Lecture 8 – Complexity theory 1

We've characterized efficient computation (whether it was classical, quantum or interference-based) as problems that we can solve with polynomial-time algorithms.

In this lecture we'll attempt to make that more precise and see where q. comp. fits in the landscape of computation.

Outline

- complexity classes
- P, EXP, NP
- reductions, hard and complete problems
- BPP, BQP, PSPACE
- oracles

Complexity classes

It's convenient to group together problems of the same type into sets called complexity classes, and then try to understand the relationships between the classes.

We will again restrict to decision problems. As such, we will view each problem as a set (called a language) containing only the Yes instances of the problem.

E.g. Primality testing

$$PRIMES = \{2, 3, 5, 7 \ldots\}$$

The complement of a language is the complementary set

$$\overline{PRIMES} = \{1, 4, 6, 8 \ldots\}$$

A complexity class is a collection of such languages.

We'll typically denote the set of all possible inputs as $\{0,1\}^*$. This represents the set of all binary strings (including the empty string).

Let's start with the most intuitive complexity class.

P – set of problems that can be solved in poly time with a deterministic algorithm

More formally

(EXP)

For a language L, we say $L \in P$ if $\exists$ a deterministic algorithm A and a polynomial $p: \mathbb{R} \to \mathbb{R}$ such that:

$$\forall x \in \{0,1\}^*, \quad A(x) = \begin{cases} 0 & \text{if } x \notin L \\ 1 & \text{if } x \in L \end{cases}$$

and the runtime of A is at most $p(|x|)$.

$(2^{p(|x|)})$

E.g. PRIMES $\in$ P

(not because of the Miller-Rabin algorithm, since that one is probabilistic; but there is a poly-time deterministic algorithm)

$$k\text{-PATH} = \{ \langle G, u, v, k \rangle \mid G \text{ is a graph}, u, v \in \text{Vertices}(G), k \geq 0; \text{ there exists a path from } u \text{ to } v \text{ in } G \text{ of length at most } k \}$$

Here the problem is to determine, given a graph G as input and nodes u, v, as well as a number $k \geq 0$, whether there is a path of length $\leq k$ from u to v.

Dijkstra's algorithm is one deterministic poly-time algorithm for this.

$$k\text{-PATH} \in P$$

What about

HAM-CYCLE = { $\langle G \rangle$ | G is a graph and there exists a cycle (closed path) in the graph that goes through each vertex exactly once }

↑ Hamiltonian cycle.

Is HAM-CYCLE $\in$ P? We don't know. No known poly-time algorithm.

However, consider the alternate problem

HAM-CYCLE-VER = { $\langle G, C \rangle$ | G is a graph and C is a Hamiltonian cycle in G }

Here we are given G and a cycle and need to check whether it's a Hamiltonian cycle. This can be done efficiently deterministically, since we just traverse the

cycle and check.

$$\text{HAM-CYCLE-VER} \in P$$

This motivates the following complexity class definition

NP

For a language L, we say $L \in NP$ if $\exists$ a deterministic algorithm A and a polynomial $\mu: \mathbb{R} \to \mathbb{R}$ such that:

$x \in L$, there exists a witness string $w \in \{0,1\}^{\mu(|x|)}$ such that $A(x,w) = 1$

$x \notin L$, there does not exist a witness w such that $A(x,w) = 1$

A's runtime is at most $\mu(|x|+|w|)$.

HAM-CYCLE $\in$ NP

When a Hamiltonian cycle exists, that cycle is the

witness string. If it doesn't exist then no witness can convince a poly-time deterministic algorithm that a cycle exist.

NP - non-deterministic polynomial time

↳ this comes from a different definition of NP that we'll explore later.

Another example problem

3-COLORING = { $\langle G \rangle$ | G is a graph and its vertices can be colored with 3 colors such that no 2 neighbouring vertices have the same color }

3-COLORING $\in$ NP

Clearly P $\subseteq$ NP. We know that problems in P can be decided in poly-time without a witness. So just make the witness the empty string.

Is $P = NP$? We don't know, but if $P = NP$ the consequences would be dramatic.

