Review for CMSC 452 Midterm: P and NP

Our Goals for Complexity Theory

We want to prove that

Our Goals for Complexity Theory

We want to prove that

1. Some languages L have **a fast program to decide them**

Our Goals for Complexity Theory

We want to prove that

1. Some languages L have **a fast program to decide them**
2. (Spoiler Alert: $L \in \mathrm{P}$.)

Our Goals for Complexity Theory

We want to prove that

1. Some languages L have **a fast program to decide them**
2. (Spoiler Alert: $L \in \mathrm{P}$.)
3. Some languages L **unlikely to have a fast program to decide them**

Our Goals for Complexity Theory

We want to prove that

1. Some languages L have **a fast program to decide them**
2. (Spoiler Alert: $L \in \mathrm{P}$.)
3. Some languages L **unlikely to have a fast program to decide them**
4. (Spoiler Alert: L is NP-complete.)

Our Goals for Complexity Theory

We want to prove that

1. Some languages L have **a fast program to decide them**
2. (Spoiler Alert: $L \in \mathrm{P}$.)
3. Some languages L **unlikely to have a fast program to decide them**
4. (Spoiler Alert: L is NP-complete.)

We first look at some problems of interest.

Sample Problems

How hard are the following problems:

Sample Problems

How hard are the following problems:

1. **SAT** and its variants.

Sample Problems

How hard are the following problems:

1. **SAT** and its variants.
2. **HAM** Given a graph G does it have a Ham Cycle? (A cycle that has every vertex exactly once.)

Sample Problems

How hard are the following problems:

1. **SAT** and its variants.
2. **HAM** Given a graph G does it have a Ham Cycle? (A cycle that has every vertex exactly once.)
3. **EUL** Given a graph G does it have a Euler Cycle? (A cycle that has every edge exactly once.)

Sample Problems

How hard are the following problems:

1. **SAT** and its variants.
2. **HAM** Given a graph G does it have a Ham Cycle? (A cycle that has every vertex exactly once.)
3. **EUL** Given a graph G does it have a Euler Cycle? (A cycle that has every edge exactly once.)
4. **CLIQ** Given G and k, is there a set of k vertices that all know each other?

Sample Problems

How hard are the following problems:

1. **SAT** and its variants.
2. **HAM** Given a graph G does it have a Ham Cycle? (A cycle that has every vertex exactly once.)
3. **EUL** Given a graph G does it have a Euler Cycle? (A cycle that has every edge exactly once.)
4. **CLIQ** Given G and k, is there a set of k vertices that all know each other?

To even ask these questions we need (1) a standard way to describe sets and a (2) model of computation.

Representing Elements of Sets

All elements (graphs, formulas, pairs of graphs and numbers) are represented by binary strings.

Representing Elements of Sets

All elements (graphs, formulas, pairs of graphs and numbers) are represented by binary strings.

The time it takes to determine if $x \in A$ is a function of $|x|$, the length of x.

Representing Elements of Sets

All elements (graphs, formulas, pairs of graphs and numbers) are represented by binary strings.

The time it takes to determine if $x \in A$ is a function of $|x|$, the length of x.

We Sometimes Cheat We may take the length of a formula to be the number of vars. We may take the length of a graph to be the number of vertices. These notions of length are poly-related to the actual length and hence is fine for our purposes.

Turing Machines Def

We will **not** define *Turing Machine* until we need to (after midterm).

Here is all you need to know:

Turing Machines Def

We will **not** define *Turing Machine* until we need to (after midterm).

Here is all you need to know:

1. Everything computable is computable by a Turing machine.

Turing Machines Def

We will **not** define *Turing Machine* until we need to (after midterm).

Here is all you need to know:

1. Everything computable is computable by a Turing machine.
2. Turing machines compute with discrete steps so one can talk about how many steps a computation takes.

Turing Machines Def

We will **not** define *Turing Machine* until we need to (after midterm).

Here is all you need to know:

1. Everything computable is computable by a Turing machine.
2. Turing machines compute with discrete steps so one can talk about how many steps a computation takes.
3. There are many different models of computation. They are all equivalent to Turing machines. And better- they are all equivalent within poly time.

Polynomial Time and Other Classes

Def

Polynomial Time and Other Classes

Def

1. $\mathrm{P} = \mathrm{DTIME}(n^{O(1)})$.

