BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

The Complexity of Problems: P and NP

Exposition by William Gasarch—U of MD

Complexity of Sets

How hard are the following problems:

Complexity of Sets

How hard are the following problems:

1. **SAT** Given a Bool fml, e.g., $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y)$ is there a sat assignment? E.g, $x = T \quad y = F \quad z = T$?

Complexity of Sets

How hard are the following problems:

1. **SAT** Given a Bool fml, e.g., $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y)$ is there a sat assignment? E.g, $x = T \quad y = F \quad z = T$?
2. **HAM** Given a graph G does it have a Ham Cycle? (A cycle that has every vertex exactly once.)

Complexity of Sets

How hard are the following problems:

1. **SAT** Given a Bool fml, e.g., $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y)$ is there a sat assignment? E.g, $x = T \quad y = F \quad z = T$?
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.)

Complexity of Sets

How hard are the following problems:

1. **SAT** Given a Bool fml, e.g., $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y)$ is there a sat assignment? E.g, $x = T \quad y = F \quad z = T$?
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?

Complexity of Sets

How hard are the following problems:

1. **SAT** Given a Bool fml, e.g., $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y)$ is there a sat assignment? E.g, $x = T \quad y = F \quad z = T$?
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.

Conventions

Conventions

1. All sets are sets of strings. E.g, we can code a graph on n vertices as an $n \times n$ adj matrix, which is a string of length n^2.

Conventions

1. All sets are sets of strings. E.g, we can code a graph on n vertices as an $n \times n$ adj matrix, which is a string of length n^2.
2. A set A is in $\mathrm{DTIME}(T(n))$ if there is an algorithm that will, on input x of length n,

Conventions

1. All sets are sets of strings. E.g, we can code a graph on n vertices as an $n \times n$ adj matrix, which is a string of length n^2.
2. A set A is in $\mathrm{DTIME}(T(n))$ if there is an algorithm that will, on input x of length n,
   - determine if $x \in A$

Conventions

1. All sets are sets of strings. E.g, we can code a graph on n vertices as an $n \times n$ adj matrix, which is a string of length n^2.
2. A set A is in $\mathrm{DTIME}(T(n))$ if there is an algorithm that will, on input x of length n,
   - determine if $x \in A$
   - terminate in $\leq T(n)$ steps.

Conventions

1. All sets are sets of strings. E.g, we can code a graph on n vertices as an $n \times n$ adj matrix, which is a string of length n^2.
2. A set A is in $\mathrm{DTIME}(T(n))$ if there is an algorithm that will, on input x of length n,
   - determine if $x \in A$
   - terminate in $\leq T(n)$ steps.
3. To define **Algorithm** we need a model of computation.

Turing Machines Def

Def A *Turing Machine* is a tuple $(Q, \Sigma, \delta, s, h)$ where

Turing Machines Def

Def A *Turing Machine* is a tuple $(Q, \Sigma, \delta, s, h)$ where

We are busy people!

Turing Machines Def

Def A *Turing Machine* is a tuple $(Q, \Sigma, \delta, s, h)$ where

We are busy people!

We are not going to bother defining Turing Machines!

Turing Machines Def

Def A *Turing Machine* is a tuple $(Q, \Sigma, \delta, s, h)$ where

We are busy people!

We are not going to bother defining Turing Machines!

Here is all you need to know:

Turing Machines Def

Def A *Turing Machine* is a tuple $(Q, \Sigma, \delta, s, h)$ where

We are busy people!

We are not going to bother defining Turing Machines!

Here is all you need to know:

1. Everything computable is computable by a Turing machine.

Turing Machines Def

Def A *Turing Machine* is a tuple $(Q, \Sigma, \delta, s, h)$ where

We are busy people!

We are not going to bother defining Turing Machines!

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

Def A *Turing Machine* is a tuple $(Q, \Sigma, \delta, s, h)$ where

We are busy people!

We are not going to bother defining Turing Machines!

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 models of computation. They are all equiv up to **poly time.** Hence **poly time** can be defined without getting into the details of a Turing machine or other models.

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

Why Polynomial Time? Reason I

Consider **SAT**.

