Lecture 24 - Quantum Complexity

[[P, NP, and all that - where does quantum fit in?]]

def: BQP is the complexity class of all "decision problems" φ such that:

$\forall$ input lengths n, $\exists$ quantum circuit C_n

s.t.: (a) C_n has n input bits (+ancillas)

(b) C_n has 1 output bit (measurement on first qubit)

(c) C_n has $\leq$ poly(n) gates

(d) [[something about allowed gates]]

(e) [[something about "uniformity"]]

and, C_n "solves" φ in that:

(i) $\forall x \in \{0,1\}^n$ with $\varphi(x) =$ YES,

$\Pr[C_n(x) \text{ outputs } 1] \geq \frac{3}{4}$

(ii) $\forall x \in \{0,1\}^n$ with $\varphi(x) =$ NO,

$\Pr[C_n(x) \text{ outputs } 1] \leq \frac{1}{4}$.

Commentary

- Apologies about the name "BQP".
  "B" stands for "Bounded error"; refers to "2-sided error" where 3/4, 1/4 are "bounded away from $\frac{1}{2}$"; as we've seen, any 2 numbers here, like .51 vs. .49, or .99 vs. .01, are equally OK, as you can "amplify success probability". OTOH, just $\geq \frac{1}{2}$ vs. $< \frac{1}{2}$ would not be cool, as you could not realistically tell the difference between probability $\frac{1}{2} \pm 2^{-n}$, say, without exponentially many repetitions.

  "Q" is for "Quantum", of course

  "P" is for polynomial time. We never discussed quantum Turing machines/algs. because they're annoying, but in general there's a close connection between algorithmic time (# of steps) and circuits' # of gates.

"Decision problems" = problems with a yes/no answer. It's technically convenient to focus on these, and basically "without loss of generality"... but annoyingly, the most famous thing you can do with a quantum computer — factoring integers — is not a decision problem. So we formally cannot say "Factoring $\in$ BQP" but instead have to invent an artificial decision variant... ||

Factoring-Decision:

Input: (x, h) integers

Output: yes/no, does x have a prime factor $\leq h$?

[On one hand, given an efficient factoring alg., you can easily use it to solve Factoring-Decision. OTOH, if you can efficiently solve Factoring-Decision, you can actually factor x without too much more effort by binary-searching for x's factors via varying h

— exercise.

]||

[Regarding (d), the "allowed gates" issue. So far we "allowed" any 1, 2, or 3-qubit unitary gate. OK, you shouldn't allow unitary matrices with uncomputable entries or such craziness. We didn't discuss this much, but it turns out even if you're very stingy, it doesn't make much difference. There are theorems saying that even if you just allow ...

H, NOT, CNOT, CCNOT

... gates, you do not fundamentally lessen the power of quantum computation, except by constant factor slowdown. If you want the ability to approximate all unitaries, you'll need: (a) phase gate (gotta get complex #'s in somehow; (b) some rotation gate with a smaller angle, like $\pi/8$. But don't actually need these for full-fledged quantum comp. So let's stick to H, NOT, CNOT, CCNOT for simplicity of BQP's definition.]

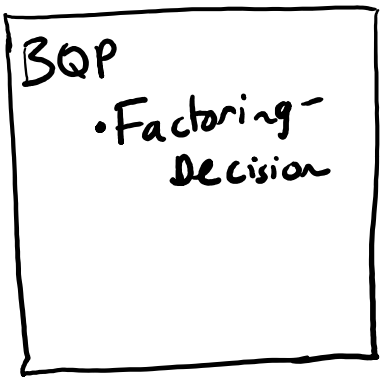
(c) "Uniformity" — a technical condition arising because we really want our model of quantum computation to be circuits. This technicality has nothing to do with quantum! it also comes up in classical complexity theory when using circuits as a model of computation...

Note that our definition merely says that for, say, $n = 1000$, the appropriate circuit C_{1000} exists. But if we don't easily know explicitly what it is, its mere existence is utterly useless, practically. So we add the following technical condition, which is essentially always satisfied in practice:

$\exists$ efficient [poly(n)-time] classical alg. (e.g., Turing machine) B that, on input "n", outputs [for Builder] the circuit diagram for C_n.

[Great, so we've finally defined....]

[Contains Factoring(-Decision), some more odds and ends----]



[Let's look inside BQP now.... As we know, Q.C. includes classical probabilistic comp.]

def: BPP : Same definition, but C_n is a classical AND/OR/NOT circuit with coin-flip gates.

