LECTURE 8 (February 12)

TODAY BQP vs PH

First, let us introduce the motivations behind this question and the complexity class PH which stands for the polynomial hierarchy and contains P, NP, coNP

We have seen some evidence in the form of oracle separations that BQP cannot solve NP-complete problems

So, if not NP-complete problems, what other practical problems might be candidates for quantum advantage ?

① NP-intermediate problems ? this includes factoring, Graph isomorphism, Lattice problems but we don't know too many NP-intermediate problems

② Something outside of NP e.g in PH ?

This is one of the first motivations for studying the BQP vs PH question

The other is related to getting better evidence that BQP cannot solve NP complete problems, for instance,

If $NP \subseteq BQP$, then a widely believed conjecture about the "collapse" of PH is false ← Note that there are no oracles in this statement

This also requires us to first understand the BQP vs PH problem

Another motivation is related to the question :
Can quantum computing survive $P = NP$?
i.e. even if $P = NP$, does $BQP \neq P$?
The answer is NO, if $BQP \subseteq PH$

So, we must seek some evidence that $BQP \not\subseteq PH$

In the next couple of lectures, we are going to see some heuristic evidence for this in the form of oracle separation : $\exists\, O$ s.t. $BQP^O \not\subseteq PH^O$.

This is one of the major results in the last five years proved by Raz & Tal and was a open problem for 30 years

What is the polynomial hierarchy?

Let us first recall P = languages decided by a poly-TM M

$$x \in L \Leftrightarrow \exists \text{ poly-TM s.t. } M(x) = 1$$

NP = languages where there is an efficient certificate that poly-TM accepts

$$x \in L \Leftrightarrow \exists\, w \in \{0,1\}^{poly(|x|)} \text{ s.t. } M(x,w) = 1$$

SAT is NP-complete where w = satisfying assignment

Now consider the following problem:

$\Sigma_2 SAT =$ " Given a boolean formula $\varphi(x_1, \ldots, x_n, y_1, \ldots, y_n)$

$\forall x \; \exists y$ s.t. $\varphi(x,y) = 1$ "

This is a very natural problem but it is not obvious if this in NP but if we define the following complexity class

Σ_2^P = languages s.t.

$$x \in L \Leftrightarrow \forall w_1 \; \exists u_1 \text{ s.t. } M(x, w_1, u_1) = 1$$

Then, $\Sigma_2 SAT \in \Sigma_2^P$ and one can also show that it is a complete problem for this class.

Σ_2^P is called the second level of the polynomial hierarchy

In general, one can define Σ_3^P = languages s.t.
$x \in L \Leftrightarrow \exists u_2 \; \forall w_1 \; \exists u_1$ s.t. $M(x, w_1, u_1) = 1$

with complete problem $\Sigma_3 SAT$ and so on for higher levels.

Polynomial Hierarchy is defined as

$$PH = \bigcup_{i=0}^{\infty} \Sigma_i^P \qquad \text{where } \Sigma_0^P = P, \; \Sigma_1^P = NP$$

It is believed that each level of the hierarchy is distinct
P = NP means the hierarchy collapses to the zeroth level
so, a weaker form of the $P \neq NP$ conjecture is that $\Sigma_i^P \neq \Sigma_{i+1}^P$ for some finite i

Another equivalent definition of PH is in terms of oracles

$$\Sigma_2^p = NP^{NP} \qquad \Sigma_3^p = NP^{NP^{NP}} \quad \text{and so on}$$

There are more equivalent definitions [consult the Arora-Barak textbook]

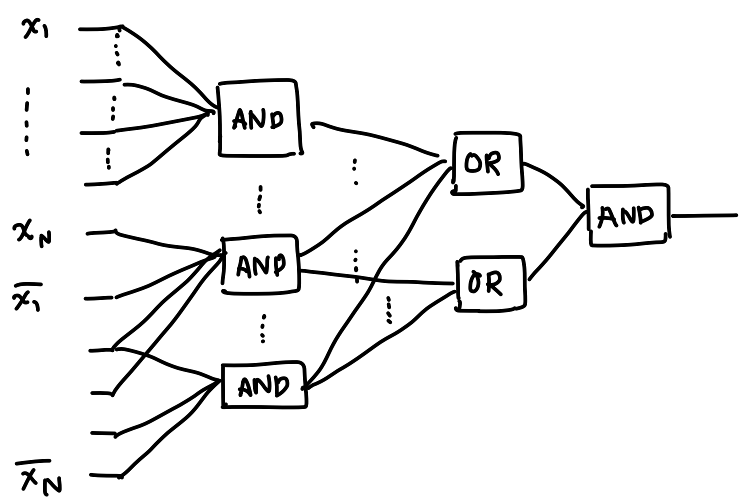
Our goal is to show that $\exists$ oracle O s.t. $BQP^O \not\subseteq PH^O$