MATH-PROOF $= \{ \langle S, k \rangle \mid S$ is a mathematical statement (in some axiomatic system); $k \geq 0$; S has a proof of at most k bits $\}$

MATH-PROOF $\in$ NP

If $P = NP$ the process of proving theorems can be efficiently automated!

FACTORING $= \{ \langle N, k \rangle \mid N, k \geq 0$, N has a a prime factor $\leq k \}$

FACTORING $\in$ NP

If $P = NP$ public-key cryptography is unsecure!

How do we understand the hardness of these problems?

Reductions

We say that L_1 reduces (in poly-time) to L_2 if $\exists$ a poly-time deterministic algorithm A ($A: \{0,1\}^* \to \{0,1\}^*$) such that

$$\forall x \in \{0,1\}^* \quad x \in L_1 \text{ iff } A(x_1) \in L_2$$

Denote this as $L_1 \leq_p L_2$

Intuition: L_2 is at least as hard as L_1.

If you can map inputs of L_1 to L_2 and decide L_2 then you can decide L_1 as well

Hardness

We say a problem L is hard for complexity class C, if $\forall L' \in C$, $L' \leq_p L$.

Then we say that L is C-hard

(note this is hardness with respect to poly-time deterministic reductions; other notions also exist)

E.g. (though we won't prove this)

HAM-CYCLE and 3-COLORING are NP-hard

But HAM-CYCLE and 3-COLORING are also in NP...

Completeness

We say that L is C-complete if

- $L \in C$
- L is C-hard

HAM-CYCLE and 3-COLORING are NP-complete!

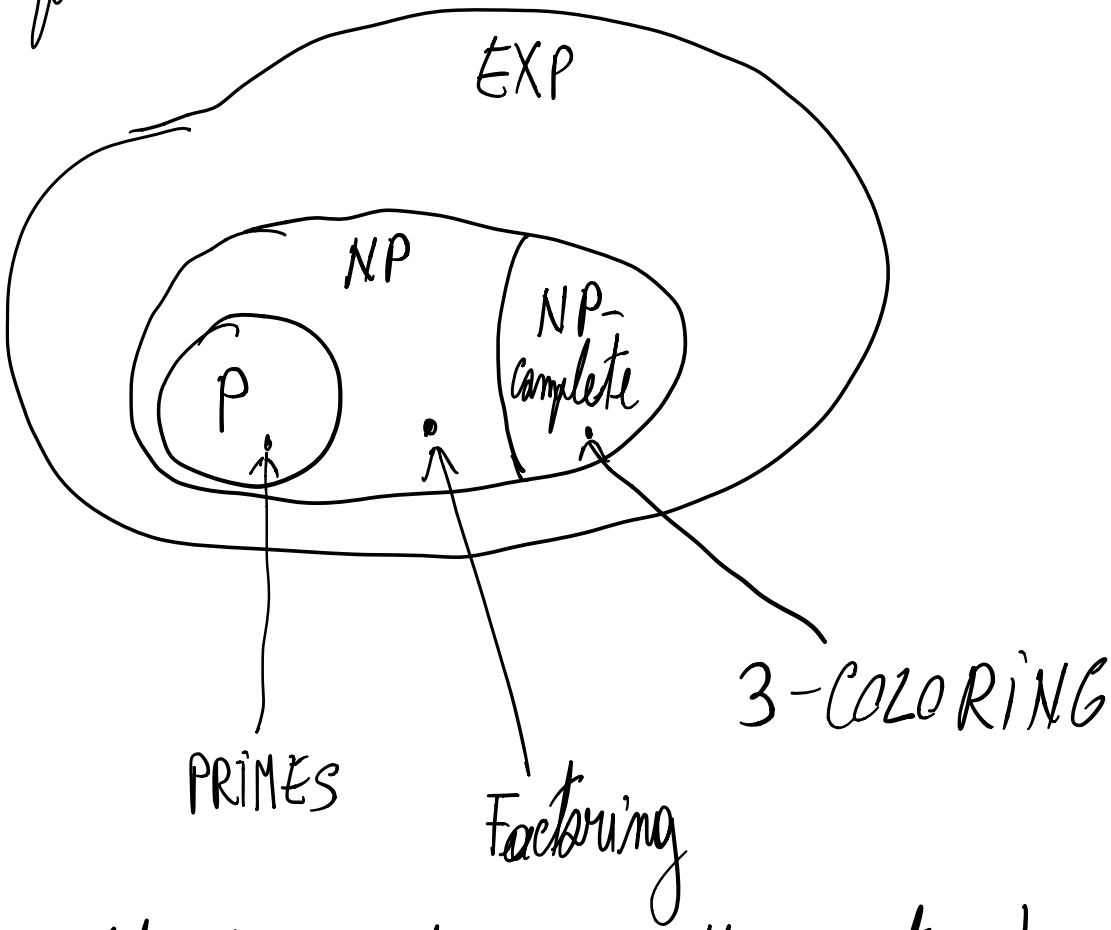
Intuition: These problems are the hardest in NP

