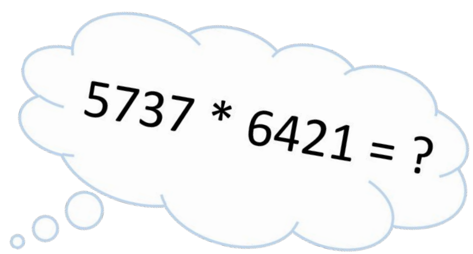
Mathematical field: We don’t do experiments. We prove theorems.

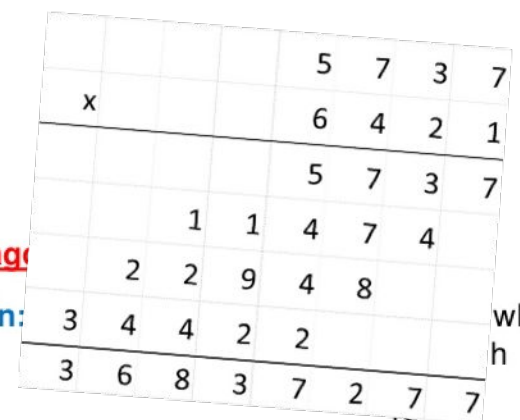
Computation:

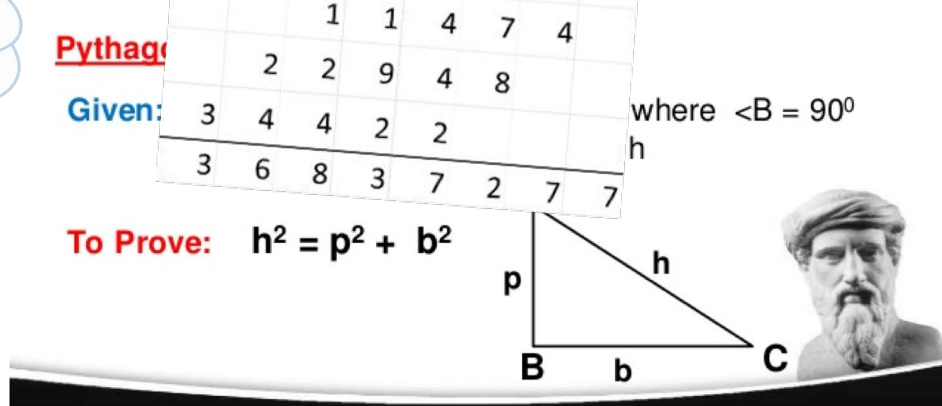
- How to solve problems / compute answers efficiently? (Algorithm)
- How fast it can be? (Complexity theory)

For example:

- Multiplication
- Factoring
- Proving theorem







Computational Lens

Evolution: Abundant life on Earth today.

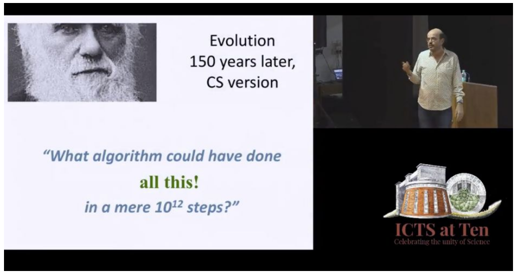
What efficiency algorithm can do it in merely 4.5 billion years?

"How do you search for a 3×10^9-long string in 3×10^9 years?"

Les Valiant 2007

Economics (game theory): Market converges to certain equilibria. How to find it?

"If your laptop can't find it, then neither can the market." Kamal Jain



Talk: "The Algorithmic Lens: How the Computational Perspective is Transforming the Sciences" by Christos H Papadimitriou

Concepts of Algorithm and Complexity

=

Algorithms and Running Time

- Problem is defined by input/output
    - E.g., Finding the largest number in {100, 200, 5, 1, 10}.

Algorithms and Running Time

Input

- Problem is defined by input/output
  - E.g., Finding the largest number in {100, 200, 5, 1, 10}.