This involves showing

(1) $\exists$ a polynomial query quantum algorithm that can query O and solve the problem

(2) No PH-machine with query access to O can solve the problem

To show (2), it suffices to study AC^0 circuits — these are constant depth circuits where the bottom layer is input or its negation, and every other alternating layer consists of AND or OR gate with unbounded fan-in



Connection between PH-oracle machines and AC^0-circuits

Consider the language $L^O = \{1^n \mid O \text{ on inputs of length } n \text{ has some property}\}$

Let M^O be a PH-oracle machine with k quantifiers making queries to oracle O on inputs of length $\leq p(n) = \text{poly}(n)$ in time $p(n) \cdot q(n) = \text{poly}(n)$

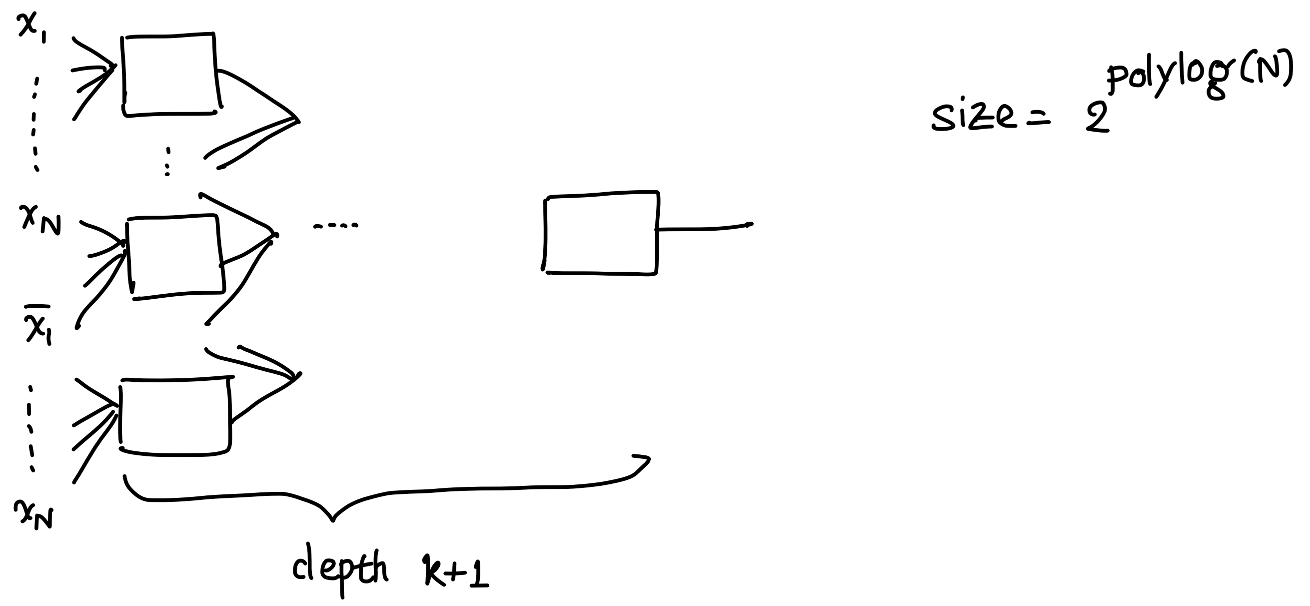
Then, $\exists$ an AC^0-circuit family with input $N = 2^{p(n)}$ bits, with depth $k+1$, size $2^{\text{polylog}(N)}$ that outputs the same answer

In particular, suppose we view M as reading bits of the truth table of O which has size $2^{p(n)} = N$

M asks for a bit $i \in [N]$ by giving its binary description and Oracle gives $O(i)$

Let us denote $x_1, \ldots, x_N = O(1), \ldots, O(N)$ to be the truth table of O

Then, the AC^0- circuit looks like



Why should this be the case ?

