1 NP

1.1 The basics

We learned about two classes in computability theory (R and RE) and about two classes in complexity theory (P and NP). The following question asks you to compare them:

Question 1. *Explain the difference between the class* R *and the class* P*. Explain the difference between the class* RE *and the class* NP*. Compare what we know about the difference between* R *and* RE *to what we know about the difference between* P *and* NP*.*

The next question should be easy, it mainly tries to verify that you understand the basic definitions of NP (what it means to be in NP, what is a witness relation).

Question 2. *For each of the following languages, prove that it is in* NP*, and show a witness-relation for it:*

- $CLIQUE = \{(G,k) : G \text{ has a clique of size at least } k\}$.
- $SUBSET\text{-}SUM = \{x_1,...,x_n,t \in \mathbb{N} : \exists S \subseteq [n], \sum_{i \in S} x_i = t\}$.
- $QUAD = \Big\{q_1,...,q_n : \text{each } q_i \text{ is a quadratic polynomial } \{0,1\}^n \to \{0,1\} \text{ and } \exists x \in \{0,1\}^n \text{ such that } \forall i \in [n],\ q_i(x) = 0\Big\}$.

The next question seems like "another one of the above" (supposedly, show that some nice language is in NP), but it is actually a trick question – be warned!

Question 3. *Fix an arbitrary polynomial* $f : \mathbb{Z}^k \to \mathbb{Z}$ *with integer coefficients. Let*

$$DIOPHANTINE_f = \{x \in \mathbb{Z} : \exists y_1,...,y_{k-1} \in \mathbb{Z},\ f(x,y_1,...,y_{k-1}) = 0\}\ .$$

Do you see a way to prove that $DIOPHANTINE_f \in$ NP*, for an arbitrary* f*?*

(Note: The language superficially looks like it's in NP*, since it's defined by an "*$\exists$*" quantifier; but are witnesses of bounded size? After you understand the problem, read the footnote for the answer.*[1]*)*

A sanity check. When defining $\mathsf{NTIME}[T]$, we considered a verifier running in time $O(T)$ that gets proofs of length at most T. What would happen if we change this and insist on getting proofs of length *exactly* T? The following exercise asks you to prove that essentially nothing would change.

Question 4. *For the sake of this question, let's say that* $L \in \mathsf{NTIME}'[T]$ *if there is a Turing machine* V *such that:*

[1] Not only is this language not in NP, for some f's it's undecidable! Proving this was a famous open problem (called Hilbert's tenth problem), and Matiyasevich resolved it in the 1970s (following Robinson, Davis, and Putnam). The idea is to represent every $L \in \mathsf{RE}$ as $L = DIOPHANTINE_f$ for some $f = f_L$.

- $x \in L \iff \exists w \in \{0,1\}^{T(|x|)}, V(x,w) = 1$.
- $x \notin L \iff \forall w \in \{0,1\}^{T(|x|)}, V(x,w) = 0$.
- *On any input* $(x,w) \in \{0,1\}^n \times \{0,1\}^{T(n)}$ *the machine* V *runs in time* $O(T(n))$.

Prove that for any time-constructible T *we have* $\mathsf{NTIME}'[T] \subseteq \mathsf{NTIME}[T]$ *and* $\mathsf{NTIME}[T] \subseteq \mathsf{NTIME}'[T']$*, for some* $T'(n) = O(T(n))$.

When solving the question above, you may work in a multitape machine model. What would change when working in the standard (single-tape) model?

How "strong" is NP? The following three questions ask you to show increasingly tighter upper-bounds on NP.

Question 5. *Is* $\mathsf{NP} \subseteq \mathsf{RE}$*? Is* $\mathsf{NP} \subseteq \mathsf{R}$*?*

Question 6. *Let* $\mathsf{EXP} = \cup_{c \in \mathbb{N}} \mathsf{TIME}[2^{n^c}]$ *be the set of languages computable in exponential time* $2^{n^{O(1)}}$*. Prove that* $\mathsf{NP} \subseteq \mathsf{EXP}$*. Do you think that* $\mathsf{NP} = \mathsf{EXP}$*?*[2]

Question 7. *Let* $\mathsf{PSPACE} = \cup_{c \in \mathbb{N}} \mathsf{SPACE}[n^c]$ *be the set of languages computable in polynomial space* $n^{O(1)}$*. Prove that* $\mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}$*. Do you think that* $\mathsf{NP} = \mathsf{PSPACE}$*?*[3]

Understanding P vs NP. The next question is a sanity check: don't get confused, and just verify that you understand what $\mathsf{P} \neq \mathsf{NP}$ means and what $\mathsf{P} = \mathsf{NP}$ means.