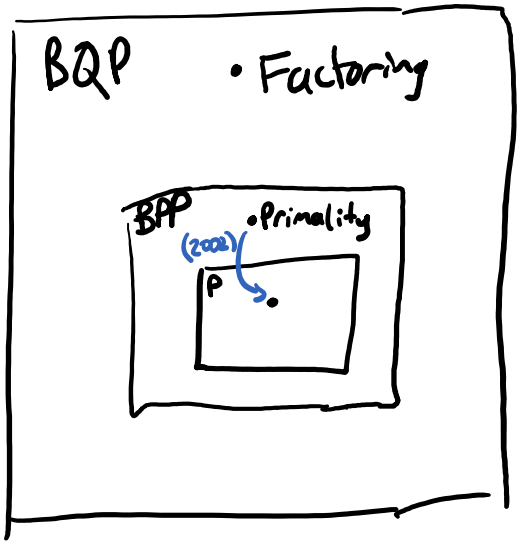
[Equivalently, forget circuits: it's just classical bounded 2-sided error randomized computation.]

[Our standard notion of "efficiently solvable" decision problems.]

def: P : same, except no coin-flips

rec: . P = BPP is generally believed.

. Factoring(-Dec) $\notin$ BPP widely believed



[What of the famous complexity class "NP"?]
[Here is a brief, somewhat nonstandard way to define it....]

def: NP = all decision probs φ s.t. there's a randomized poly-time alg. A s.t.

(i) $\forall x$ with $\varphi(x) = YES$, $Pr[A(x)=1] > 0$

(ii) $\forall x$ with $\varphi(x) = NO$, $Pr[A(x)=0] = 0$.

[Note: the >0 vs. $=0$ distinction not big enough to be exploited in practice. So why do we study NP & other "non-realistic" complexity classes? They help us understand problems & realistic classes we do care about!]

def: The (Circuit-)Sat decision problem:

Input: a classical AND/OR/NOT circuit C

[Yes, it's a "meta" problem: a computational task about computational algs....]

Output: Yes/no: $\exists ?\, x$ s.t. $C(x)=1$ [is it "satisfiable"?]

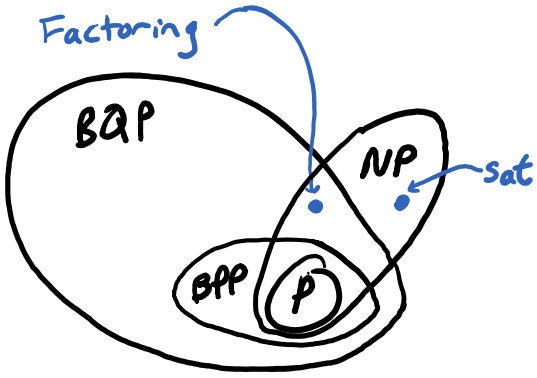
fact: Sat $\in$ NP because of:

> A("C"): · pick x at random
> · output C(x)

Poly-time? ✓ [About $\sim m$ time if C has m gates.]

(i), (ii) ✓

- Factoring (-Dec.) $\in$ NP
  (exercise – guess & check factorization of x)



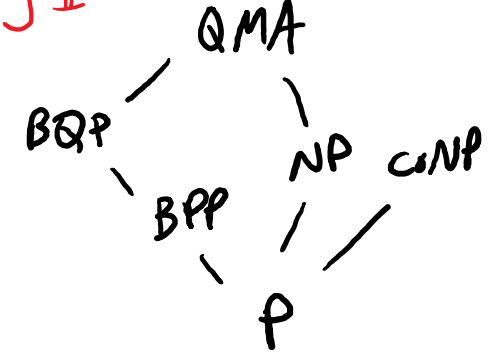
- "Sat $\notin$ BQP" is generally believed
  (tho. Grover's alg. solves Sat in $\approx \sqrt{2}^n$ time)
- "$\exists$ $\varphi \in$ BQP but not $\in$ NP" generally believed
  $\supset$ & not even in "coNP" [[we'll define in a sec.]]
  or even in "PH"
  [[if you know what the Polynomial Hierarchy is]]
  [[See [Raz-Tal '18] for evidence]]

NP variant #1: "coNP": same, except

(i) $\varphi(x) = YES \Rightarrow Pr[A(x)=1] = 1$
(ii) $\varphi(x) = NO \Rightarrow Pr[A(x)=1] < 1$

[[exercise: also contains Factoring]]

#2
"QMA": "a quantum version of NP".



