Lecture 14

- More on the class NP
- The class CoNP and NP vs CoNP
- Poly-time reducibility
- NP-complete and NP-complete languages

**Important languages:**

- $\text{HAMPATH} = \{< G, s, t > | G \text{ has a Hamiltonian path from } s \text{ to } t \}$
- $\text{HAMCYCLE} = \{< G > | G \text{ has a cycle that visits every vertex exactly once} \}$
- $\text{COMPOSITE} = \{< x > | x \text{ is a composite integer} \}$
- $\text{SAT} = \{< \phi > | \phi \text{ is a satisfiable Boolean formula} \}$
- $\text{3SAT} = \{< \phi > | \phi \text{ is a satisfiable Boolean formula in 3CNF} \}$
- $\text{TAUTOLOGY} = \{< \phi > | \phi \text{ is a Boolean formula if every assignment} \}$
- $\text{CLIQUE} = \{< G, k > | G \text{ has a clique of size } k \text{ (fully connected subgraph)} \}$
- $\text{Independent Set} = \{< G, k > | G \text{ has a subgraph of size } k \text{ such that no two vertices in the subgraph are connected} \}$
- $\text{Vertex Cover} = \{< G, k > | G \text{ has a subgraph of size } k \text{ such that all edges in } E(G) \text{ are either adjacent to at least one vertex in the subgraph} \}$
A alternative definition of NP relies on a polynomial-time verifier:

**Definition:** [poly-time verifier] A poly-time verifier for a language \( L \) is a TM \( V \) s.t. \( L = \{ w \mid \exists x \in \mathbb{F} \text{ witness } w \text{ s.t. } V(x, w) \text{ accepts} \} \) and \( V \)'s runtime is \( O(|x|) \) on input \( x \).

**Definition:** [all NP] \( L \in \text{NP} \) if \( L \) has a poly-time verifier.

Let's construct a poly-time verifier for \( \text{EUHAMPATH} \):

"On input \( (G, s, t, w) \):

1. If \( w = (v_1, \ldots, v_n) \) that are a Hamiltonian path s.t. \( v_1 = s \) and \( v_n = t \), accept
2. Else reject."

**Analysis:** Correctness.

Suppose that \( (G, s, t) \in \text{EUHAMPATH} \). Then \( u \), the Hamiltonian path, \( \exists v \in V \) s.t. \( V(x, u) \) accepts.

Suppose that \( (G, s, t) \notin \text{EUHAMPATH} \). Then no such \( w \) exists, so \( \forall w \), \( V(x, w) \) rejects.

**Runtime:** For each \( v_i, v_{i+1} \), check if \( (v_i, v_{i+1}) \in E(G) \). \( \forall \)

**Poly-time verifier for COMPOSITES**

"On input \( (x, w) \):

1. If \( w: \{ w \cup x \text{ and } w \setminus x \text{, then accept} \}
2. Else reject."

**Analysis:** Correctness.

\( x \in \text{COMPOSITES} \) iff \( \exists \text{ witness } w \in \mathbb{F} \text{ s.t. } V(x, w) \text{ accepts} \).

**Polytime:** all basic arithmetic operations are polytime (proof is left to the reader).

**Note:** that the verifier rejecting \( (x, w) \) for some \( w \) does NOT mean \( \langle x \rangle \) is not a composite. All choices of \( w \) must lead to \( V \) rejecting \( (x, w) \).
Theorem: $NP = AHt NP$ (two-directional proof)

Proof: (i) $L \in NP \implies L \in AHt NP$

Suppose $L \in NP$. Then $\exists$ a NTHN $N$ that decides $L$ in polynomial time. Let $V$ be as follows:

"On input $(x, w)$

Simulate $N$ on $x$ by using $w$ to choose between branches of $N$'s computation.

If $N$ accepts, accept

else reject"

Analysis: correctness.

Suppose $x \in L$. Then $\exists$ a branch of computation in $N$ such that $N$ accepts

1) $\exists$ a set of choices that $N$ can non-deterministically make to accept $x$.

Let $w$ be a string that encodes these choices. Then $V(x, w)$ accepts.

Suppose $x \not\in L$. Then $N$ never accepts $\implies \exists$ a set of choices that leads $N$ to accept $x$ $\iff$ no such $w$ accepts.

polynomial time: same as $N$ up to some polynomial factors $\implies$ polynomial

(ii) $L \in AMTHNP \implies L \in NP$.

Suppose $L \in AMTHNP$. Then $\exists$ a polynomial verifier $V$ that decides $L$:

Consider $N$:

"On input $x$,

Non-deterministically choose $w$ of length $p(|x|)$

Run $V(x, w)$, accept if it accepts

reject otherwise"

It is sufficient to choose $w$ of length $p(|x|)$, as $V(x, w)$ does not have time to consider longer strings than $p(|x|)$ before the computation halts.

correctness

polynomial time
We introduce a new complexity class $\text{coNP}$:

$$\text{coNP} = \{ L \mid L^c \in \text{NP} \}$$

Note that $P \subseteq \text{NP}$ and $P \subseteq \text{coNP}$.

Another famous open problem is $\text{NP} = \text{coNP}$.

