COMPLEXITY

LECTURE 6

May 30, 2022

Brom WESTERBAAN

Institute for Computing and Information Sciences
Radboud University Nijmegen

NWI-IBC028

This week

1. CNF is **NP**-complete

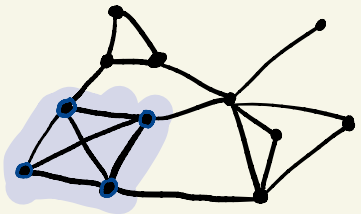
2. Tips for the exam

Refresher

P is the class of problems that's solvable in polynomial time.

NP is the class of problems that's solvable in polynomial time, given the right 'hint'.

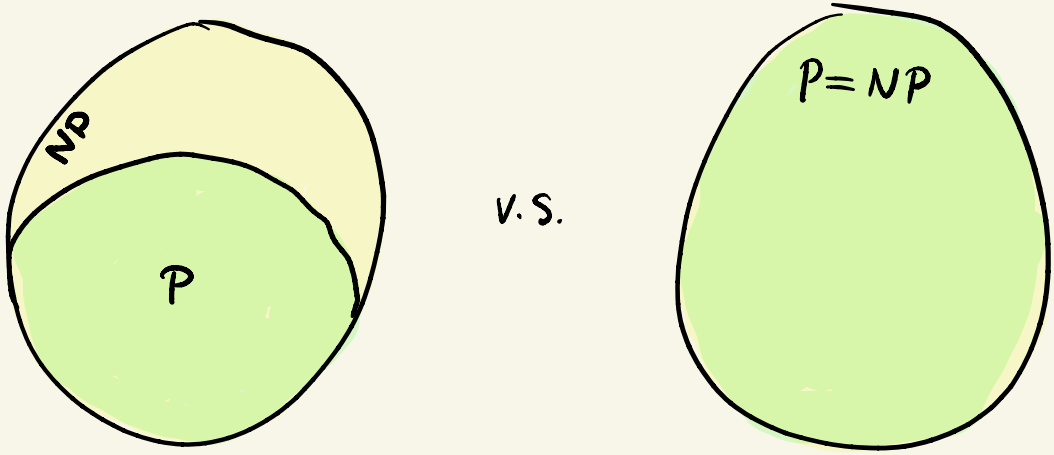
EXAMPLE While it's difficult (unknown to be in **P**) to determine if a graph has a clique of size k, it's easy to verify that whether a given set of nodes clique of size k.



a clique of size 4

Refresher: P v.s. NP

While clearly $\mathbf{P} \subseteq \mathbf{NP}$,
it's unknown whether $\mathbf{P} = \mathbf{NP}$,
but $\mathbf{P} \neq \mathbf{NP}$ is generally expected.



Refresher: NP-completeness

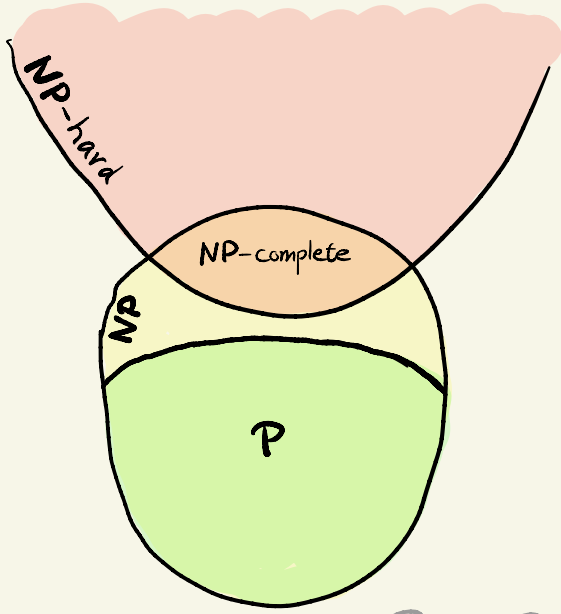
While no problem $A \in \mathbf{NP} \setminus \mathbf{P}$ is known, there are problems in **NP** 'harder' than all others.

A problem A is **NP-hard** when $B \leq_p A$ for all $B \in \mathbf{NP}$.

An **NP**-hard problem that's in **NP** is called **NP-complete**.