[Another odd variant... An early "wrong" attempt at defining BPP]

def: PP = all decision probs φ s.t. $\exists$ poly-time randomized alg. A s.t.

(i) $\varphi(x) = YES \Rightarrow Pr[A(x)=1] > \frac{1}{2}$

(ii) $\varphi(x) = NO \Rightarrow Pr[A(x)=1] \leq 1/2$

[Again, not useful in practice as it could be $\frac{1}{2} \pm 2^{-n}$!]

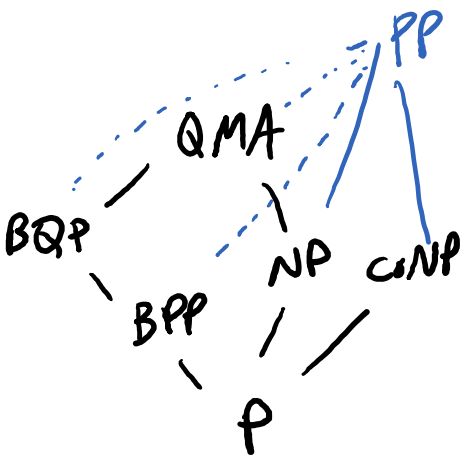
- Clearly $BPP \subseteq PP$
- Also $NP \subseteq PP$:

Say A has $\varphi(x)=YES \Rightarrow Pr[A(x)=1] > 0$

$\varphi(x)=NO \Rightarrow Pr[A(x)=1] = 0$.

Let $A'(x) =$ { Flip coin. If Heads, output 1. Else do $A(x)$.

$\rightarrow$ satisfies (i), (ii) above. $\Rightarrow \varphi \in PP$.



[Also not hard to show coNP $\subseteq$ PP. Exercise.]

Today: BQP $\subseteq$ PP.

[Adleman-Demarrais-Huang '97]

[In fact, once you know this proof, and the def. of QMA, even QMA $\subseteq$ PP is not hard.]

[[So let's get to "upper-bounding" BQP. We'll start with the most basic question of all — surely not all dec. probs. are in BQP. Surely the Halting Problem is not in BQP!]]

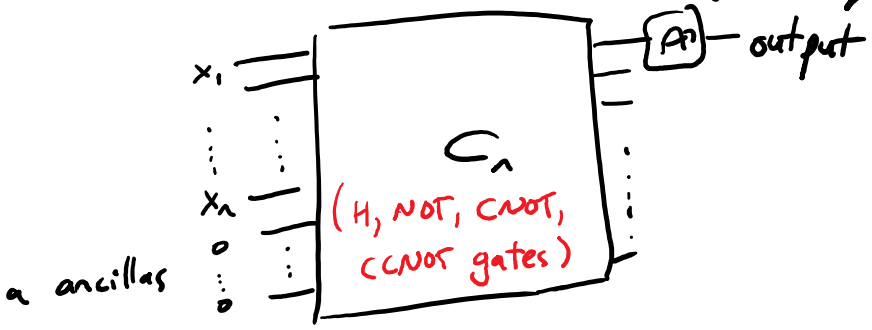
Thm: Suppose $\varphi \in BQP$. Then there is a classical (deterministic) alg. solving φ.

[[In some finite time. In fact, in at most exponential time, as we'll see.]]

proof sketch: Given input x....

Step 0: Compute n, # bits of x.

Run $B(n)$ to get quantum circuit C_n.



Step 1: Compute quantum state, $\sum_{y \in \{0,1\}^{n+a}} \alpha_y |y\rangle$, after each gate.

(You literally implemented this in an early homework!)

$\Pr[C_n(x)=1] = \sum_{y : y_1 = 1} \alpha_y^2$ at end is exactly computable.