Why Polynomial Time? Reason I

Consider **SAT**.

1. $\mathrm{SAT} \in \mathrm{EXP}$, time 2^n, by brute force.

Why Polynomial Time? Reason I

Consider **SAT**.

1. $\mathrm{SAT} \in \mathrm{EXP}$, time 2^n, by brute force.
2. If I came up with a $(1.5)^n$ algorithm that's **just brute force** with some tricks.

Why Polynomial Time? Reason I

Consider **SAT**.

1. $\mathrm{SAT} \in \mathrm{EXP}$, time 2^n, by brute force.
2. If I came up with a $(1.5)^n$ algorithm that's **just brute force** with some tricks.
3. If I came up with an n^{1000} algorithm then it's **NOT brute force**. I would have found something **very clever**. Not practical, but that cleverness can probably be exploited to get a practical algorithm.

Why Polynomial Time? Reason II

A contrast to quadratic time.

Why Polynomial Time? Reason II

A contrast to quadratic time.

1. Quadratic Time. Different models of comp yield diff notions.

Why Polynomial Time? Reason II

A contrast to quadratic time.

1. Quadratic Time. Different models of comp yield diff notions.
2. P. Different models of comp yield same P.

Why Polynomial Time? Reason II

A contrast to quadratic time.

1. Quadratic Time. Different models of comp yield diff notions.
2. P. Different models of comp yield same P.
3. Quadratic time not closed under composition: if $f(n), g(n)$ are quadratic then $f(g(n))$ is quartic, not quadratic.

Why Polynomial Time? Reason II

A contrast to quadratic time.

1. Quadratic Time. Different models of comp yield diff notions.
2. P. Different models of comp yield same P.
3. Quadratic time not closed under composition: if $f(n), g(n)$ are quadratic then $f(g(n))$ is quartic, not quadratic.
4. P is closed under composition: if $f(n), g(n)$ are poly then $f(g(n))$ is poly.

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

We rewrite these problems.

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

We rewrite these problems.

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

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

We rewrite these problems.

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

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

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

We rewrite these problems.

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

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

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

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

We rewrite these problems.

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

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

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

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

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

We rewrite these problems.

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

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

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

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

Why is this interesting?

We Look At *CLIQ*

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

We Look At *CLIQ*

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

If $(G,k) \in \mathrm{CLIQ}$ then the $(v_1,\ldots,v_k)$ is a **witness** of this.
Note $(v_1,\ldots,v_k)$ is short: length is poly in the length of (G,k).

We Look At *CLIQ*

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

If $(G,k) \in \mathrm{CLIQ}$ then the $(v_1,\ldots,v_k)$ is a **witness** of this.
Note $(v_1,\ldots,v_k)$ is short: length is poly in the length of (G,k).

Note Verifying a witness is fast:
If $(v_1,\ldots,v_k)$ is a **potential witness** then **verifying** that $(v_1,\ldots,v_k)$ is a witness is **fast**: time poly in the length of (G,k).

We Look At *CLIQ*

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

If $(G,k) \in \mathrm{CLIQ}$ then the $(v_1,\ldots,v_k)$ is a **witness** of this.
Note $(v_1,\ldots,v_k)$ is short: length is poly in the length of (G,k).

Note Verifying a witness is fast:
If $(v_1,\ldots,v_k)$ is a **potential witness** then **verifying** that $(v_1,\ldots,v_k)$ is a witness is **fast**: time poly in the length of (G,k).

SAT, HAM, EUL are similar.

NP

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

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

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

Our Plan for NP

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

Our Plan for NP

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

1. This does not mean that any of these problems are easy.

Our Plan for NP

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

1. This does not mean that any of these problems are easy.
2. This does not mean that any of these problems are hard.

Our Plan for NP

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

1. This does not mean that any of these problems are easy.
2. This does not mean that any of these problems are hard.
3. SAT, HAM, CLIQ (but NOT EUL) are **equivalent** and hence one of the following holds:

Our Plan for NP

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

1. This does not mean that any of these problems are easy.
2. This does not mean that any of these problems are hard.
3. SAT, HAM, CLIQ (but NOT EUL) are **equivalent** and hence one of the following holds:
   - SAT, HAM, CLIQ are all in P.