Note

If any **NP**-complete problem is in **P**, then **P** = **NP**.
So **NP**-complete problems are likely to be difficult to solve.



situation when $P \neq NP$

Refresher: proving NP-completeness

To prove that a problem A is **NP**-complete one generally proceeds as follows:

1. show that $A \in \mathbf{NP}$ directly,
2. show that A is **NP**-hard by reducing an **NP**-hard problem B to A.

This leads to a tree of reductions, such as:

$$\text{CNF} \leq_p \text{3CNF} \leq_p \text{Clique} \leq_p \text{VertexCover} \leq_p \text{HamPath} \leq_p \text{HamCycle} \equiv_p \text{TSP}$$

$\text{CNF} \leq_p \text{SAT}$

$\text{3CNF} \leq_p \text{3Color}$

! For this scheme to work, we must start with one **NP**-hard problem at the root, which is CNF, and show it's **NP**-hard, directly.

Why CNF instead of SAT?

We'll prove that CNF is NP-hard, but the famous theorem is:

THEOREM (COOK-LEVIN, 1971, 1973) SAT is NP-complete.

Of course, since $CNF \leq_p SAT$, COOK-LEVIN follows when CNF is NP-hard, but getting $SAT \leq_p CNF$ is not simply a matter of putting a formula φ into conjunctive normal form:

Why CNF instead of SAT?

We'll prove that CNF is NP-hard, but the famous theorem is:

THEOREM (COOK–LEVIN, 1971, 1973) SAT is NP-complete.

Of course, since $CNF \leq_p SAT$, COOK–LEVIN follows when CNF is NP-hard, but getting $SAT \leq_p CNF$ is not simply a matter of putting a formula φ into conjunctive normal form:

$$(a_{11} \wedge a_{12}) \vee (a_{21} \wedge a_{22}) \longrightarrow (a_{11} \vee a_{21}) \wedge (a_{11} \vee a_{22}) \wedge (a_{12} \vee a_{21}) \wedge (a_{12} \vee a_{22})$$

$$\bigvee_{i \in I} a_{i1} \wedge a_{i2} \longrightarrow \bigwedge_{f : I \to \{1,2\}} a_{1f(1)} \vee a_{2f(2)}$$

Why CNF instead of SAT?

We'll prove that CNF is NP-hard, but the famous theorem is:

THEOREM (COOK–LEVIN, 1971, 1973) SAT is NP-complete.

Of course, since $CNF \leq_p SAT$, COOK–LEVIN follows when CNF is NP-hard, but getting $SAT \leq_p CNF$ is not simply a matter of putting a formula φ into conjunctive normal form:

$$(a_{11} \wedge a_{12}) \vee (a_{21} \wedge a_{22}) \longrightarrow (a_{11} \vee a_{21}) \wedge (a_{11} \vee a_{22}) \wedge (a_{12} \vee a_{21}) \wedge (a_{12} \vee a_{22})$$

$$\bigvee_{i \in I} a_{i1} \wedge a_{i2} \longrightarrow \bigwedge_{f : I \to \{1,2\}} a_{1f(1)} \vee a_{2f(2)}$$

← $2^{|I|}$ clauses! (so not polynomial)

Why CNF instead of SAT?

We'll prove that CNF is NP-hard, but the famous theorem is:

THEOREM (COOK–LEVIN, 1971, 1973) SAT is NP-complete.

Of course, since $CNF \leq_p SAT$, COOK-LEVIN follows when CNF is NP-hard, but getting $SAT \leq_p CNF$ is not simply a matter of putting a formula φ into conjunctive normal form:

$$(a_{11} \wedge a_{12}) \vee (a_{21} \wedge a_{22}) \longrightarrow (a_{11} \vee a_{21}) \wedge (a_{11} \vee a_{22}) \wedge (a_{12} \vee a_{21}) \wedge (a_{12} \vee a_{22})$$

$$\bigvee_{i \in I} a_{i1} \wedge a_{i2} \longrightarrow \bigwedge_{f : I \to \{1,2\}} a_{1f(1)} \vee a_{2f(2)}$$

$\leftarrow 2^{|I|}$ clauses! (so not polynomial)

While it is possible to get $SAT \leq_p CNF$ directly, Richard KARP decided to modify COOK's proof instead to show that CNF is NP-hard, and we'll follow his lead.

THEOREM (KARP, 1972) CNF is **NP**-hard.

PROOF Let $X \in \mathbf{NP}$; we must show that $X \leq_P \mathrm{CNF}$.

