Quantum Computation (Preskill ch. 6)

Quantum Computation

Classical Circuits

* Universal Gates
* Circuit Complexity

Quantum Circuits

* Quantum Complexity
* Universal Quantum Gates

Review of Classical Circuit Theory

We can think of a computation as a function that maps *n* bits to *m* bits

Computation

$$\boxed{F:\{0,1\}^n \to \{0,1\}^m}$$

Equivalent

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

m of these simpler functions

Function evaluation ⟷ sequence of logic operations

Given a binary input $x = x_1 x_2 \ldots x_n$

⇨ separate in sets $\begin{cases} f(x)=1 \\ f(x)=0 \end{cases}$

Consider the input

$x^{(a)}: f(x^{(a)}) = 1$ ⇨ define $f^{(a)}(x) = \begin{cases} 1 & \text{for } x = x^{(a)} \\ 0 & \text{for } x \neq x^{(a)} \end{cases}$

one of the *m* simple functions

n of these

Given, for example, we implement $f^{(a)}$ w/logic operations

$$x = \begin{cases} 111\ldots \to f(x) = x_1 \wedge x_2 \wedge x_3 \ldots \wedge x_n \\ 0110\ldots \to f(x) = (\neg x_1) \wedge x_2 \wedge x_3 \wedge (\neg x_4) \ldots \end{cases}$$

And finally, given the $f^{(a)}(x)$'s we can implement the *m* $f(x)$'s

$$\boxed{f(x) = f^{(1)}(x) \vee f^{(2)}(x) \vee \ldots \vee f^{(n)}(x)} \Rightarrow \boxed{F(x)}$$

equivalent to m $f(x)$'s

Quantum Computation (Preskill ch. 6)

Given a binary input $x = x_1 x_2 \ldots x_n$

⇨ separate in sets $\begin{cases} f(x) = 1 \\ f(x) = 0 \end{cases}$

Consider the input

$x^{(a)} : f(x^{(a)}) = 1$ ⇨ define $f^{(a)}(x) = \begin{cases} 1 & \text{for } x = x^{(a)} \\ 0 & \text{for } x \neq x^{(a)} \end{cases}$

one of the m simple functions

n of these

Given, for example, we implement $f^{(a)}$ w/logic operations

$$x = \begin{cases} 111\ldots \rightarrow f(x) = x_1 \wedge x_2 \wedge x_3 \ldots \wedge x_n \\ 0110\ldots \rightarrow f(x) = (\neg x_1) \wedge x_2 \wedge x_3 \wedge (\neg x_4) \ldots \end{cases}$$

And finally, given the $f^{(a)}(x)$'s we can implement the m $f(x)$'s

$$\boxed{f(x) = f^{(1)}(x) \vee f^{(2)}(x) \vee \ldots \vee f^{(m)}(x)} \Rightarrow \boxed{F(x)}$$

equivalent to m $f(x)$'s

Note: This approach

* Reduces the problem of evaluating $F(x)$ to bitwise $\vee, \wedge, \neg$
* We have implicitly used COPY
* These gates suffice to implement any computation

Conclusion: The following make up a Universal Gate Set

$$\boxed{\text{OR, AND, NOT, COPY}}$$

Note: Universal gate sets are not unique

Example of a simpler Set: $\boxed{\text{NAND, COPY}}$

Quantum Computation (Preskill ch. 6)

Note: This approach

- Reduces the problem of evaluating $F(x)$ to bitwise $\vee, \wedge, \neg$
- We have implicitly used COPY
- These gates suffice to implement any computation

Conclusion: The following make up a Universal Gate Set

$\boxed{\text{OR, AND, NOT, COPY}}$

Note: Universal gate sets are not unique

Example of a simpler Set: $\boxed{\text{NAND, COPY}}$

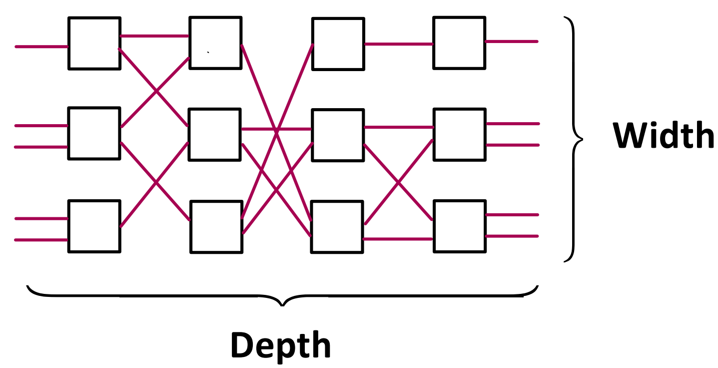
Circuit Complexity

(Pick a universal gate set)

Central Question: How hard is it to solve PROBLEM ?

