NP and NP Completeness

Lecture 23
Tuesday, December 1, 2020
23.1

NP-Completeness: Cook-Levin Theorem
23.1.1 Completeness
NP: Non-deterministic polynomial

**Definition 23.1.**

A decision problem is in $\textbf{NP}$, if it has a polynomial time certifier, for all the YES instances.

**Definition 23.2.**

A decision problem is in $\textbf{co-NP}$, if it has a polynomial time certifier, for all the NO instances.

**Example 23.3.**

1. $3\text{SAT}$ is in $\textbf{NP}$.
2. But $\textbf{Not3SAT}$ is in $\textbf{co-NP}$. 
In the beginning...
In the beginning...

Undecidable
In the beginning...

Undecidable

$EXP$
In the beginning...

- Undecidable

- EXP

- PSPACE
In the beginning...
In the beginning...
In the beginning...

- Undecidable
- NP
- co-NP
- NP-Hard
- P
- PSPACE
- EXP
In the beginning...
In the beginning...
In the beginning...

Undecidable

NP – Hard

NP

NP- Hard

NPC

P

co-NP

PSPACE

EXP
“Hardest” Problems

Question
What is the hardest problem in \textbf{NP}? How do we define it?

Towards a definition
1. Hardest problem must be in \textbf{NP}.
2. Hardest problem must be at least as “difficult” as every other problem in \textbf{NP}.
NP-Complete Problems

Definition 23.4.
A problem $X$ is said to be **NP-Complete** if

1. $X \in \text{NP}$, and
2. (Hardness) For any $Y \in \text{NP}$, $Y \leq_P X$. 


Proposition 23.5.

Suppose $X$ is NP-Complete. Then $X$ can be solved in polynomial time $\iff P = NP$.

Proof.

$\Rightarrow$ Suppose $X$ can be solved in polynomial time

0.1 Let $Y \in NP$. We know $Y \leq_P X$.
0.2 We showed that if $Y \leq_P X$ and $X$ can be solved in polynomial time, then $Y$ can be solved in polynomial time.
0.3 Thus, every problem $Y \in NP$ is such that $Y \in P$.
0.4 $\iff NP \subseteq P$.
0.5 Since $P \subseteq NP$, we have $P = NP$.

$\Leftarrow$ Since $P = NP$, and $X \in NP$, we have a polynomial time algorithm for $X$. $\square$
Definition 23.6.

A problem $X$ is said to be **NP-Hard** if

1. (Hardness) For any $Y \in \text{NP}$, we have that $Y \leq_p X$.

An **NP-Hard** problem need not be in **NP**!

Example: Halting problem is **NP-Hard** (why?) but not **NP-Complete**.
Consequences of proving NP-Completeness

If \( X \) is NP-Complete

1. Since we believe \( P \neq NP \),
2. and solving \( X \) implies \( P = NP \).

\( X \) is unlikely to be efficiently solvable.

At the very least, many smart people before you have failed to find an efficient algorithm for \( X \).
(This is proof by mob opinion — take with a grain of salt.)
Consequences of proving **NP-Completeness**

If $X$ is **NP-Complete**

1. Since we believe $P \neq NP$,
2. and solving $X$ implies $P = NP$.

$X$ is unlikely to be efficiently solvable.

At the very least, many smart people before you have failed to find an efficient algorithm for $X$.

(This is proof by mob opinion — take with a grain of salt.)
Consequences of proving NP-Completeness

If $X$ is NP-Complete

1. Since we believe $P \neq NP$,
2. and solving $X$ implies $P = NP$.

$X$ is unlikely to be efficiently solvable.

At the very least, many smart people before you have failed to find an efficient algorithm for $X$.

(This is proof by mob opinion — take with a grain of salt.)
Consequences of proving NP-Completeness

If $X$ is NP-Complete

1. Since we believe $P \neq NP$,
2. and solving $X$ implies $P = NP$.

$X$ is unlikely to be efficiently solvable.

At the very least, many smart people before you have failed to find an efficient algorithm for $X$.
(This is proof by mob opinion — take with a grain of salt.)
THE END

...  

(for now)