Question 8. *Mark each of the following statements as correct or incorrect:*

1. *To prove* $\mathsf{P} = \mathsf{NP}$*, it suffices to prove that there is* $L \in \mathsf{NP}$ *such that* $L \in \mathsf{P}$*.*
2. *If for every* $L \in \mathsf{NP}$ *it holds that* $L \in \mathsf{P}$*, then* $\mathsf{P} = \mathsf{NP}$*.*
3. *If there is* $L \in \mathsf{NP}$ *such that* $L \notin \mathsf{P}$*, then* $\mathsf{P} \neq \mathsf{NP}$*.*
4. *If* $\mathsf{P} \neq \mathsf{NP}$ *then for all* $L \in \mathsf{NP}$ *it holds that* $L \notin \mathsf{P}$*.*

Now, to the bottom line: Which of the above do we need to show to prove $\mathsf{P} \neq \mathsf{NP}$*? Which of the above do we need to show to prove* $\mathsf{P} = \mathsf{NP}$*?*

1.2 Search vs decision

In class we saw that $\mathsf{P} = \mathsf{NP}$ if and only if every polytime verifiable search problem is also solvable in polynomial time.

Question 9. *Prove or disprove: If* $\mathsf{P} = \mathsf{NP}$*, then for every* $L \in \mathsf{NP}$ *and every witness relation* R *for* L*, we can solve the search problem* R *in polynomial time.*

[2] Amazingly, resolving NP vs EXP is an open problem.
[3] Alas, resolving NP vs PSPACE is also an open problem.

Question 10. *Show a search problem that cannot be solved in polynomial time (even if* $\mathsf{P} = \mathsf{NP}$*). Is your search problem polytime verifiable?*

Question 11. *Show a specific polytime verifiable search problem R such that if* $\mathsf{P} \neq \mathsf{NP}$*, then R cannot be solved in polynomial time.*

1.3 NP-completeness and the Cook-Levin theorem

The following questions are a sanity check for your understanding of the definitions of "NP-hard" and "NP-complete" and of reductions.

Question 12. *Show that the universal problem is* NP*-hard.*

Question 13. *Is every* NP*-hard problem also* NP*-complete? Either prove this, or show an explicit counter-example (proving it's not* NP*-complete).*

Question 14. *Assume that there is a polynomial-time reduction of some problem L to an* NP*-complete problem L'. Does this mean that L is* NP*-complete?*

Question 15. *Prove or disprove: Every problem in* P *reduces to $CircuitSAT$ in polynomial time.*

Question 16. *What would be the consequence of a polynomial-time reduction of $CircuitSAT$ to a problem in* P*?*

Question 17. *Let L and L' be two* NP*-complete problems. Which of them is reducible in polynomial time to which?*

Question 18. *Consider the language $4CSP$ which consists of satisfiable lists of 4-constraints.*[4] *Is it* NP*-complete?*

The following non-technical questions are based on many true stories, and may help you to develop intuition about the meaning of NP-completeness.

Question 19. *Dr. Underwood Dudley has emailed you a 370-page PDF describing what he claims to be a polynomial-time algorithm solving $3CSP$. The text is hard to read and uses vaguely defined terminology. What is your guess: Is Dr. Dudley's algorithm correct?*

Question 20. *Your company's upper management has instructed your team to design a very fast algorithm that solves $3CSP$ by next Monday. What do you tell them?*

The following questions are more complicated, and some of them require you to go into the details of the proof of the Cook-Levin theorem.

Question 21. *Assume that Dr. Dudley was correct after all, and that there is a polynomial-time algorithm for $3CSP$, and moreover, Dr. Dudley's algorithm* finds *a satisfying assignment when one exists. Let $L \in$* NP *and let V be an* NP*-verifier for L. Describe two* different *polynomial-time algorithms that, given $x \in L$, find a witness w such that $V(x, w) = 1$.*

[4] That is, the problem is defined identically to $3CSP$, except that we allow constraints over four variables instead of three variables.

