Lecture 1 : Introduction to Computation & Complexity

Quantum computing is the flagship quantum technology. Peter Shor's 1994 quantum factoring algorithm kicked off a wave of interest & excitment about q. computing, & is in large part responsible for the field of quantum information going mainstream.

A new wave of excitment has been kicked off by the near-term prospect of NISQ hardware (noisy intermediate-scale quantum computers) – i.e. quantum computing hardware that is not obviously useful for anything ... but no longer obviously useless either!

However, there is also currently a lot of hype and unrealistic expectations.

Aim of this course is to teach you what the excitment derives from, rigorously and in detail; and to innoculate you against the over-hype.

You often (used to?) hear:
"There are only two quantum algorithms"
(meaning Shor & Grover).
That's out of date: there are 3 now!
But that's OK. There are only 4 classical algorithms,* so quantum is not doing badly.

Shor (a.k.a QFT) & Grover (a.k.a. amplitude amplification) are more general algorithmic techniques than specific algorithms. In the same vein, the vast majority of classical algorithms are based on divide-and-conquor, dynamic programming, convex optimisation or one more I'm leaving open as a safety margin.

By the end of the lecture course, you will understand Shor's alg., Grover's alg. & some others in detail. But more importantly, you will hopefully understand better what the exciting potential of quantum computation is & also what are some of its theoretical challenges & limitations.

Classical Computation (in the circuit model)

Def. Logic gate (or "gate")
Boolean function on constant # bits
$G: \{0,1\}^a \longrightarrow \{0,1\}^b$ typically $a, b = 1$ or 2

E.g.

$$AND(x,y) = \begin{cases} 1 & x = y = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$NOT(x) = \begin{cases} 0 & x = 1 \\ 1 & x = 0 \end{cases}$$

$FANOUT(x) = x\,x$ copies input bit to 2 output bits

$XOR(x,y) = x \oplus y$

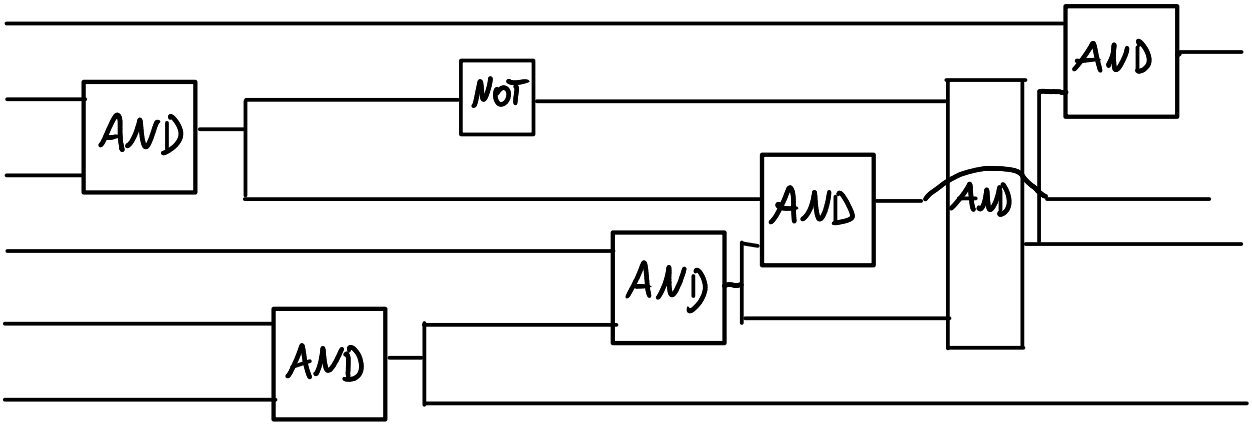
(classical) $CNOT(x,y) = x, x \oplus y$

$TOFFOLI(a,b,c) = a, b, c \oplus (a \wedge b)$

Circuit:

Finite sequence of gates each acting on specified subset of bits.

E.g. (circuit diagram):



(Gates don't have to only act on "adjacent" bits — just easier to draw!)

Theorem

{AND, NOT, FANOUT} is universal.