Our Plan for NP

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

1. This does not mean that any of these problems are easy.
2. This does not mean that any of these problems are hard.
3. SAT, HAM, CLIQ (but NOT EUL) are **equivalent** and hence one of the following holds:
    - SAT, HAM, CLIQ are all in P.
    - None of SAT, HAM, CLIQ are in P.

Reductions

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

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

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

Reductions

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

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

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

Reductions are transitive.
Easy Lemma 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

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

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

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

Reductions are transitive.
Easy Lemma 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.)

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

Def of NP-Complete

Def A set Y is **NP-complete** 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** 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** 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 $\mathrm{P} = \mathrm{NP}$.

Honesty When I first saw the definition of NP-completeness I thought (1) there are no NP-complete sets or (2) there are no natural NP-complete sets.

Def of NP-Complete

Def A set Y is **NP-complete** 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 $\mathrm{P} = \mathrm{NP}$.

Honesty When I first saw the definition of NP-completeness I thought (1) there are no NP-complete sets or (2) there are no natural NP-complete sets.

The condition:

$$\text{for EVERY } X \in \mathrm{NP},\ X \leq Y?$$

seemed very hard to meet.

Variants of SAT

We define several variants of SAT:

Variants of SAT

We define several variants of SAT:

1. SAT is the set of all boolean formulas that are satisfiable.

Variants of SAT

We define several variants of SAT:

1. SAT is the set of all boolean formulas that are satisfiable. That is, $\phi(\vec{x}) \in SAT$ if there exists a vector $\vec{b}$ such that $\phi(\vec{b}) = TRUE$.

Variants of SAT

We define several variants of SAT:

1. SAT is the set of all boolean formulas that are satisfiable. That is, $\phi(\vec{x}) \in SAT$ if there exists a vector $\vec{b}$ such that $\phi(\vec{b}) = TRUE$.
2. CNF-SAT is the set of all boolean formulas in SAT of the form $C_1 \wedge \cdots \wedge C_m$ where each C_i is an $\vee$ of literals.

Variants of SAT

We define several variants of SAT:

1. SAT is the set of all boolean formulas that are satisfiable. That is, $\phi(\vec{x}) \in SAT$ if there exists a vector $\vec{b}$ such that $\phi(\vec{b}) = TRUE$.
2. CNF-SAT is the set of all boolean formulas in SAT of the form $C_1 \wedge \cdots \wedge C_m$ where each C_i is an $\vee$ of literals.
3. $\mathrm{3SAT}$ is CNF-SAT where each clause has ≤ 3 literals.

SAT is NP-Complete

Cook (1971) and Levin (1973) independently showed:

CNF-SAT is NP-complete

SAT is NP-Complete

Cook (1971) and Levin (1973) independently showed:

CNF-SAT is $\mathbf{NP}$-complete

Thoughts on this:

SAT is NP-Complete

Cook (1971) and Levin (1973) independently showed:

CNF-SAT is NP-complete

Thoughts on this:

1. The proof is not hard, but it involves looking at actual TMs. SAT was the **first** NP-complete problem. You could not use some other problem. 3SAT was the second by an easy reduction.

SAT is NP-Complete

Cook (1971) and Levin (1973) independently showed:

CNF-SAT is NP-complete

Thoughts on this:

1. The proof is not hard, but it involves looking at actual TMs. SAT was the **first** NP-complete problem. You could not use some other problem. 3SAT was the second by an easy reduction.
2. Once we have 3SAT is NP-complete we will NEVER use Turing machines again. To show Y NP-complete: (1) $Y \in \mathrm{NP}$, (2) $A \leq Y$ for a known A that is NPC, often 3SAT.