Question 22. *Let Circuit2SAT be the language of descriptions of Boolean circuits that have* at least two *satisfying assignments. Is this language* NP*-complete? Prove your answer.*

Question 23. *A Boolean circuit with NAND gates is defined similarly to standard Boolean circuits, except that instead of* $\{\wedge, \vee, \neg\}$ *gates, we only allow* NAND *gates (over two variables) and* $\neg$ *gates (over one variable). Let* $Circuit_{NAND}SAT$ *be the language of satisfiable Boolean circuits with NAND gates. Is this language* NP*-complete? Prove your answer.*

Question 24. *In the reduction we saw of an arbitrary* $L \in \mathsf{NTIME}[T]$ *to CircuitSAT, given input* $x \in \{0,1\}^n$*, what is the size of the circuit* C_x *that we constructed?*

Question 25. *We say that L is decidable by a* T*-time right-left-walking verifier if there is an* NP*-verifier for L that runs in time T and that on any input x and witness w first walks* $O(T)$ *steps to the right, then walks all the way back to the left, then halts.*[5] *Show a reduction of any such L to CircuitSAT that maps an input x to a circuit* C_x *of size* $O(T)$*.*

Question 26. *Generalize the claim in the previous question to a broader class of verifiers, rather than only right-left-walking verifiers.*

Question 27. *A Boolean circuit with arbitrary fan-in-10 gates is defined similarly to standard Boolean circuits, except that instead of* $\{\wedge, \vee, \neg\}$ *gates of fan-in two, the circuit can have gates computing* any function over 10 input gates/variables*. Let* $Circuit_{\mathsf{FanIn10}}SAT$ *be the language of satisfiable Boolean circuits with arbitrary fan-in-*10 *gates. Show a reduction from this language to kCSP for a constant k.*[6]

1.4 NP-completeness of natural problems

Recall that $3CSP$ is NP-complete but $2CSP \in \mathsf{P}$.[7] The next question refers to a generalization of $2CSP$, which is actually NP-complete. The main difference is that in $2CSP$ the variables may take values either 0 or 1, whereas in the generalization the variables may come from a larger alphabet (of size three or more).

Question 28. *A* $(2,3)$*-constraint is a function* $f\colon \{0,1,2\}^2 \to \{0,1\}$ *(i.e., the constraint is defined over two input variables* x_i, x_j *each taking values in* $\{0,1,2\}$*, and the constraint reads the variables and outputs "yes" or "no"). Let* $(2,3)$*-CSP be the language of satisfiable lists of* $(2,3)$*-constraints. Show that* $(2,3)$*-CSP is* NP*-complete.*

The next two questions ask you to come up with new Karp reductions (that you haven't seen in class), showing that certain natural problems are NP-complete.

Question 29. *In the problem SetCover we are given a collection of sets* $S_1, ..., S_m$ *and an integer* $k \leq m$*, and need to decide if there are k sets whose union covers all possible elements* $\cup_{i \in [m]} S_i$*.*[8] *Show that SetCover is* NP*-complete.*

[5] The walking pattern of the machine is simple (i.e., right then left), but it may still perform non-trivial computation. For simplicity, let us ignore the question of how it is possible for the machine to walk exactly $O(T)$ steps to the right without counting its steps.

[6] The problem $kCSP$ is defined identically to $3CSP$ except that constraints now involve k variables rather than 3.

[7] We mentioned that $2CSP \in \mathsf{P}$ in class, although we haven't seen proof of that.

[8] That is, where there are indices $i_1, ..., i_k$ such that $\cup_{j \in [k]} S_{i_j} = \cup_{i \in [k]} S_i$.

Hint: Use a Karp reduction from 3SAT over n variables and with m clauses. Create $2n$ subsets of $[m+n]$ such that the i^{th} set contains $m+i$ and the indices of clauses that are satisfied when $x_i = 1$, and the $(n+i)^{th}$ set contains $m+i$ and the indices of clauses that are satisfied when $x_i = 0$.

Question 30. *A 1-in-3 constraint is a function that gets as input three literals as outputs "yes" if and only if exactly one of the literals is true. Let* 1-in-3SAT *be the language of satisfiable lists of 1-in-3 constraints. Show that* 1-in-3SAT *is* NP*-complete.*

Hint: Use a Karp reduction from 3SAT. The main technical trick is to notice that $\ell_1 \vee \ell_2 \vee \ell_3$ is true if and only if $ExactlyOne(\neg\ell_1, a, b) \wedge ExactlyOne(b, \ell_2, c) \wedge ExactlyOne(c, d, \neg\ell_3)$, where ExactlyOne is the 1-in-3 function and a, b, c, d are auxiliary variables.