I.e. any boolean function can be computed by a circuit composed only of these gates.

Exercise: Prove this!

Reversible Computation

A gate is reversible if the input can be recovered from the output

NOT is reversible (self-inverse)

AND is not reversible (00, 01, 10 all give same output 0

For a gate to be reversible, must have same number of input & output bits (obvious!)

Is reversible computation universal? I.e. can any Boolean function be computed by a circuit constructed from reversible gates?

Yes! If we allow ancillas (extra input bits initialised to 0).

If $f : \{0,1\}^n \to \{0,1\}^m$ is not an invertible function, how can we possibly compute it reversibly?

Instead, compute:

$$f_R : \{0,1\}^{n+m} \longrightarrow \{0,1\}^{n+m}$$

$$f(x,y) = x, y \oplus f(x)$$

f_R always invertible (in fact, self-inverse).

$f_R(x,0) = x, f(x)$ $\longrightarrow$ outputs $f(x)$ along with copy of input

We say that f_R "reversibly-computes" f.

Can f_R always be computed by a reversible circuit?

Theorem

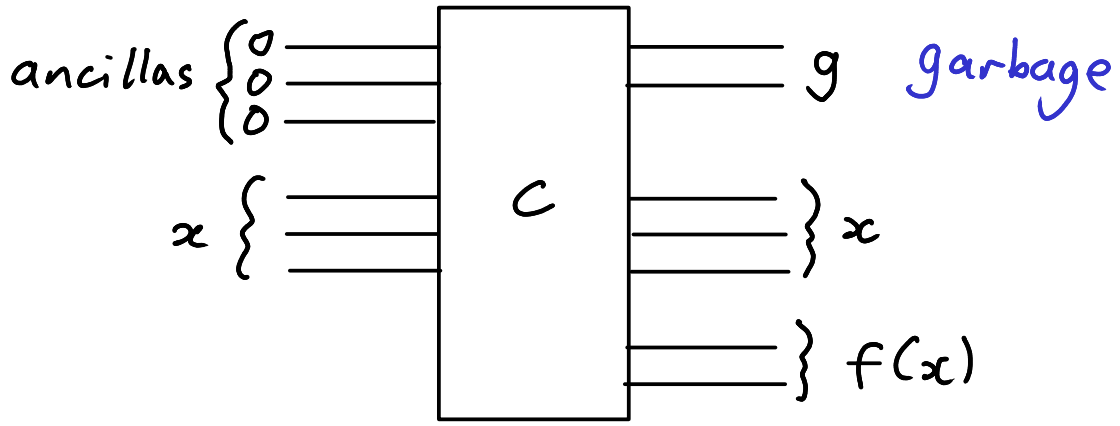
TOFFOLI is universal for reversible computation

Proof

TOFFOLI can reversibly-compute NOT, AND & FANOUT (Exercise!)

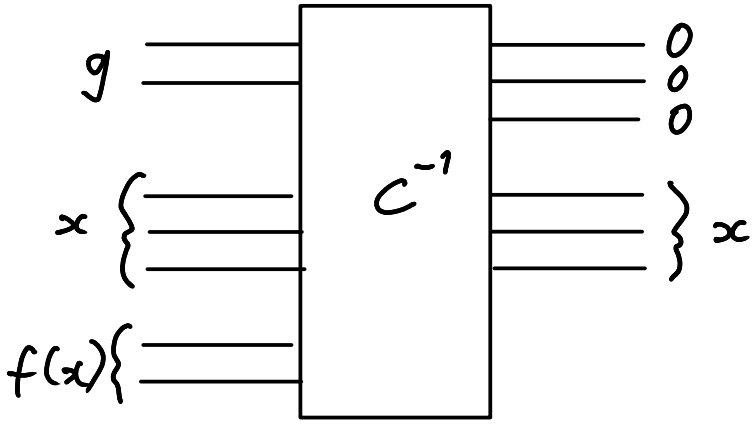
Given (non-reversible) circuit for a Boolean function f, replace all gates by TOFFOLI equivalents.