- One measure is the size of the smallest circuit that solves it

$\boxed{\text{Size = Width x Depth}}$



Quantum Computation (Preskill ch. 6)

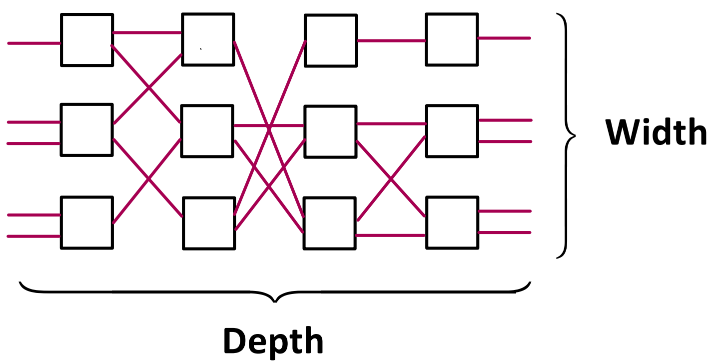
Circuit Complexity

(Pick a universal gate set)

Central Question: How hard is it to solve PROBLEM ?

* One measure is the size of the smallest circuit that solves it

Size = Width x Depth



Consider a circuit family $\{C_n\}$ that solves a decision problem $f: \{0,1\}^n \to \{0,1\}$

Examples

FACTORING $f(x,y) = \begin{cases} 1 & \text{if integer } x \text{ has divisor } < y \\ 0 & \text{otherwise} \end{cases}$

HAMILTONIAN PATH $f(x,y) = \begin{cases} 1 & \text{if graph } x \text{ has Hamiltonian Path} \\ 0 & \text{otherwise} \end{cases}$

We define:

Easy Problems: $\text{size}(C_n) \leq \text{poly}(n)$

Hard Problems: $\text{size}(C_n) > \text{poly}(n)$

This distinction allows us to define Complexity Classes, for example

Problem Class P = { Decision Problems solved by a polynomial-sized circuit }

Quantum Computation (Preskill ch. 6)

Consider a circuit family $\{C_n\}$ that solves a <u>decision problem</u> $f: \{0,1\}^n \to \{0,1\}$

Examples

FACTORING $f(x,y) = \begin{cases} 1 & \text{if integer } x \text{ has divisor } < y \\ 0 & \text{otherwise} \end{cases}$

HAMILTONIAN PATH $f(x,y) = \begin{cases} 1 & \text{if graph } x \text{ has Hamiltonian Path} \\ 0 & \text{otherwise} \end{cases}$

We <u>define</u>:

Easy Problems:	$\text{size}(C_n) \le \text{poly}(n)$
Hard Problems:	$\text{size}(C_n) > \text{poly}(n)$

This distinction allows us to define <u>Complexity Classes</u>, for example

Problem Class P = { Decision Problems solved by polynomial-sized circuit }

* Whether PROBLEM $\in P$ is independent of circuit design, universal gate set & other specifics

* Problems in P are special – they have structure that allows efficient computation

<u>Note</u>: The majority of functions $\notin P$

For example, if the output $f(x) \sim$ random we must compute $f(x)$ by lookup table with 2^n entries

Circuit that does lookup has exponential size

<u>Special Class</u>:

One-Way Function

Problem Class NP = { PROBLEM is easy or hard, but the answer is easy to check }

Stands for "Non-deterministic Polynomial Time

<u>Examples</u>: FACTORING $\in$ NP

HAMILTONIAN PATH $\in$ NP

<u>Note</u>: <u>Clearly</u> $P \subseteq NP$, <u>Conjecture</u> that $P \neq NP$

Quantum Computation (Preskill ch. 6)

* Whether PROBLEM $\in \mathcal{P}$ is independent of circuit design, universal gate set & other specifics

* Problems in $\mathcal{P}$ are special – they have structure that allows efficient computation