Algorithms and Running Time

Output

- Problem is defined by input/output
    - E.g., Finding the largest number in {100, $\boxed{200}$, 5, 1, 10}.

Algorithms and Running Time

- Problem is defined by input/output
  - E.g., Finding the largest number in {100, 200, 5, 1, 10}.
- Algorithm: given the input, find the output.
  - E.g., Read/compare numbers one by one.

Algorithms and Running Time

- Problem is defined by input/output
  - E.g., Finding the largest number in {100, 200, 5, 1, 10}.
- Algorithm: given the input, find the output.
  - E.g., Read/compare numbers one by one.
- Evaluate the running time of an Alg. in terms of input size.
  - E.g., Given **n** numbers, find the largest number.
  - <u>Input size: n</u> / <u>Running time: O(n)</u>

Running Time & Input Size

Evaluate the running time of an Alg. in terms of input size.

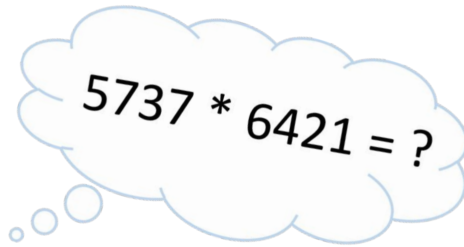
Want the running time to grow “slowly” with the input size.

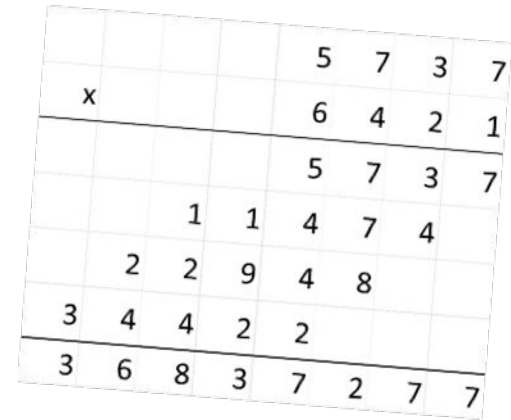
- An algorithm is efficient if its running time is in poly(n).
  - E.g., n^2, n^{100}, 10000n.
- An algorithm is inefficient if its running time is >poly(n).
  - E.g., 2^n and $n^{\log n}$.

Examples

Multiplication

- Input: two n digits numbers p, q
- Output: number N = p * q
- Textbook multiplication: $O(n^2)$ time





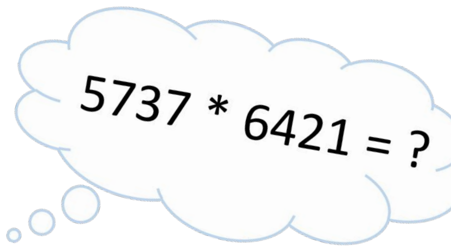
				5	7	3	7
x				6	4	2	1
				5	7	3	7
		1	1	4	7	4	
	2	2	9	4	8		
3	4	4	2	2			
3	6	8	3	7	2	7	7

Factoring

- Input: an n digit number N
- Output: factors p, q s.t. N = p * q
- Best classical algorithm: $O\left(2^{\sqrt[3]{n}}\right)$ time
- Shor's quantum algorithm: $O(n^3)$ time

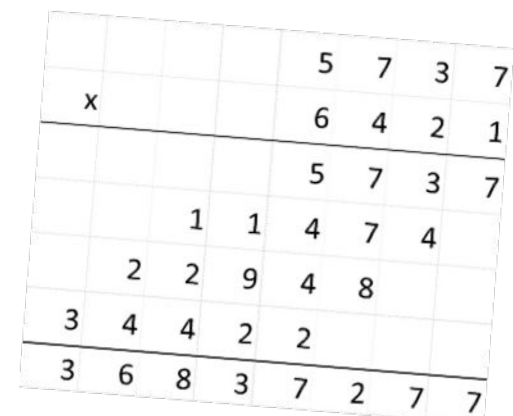
Mathematical Methodology

- Design an algorithm A to solve the problem
- Correctness: Show A always outputs correct answers
- Efficiency: Show A always terminate in desired runtime



Multiplication

- Algorithm: textbook multiplication
- Correctness: output = p * q
- Efficiency: terminate in $O(n^2)$ time



A needs to work **for all** input instances

- There are infinite instances!
- Can't rely on experiments --- rigorous proof needed!