That is, we must find $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ that runs in polynomial time, such that,

$$\forall w \in \{0,1\}^* \Big[\quad w \in X \iff f(w) \in \underset{\cup\!\cup}{\{0,1\}^*}\ \mathrm{CNF} \quad \Big]$$

RECALL that decision problems are formally subsets of $\{0,1\}^*$.
So we have implicitly assumed a sensible encoding

$$\mathcal{C} := \{ \varphi \text{ formula in CNF} \} \xhookrightarrow{e} \{0,1\}^*.$$

E.g. $(A \vee \beta) \wedge \neg\gamma \rightsquigarrow$ ASCII-encoding of "(A \/ beta) /\ ~ gamma"

THEOREM CNF is **NP**-hard.

PROOF Let $X \in \mathbf{NP}$; we must show that $X \leq_p \mathrm{CNF}$.

That is, it suffices to find $f : \{0,1\}^* \longrightarrow \mathcal{C}$ that (or rather ecf) runs in polynomial time, such that,

$$\forall w \in \{0,1\}^* \left[\quad w \in X \iff f(w) \in \underbrace{\mathcal{C}}_{=\mathrm{CNF}} \quad \right]$$

RECALL that decision problems are formally subsets of $\{0,1\}^*$.
So we have implicitly assumed a sensible encoding

$$\mathcal{C} := \{ \varphi \text{ formula in CNF} \} \xhookrightarrow{e} \{0,1\}^*.$$

E.g. $(A \vee \beta) \wedge \neg\gamma \rightsquigarrow$ ASCII-encoding of "(A \/ beta) /\ ~ gamma"

What do we know about $X \in NP$?

What do we know about $X \in NP$?

There is a polynomial time algorithm $A : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ (the "verifier") and some $C > 0$ and $k \in \mathbb{N}$ such that for all $w \in \{0,1\}^*$,

$$w \in X \iff \exists y \in \{0,1\}^*,\ |y| \leq C|w|^k \left[\ A(w,y) = 1\ \right]$$

(y: certificate, $|y| \leq C|w|^k$: not too big)

What do we know about $X \in NP$?

There is a polynomial time algorithm $A : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ (the "verifier") and some $C > 0$ and $k \in \mathbb{N}$ such that for all $w \in \{0,1\}^*$,

$$w \in X \iff \exists y \in \{0,1\}^*,\ |y| \leq C|w|^k \left[\ A(w,y) = 1\ \right]$$

(y: certificate, $|y| \leq C|w|^k$: not too big)

We must somehow use A to map $w \in \{0,1\}^*$ in polynomial time to a CNF-formula φ_w such that $w \in X \iff \varphi_w$ is satisfiable.

What do we know about $X \in NP$?

There is a polynomial time algorithm $A : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ (the "verifier") and some $C > 0$ and $k \in \mathbb{N}$ such that for all $w \in \{0,1\}^*$,

$$w \in X \iff \exists y \in \{0,1\}^*,\ |y| \leq C|w|^k \left[\ A(w,y) = 1\ \right]$$

(y: certificate, $|y| \leq C|w|^k$: not too big)

We must somehow use A to map $w \in \{0,1\}^*$ in polynomial time to a CNF-formula φ_w such that $w \in X \iff \varphi_w$ is satisfiable.

IDEA: 'simulate' A using φ_w, using part of φ_w variables to encode y.

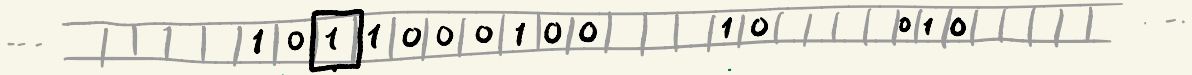
To simulate the algorithm $A : \{0,1\}^* \times \{0,1\}^* \longrightarrow \{0,1\}$, we need to be more precise about the word "algorithm".

Book (Lemma 34.6): Random Access Machines
Correct, but vague.

These slides, and ZANTEMA's lecture notes (see page 14): Turing Machines
More concrete

TURING MACHINE *

1 *infinite tape* containing 0s, 1s, and blanks, "□"s. $\Sigma := \{0, 1, \square\}$.



