Aim of complexity theory: understand & classify difficulty of problems.

Difficulty typically measured in terms of resources needed (time, memory) to solve problem, as function of the problem size $n$ (= number needed to specify the specific instance of the problem, i.e., the input).

Focus on decision problems (= yes/no problems):

Problem: $f : \{0,1\}^n \rightarrow \{0,1\}$

$$f : x \mapsto f(x) = \text{yes/no}$$

Note: Not all decision problems can be solved by a few calls to a decision problem.

CS terminology: $L = \{ x \in \{0,1\}^* \mid f(x) = 1 \}$ is called language.
1. Classical complexity classes

Which comp. model? — Church-Turing thesis:
all comp. models equiv. w/ poly overhead
→ equiv. for our purposes.

Class $P$ ("polynomial time"): $f(x)$ can be computed in time ($=\#$ of operations) $\text{poly}(|x|)$ → length $|x| 

(These are commonly considered efficiently solvable.)

Examples in $P$:
- multiplication, addition... (or decide versions)
- primality testing
- $\text{gcd}$

(Note: We can also define other time classes, e.g., $\text{EXP}$)
NP ("non-deterministic polynomial time")

Problem can be solved in time \( \text{poly}(|x|) \) by a non-deterministic computer, i.e., which can check many inputs \( y \) in parallel.

Equiv: If \( f(x) = 1 \), there exists a proof (witness) \( y \) which can be checked in time \( \text{poly}(|x|) \) with a verifier \( v(x, y) = 1 \) (and only if \( f(x) = 1 \)):

\[\exists v(x, y) \text{ s.t.},\]

\[x \in L \rightarrow \exists y : v(x, y) = 1 \quad (\text{"proof accepted"})\]
 \[x \notin L \rightarrow \forall y v(x, y) = 0 \quad (\text{"proof rejected"})\]

\( y \) is the "solution" to the problem.

Examples:

* Graph coloring (color graph w/out eg, adjacent colors)

  \( y \) is a valid coloring \( \rightarrow \) proof = a valid coloring

  \( \neg y \) is no coloring exists \( \rightarrow \) no valid proof

* \( k \)-SAT: variables \( x_1, \ldots, x_k \)

  \( k \)-clause \({ g_j = x_j \lor \bar{x}_j \lor \bar{x}_j \lor \ldots \lor \bar{x}_j \text{ etc.} }\)

  (i.e., each \( g_j \) is \( x_j \) or \( \bar{x}_j \), and \( g_j \) is or \( 3 \) variables)
Question: Do there exist a satisfying assignment $x_1, \ldots, x_k$ s.t. $g_j = 1 \forall j$?

Yes instance: Proof = $(x_1, \ldots, x_k)$

No instance: No proof exists

* (Prime) factor decomposition
* graph isomorphism: are two graphs isomorphic?

Can problems in NP be arbitrarily hard?

→ No, e.g., some are typ. w.r.t. NP (due to their branching structure).

→ NP contains "reasonably hard" problems.

→ Generally (?) believed: $P \neq NP$ (ask: $P \subset NP$).

Classes beyond NP:

PSPACE (polynomial space):
Problems which can be solved using only $\text{poly}(|x|)$ memory (no time limited, but note that time \( \leq \text{exp}(\text{space}) \))
If $P \neq NP$: The "hardest" problems in $NP$ should be interesting class of hard problems.

How can we identify "hardest" problems in $NP$?

**NP-complete**: A problem is $NP$-complete if we can map ("reduce") any problem in $NP$ to it in poly time. (i.e.: if we have a way to solve an $NP$-complete problem in poly time, we can solve any $NP$-problem in poly time)

**Examples for NP-complete problems**

- graph coloring, $k$-$SAT$ for $k \geq 3$

**NP-complete (NP-intermediate) problems**

- factoring, graph isomorphism

The Cook-Levin-Theorem: $3$-$SAT$ is NP-complete

Aim: reduce any NP-problem to $3$-$SAT$.

General NP-problem: Given by relation $r(x, y)$,

yes instance $\iff \exists y : r(x, y) = 1$. 
\[ u(x, y) = \text{circuit:} \]

\[
\begin{align*}
x_1 & \rightarrow 2_1 & 2_2 & 2_3 & 2_4 & 2_5 \\
x_2 & \rightarrow 3_0 & 3_1 & 3_2 & 3_3 & 3_4 & 3_5 \\
x_3 & \rightarrow 4_0 & 4_1 & 4_2 & 4_3 & 4_4 & 4_5 \\
(x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5), \text{ etc.} 
\end{align*}
\]

"History state" \( (\overrightarrow{z_0}, \overrightarrow{z_1}, \ldots, \overrightarrow{z_5}) \) of verifier checking proof \( y \)

Construct 3-SAT instance which ensures \( (\overrightarrow{z_0}, \overrightarrow{z_1}, \ldots) \) is a valid history of \( z \) for instance \( x \) and some proof \( y \).

- For each input \( x_i \), add clause which ensures that \( (\overrightarrow{z_0}_i) = x_i \) (i.e., \( z_0_i = x_i \) or \( z_0_i = \overline{x_i} \))

- For each gate (ref. copy), add clauses which ensure that the gate is done correctly (i.e., pencil/typical copy) - acts as 3-SAT case

- Add a clause requiring the output \( z_5 \) to be "yes."

\[ \exists y : u(x, y) = 1 \iff \text{3-SAT satisfiable.} \]
Central idea of proof: Construct three-coloring of verifier $v$ and write $k$-SAT problem for it.

Note: This $k$-SAT corresponds to a classical local Hamiltonian on a $D$-lattice, finding ground state of 2D lattices is NP-complete.

(Note: NP-completeness does not tell us about average case complexity.
- Many hard problems are only hard in certain parameter regimes.)

Quantum Complexity Classes:

BQP ("Bounded-error quantum polynomial time")

The class of problems which can be solved by a (qubit-based) quantum computer in time poly$(1/x)$ with bounded error.

Note: Bounded error $\iff$ yes-instance: $p(output = 1) \geq 2/3$
no-instance: $p(output = 1) < 1/3$. 
We can use amplification (i.e., running an algorithm poly # of times and using majority vote) to get this:

\[ p(\text{output} = 1 | y_0) \geq 1 - 2^{-|x|} \]

\[ p(\text{output} = 1 | u_0) \leq 2^{-|x|} \]

Similarly,

\[ p(\text{output} = 1 | y_0) \geq 1/2 + \frac{1}{\text{poly}(|x|)} \]

\[ p(\text{output} = 1 | u_0) \leq 1/2 - \frac{1}{\text{poly}(|x|)} \]

We have that $P \subseteq BQP$ (and $\text{BPP} \subseteq \text{BQP}$) with class. rand. poly. time.

Problems in $BQP$ not known to be in $P$

* Factoring
* Simon's $\mathbb{C}^2$ systems

What is a classical upper bound on $BQP$?

(i.e., how hard is it to simulate a $Q$-Comp. classically?)

$\Rightarrow Q$-Comp. can be simulated in polynomial space, i.e., in $PSPACE$. 