Polynomial Time and Other Classes

Def

1. $\mathrm{P} = \mathrm{DTIME}(n^{O(1)})$.
2. $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$.

Polynomial Time and Other Classes

Def

1. $\mathrm{P} = \mathrm{DTIME}(n^{O(1)})$.
2. $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$.
3. PF is the set of a **functions** computable in poly time.

Polynomial Time and Other Classes

Def

1. $\mathrm{P} = \mathrm{DTIME}(n^{O(1)})$.
2. $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$.
3. PF is the set of a **functions** computable in poly time.

These definitions are model independent.

3SAT, HAM, EUL, CLIQ, 3COL All Walk into a Bar

We rewrite 3SAT, HAM, EUL.

$\mathrm{3SAT}, \mathrm{HAM}, \mathrm{EUL}, \mathrm{CLIQ}, \mathrm{3COL}$ All Walk into a Bar

We rewrite 3SAT, HAM, EUL.

$$\mathrm{3SAT} = \{\phi : (\exists \vec{b})[\phi(\vec{b}) = T]\}$$

3SAT, HAM, EUL, CLIQ, 3COL All Walk into a Bar

We rewrite 3SAT, HAM, EUL.

$$\mathrm{3SAT} = \{\phi : (\exists \vec{b})[\phi(\vec{b}) = T]\}$$

$$\mathrm{HAM} = \{G : (\exists v_1, \ldots, v_n)[v_1, \ldots, v_n \text{ is a Ham Cycle}]\}.$$

$\mathbf{3SAT}, \mathbf{HAM}, \mathbf{EUL}, \mathbf{CLIQ}, \mathbf{3COL}$ All Walk into a Bar

We rewrite 3SAT, HAM, EUL.

$$\mathrm{3SAT} = \{\phi : (\exists \vec{b})[\phi(\vec{b}) = T]\}$$

$$\mathrm{HAM} = \{G : (\exists v_1, \ldots, v_n)[v_1, \ldots, v_n \text{ is a Ham Cycle}]\}.$$

$$\mathrm{EUL} = \{G : (\exists v_1, \ldots, v_n)[v_1, \ldots, v_n \text{ is an Eul Cycle}]\}.$$

$\mathrm{3SAT}, \mathrm{HAM}, \mathrm{EUL}, \mathrm{CLIQ}, \mathrm{3COL}$ All Walk into a Bar

We rewrite 3SAT, HAM, EUL.

$$\mathrm{3SAT} = \{\phi : (\exists \vec{b})[\phi(\vec{b}) = T]\}$$

$$\mathrm{HAM} = \{G : (\exists v_1, \ldots, v_n)[v_1, \ldots, v_n \text{ is a Ham Cycle}]\}.$$

$$\mathrm{EUL} = \{G : (\exists v_1, \ldots, v_n)[v_1, \ldots, v_n \text{ is an Eul Cycle}]\}.$$

$$\mathrm{CLIQ} = \{(G, k) : (\exists v_1, \ldots, v_k)[v_1, \ldots, v_k \text{ are a Clique}]\}.$$

$\mathrm{3SAT}, \mathrm{HAM}, \mathrm{EUL}, \mathrm{CLIQ}, \mathrm{3COL}$ All Walk into a Bar

We rewrite 3SAT, HAM, EUL.

$$\mathrm{3SAT} = \{\phi : (\exists \vec{b})[\phi(\vec{b}) = T]\}$$

$$\mathrm{HAM} = \{G : (\exists v_1, \ldots, v_n)[v_1, \ldots, v_n \text{ is a Ham Cycle}]\}.$$

$$\mathrm{EUL} = \{G : (\exists v_1, \ldots, v_n)[v_1, \ldots, v_n \text{ is an Eul Cycle}]\}.$$

$$\mathrm{CLIQ} = \{(G, k) : (\exists v_1, \ldots, v_k)[v_1, \ldots, v_k \text{ are a Clique}]\}.$$

For the above sets: If x is a member then there is a short verifiable witness of this.

NP

Def A is in NP if there exists a set $B \in \mathrm{P}$ and a polynomial p such that

$$A = \{x : (\exists y)[|y| = p(|x|) \land (x, y) \in B]\}.$$

NP

Def A is in NP if there exists a set $B \in \mathrm{P}$ and a polynomial p such that

$$A = \{x : (\exists y)[|y| = p(|x|) \wedge (x, y) \in B]\}.$$

Intuition. Let $A \in \mathrm{NP}$.