Complexity Measure

- Time complexity: # steps to finish the computation
- Space complexity: required memory for the computation
- Computation Model:
  - Conventional computer (RAM machine)
  - Turing machine
  - Quantum circuit (Clifford, T, Toffoli gates, etc)
  - Classical circuit (AND, OR, NOT gates, etc)
  - Classical-quantum hybrid model
- Depth complexity:
  - # parallel steps to finish the computation

Nai-Hui's Talk: On the Need for Large Quantum Depth

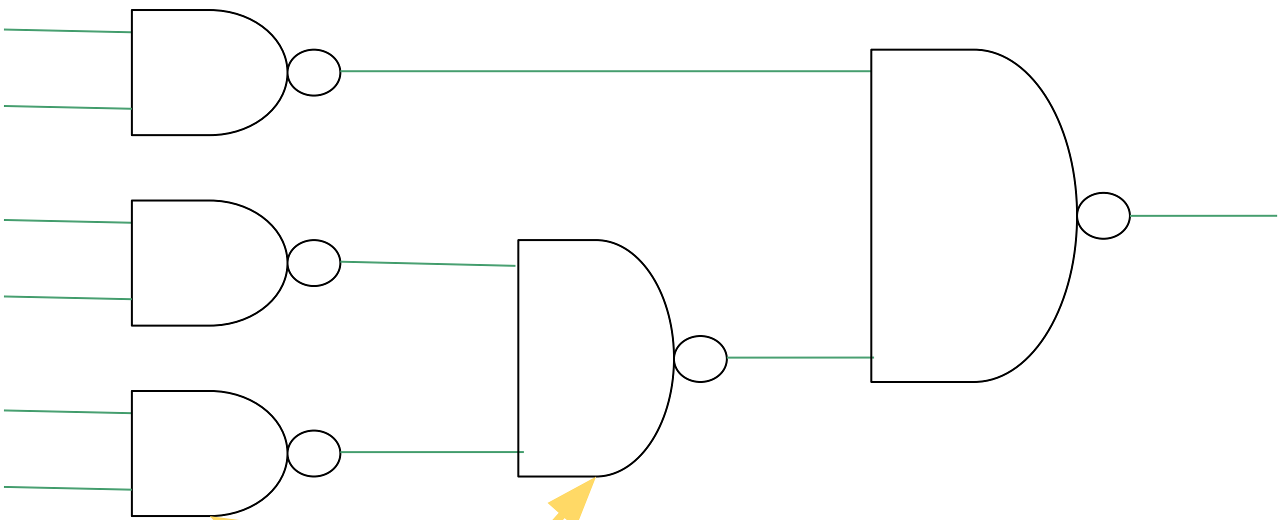
- Motivation: near-term quantum computer has small depth
  - What can near-term quantum computers do?

- Computation Models:
  1. Classical + small-depth quantum computation
  2. Polynomial time quantum computation (no depth limits)

- Do they have the same computational power?
  - Can solve the same set of computation problems? or
  - ∃ a problem s.t. it can be solved by 2 but not 1?