Doesn't quite achieve what we want, namely $f_R(x, y) = x, y \oplus f(x)$, as it may produce some additional "garbage" bits:

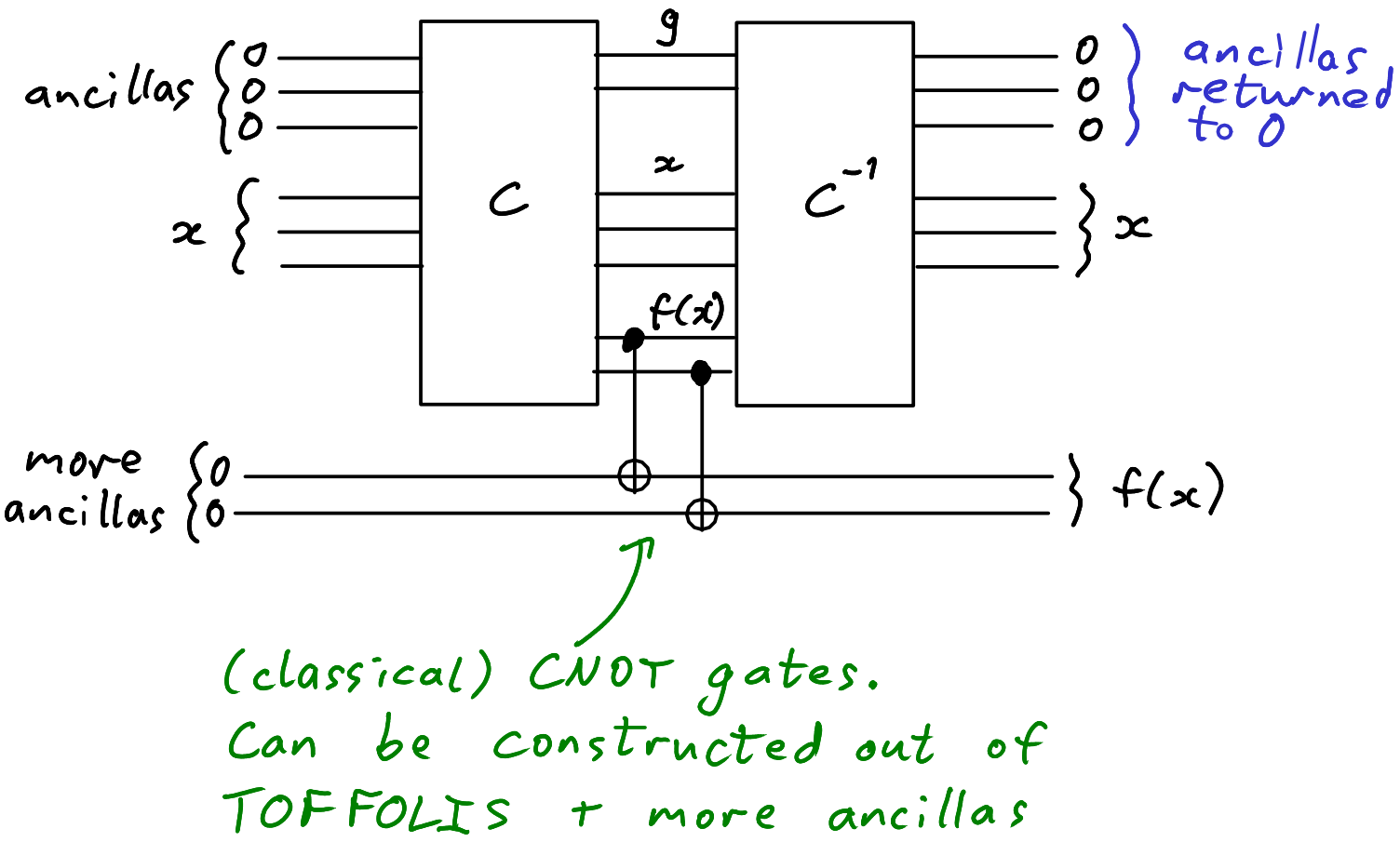


Can fix this by "uncomputing" the garbage:

Since TOFFOLI is self-inverse, can construct C^{-1} just by reversing the circuit:



Then construct

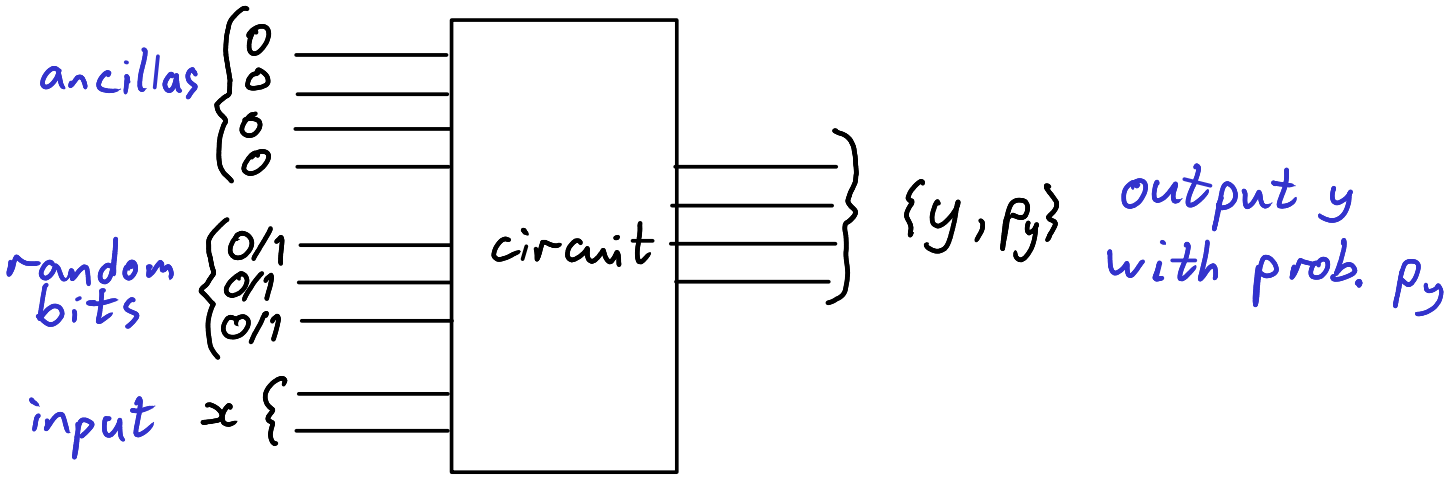


(Classical) Probabilistic computation

Computation can now additionally draw from a source of random bits during computation.

→ Outcome of compution no longer deterministic, but a probability distribution over possible output bit-strings

Can model this as an additional set of ancilla bits set to random values (instead of 0) at input to computation:



Note: if we sample from probability distribution $\{z, p_z\}$ on bits part way through the circuit, & then feed sampled value as input to rest of the circuit, by Bayes' rule we get exactly the same output distribution as before:

$$Pr(y|z)\,Pr(z) = Pr(y) = p_y$$

This is not true in quantum computation!

Quantum computation

Quantum gate

Unitary operator on 1 or 2 qubits

$U_1 : \mathbb{C}^2 \longrightarrow \mathbb{C}^2$

$U_2 : \mathbb{C}^2 \otimes \mathbb{C}^2 \longrightarrow \mathbb{C}^2 \otimes \mathbb{C}^2$

E.g.

$$U_{CNOT} = |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11| \cong \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{pmatrix}$$

$$U_Z = |0\rangle\langle 0| - |1\rangle\langle 1| \cong \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Note: unitarity $\Rightarrow$ reversibility:
$U^\dagger U = \mathbb{1} \Rightarrow U$ has inverse gate $U^\dagger$

Quantum circuit:

Finite sequence of quantum gates each acting on specified subset of qubits

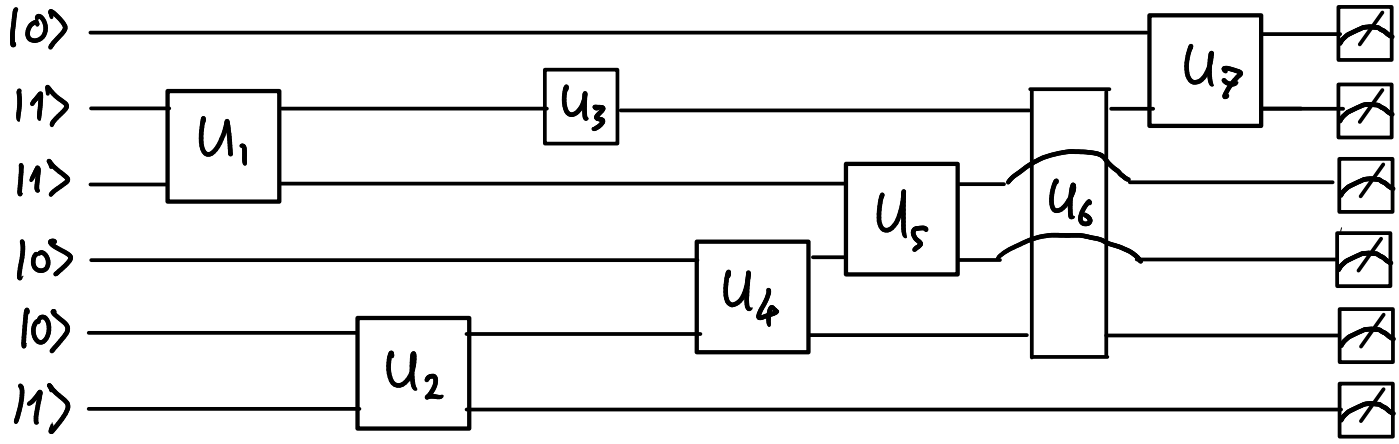
Input $|\psi\rangle$ is "computational basis" state, i.e.
$|\psi\rangle = \bigotimes_i |x_i\rangle$, $|x_i\rangle \in \{|0\rangle, |1\rangle\}$.

Output is outcome of measuring qubits in computational basis, i.e.
measure $\{\Pi^{(0)}, \Pi^{(1)}\}$ on each qubit.

$\longrightarrow$ outcome probabilistic

E.g. (quantum circuit diagram):

comp. basis measurements



Note: if we measure qubits part way through circuit, then feed resulting computational basis state into remainder of circuit, we do <u>not</u> necessarily get same output

(<u>Exercise</u>: construct an example)

This cuts to the heart of the difference between classical probabilistic & quantum computation.

Note: We can restrict to performing all measurements at very end of the circuit without loss of generality.

<u>Exercise</u>: Prove this!
(Hint: simulate computational basis measurement using a CNOT gate. "Coherent measurement")

Are there universal quantum gate sets?

Theorem

$\{U_1, U_{CNOT}\}$ is a universal quantum gate set, where U_1 includes *all* single-qubit unitaries.

I.e. can generate any unitary $SU(2^n)$ on any number of qubits n.

(Unfortunately we don't have time to prove this in lectures :-(See references.)

But this gate set contains an (uncountably) infinite # gates. Are there *finite* universal quantum gate sets?

Problem 1: there are an *uncountably* infinite # unitaries even on a single qubit (continuous families, e.g. e^{-iZt}).

Any finite gate set can only generate a countably infinite # of unitaries.

→ Finite universal gate sets cannot exist.

However, *exactly* generating all unitaries isn't physical anyway. In reality, we can only expect to approximate desired unitary.

Def (Approximately universal)

Gate set $\{U_i\}$ is **approximately universal** if $\forall n, \varepsilon > 0$ & $U \in SU(2^n)$ $\exists$ circuit U_C composed out of $\{U_i\}$ s.t. $\| U - U_C \| \leq \varepsilon$.

Theorem

$\{ H, T, U_{CNOT} \}$ is approximately universal, where

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$ "Hadamard"

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$ "T gate"

(We won't have time to prove this either :-()

Problem 2: Some (actually, almost all!) unitaries $U \in SU(2^n)$ require $\Omega(2^n \log(1/\varepsilon)/\log(n))$ gates to approximate to within ε.