Note: The majority of functions $\notin \mathcal{P}$

For example, if the output $f(x) \sim$ random we must compute $f(x)$ by lookup table with 2^n entries

Circuit that does lookup has exponential size

Special Class:

One-Way Function

Problem Class NP = { PROBLEM is easy or hard, but the answer is easy to check }

Stands for "Non-deterministic Polynomial Time

Examples: FACTORING $\in$ NP

HAMILTONIAN PATH $\in$ NP

Note: **Clearly** $P \subseteq NP$, **Conjecture** that $P \neq NP$

Special Problem: CIRCUIT-SAT $\in$ NP

Input = Circuit w/*n* gates, *m* input bits

Problem = is there an *m*-bit input w/output = 1

$$f(C) = \begin{cases} 1 & \text{if } \exists\, x^{(m)} \text{ so } C(x^{(m)}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Easy to check solution because if we have the input circuit C we can run it with the input $x^{(m)}$ and determine if it evaluates to 1.

Cooks Theorem: Every PROBLEM $\in$ NP is polynomially reducible to CIRCUIT-SAT

$\left.\begin{array}{c}\text{CIRCUIT-SAT} \\ \text{HAMILTONIAN PATH}\end{array}\right\} \in \boxed{\text{NP- Complete}}$ NPC ≠ NP

Quantum Computation (Preskill ch. 6)

Special Problem: CIRCUIT-SAT $\in$ NP

Input = Circuit w/n gates, m input bits

Problem = is there an m-bit input w/output = 1

$$f(C) = \begin{cases} 1 & \text{if } \exists\, x^{(m)} \text{ so } C(x^{(m)}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

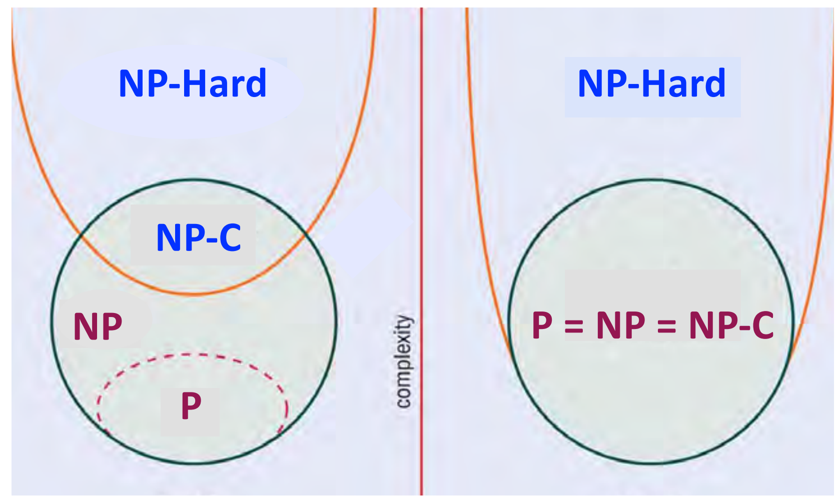
Easy to check solution because if we have the input circuit C we can run it with the input $x^{(m)}$ and determine if it evaluates to 1.

Cooks Theorem: Every PROBLEM $\in$ NP is polynomially reducible to CIRCUIT-SAT

$$\left.\begin{array}{c} \text{CIRCUIT-SAT} \\ \text{HAMILTONIAN PATH} \end{array}\right\} \in \boxed{\text{NP- Complete}} \qquad \text{NPC} \neq \text{NP}$$

Complexity Hierarchy

* **Conjecture:** P $\in$ NP
* $\exists$ Problems in NP that are neither P or NPC
* NPI: Problems of intermediate difficulty
* Conjecture: Factoring $\in$ NPI