NP

Def A is in NP if there exists a set $B \in \mathrm{P}$ and a polynomial p such that

$$A = \{x : (\exists y)[|y| = p(|x|) \wedge (x, y) \in B]\}.$$

Intuition. Let $A \in \mathrm{NP}$.

- If $x \in A$ then there is a SHORT (poly in $|x|$) proof of this fact, namely y, such that x can be VERIFIED in poly time.

NP

Def A is in NP if there exists a set $B \in \mathrm{P}$ and a polynomial p such that

$$A = \{x : (\exists y)[|y| = p(|x|) \wedge (x, y) \in B]\}.$$

Intuition. Let $A \in \mathrm{NP}$.

- If $x \in A$ then there is a SHORT (poly in $|x|$) proof of this fact, namely y, such that x can be VERIFIED in poly time.
- So if I wanted to convince you that $x \in A$, I could give you y. You can verify $(x, y) \in B$ easily and be convinced.

NP

Def A is in NP if there exists a set $B \in \mathrm{P}$ and a polynomial p such that

$$A = \{x : (\exists y)[|y| = p(|x|) \wedge (x, y) \in B]\}.$$

Intuition. Let $A \in \mathrm{NP}$.

- If $x \in A$ then there is a SHORT (poly in $|x|$) proof of this fact, namely y, such that x can be VERIFIED in poly time.
- So if I wanted to convince you that $x \in A$, I could give you y. You can verify $(x, y) \in B$ easily and be convinced.
- If $x \notin A$ then there is NO proof that $x \in A$.

Note $\mathrm{3SAT}$, HAM, EUL, CLIQ are all in NP.

Our Plan for NP

3SAT, HAM, EUL, CLIQ are all in NP.

Our Plan for NP

3SAT, HAM, EUL, CLIQ are all in NP.

So is

$$\mathrm{IS} = \{(G, k) : G \text{ has an Ind Set of size } k\}.$$

If IS $\in$ P then 3SAT $\in$ P: Plan

We (the slides from Stanford) gave an algorithm that does the following:

If IS $\in$ P then 3SAT $\in$ P: Plan

We (the slides from Stanford) gave an algorithm that does the following:

1. **Input** ϕ, a formula in 3-CNF form.

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

We (the slides from Stanford) gave an algorithm that does the following:

1. **Input** ϕ, a formula in 3-CNF form.
2. **Output** (G, k) such that

$$\phi \in \mathrm{3SAT} \text{ iff } (G, k) \in \mathrm{IS}.$$

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

We (the slides from Stanford) gave an algorithm that does the following:

1. **Input** ϕ, a formula in 3-CNF form.
2. **Output** (G, k) such that

$$\phi \in \mathrm{3SAT} \text{ iff } (G, k) \in \mathrm{IS}.$$

3. The algorithm runs in time $p(|\phi|)$ (p is a poly).

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

We (the slides from Stanford) gave an algorithm that does the following:

1. **Input** ϕ, a formula in 3-CNF form.
2. **Output** (G, k) such that

$$\phi \in \mathrm{3SAT} \text{ iff } (G, k) \in \mathrm{IS}.$$

3. The algorithm runs in time $p(|\phi|)$ (p is a poly).
4. Produces (G, k) where $|(G, k)| \leq q(|\phi|)$ (q is a poly).

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

We (the slides from Stanford) gave an algorithm that does the following:

1. **Input** ϕ, a formula in 3-CNF form.
2. **Output** (G, k) such that

$$\phi \in \mathrm{3SAT} \text{ iff } (G, k) \in \mathrm{IS}.$$

3. The algorithm runs in time $p(|\phi|)$ (p is a poly).
4. Produces (G, k) where $|(G, k)| \leq q(|\phi|)$ (q is a poly).

Call this algorithm **ALG**. On next slide we use **ALG** to show that $\mathrm{IS} \in \mathrm{P}$ implies $\mathrm{3SAT} \in \mathrm{P}$.

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

Assume $\mathrm{IS} \in \mathrm{P}$ via program M which runs in $r(|(G, k)|)$.

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

Assume $\mathrm{IS} \in \mathrm{P}$ via program M which runs in $r(|(G, k)|)$.

1. **Input** ϕ, a formula in 3-CNF form of length L.