Exercise: Prove this!

Does this mean exponential quantum speedups like Shor's algorithm, let alone polynomial speedups like Grover's algorithm, depend on having exactly the right gate set? → No!

Theorem (Solovay-Kitaev)

Let G_1 be an approximately universal gate set closed under inverses (i.e. $U \in G \Leftrightarrow U^{-1} \in G$).
Let G_2 be another universal gate st.

Then any circuit constructed from t gates in G_2 can be approximated to any desired precision $\varepsilon > 0$ by a circuit with $O(t \cdot \text{poly}(\log \frac{t}{\varepsilon}))$ gates from G_1.
(And there is a poly-time classical algorithm for constructing this circuit.)

The Solovay-Kitaev Thm. is crucial in establishing that the run-time of quantum algorithms is independent (up to polylog factors) of the choice of gate set $\longrightarrow$ complexity of quantum circuits is well-defined; even polynomial speedups are genuine, not an artefact of the choice of gates.

The proof of Solovay-Kitaev is beautiful & non-trivial — please go and read it for your own pleasure and edification!

(Brief) Introduction to Complexity Theory

Complexity theory is the rigorous analysis of the computational resources (space, time, ...) required to solve a problem.

More precisely, it concerns the scaling of the resources as a function of the size of the problem.

Computational Problems

Problem size = amount of information to specify problem

E.g:

Factoring problem

Input: $n \in \mathbb{N}$

Output: factors of n

Problem size $\sim$ # of digits in n $\sim \log(n)$

SAT

Input: boolean expression using vars $\{b_i\}$, operations $\wedge$, $\vee$, $\neg$

Output: $\begin{cases} \text{YES} & \text{if } \exists \text{ values } b_i \in \{0,1\} \text{ s.t. expression evaluates to 1} \\ \text{NO} & \text{otherwise} \end{cases}$

Problem size $\sim$ # boolean variables

Often interested in decision problems: problems with binary output YES or NO (1 or 0).

Def (Decision Problem)

$f : \{0,1\}^* \longrightarrow \{0,1\}$

↑ input written as bit-string

↑ binary output

(In literature, often see terminology
"Language" $L = \{s \in \{0,1\}^* : f(s) = 1\}$
"Deciding the language" = solving decision problem)

Computational problems often have natural equivalent decision variants,

↖ will make this rigorous later!

E.g:

Factoring (decision variant)

Input: $n, k \in \mathbb{N}$

Output: YES if n has factor $< k$
NO otherwise

Exercise

Show that if you can solve factoring decision problem, you can also find factors.

Complexity classes

Classify computational problems according to computational resources required to solve them.

Def. P (polynomial-time)
Decision problems solvable with polynomial-sized (classical) circuit.

(Problem size n
$\longrightarrow$ # gates in circuit $= O(\text{poly}(n))$.)

Examples: addition, multiplication, computing eigenvalues, primality testing (2000)

ignore name!

Def. NP (non-deterministic poly-time)
Decision problems:
if answer for problem input x is:

YES:
$\exists$ polynomial-sized "witness" $w \in \{0,1\}^{\text{poly}(n)}$
& polynomial-sized "verifier" circuit C
s.t. $C(x,w) = 1$

NO:
$\forall$ witnesses w, $C(x,w) = 0$.

Think of NP as game between all-powerful (but untrustworthy) Merlin, and less powerful Arthur who can only run polynomial-time computations.

Arthur asks Merlin question x.
Merlin replies with answer YES/NO.
But Arthur doesn't trust Merlin, so Merlin also gives Arthur a simple (poly-time checkable) proof w that his answer is correct.

NP is class of all problems for which Merlin can <u>convince</u> Arthur of a YES answer ($\exists w$), and <u>cannot trick</u> him into believing a wrong NO answer ($\forall w$).

(Note asymmetry between YES & NO.
Class co-NP is very similar, but with roles of YES & NO swapped.)