2 *head* ↗

3 a finite set Q of *states*, one of which is the current state

4 an *initial state* $q_0 \in Q$ and a partial *transition function*:

$$\delta : Q \times \Sigma \dashrightarrow Q \times \Sigma \times \{\leftarrow, \rightarrow\}$$

```
let q = q0
while let (q', b, d) = δ(q, symbol at the head)
      write the symbol b at the head
      move the head in direction d
      let q = q'
else "halt"
```

* There are many different definitions of Turing Machine that achieve the same result. We use one that is convenient for the proof.

In the lecture notes, the tape is infinite only in one direction, and more symbols are allowed on the tape.

EXAMPLE: DOUBLING

OBJECTIVE

TURING MACHINE

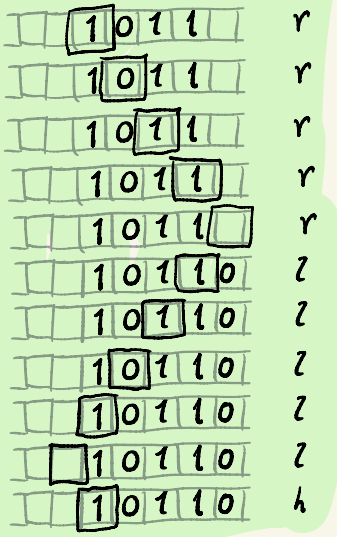
$Q = \{r, l, h\}$

$q_0 := r$

transitions

state	read	new state	write	move
r	0	r	0	→
r	1	r	1	→
r	□	l	0	←
l	0	l	0	←
l	1	l	1	←
l	□	h	□	→

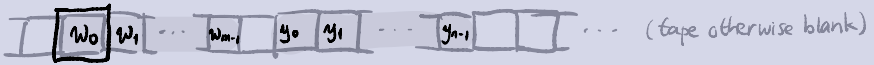
COMPUTATION



TURING MACHINE* FOR

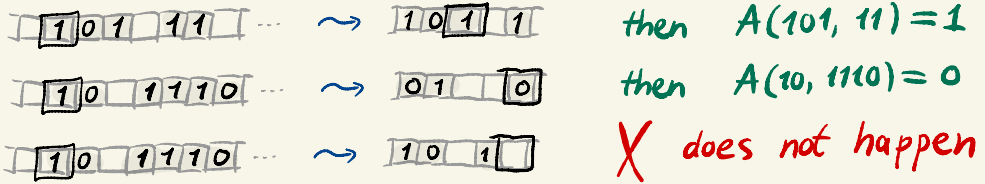
$A : \{0,1\}^* \times \{0,1\}^* \longrightarrow \{0,1\}$

Given $y \equiv y_0 y_1 \cdots y_{n-1}$ and $w \equiv w_0 w_1 \cdots w_{m-1}$ from $\{0,1\}^*$, our Turing Machine for A does, when started in configuration



halt in a configuration with $A(w,y)$ under its head:

EXAMPLES:



* Again, multiple definitions exist.

In the lecture notes, a special marking symbol is used to separate the inputs, and a special state is used to indicate $A(w,y) = 1$.

SITUATION

- Given $X \in \mathbf{NP}$, we must map $w \in \{0,1\}^*$ in polynomial time to a CNF-formula φ_w with: φ_w satisfiable $\iff$ $w \in X$.

- Since $X \in \mathbf{NP}$, there is a Turing Machine $(Q, q_0, \delta, \dots)$ computing a map $A : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in polynomial time such that

$$X = \{ w \in \{0,1\}^* \mid \exists y \in \{0,1\}^*,\ |y| \leq c|w|^k [\ A(w,y) = 1\ ] \},$$

for some $c > 0$, $k \in \mathbb{N}$.

- We'll define φ_w such that:

$$\varphi_w \text{ satisfiable} \iff \exists y \in \{0,1\}^*,\ |y| < c|w|^k\ [\ A(w,y) = 1\ ]$$

MAIN DIFFICULTY: how to deal with the infinite tape?

DEFINITION OF φ_w: BOUNDS

Let $w \in \{0,1\}^*$ be given. Since our Turing Machine for A has polynomial runtime, there are $D > 0$ and $\ell \in \mathbb{N}$ with

$$T_A(n) \leq D n^\ell$$

↖ worst-case runtime of our TM on an input of size n

The input to A will be w and a certificate $y \in \{0,1\}^*$ with size

$$|y| \leq C|w|^k,$$

so the T.M. will take at most $N := D(|w| + C|w|^k)^\ell$ steps before halting, and its head will move at most N positions so any cell on the tape beyond that will remain untouched by our T.M., and need not be simulated.