Memory: $\approx 2^{n+a}$ #'s

Time: $\approx m \cdot 2^{n+a}$ steps. □

$\therefore$ BQP $\subseteq$ "EXP" ‖ problems decidable in at most exponential time ‖

‖We'll now see a better algorithm: you can get the memory/space usage down to polynomial, though the time is still exponential. ‖

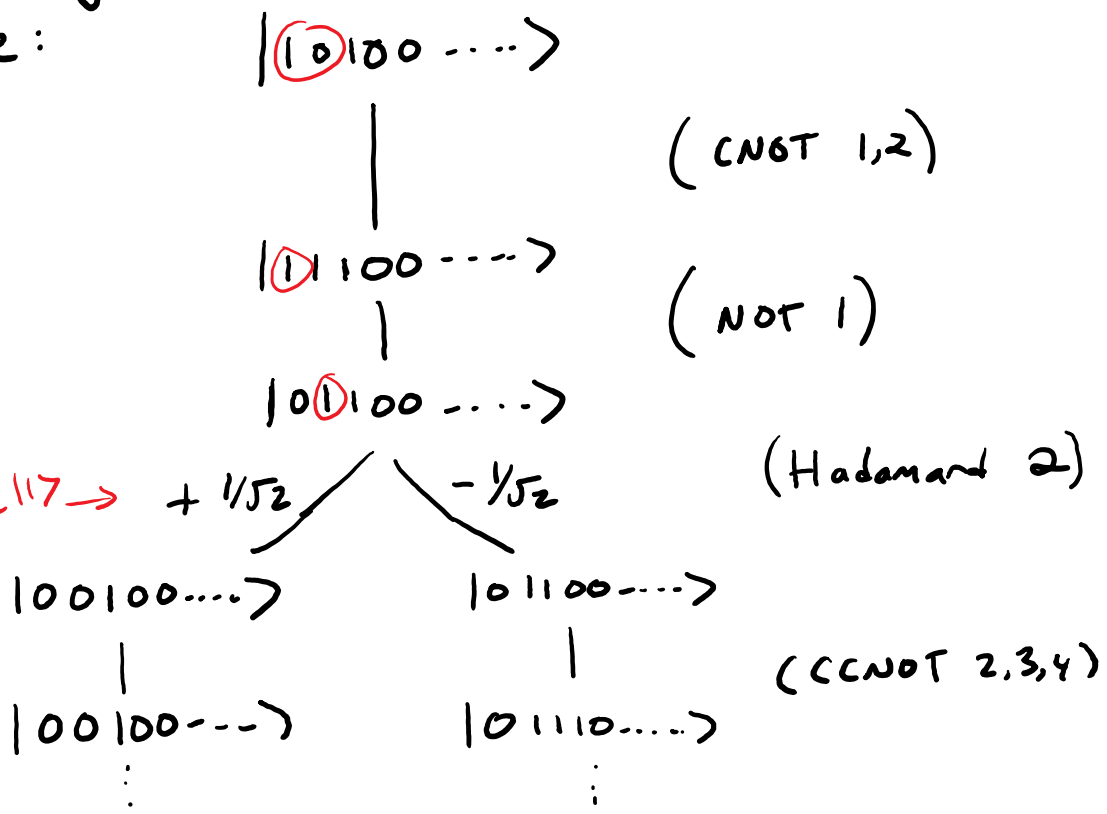
thm: BQP $\subseteq$ "PSPACE"

proof: Given x, get C_n as before.

Say, e.g. $x = 10100\cdots$,

C_n = "· CNOT qubits 1&2
· NOT qubit 1
· Hadamard qubit 2
· CCNOT 2,3,4 ----"

Algorithm imagines ‖but does not explicitly write down‖ this tree:



[[For each NOT, CNOT, or CCNOT gate, each node only has 1 child, and should actually label the edge by the amplitude "+1".]]

Say C_n has m gates, of which h are Hadamards.

[[For each Hadamard, nodes will have fanout 2, with edge labels either $+1/\sqrt{2}, +1/\sqrt{2}$, if H applied to a $|0\rangle$, or $+1/\sqrt{2}, -1/\sqrt{2}$ if H applied to a $|1\rangle$.]]