P ≠ NP

P = NP

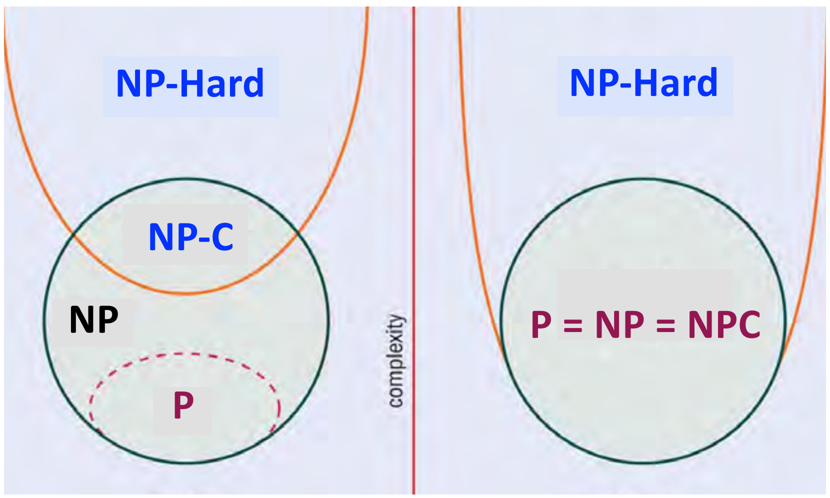
Takeaway Message

* Complexity theory is a rich field with many known complexity classes
* Many foundational conjectures remain unproven
* As we will see, switching to Quantum Circuits changes things

Quantum Computation (Preskill ch. 6)

Complexity Hierarchy

- **Conjecture**: P ∈ NP
- ∃ Problems in NP that are neither P or NPC
- NPI: Problems of intermediate difficulty
- Conjecture: Factoring ∈ NPI



P ≠ NP   P = NP

Takeaway Message

- Complexity theory is a rich field with many known complexity classes
- Many foundational conjectures remain unproven
- As we will see, switching to Quantum Circuits changes things

Aside: Classical Reversible Computation

Motivation:

Quantum Computation = Unitary Transformation

Reversible !

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

Repackage $f: \{0,1\}^n \to \{0,1\}^m$ as reversible

$$\boxed{\begin{array}{l} f: \{0,1\}^{n+m} \to \{0,1\}^{n+m} \\ f(x, 0^{(m)}) = (x; f(x)) \end{array}}$$

we separate $n + m$ qubit register into input and output so no information is lost

Note: Not all 1 & 2-bit gates are reversible, e. g., AND, OR, ERASE

Quantum Computation (Preskill ch. 6)

Aside: Classical Reversible Computation

Motivation:

Quantum Computation = Unitary Transformation

Reversible !

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

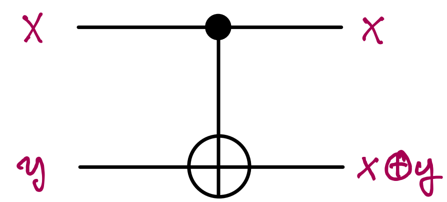
Repackage $f: \{0,1\}^n \to \{0,1\}^m$ as reversible

$$\boxed{\begin{array}{l} f: \{0,1\}^{n+m} \to \{0,1\}^{n+m} \\ f(x, 0^{(m)}) = (x; f(x)) \end{array}}$$

we separate $n + m$ qubit register into input and output so no information is lost

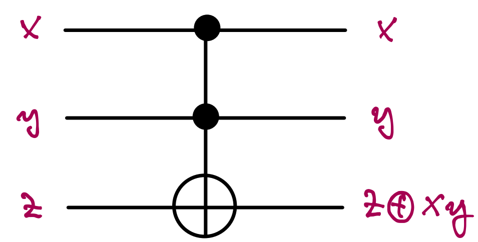
Note: Not all 1 & 2-bit gates are reversible, e. g., AND, OR, ERASE

Example of reversible gate: XOR (CNOT)



Note: One can show that 1 & 2 bit reversible gates are non-universal – they can only do linear maps between input and output.

Toffoli Gate: Example of universal 3-bit gate



Flips z if x = y = 1

* Can show we can build a circuit to compute any reversible function using only Toffoli gates

* **Lesson:** The nature of universal gate sets depends on the nature of the transformations done by the device at hand

Quantum Computation (Preskill ch. 6)

Quantum Circuits

Classical Computer = finite set of gates acting on bits

Quantum Computer = finite set of quantum gates acting on quantum bits

Quantum Computation:

$U: |000\ldots0\rangle \rightarrow |\Psi\rangle$

unitary composed of finite # of gates

n qubit input

output = outcome of Orthog. Measurement in basis $\{|0\rangle, |1\rangle\}^n$

Note:

* The Hilbert space of the Quantum Computer has a preferred decomposition into tensor producs of low dimensional spaces (qubits), respected by gates which act on only a few qubits at a time.