To see this, we consider Σ_1^P- oracle machines, aka, NP-oracle machines

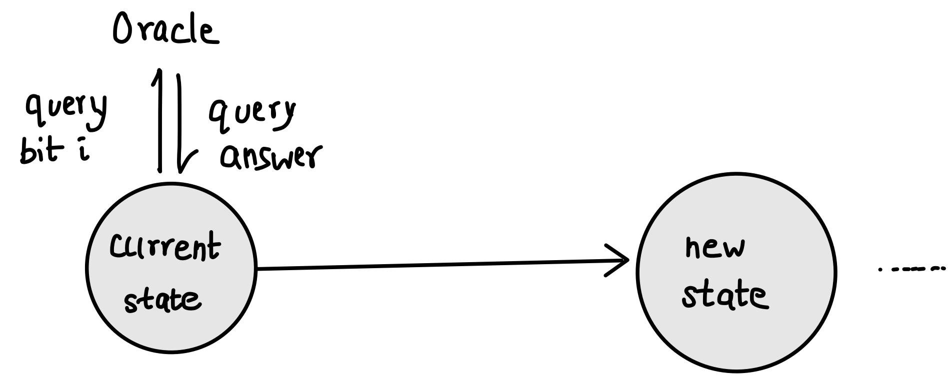
These use a single $\exists$ quantifier and give rise to a depth- 2 AC^0-circuit

This will Introduce the key idea. In the general case – when there are k quantifiers one can easily extend the ideas here to get a depth $k+1$ circuit.

What does an NP^O machine do on input 1^n ?

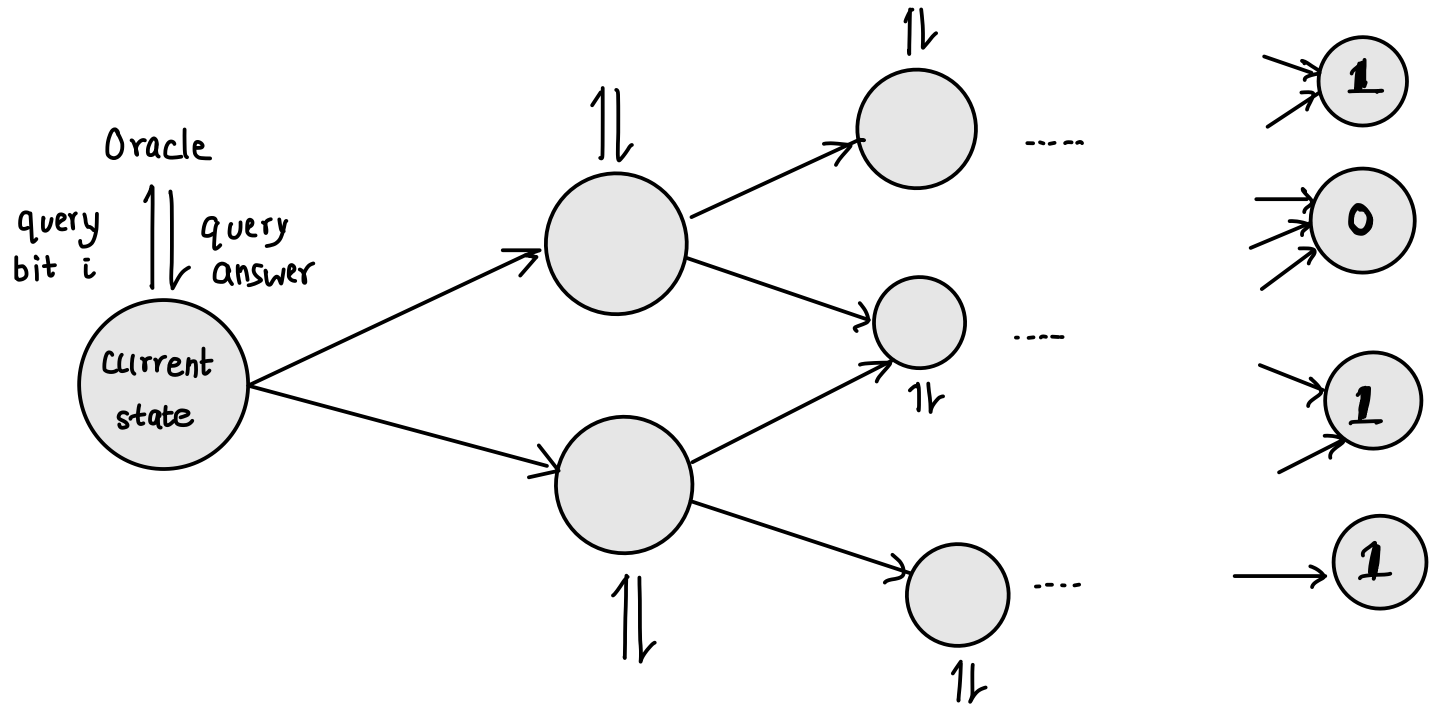
We have mostly use the characterization of NP- machines in terms of certificates but here we will need to use the non-determinism characterization