(though they're not the only ones)

As a simple observation, note that if

> HAM-CYCLE or 3-COLORING $\in P$
>
> then $P = NP$

Picture so far



There are a lot of conjectures in this picture!

We don't know that $P \neq NP$, but we suspect it's true. We don't know that Factoring $\notin P$ or

that Factoring is not NP-complete, but we suspect both are true.

What about probabilistic computation?

BPP - bounded-error probabilistic polynomial time

For a language L, we say $L \in BPP$ if $\exists$ a deterministic algorithm A and polynomials p, q, s such that:

$$x \in L, \quad \Pr_{r \leftarrow \{0,1\}^{p(|x|)}}\left(A(x,r)=1\right) \geq a$$

$$x \notin L, \quad \Pr_{r \leftarrow \{0,1\}^{p(|x|)}}\left(A(x,r)=1\right) \leq b$$

A's runtime is at most $q(|x|+|r|)$, and

$$a-b \geq \frac{1}{s(|x|+|r|)}$$

Here a-b is the acceptance-rejection (or completeness-soundness) gap. This can be boosted to arbitrarily close to 1 by repeating A many times (at least $s(|x|+|r|)$ times) and taking the majority outcome. This is why we typically take $a = 2/3$, $b = 1/3$.

Note that $P \subseteq BPP$ (just have A ignore the randomness)

BPP is probabilistic version of P. Is there a probabilistic version of NP?

Yes, just make A probabilistic in def. of NP.

The resulting class is called MA (Merlin-Arthur)

Finally, efficient quantum computation is captured by the class

BQP - Bounded-error quantum polynomial time

For a language L, we say $L \in BQP$ if $\exists$ a quantum algorithm A and polynomials p, q such that:

$$x \in L, \quad \Pr(A(x) = 1) \geq a$$

$$x \notin L, \quad \Pr(A(x) = 1) \leq b$$

A's runtime is at most $p(|x|+|r|)$, and

$$a - b \geq \frac{1}{q(|x|+|r|)}$$

Here the randomness is the result of measuring quantum states.

We saw in previous lectures that any boolean function can be implemented using only Toffoli gates. In addition, it's easy to reproduce randomness