  - This helps establish notion of Quantum Complexity

* Decomposition into subsystems and local manipulations means gates act on qubits in a bounded region.

* It is suspected, but not proven, that the power of Q. C. derives from this decomposition:

$\boxed{n \text{ qubits} \rightarrow 2^n \text{ dimensional } \mathcal{H} \text{ resource grows} \sim 2^n}$

* Unitaries form a continuum, but we restrict to discrete gate sets. This is necessary for Fault Tolerance

* Quantum Gates could be Superoperators, and readout could be POVM's

However:

we can simulate | Superoperators as unitaries / POVM's as Orthog. Meas | in larger $\mathcal{H}$

Our simpler conceptualization is general

Quantum Computation (Preskill ch. 6)

* It is suspected, but not proven, that the power of Q. C. derives from this decomposition:

$\boxed{n \text{ qubits} \rightarrow 2^n \text{ dimensional } \mathcal{H} \text{ resource grows} \sim 2^n}$

* Unitaries form a continuum, but we restrict to discrete gate sets. This is necessary for Fault Tolerance

* Quantum Gates could be Superoperators, and readout could be POVM's

However:

we can simulate | Superoperators as unitaries / POVM's as Orthog. Meas | in larger $\mathcal{H}$

Our simpler conceptualization is general

* Final readout could be collective or in a basis ≠ the standard logical basis →

Unitary maps to standard basis $\{|0\rangle, |1\rangle\}^n$ with overhead included in complexity

* We could do measurements during computation, then condition later steps on the outcomes. But one can show the same results can be achieved by measuring at the end of the computation

- In practice measurement during computation is essential for error correction

Note: None of the above changes notion of complexity

Quantum Computation (Preskill ch. 6)

* Final readout could be collective or in a basis $\neq$ the standard logical basis →

  Unitary maps to standard basis $\{|0\rangle, |1\rangle\}^n$ with overhead included in complexity

* We could do measurements during computation, then condition later steps on the outcomes. But one can show the same results can be achieved by measuring at the end of the computation

  - In practice measurement during computation is essential for error correction

Note: None of the above changes notion of complexity

At this point we are left with 3 main issues

(1) Universality: we must be able to implement the most general unitary $\in SU(2^n)$

group of unitaries in $\mathcal{H}$, $\mathrm{Dim}\,\mathcal{H} = 2^n$

→ Circuit of chosen gates must approx. any $U \in SU(2^n)$

(2) Quantum Complexity:

New class **BQP** [1)]

Decision problems solved w/high prob. by poly-sized quantum circuits

(3) Accuracy: BQP is defined assuming perfect gates. What happens if circuit elements do not have exponential accuracy?

Can show noisy gates are OK:

T - gate circuit requires error prob. $\propto 1/T$

[1)] BQP = Bounded-error Quantum Polynomial time

Quantum Computation (Preskill ch. 6)

At this point we are left with 3 main issues

(1) **Universality**: we must be able to implement the most general unitary $\in SU(2^n)$

group of unitaries in $\mathcal{H}$, $\text{Dim}\,\mathcal{H} = 2^n$

→ Circuit of chosen gates must approx. any $U \in SU(2^n)$

(2) **Quantum Complexity**: $\boxed{\text{New class BQP}}$

Decision problems solved w/high prob. by poly-sized quantum circuits

(3) **Accuracy**: BQP is defined assuming perfect gates. What happens if circuit elements do not have exponential accuracy?

Can show noisy gates are OK: $\boxed{\text{T - gate circuit requires error prob.} \propto 1/T}$

[1] BQP = Bounded-error Quantum Polynomial time

Note on Quantum Complexity:

A QC can simulate a probabilistic classical computer (most general class)

→ $\boxed{BPP^{2)} \subseteq BQP}$

Open Question: Is $BPP \neq BQP$? Seems reasonable, as a prob. C.C. cannot easily simulate QM in a 2^n – dimensional Hilbert space.

If so, a QC will negate the Strong Church-Turing Thesis which holds that any physically reasonable model of computation can be simulated on a probabilistic classical computer with only polynomial slowdown.

[1] BQP = Bounded-error Quantum Polynomial time

[2] BPP = Bounded-error Probabilistic Polynomial time

Quantum Computation (Preskill ch. 6)