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

Assume $\mathrm{IS} \in \mathrm{P}$ via program M which runs in $r(|(G,k)|)$.

1. **Input** ϕ, a formula in 3-CNF form of length L.
2. Compute **ALG** on ϕ to get (G,k). Takes time $p(|\phi|)$ and produces (G,k) where $|(G,k)| \leq q(|\phi|)$.

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

Assume $\mathrm{IS} \in \mathrm{P}$ via program M which runs in $r(|(G,k)|)$.

1. **Input** ϕ, a formula in 3-CNF form of length L.
2. Compute **ALG** on ϕ to get (G,k). Takes time $p(|\phi|)$ and produces (G,k) where $|(G,k)| \le q(|\phi|)$.
3. Run M on (G,k) (takes time $r(q(|\phi|))$). Recall that

$$\phi \in \mathrm{3SAT} \text{ iff } (G,k) \in \mathrm{IS}.$$

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

Assume $\mathrm{IS} \in \mathrm{P}$ via program M which runs in $r(|(G,k)|)$.

1. **Input** ϕ, a formula in 3-CNF form of length L.
2. Compute **ALG** on ϕ to get (G,k). Takes time $p(|\phi|)$ and produces (G,k) where $|(G,k)| \le q(|\phi|)$.
3. Run M on (G,k) (takes time $r(q(|\phi|))$). Recall that

$$\phi \in \mathrm{3SAT} \text{ iff } (G,k) \in \mathrm{IS}.$$

   So just output the output of $M(G,k)$.

If $\mathrm{IS} \in \mathrm{P}$ then $\mathrm{3SAT} \in \mathrm{P}$: Plan

Assume $\mathrm{IS} \in \mathrm{P}$ via program M which runs in $r(|(G,k)|)$.

1. **Input** ϕ, a formula in 3-CNF form of length L.
2. Compute **ALG** on ϕ to get (G,k). Takes time $p(|\phi|)$ and produces (G,k) where $|(G,k)| \le q(|\phi|)$.
3. Run M on (G,k) (takes time $r(q(|\phi|))$). Recall that

   $$\phi \in \mathrm{3SAT} \text{ iff } (G,k) \in \mathrm{IS}.$$

   So just output the output of $M(G,k)$.

This is an algorithm for $\mathrm{3SAT}$ that takes time

$$p(|\phi|) + r(q(|\phi|))$$

By the Cook-Levin Theorem Have the Converse

From the above we have
$\mathrm{IS} \in \mathrm{P}$ implies $3\mathrm{SAT} \in \mathrm{P}$.

By the Cook-Levin Theorem Have the Converse

From the above we have
$\mathrm{IS} \in \mathrm{P}$ implies $\mathrm{3SAT} \in \mathrm{P}$.

By the Cook-Levin theorem (after the midterm) we will have
$\mathrm{3SAT} \in \mathrm{P}$ implies $\mathrm{IS} \in \mathrm{P}$.

By the Cook-Levin Theorem Have the Converse

From the above we have
$\mathrm{IS} \in \mathrm{P}$ implies $\mathrm{3SAT} \in \mathrm{P}$.

By the Cook-Levin theorem (after the midterm) we will have
$\mathrm{3SAT} \in \mathrm{P}$ implies $\mathrm{IS} \in \mathrm{P}$.

Hence we will have
$\mathrm{3SAT} \in \mathrm{P}$ iff $\mathrm{IS} \in \mathrm{P}$.

By the Cook-Levin Theorem Have the Converse

From the above we have
$\mathrm{IS} \in \mathrm{P}$ implies $\mathrm{3SAT} \in \mathrm{P}$.

By the Cook-Levin theorem (after the midterm) we will have
$\mathrm{3SAT} \in \mathrm{P}$ implies $\mathrm{IS} \in \mathrm{P}$.

Hence we will have
$\mathrm{3SAT} \in \mathrm{P}$ iff $\mathrm{IS} \in \mathrm{P}$.

Much More is Known The following are all in P or all NOT in P:
HAM, $\mathrm{3SAT}$, IS, $\mathrm{3COL}$, CLIQ.

Reductions

We now generalize what we did for $3\mathrm{SAT}$ and IS.

Reductions

We now generalize what we did for $\mathrm{3SAT}$ and IS.
Def Let X, Y be sets. A **reduction** from X to Y is a polynomial-time computable function f such that

$$x \in X \text{ iff } f(x) \in Y.$$

Reductions

We now generalize what we did for $\mathrm{3SAT}$ and IS.