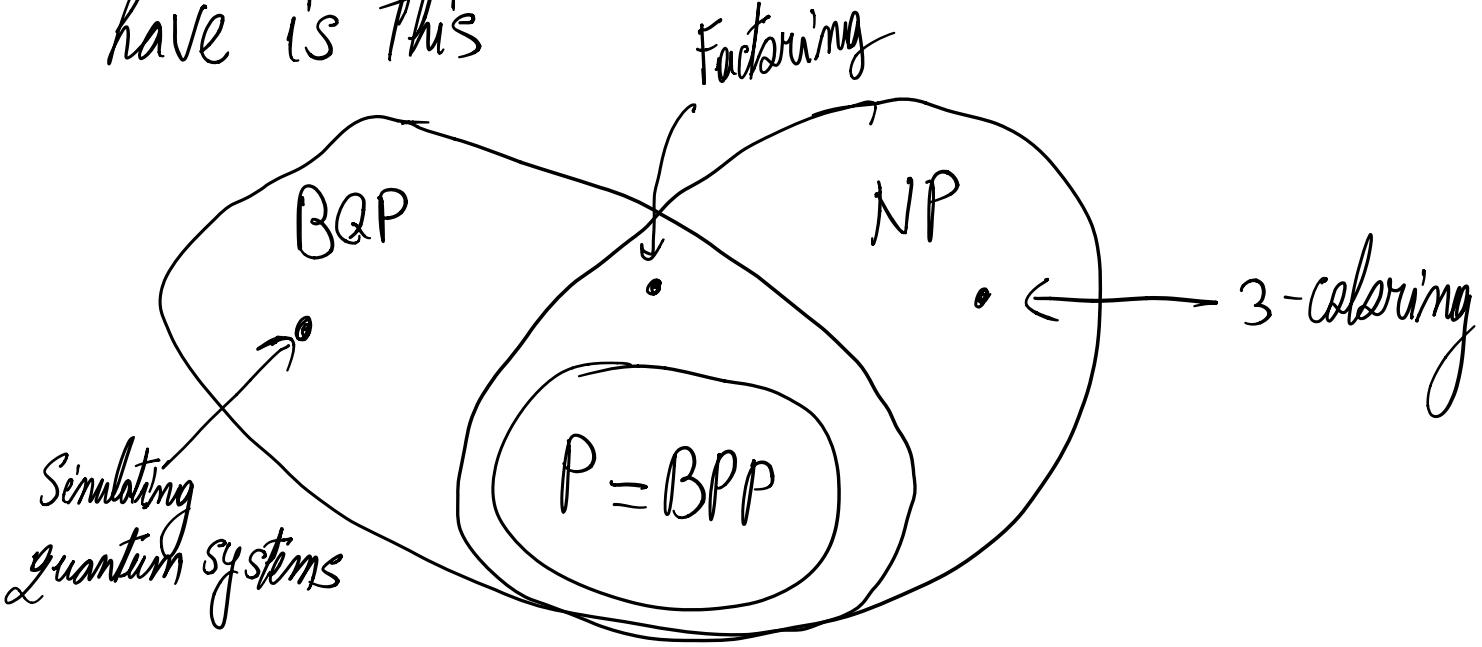
quantumly by measuring $H^{\otimes n}|0^{\otimes n}\rangle$.

Thus $BPP \subseteq BQP$

Again, we believe $BPP \subset BQP$, but we have no proof of this.

We do know that Factoring $\in BQP$ and Factoring $\in NP$. So what is the relation between BQP and NP?

It's conjectured that $BQP \not\subseteq NP$ and $NP \not\subseteq BQP$. So the picture we think we have is this



Do we know of BQP-complete problems?

Yes, here's a simple example

$$\text{Q-CIRCUIT} = \{ \langle C \rangle \mid C \text{ is a } q \text{ circuit s.t. } |\langle 0^n | C | 0^n \rangle|^2 \geq 2/3 \}$$

↑ Prob of seeing output $|0^n\rangle$ when measuring $C|0^n\rangle$

(there's a technical point here that we won't delve into too much; the way we defined BQP Yes instances occur with prob $\geq 2/3$ and No instances with prob $\leq 1/3$; but for Q-CIRCUIT No instances have prob $< 2/3$. In fact what we have is a promise problem, a problem where we have a Yes set and a No set.

So Q-CIRCUIT $= (L_{YES}, L_{NO})$

$$L_{YES} = \{ \langle C \rangle \mid |\langle 0^n | C | 0^n \rangle|^2 \geq 2/3 \}$$

$$L_{NO} = \{ \langle C \rangle \mid |\langle 0^n | C | 0^n \rangle|^2 \leq 1/3 \}$$