≈ "probabilistic P"

Def. BPP (bounded-error poly-time)

Decision problems:

∃ poly-sized probabilistic circuit C s.t.

$\Pr(C \text{ outputs } 1)$ $\begin{cases} \geq 2/3 & \text{YES instance} \\ \leq 1/3 & \text{NO instance} \end{cases}$

Note: Probs. $2/3$, $1/3$ arbitrary. Could choose $1-\varepsilon$, ε for any $\frac{1}{2^n} \leq \varepsilon \leq \frac{1}{2} - \frac{1}{\text{poly}(n)}$.

Exercise

Prove class BPP is independent of choice of ε as above.

≈ "quantum P"

Def. BQP (bounded-error q. poly-time)

Decision problems:

∃ poly-sized quantum circuit U s.t.

$\Pr(U \text{ outputs } "1")$ $\begin{cases} \geq 2/3 & \text{YES instance} \\ \leq 1/3 & \text{NO instance} \end{cases}$

(Recall

$\Pr(U \text{ outputs } "1")$ input $|x\rangle \to$ density matrix $|x\rangle\langle x|$

$= \text{tr}[\Pi_1^{(1)} \otimes \mathbb{1}_{2\ldots n}\, U |x\rangle\langle x| U^\dagger]$

$= \langle x| U^\dagger \Pi_1^{(1)} \otimes \mathbb{1}_{2\ldots n}\, U |x\rangle$)

↖ wlog output is given by 1st qubit

Def. MA (Merlin-Arthur)

Decision problems.

$\exists$ poly-sized probabilistic verifier circuit C s.t. if answer on input x is:

YES:

$\exists$ poly-sized witness w s.t.

$$\Pr(C(x, w) = 1) \geq \frac{2}{3}$$

NO:

$\forall$ input strings w, $\Pr(C(x,w)=1) \leq \frac{1}{3}$.

Note: again, probs. $\frac{2}{3}$, $\frac{1}{3}$ arbitrary.

I.e. probability of correctly accepting or rejecting the witness is $\geq \frac{2}{3}$.

Def. QMA (Quantum Merlin-Arthur)

Decision problems.

$\exists$ poly-sized quantum verifier circuit U s.t. if answer on (classical) input x is:

YES:

$\exists$ poly-sized quantum witness $|w\rangle \in \mathbb{C}^{poly(n)}$ s.t.

$Pr(U \text{ outputs "1" on input } |x\rangle|w\rangle) \geq \frac{2}{3}$

NO:

$\forall$ states $|w\rangle$,

$Pr(U \text{ outputs "1" on input } |x\rangle|w\rangle) \leq \frac{1}{3}$

Note: again, probs. $\frac{2}{3}$, $\frac{1}{3}$ arbitrary.

Exercise (hard!): Prove this.

This course (and applications of quantum computing in general!) is largely concerned with BPP, BQP & the diffences between them.

Reduction

Lets us compare <u>difficulty</u> of different computational problems.

⟶ rigorously say problem B is harder than problem A.

// "Karp reduction"

<u>Def</u> (poly-time reduction)

A <u>reduces</u> to B if

$\exists$ map $A \longmapsto B$

instance$_a$ ⟶ instance$_b$

s.t. b has answer YES iff a has answer YES.

<u>and</u> map $A \to B$ can be computed by poly-size circuit.

I.e. can solve A by transforming it into B & solving B.

Write "$A \leq B$".

$A \leq B$ & $B \leq A$ $\Rightarrow$ A, B <u>equivalent</u>

⟶ write "$A = B$".

Def. (NP-hard)

Problem B is <u>NP-hard</u> if
$\forall A \in NP \quad A \leq B$.

Def. (NP-complete)

Problem A is NP-complete if
$A \in NP\text{-hard}$ & $A \in NP$.