Def Let X, Y be sets. A **reduction** from X to Y is a polynomial-time computable function f such that

$$x \in X \text{ iff } f(x) \in Y.$$

(Example: Our function that took ϕ to (G, k).)

Reductions

We now generalize what we did for $\mathrm{3SAT}$ and IS.

Def Let X, Y be sets. A **reduction** from X to Y is a polynomial-time computable function f such that

$$x \in X \text{ iff } f(x) \in Y.$$

(Example: Our function that took ϕ to (G, k).)

We express this by writing $X \leq Y$.

Reductions

We now generalize what we did for $3\mathrm{SAT}$ and IS.
Def Let X, Y be sets. A **reduction** from X to Y is a polynomial-time computable function f such that

$$x \in X \text{ iff } f(x) \in Y.$$

(Example: Our function that took ϕ to (G, k).)
We express this by writing $X \leq Y$.

Reductions are transitive.
Lemma (HW) If $X \leq Y$ and $Y \in \mathrm{P}$ then $X \in \mathrm{P}$. (We use that if $f(n), g(n)$ are poly then $f(g(n))$ is poly.)

Reductions

We now generalize what we did for $\mathrm{3SAT}$ and IS.

Def Let X, Y be sets. A **reduction** from X to Y is a polynomial-time computable function f such that

$$x \in X \text{ iff } f(x) \in Y.$$

(Example: Our function that took ϕ to (G, k).)
We express this by writing $X \le Y$.

Reductions are transitive.

Lemma (HW) If $X \le Y$ and $Y \in \mathrm{P}$ then $X \in \mathrm{P}$. (We use that if $f(n), g(n)$ are poly then $f(g(n))$ is poly.)

Contrapositive If $X \le Y$ and $X \notin \mathrm{P}$ then $Y \notin \mathrm{P}$.

Def of NP-Complete

Def A set Y is **NP-complete (NPC)** if the following hold:

- $Y \in \mathrm{NP}$
- If $X \in \mathrm{NP}$ then $X \leq Y$.

Def of NP-Complete

Def A set Y is **NP-complete (NPC)** if the following hold:

- $Y \in \mathrm{NP}$
- If $X \in \mathrm{NP}$ then $X \leq Y$.

Easy Lemma If Y is NP-complete and $Y \in \mathrm{P}$ then $\mathrm{P} = \mathrm{NP}$.

Def of NP-Complete

Def A set Y is **NP-complete (NPC)** if the following hold:

- $Y \in \mathrm{NP}$
- If $X \in \mathrm{NP}$ then $X \leq Y$.

Easy Lemma If Y is NP-complete and $Y \in \mathrm{P}$ then $\mathrm{P} = \mathrm{NP}$.

Cook-Levin Theorem $\mathrm{3SAT}$ is NP-complete.

Def of NP-Complete

Def A set Y is **NP-complete (NPC)** if the following hold:

- $Y \in \mathrm{NP}$
- If $X \in \mathrm{NP}$ then $X \leq Y$.

Easy Lemma If Y is NP-complete and $Y \in \mathrm{P}$ then $\mathrm{P} = \mathrm{NP}$.

Cook-Levin Theorem $3\mathrm{SAT}$ is NP-complete.

Since then thousands of problems have been shown NP-complete.

SAT, HAM, CLIQ, 3COL Walk into a Bar

SAT, HAM, CLIQ, 3COL Walk into a Bar

1. SAT is NP-complete by Cook-Levin Theorem.

SAT, HAM, CLIQ, 3COL Walk into a Bar

1. SAT is NP-complete by Cook-Levin Theorem.
2. IS is NP-complete. We proved this by showing 3SAT ≤ IS.

SAT, HAM, CLIQ, 3COL Walk into a Bar

1. SAT is NP-complete by Cook-Levin Theorem.
2. IS is NP-complete. We proved this by showing $3\text{SAT} \le \text{IS}$.
3. 3COL is NP-complete. We proved this.

SAT, HAM, CLIQ, 3COL Walk into a Bar

1. SAT is NP-complete by Cook-Levin Theorem.
2. IS is NP-complete. We proved this by showing $3SAT \leq IS$.
3. 3COL is NP-complete. We proved this.
4. HAM is NP-complete. Just take my word for it.