Classical Circuit

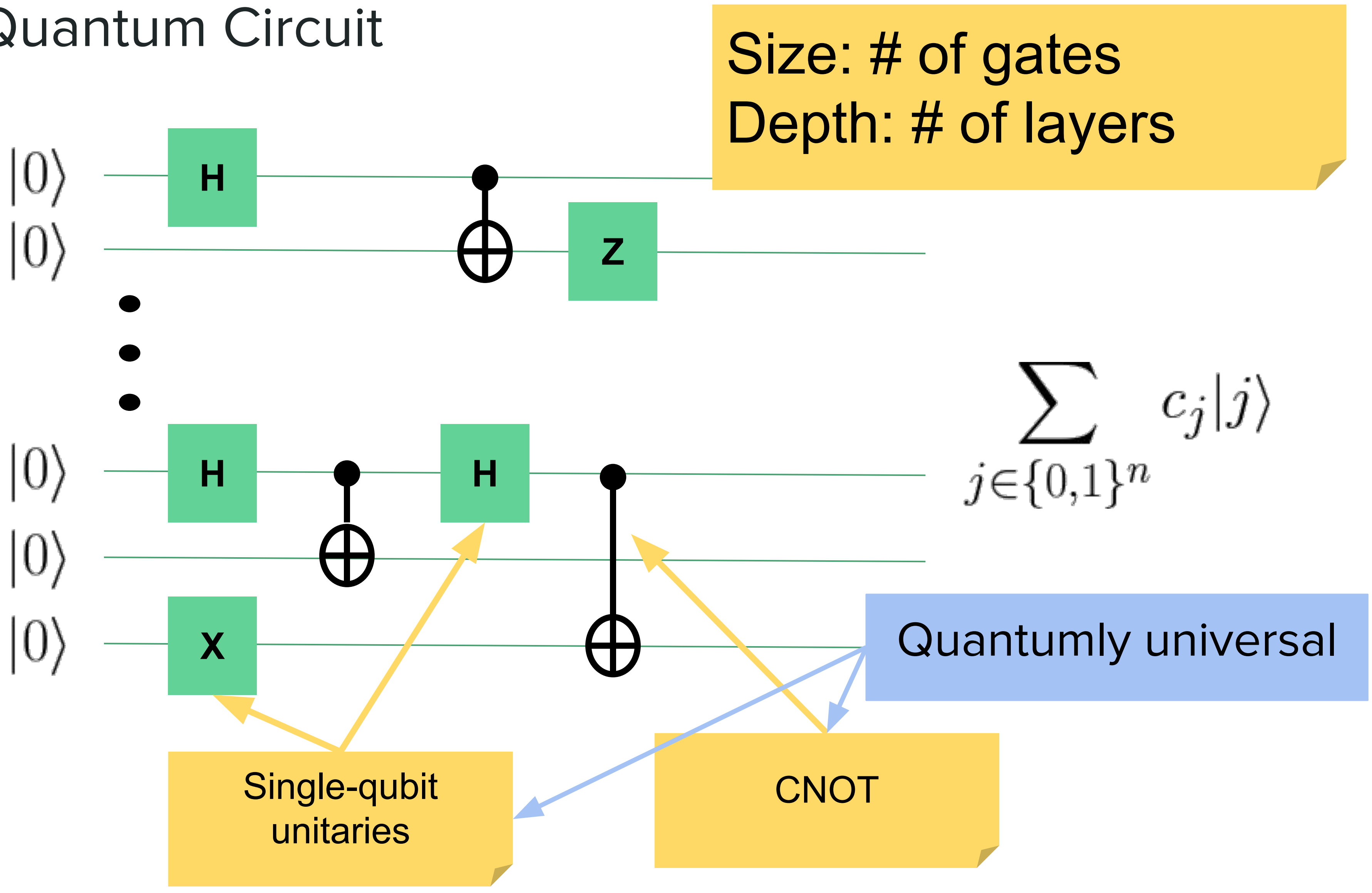
Size: # of gates
Depth: # of layers



NAND

Classically universal

Quantum Circuit



Han-Hsuan’s: Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning

- Quantum-inspired / dequantized classical algorithms
    - Unexpected classical algorithms match quantum ones

- Sublinear time algorithms for several problems
    - E.g., Semi-definite programming (SDP), matrix inversion

- *Sublinear time:* do not have time to read all input!
    - Quantum: QRAM access to the input
    - Classical: sampling access to the input

Provable vs. Heuristic Algorithms

- Not all algorithms can be analyzed provably
- Heuristic algorithms: lack of provable guarantees, but can be very powerful and play important role!
  - Machine learning, SAT solver, genetic algorithms, simulation annealing
- Quantum heuristic algorithms
  - Quantum annealing, VQE, QAOA, adiabatic alg., etc
  - May lead to first near-term application for QC
- Research methodology
  - Rely on solid experimental evidence, and sometimes, good theoretical explanations (e.g., for SAT solvers)
  - Caution: current QC is still too small for exp. evidence...