16:25

Definition of φ_w: Atoms

The atoms (= variables) of our formula φ_w are:

ATOM	INTENDED MEANING
$T^t_{x,s}$	at step t of the computation, the symbol s is on the tape at position x.
H^t_x	at step t of the computation, the head is at position x.
S^t_q	at step t of the computation, the machine is in state q.

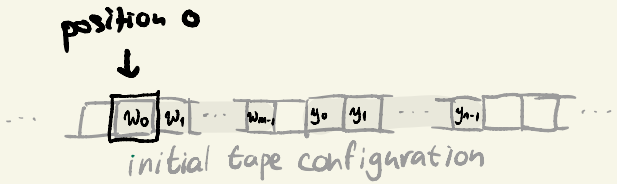
where $t \in \{0, \dots, N\}$, $x \in \{-N, \dots, 0, \dots, N\}$, $s \in \{0, 1, \square\} = \Sigma$, $q \in Q$.

Note that the number of atoms is polynomial in N, and thus in $|w|$.

Definition of φ_w: Clauses

φ_w is the conjunction of the following clauses:

- $H_0^0 \wedge S_{q_0}^0$ — the T.M. starts in state q_0 with the head at position 0, ...
- $\bigwedge_{x=-N}^{-1} T_{x,\square}^0$ — the tape blank to the left of the head, ...
- $\bigwedge_{x=0}^{|w|-1} T_{x,w_i}^0$ — and w written to the right, ...
- $T_{|w|,\square}^0$ — followed by a blank, ...



- $\bigwedge_{x=|w|+1}^{N} \left(T_{x,0}^0 \vee T_{x,1}^0 \vee T_{x,\square}^0\right) \wedge \left(T_{x,\square}^0 \rightarrow T_{x+1,\square}^0\right)$ — followed by a string of 0 and 1s, the certificate y, and then only $\square$s (blanks), ...
- $\bigwedge_{x=|w|+1+C|w|^k}^{N} T_{x,\square}^0$ — where $|y| \leq C|w|^k$.
- (next page)

$A \rightarrow B \equiv \neg A \vee B$

Note that we do not fix any value for y, but only restrict its length. Here we tap into the difficulty of solving CNF.

Definition of φ_w: Clauses. II

Proof that CNF is NP-hard, continued

$\neg(S_q^t \wedge S_{q'}^t) \equiv \neg S_q^t \vee \neg S_{q'}^t$

- $\bigwedge_{t=0}^{N} \bigwedge_{\substack{q,q' \in Q \\ q \neq q'}} \neg S_q^t \vee \neg S_{q'}^t$

  The T.M. is not at two states at once.

- $\bigwedge_{t=0}^{N} \bigwedge_{\substack{x,y=-N \\ x \neq y}}^{N} \neg H_x^t \vee \neg H_y^t$

  The head is not in two positions at the same time.

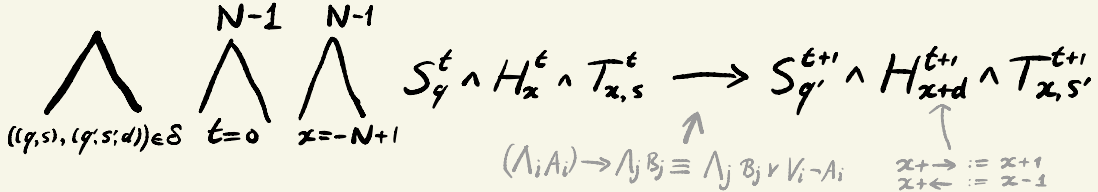
- $\bigwedge_{t=0}^{N} \bigwedge_{x=-N}^{N} \bigwedge_{\substack{s,s' \in \Sigma \\ s \neq s'}} \neg T_{x,s}^t \vee \neg T_{x,s'}^t$

  There is at most one symbol on each position of the tape.