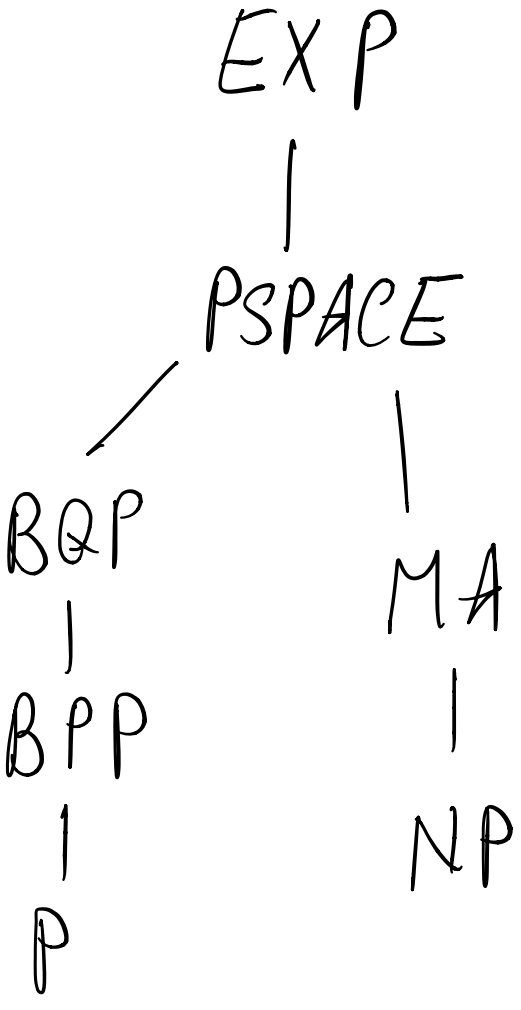
and this problem is complete for Promise BQP, which is a set of promise problems, rather than just languages.

But we'll ignore this for now and work with just BQP)

The last complexity class we'll mention is PSPACE.

PSPACE - set of languages that can be decided by a deterministic algorithm that uses polynomial memory/space (though it need not necessarily run in polynomial time)

Here are all the relations we know (an edge going from bottom to top means containment)



We conjecture that all these containments are strict, except for $P \subseteq BPP$ and $NP \subseteq MA$, since here we believe $P = BPP$ and $NP = MA$ (i.e. the conjecture that randomized algorithms can be de-randomized using pseudo-random

number generators)

The only strict containment we know is $P \subset EXP$.

Oracles

We've talked about oracle problems where we're given black box access to a function. How are these treated in complexity theory?

For all the definitions covered so far, we can give oracle access to the algorithm. That is, the algorithm is allowed to query the oracle.

So for some oracle $O: \{0,1\}^* \to \{0,1\}^*$, P^O has the same definition as P, except the poly-time algorithm A has access to O (we denote it as A^O).

We can then also define languages relative to the oracle. For instance for some $O: \{0,1\}^* \to \{0,1\}$ we can simply define

$$L^O = \{ x \mid O(x) = 1 \}$$

Baker-Gill-Solovay showed that

1) $\exists$ an oracle O, s.t

$$P^O \neq NP^O$$

2) $\exists$ an oracle O, s.t.

$$P^O = NP^O$$

So while oracle results can give us some insights, ultimately they can't be used to resolve questions like P vs NP.

The same is true for BPP and BQP.

For instance $\text{Simon} = (L^O_{Yes}, L^O_{No})$

$$L^O_{Yes} = \{ 1^n \mid n > 0 \text{ s.t } O(1^n, x) = f(x), \text{ where } f \text{ is a Simon function, so } f(x) = f(x \oplus s) \text{ for some } s \in \{0,1\}^n \setminus \{0^n\} \}$$

$$L^O_{No} = \{ 1^n \mid n > 0 \text{ st } O(1^n, x) = f(x), \text{ where } f \text{ is a 1-to-1 function} \}$$

Simon $\notin BPP^O$ but Simon $\in BQP^O$

However there also exists O s.t $BPP^O = BQP^O$

One can provide an oracle to a complete problem.

Recall $3\text{-COLORING} = \{ \langle G \rangle \mid G \text{ is 3-colorable} \}$

Let $O(x) = \begin{cases} 1 & \text{if } x \text{ encodes a graph } G \text{ that is 3-colorable} \\ 0 & \text{otherwise} \end{cases}$

Then $P^Q = P^{3\text{-COLORING}}$ and we denote it as P^{NP}, since 3-COLORING is NP-complete.

We can do the same for the other classes.

Note that $P \subseteq P^{NP}$, $NP \subseteq P^{NP}$

But P^{NP} is not necessarily equal to NP. In fact it's believed that they're different.

More on that next time.