Complexity Theory 101 – Formal Definitions

=

Computational Problems

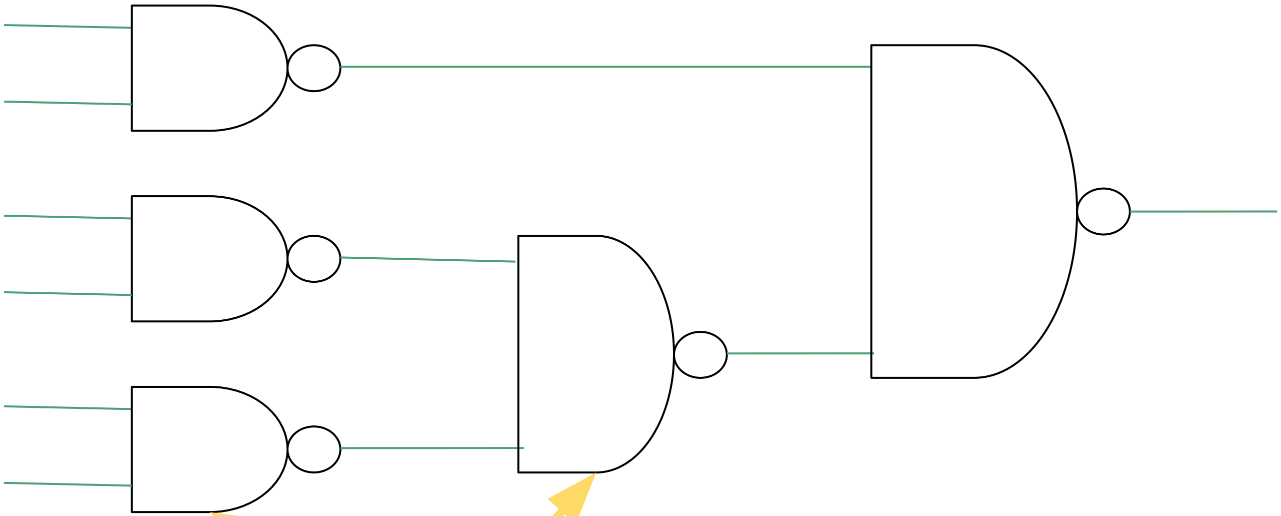
Just a function from bit strings to bit strings $f: \{0,1\}^* \rightarrow \{0,1\}^*$

- For each input size n: $f_n: \{0,1\}^n \rightarrow \{0,1\}^{m(n)}$
- Goal: understand the difficult of computing f
  - Input: $x \in \{0,1\}^n$
  - Output: $f_n(x) \in \{0,1\}^{m(n)}$
  - Complexity: e.g., time $T(n)$, space $S(n)$, depth $D(n)$
  - E.g., Multiplication, x = two numbers, $f(x)$ = their multiplication