In particular, a deterministic Turing machine or algorithm on input 1^n queries the oracle and depending on the answer chooses its next step deterministically



A non-deterministic algorithm can choose among several different "next steps"

This leads to many paths in the underlying state space graph that end up in 0 or 1 (the answer)



If input 1^n to the machine is in the language, then the guarantee is that there is a path that ends in the answer 1

The path is of $p(n) \cdot q(n) = \text{poly}(n)$ length and serves as a witness / certificate

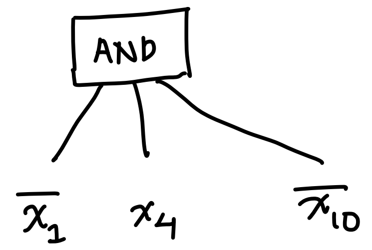
If input 1^n is not in the language, then all paths end in 0

Now to convert this into depth-2 AC^0 circuit

Let us take any path in the "state space" graph

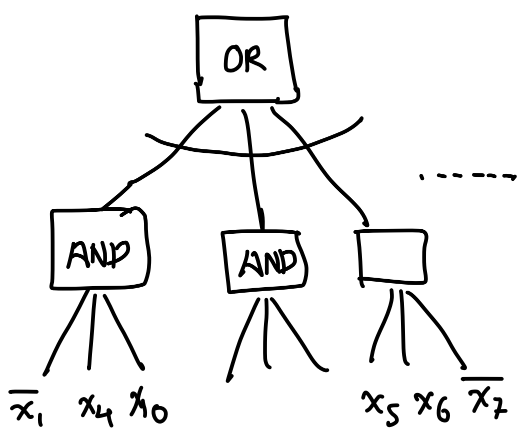
If the path queries bits say 1, 4, 10 and recieved answers $x_1 = 0$, $x_4 = 1$, $x_{10} = 0$

then we add a AND gate as follows



This gate outputs 1 iff $x_1 = 0$, $x_4 = 1$, $x_{10} = 0$

We add such AND gates for each accepting path and then add a single OR gate in the top layer connected to all the AND gates



← The OR gate checks if $\exists$ path that outputs 1

← Each AND gate corresponds to checking if all queries along a path are consistent with what the oracle says.

All AND gates together remove all accepting paths not consistent with oracle answers

Overall, # input bits $N = 2^{p(n)}$

$$\# \text{ gates} \approx \# \text{ paths} = 2^{p(n)\cdot q(n)} = N^{q(n)} = N^{\text{polylog}(N)} = 2^{\text{poly}(N)}$$

$$\text{Depth} = 2$$

In general, the same idea works if there are k quantifiers with each quantifier giving us a layer of AND or OR gates

To prove it formally, one needs the concept of alternating Turing machines which are generalization of non-deterministic TMs where steps are labeled with quantifiers, so we are not going to cover it here

Separating Quantum Algorithms from AC^0 circuits

With this connection, Raz & Tal showed the following

Let $x_1, \ldots, x_N$ be the truth-table of an oracle

A quantum algorithm can query bits in a superposition via the phase oracle

$$|i\rangle \longrightarrow x_i |i\rangle \quad \text{where} \quad x_i \in \{\pm 1\} \qquad \# \text{qubits} = \log N$$