Tree has 2^h leaves/paths.

$$\text{Final state} = \sum_{\text{paths } p} |\text{leaflabel}(p)\rangle \cdot \underbrace{\prod(\text{amplitudes along } p)}_{= \left(\frac{1}{\sqrt{2}}\right)^h \cdot \underbrace{\prod(\text{signs along } p)}_{\text{"sign}(p)\text{"}}}$$

$$\therefore \text{Final amplitude on } |y\rangle = \left(\frac{1}{\sqrt{2}}\right)^h \cdot \sum_{p:\ \text{leaflabel}(p) = y} \text{sign}(p)$$

$$\therefore \text{Final ampl.}^2 \text{ on } |y\rangle = 2^{-h} \cdot \sum_{p, p':\ \text{label}(p) = \text{label}(p') = y} \text{sign}(p)\,\text{sign}(p')$$

$$\therefore \text{final ampl.}^2 \text{ on } |y\rangle = 2^{-h} \cdot \sum_{p,p':\ \text{label}(p)=\text{label}(p')=y} \text{sign}(p)\,\text{sign}(p')$$

$$\Rightarrow \Pr[C_n(x)=1] = 2^{-h} \sum_{p,p':\ \text{label}(p)=\text{label}(p') \text{ \& first bit is } 1} \text{sign}(p)\,\text{sign}(p')$$

$$\Rightarrow \Pr[C_n(x)=1] - \Pr[C_n(x)=0] = 2^{-h} \sum_{p,p':\ \text{label}(p)=\text{label}(p')} \text{sign}(p)\,\text{sign}(p')(-1)^{\overline{\text{label}(p)}_1} \quad (*)$$

Key: $\varphi(x)=\text{YES} \Rightarrow (*) \geq \frac{3}{4} - \frac{1}{4} = 1/2$

$\varphi(x)=\text{NO} \Rightarrow (*) \leq \frac{1}{4} - \frac{3}{4} = -1/2.$

And in fact, $(*)$ easy to compute exactly in poly(n) space, exp(n) time:

[Enumerate all pairs of paths p, p', reusing space. For each, compute $\text{sign}(p)\,\text{sign}(p')(-1)^{\overline{\text{label}(p)}_1}$, keep running total. Decide if $\geq 2^{h-1}$ or $\leq -2^{h-1}$!]

Cor: $BQP \subseteq PP$

proof: · On input x, obtain C_n, & "imagine" the tree.

· Randomly, indep'ly, choose 2 paths p, p'.

· Compute label(p), label(p').

· If unequal, output "+1" with prob. 1/2,
output "−1" " " 1/2.

· If equal, compute and output
$\text{sign}(p)\,\text{sign}(p')(-1)^{\overline{\text{label}(p)}}$, $\in \{\pm 1\}$.

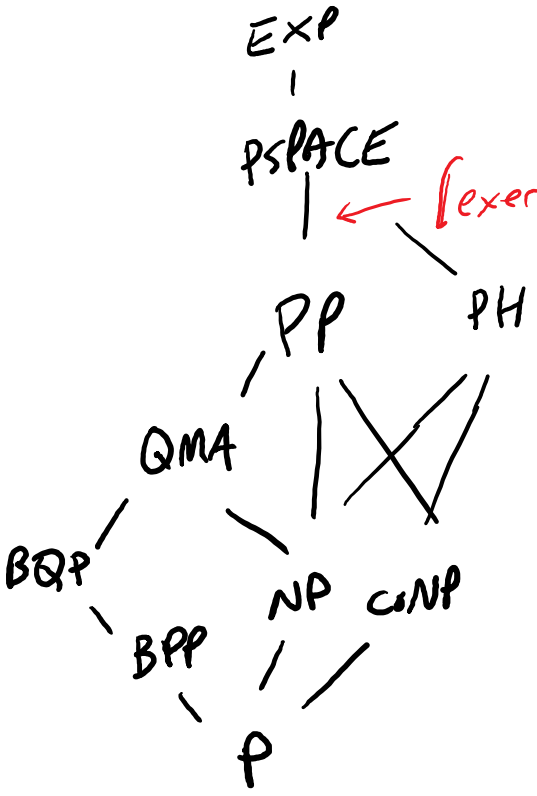
Say this has probability q.

[Don't really know what q is, but it's some nonzero #.]

$$E[\text{alg output}] = (1-q)\cdot\underbrace{\left\{\tfrac{1}{2}(+1)+\tfrac{1}{2}(-1)\right\}}_{0} + q\cdot(\text{something proportional to } \circledast)$$

$\Pr[\text{alg. outputs } +1] - \Pr[\text{outputs } -1]$ > 0 if and only if $\circledast > 0$.

$\therefore \Pr[\text{alg outputs } +1] > \frac{1}{2}$ iff $\circledast > 0$ iff $\varphi(x) = \text{yes}$. ☺ ■



Final lecture:

"Quantum supremacy"

[Will involve
- real-world Q.C.
- PP & time travel... !!]