Decision problem (Language): $L: \{0,1\}^* \rightarrow \{0,1\}$

Classical Circuit

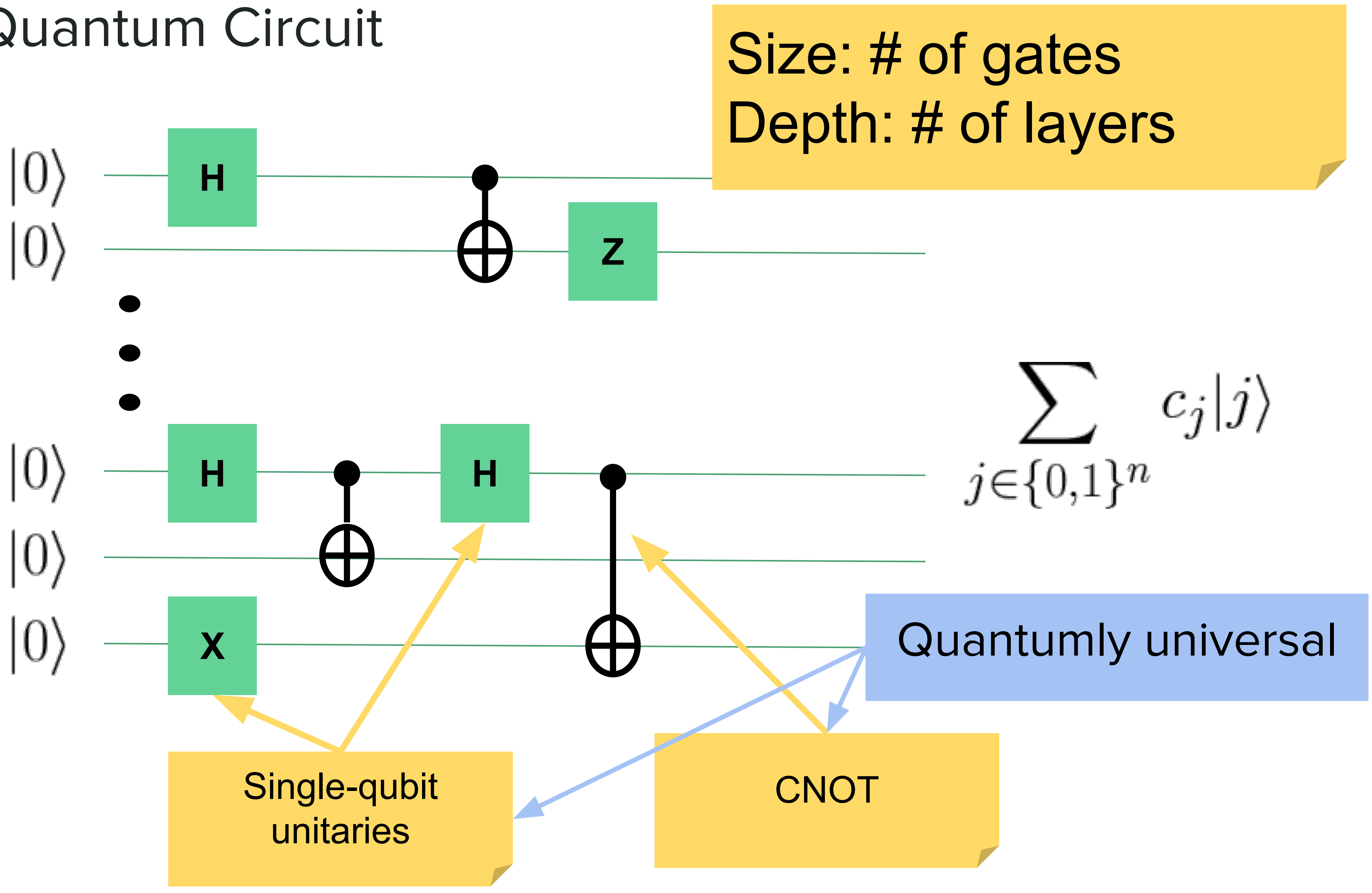
Size: # of gates
Depth: # of layers



NAND

Classically universal

Quantum Circuit



Circuit Complexity

Given a computation problem $f: \{0,1\}^* \to \{0,1\}^*$

We say a sequence of circuits $\{C_n\}$ computes f if

- For all n, for all $x \in \{0,1\}^n$, $C_n(x) = f_n(x)$

The circuit size complexity of f is $s(n)$ if

- $\exists \{C_n\}$ computes f s.t. $SIZE(C_n) \leq s(n)$ for all n

Complexity Class: classify comp. prob. by its complexity

- E.g., P/poly = { f: f has circuit complexity $s(n) \leq poly(n)$}

Challenge: prove circuit complexity lower bound

- E.g., show some f has circuit size complexity $s(n) > n^2$

Millennium Prize Problems: P vs NP

Other Computation Model & Complexity Measure

Computation Models

- Turing Machine, Classical-Quantum Hybrid

Complexity Measure

- Time/Size, Space (memory), Depth