An AC^0-circuit takes in input $x_1, \ldots, x_N$

Then, $\exists$ a problem s.t. $\longrightarrow$ called the Fourier Correlation problem

(1) A quantum algorithm can solve it with one query with success probability

$$\frac{1}{2} + \frac{1}{\text{polylog}(N)}$$ ← One can make this $\frac{1}{2} + 0.1$ but its more complicated and we won't cover it here

(2) Any AC^0 circuit of size $2^{\text{polylog}(N)}$ has success probability

$$\text{atmost} \quad \frac{1}{2} + \frac{\text{polylog}(N)}{\sqrt{N}} \ll \frac{1}{2} + \frac{1}{N^{1/2 - o(1)}}$$

$\Longrightarrow$ Using diagonization and above connection between PH-oracle machines and AC^0 circuit this implies that

$$\exists\, O \text{ s.t. } BQP^O \not\subseteq PH^O$$

Fourier Correlation or Forrelation Problem introduced by Aaronson

Input $x_1, \dots, x_N, y_1, \dots, y_N \in \{\pm 1\}^{2N}$ $\Rightarrow$ One can encode this with $2n$ qubits where $N = 2^n$

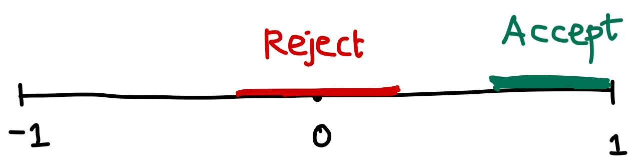
Decide if $\frac{\langle x, Hy \rangle}{N} \geq \frac{1}{32 \cdot \log N}$ "Accept"

$\frac{|\langle x, Hy \rangle|}{N} \leq \frac{1}{64 \cdot \log N}$ "Reject"

$H = H^{\otimes n}$ is the Hadamard matrix of size $2^n \times 2^n = N \times N$

Note, $\frac{x}{\sqrt{N}}$ and $\frac{y}{\sqrt{N}}$ are unit vectors and H is a unitary matrix

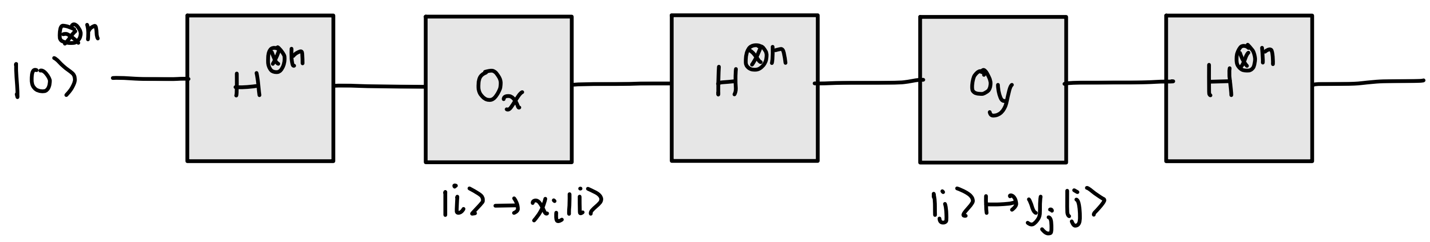
so, $\frac{\langle x, Hy \rangle}{N} \in [-1, 1]$ Also, note $\frac{\langle x, Hy \rangle}{N} = \sum_{ij} x_i y_j \frac{H_{ij}}{N}$



Why the name? H is also called the Fourier Transform matrix and Hy is the Fourier Transform of y

So, we are checking if x is correlated with the Fourier transform of y

Connection to Quantum Circuits



The final state of this circuit (before measurement) in the computational basis $|0\rangle, |1\rangle, \dots |N\rangle$ looks like

$$\frac{\langle x, Hy \rangle}{N} |0\rangle + \_\_ |1\rangle + \_\_ |2\rangle + \dots \quad \text{(Exercise)}$$

The amplitude of $|0\rangle$ is exactly the quantity we are interested in

One can use this to come up with a 1-query algorithm for this problem that succeeds with probability

$$\frac{1}{2} + \frac{1}{2} \frac{\langle x, Hy \rangle}{N} \quad \text{(Exercise)}$$

NEXT TIME Classical Lower Bound for Forrelation