To see why this is not trivial, let's assume we have a language $L \subseteq \text{NP}$ with a polytime decider $\text{NTM } N$.

$$N \text{ on input } x$$

```
\text{accept?}
\text{if yes } x \in L
```

These branches do not know of each other.

N on input $x \notin L$ rejects if all branches reject. Could we just swap the final decision at each branch? No! because $\neg \phi$ for an $x \in L$, there may only be one accepting branch.

If we have a lot of rejecting branches and one accepting branch =) a rejecting branch at the decision gives a lot of accepting branches and one rejection => $N$ accepts $x$ although $x \notin L$.

**Polytime reducibility**

Define: $f : \Sigma^* \rightarrow \Sigma^*$ is polytime if $f$ is computable by a polytime TM.

Define: Let $A, B$ be languages. $A \leq_p B$ ("$A$ is polytime reducible to $B$") if $\exists$ a polytime computable $f$ of $A \times x$, $x \in A$ iff $f(x) \in B$.

**Example:**

(Trivial) $3 \text{SAT} \leq_p \text{SAT}$

Let $f$ be the identity function ($f(\phi) = \phi$). If $\phi$ is satisfiable and $\exists \chi \in f(\phi)$, then $\phi$ is satisfiable. Similarly for the other direction.
HAMPATH $\cup$ HAMCYCLE

Want to do: take an instance of HAMPATH $(G, s, t)$ and convert to a $(G')$ with a cycle $(G', s, t) \in$ HAMPATH.

Idea: add a new vertex $V_{new}$ with edges to $s, t$.

On input $(G, s, t)$,
- Compute the graph $G'$ such that:
  - $V(G') = V(G) \cup \{V_{new}\}$
  - $E(G') = E(G) \cup \{(s, V_{new}), (V_{new}, t)\}$

Output $G'$.

Analysis: Correctness.

Suppose $(G, s, t) \in$ HAMPATH. Then consider $V_{new}, V_1, V_2, \ldots, V_n, V_{new}$ such that $V_1 = s$, $V_n = t$. Then this is a cycle that visits every vertex in $G'$ once.

Suppose $2(G') \in$ HAMCYCLE. Let $V_{new}, V_1, V_2, \ldots, V_n, V_{new}$ be a cycle Hamiltonian cycle in $G'$. Then let $s = V_1$, $t = V_n$, and we have a Hamiltonian cycle in $G$.

$\square$ is showing that $(G, s, t) \in$ HAMPATH $\implies (G') \in$ HAMCYCLE

EQUIVALENT AND $(G') \in$ HAMCYCLE $\iff 2(G, s, t) \in$ HAMPATH equivalent to showing $(G)$ and $(G', s, t) \not\in$ HAMPATH $\iff (G') \not\in$ HAMCYCLE?

A: Yes! In general, you need to show one of the following point carefully.

1. $x \in A \implies f(x) \in B$ or
2. $x \in A \implies f(x) \not\in B$

and $x \in B \iff x \in A$ and $f(x) \in B$. $f(x) \not\in B$. 

$\square$ is also called being NP-hard.

NP-COMPLETENESS

Definition: $L$ is NP-COMPLETE if
1. $L \in$ NP
2. $A \in$ NP, $A \leq P L$ (this means every language in NP is in some sense easier than $A \leq L$)
Theorem: If \( A \approx P \) and \( B \approx P \), then \( A \approx P \).

Proof: If \( A \approx P \) and \( B \approx P \), and let \( f \) be the reduction to decide \( A \) in polynomial time, do:

1. On input \( x \):
   a. Compute \( f(x) \).
   b. Run \( M_B \) on \( y \) accept if it accepts, otherwise reject.

This is polynomial as \( f(x) \) is polynomial, \( M_B \) is polynomial, and \( |y| \) is polynomial in \( |x| \).

Definition 2: You are \( \text{NP-complete} \) if finding a polynomial algorithm for \( L \), i.e., showing that \( L \in \text{P} \), solves the problem \( \text{P=NP} \).

Now, how do we show that any language \( L \) is \( \text{NP-complete} \)?

Thm (Cook-Levin): \( \text{SAT} \) is \( \text{NP-complete} \) \( \iff \) proof next lecture!

Thm: If \( A \approx P \) and \( A \) is \( \text{NP-complete} \), and \( B \in \text{NP} \), then \( B \) is also \( \text{NP-complete} \).

Proof: Let \( L \) be a language \( \in \text{NP} \).
   - Let \( f_1 \) be the reduction \( L \leq A \) b/c \( A \) is \( \text{NP-complete} \).
   - Let \( f_2 \) be the reduction \( A \leq P \).
   - Consider \( f(x) = f_2(f_1(x)) \). \( f \) is polynomial from \( L \) to \( B \).
   - \( f(x) \in B \) \( \iff \) \( f_1(x) \in A \) \( \iff \) \( x \in L \).

On the next HW (5), \( \text{SAT} \leq \text{SAT} \), and we will show \( 3\text{SAT} \in \text{NP} \). \( 3\text{SAT} \) is \( \text{NP-complete} \).

In the book: \( 3\text{SAT} \leq \text{CLIQUE} \).