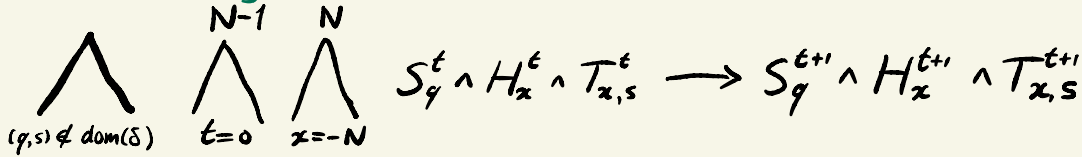
- (next page)

Definition of φ_w: Clauses, III

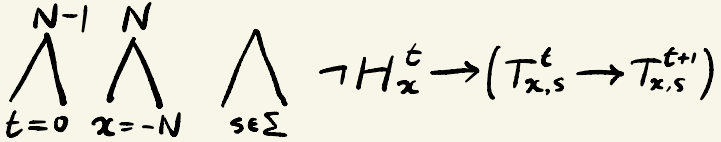
- The state changes, the head moves, and the tape is modified according to the instructions in the transition function δ:



- When the Turing Machine has halted, nothing happens:



- The tape remains otherwise unchanged:



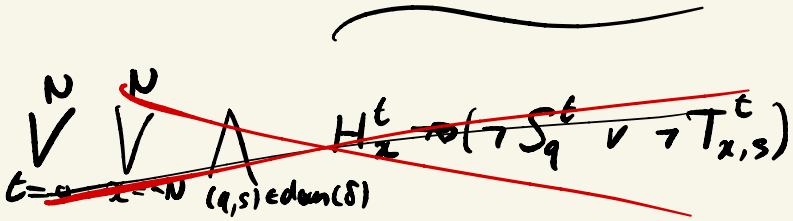
- (next page)

Definition of φ_w: Clauses, IV

Proof that CNF is NP-hard, continued

and, finally:

- $\bigwedge_{x=-N}^{N} H_x^N \to T_{x,1}^N$ At the end there is a 1 under the head.



Definition of φ_w: Clauses, IV

and, finally:

- $\bigwedge_{x=-N}^{N} H_x^t \to T_{x,1}^t$ At the end there is a 1 under the head.

Then: φ_w is satisfiable iff $\exists y \in \{0,1\}^*, |y| \leq c|w|^k [A(w,y)=1]$

iff $w \in X$.

Definition of φ_w: Clauses, IV

and, finally:

- $\bigwedge_{x=-N}^{N} H_x^t \to T_{x,1}^t$ At the end there is a 1 under the head.

Then: φ_w is satisfiable iff $\exists y \in \{0,1\}^*,\ |y| \le C|w|^k [\, A(w,y) = 1 \,]$

iff $w \in X$.

Thus $w \mapsto \varphi_w$ gives a reduction from X to CNF, and CNF is NP-hard, right?

Complexity of φ_w

Almost... it remains to be shown that φ_w can be computed in polynomial time.

To this end, note that φ_w is a conjunction of

- $\Theta(N^2)$ clauses, each of which is a disjunction of
- $\Theta(1)$ literals, each of which contains some of the
- $\Theta(N^2)$ atoms,

so φ_w can clearly be encoded with a number of characters that is polynomial in N, and thus in $|w|$ (since $N := D(|w| + C|w|^k)^e$.)

Since computing these clauses requires no additional complex computation, φ_w can be computed in polynomial time.

Whence CNF is NP-hard. □

SAT solvers

Although SAT is **NP**-complete,

- there are powerful tools called SAT solvers that can solve many large SAT problems (10^5 variables, 10^6 clauses) in reasonable time — just not all of them.
- When you encounter a problem with many distinct cases, it might be worth translating it to SAT, and apply a SAT solver.
- Also used in Mathematics: in 2016, Marijn HEULE e.a. solved the Boolean Pythogorean triples problem using a SAT solver, which was an open problem since the 1980s.

PROBLEM Is there $S \subseteq \mathbb{N}$ such that there are no distinct $a, b, c \in \mathbb{N}$ with $a^2 + b^2 = c^2$ and $a, b, c \in S$ or $a, b, c \in \mathbb{N} \setminus S$. SOLUTION: No.

Hints for the Exam

- Study the weekly assignments and their solutions.

- Practise:
  - solving recurrence relations (using the substitution method and Master's Theorem,)
  - and proving NP-completeness of problems,

  and you'll greatly increase your chances to pass.

- Memorize NP-complete problems from the lectures, and from the assignments:

  SAT, 3CNF, 3Color, TSP, SubsSum, Clique, VertexCover, ...

Good luck!

(and happy holidays)