NP-complete problems are "hardest problems in NP": if you can solve one, you can solve <u>all</u> NP problems

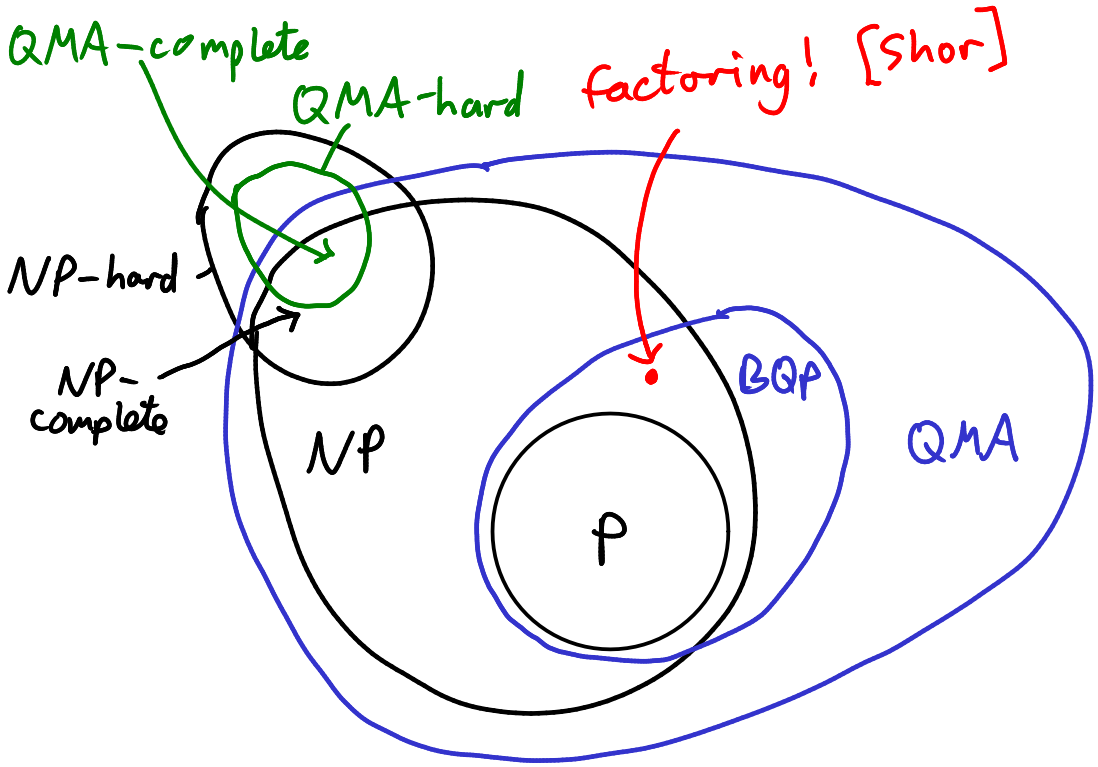
Define QMA-hard & QMA-complete analogously.

Complexity Zoo

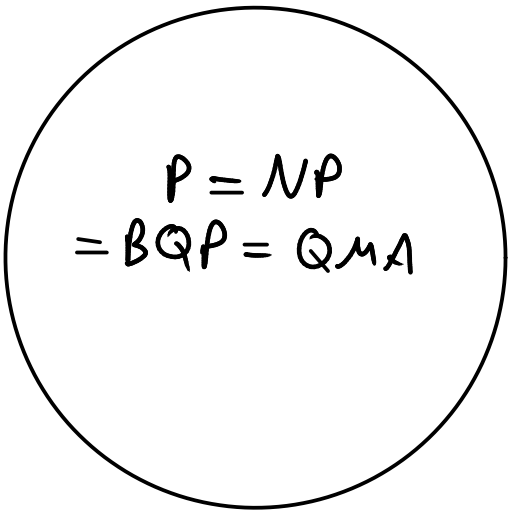
Relationship between P, NP, BQP, QMA is an open problem.

- $P \subseteq BQP$ (classical = special case of quantum)
  $NP \subseteq QMA$
- $P \stackrel{?}{=} NP$ infamous! $1,000,000
- $P \stackrel{?}{=} BQP$ "are quantum computers useful?"
- $BQP \stackrel{?}{=} QMA$ quantum "P vs. NP"
- $NP \stackrel{?}{=} QMA$ unlikely

General belief...



... but could be



!