SAT is NP-Complete

Cook (1971) and Levin (1973) independently showed:

CNF-SAT is NP-complete

Thoughts on this:

1. The proof is not hard, but it involves looking at actual TMs. SAT was the **first** NP-complete problem. You could not use some other problem. 3SAT was the second by an easy reduction.
2. Once we have 3SAT is NP-complete we will NEVER use Turing machines again. To show Y NP-complete: (1) $Y \in \mathrm{NP}$, (2) $A \le Y$ for a known A that is NPC, often 3SAT.
3. Thousands of problems are NP-complete. If any are in P then they are all in P.

SAT is NP-Complete

Cook (1971) and Levin (1973) independently showed:

CNF-SAT is NP-complete

Thoughts on this:

1. The proof is not hard, but it involves looking at actual TMs. SAT was the **first** NP-complete problem. You could not use some other problem. 3SAT was the second by an easy reduction.
2. Once we have 3SAT is NP-complete we will NEVER use Turing machines again. To show Y NP-complete: (1) $Y \in \mathrm{NP}$, (2) $A \leq Y$ for a known A that is NPC, often 3SAT.
3. Thousands of problems are NP-complete. If any are in P then they are all in P.
4. Most Computer Scientists and Mathematicians think $\mathrm{P} \neq \mathrm{NP}$.

History: HAM and EUL

1736 Euler shows the Konigsberg bridge problem is unsolvable by proving, in modern terms,
A graph is EUL iff every vertex has even degree. So $\mathrm{EUL} \in \mathrm{P}$.

History: HAM and EUL

1736 Euler shows the Konigsberg bridge problem is unsolvable by proving, in modern terms,
A graph is EUL iff every vertex has even degree. So $\mathrm{EUL} \in \mathrm{P}$.

1850? Hamilton poses, in modern terms, the question of characterizing when graphs are HAM.

History: HAM and EUL

1736 Euler shows the Konigsberg bridge problem is unsolvable by proving, in modern terms,
A graph is EUL iff every vertex has even degree. So $\mathrm{EUL} \in \mathrm{P}$.

1850? Hamilton poses, in modern terms, the question of characterizing when graphs are HAM.

Note Mathematicians wanted a **characterization of HAM graphs similar to the characterization of EUL graphs**.

History: HAM and EUL

1736 Euler shows the Konigsberg bridge problem is unsolvable by proving, in modern terms,
A graph is EUL iff every vertex has even degree. So $\mathrm{EUL} \in \mathrm{P}$.

1850? Hamilton poses, in modern terms, the question of characterizing when graphs are HAM.

Note Mathematicians wanted a **characterization of HAM graphs similar to the characterization of EUL graphs**.
They didn't have the notion of algorithms to state what they wanted more rigorously.

History: HAM and EUL

1736 Euler shows the Konigsberg bridge problem is unsolvable by proving, in modern terms,
A graph is EUL iff every vertex has even degree. So $\mathrm{EUL} \in \mathrm{P}$.

1850? Hamilton poses, in modern terms, the question of characterizing when graphs are HAM.

Note Mathematicians wanted a **characterization of HAM graphs similar to the characterization of EUL graphs**.
They didn't have the notion of algorithms to state what they wanted more rigorously.

The theory of NP-completeness enabled mathematicians to **state** what they wanted rigorously ($\mathrm{HAM} \in \mathrm{P}$) and also gave the basis for proving likely it **cannot** be done (since HAM is NP-Complete).

$\mathrm{SAT}, \mathrm{HAM}, \mathrm{CLIQ}$ Walk into a Bar

SAT, HAM, CLIQ Walk into a Bar

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

SAT, HAM, CLIQ Walk into a Bar

1. SAT is NP-complete by Cook-Levin Theorem.
2. CLIQ is NP-complete. We prove that on next few slides.

SAT, HAM, CLIQ Walk into a Bar

1. SAT is NP-complete by Cook-Levin Theorem.
2. CLIQ is NP-complete. We prove that on next few slides.
3. HAM is NP-complete. Just take my word for it.

$3SAT \leq CLIQ$

1) Input $\phi = C_1 \wedge \cdots \wedge C_k$ where each C_i is a 3-clause.

$\mathbf{3SAT} \le \mathbf{CLIQ}$

1) Input $\phi = C_1 \wedge \cdots \wedge C_k$ where each C_i is a 3-clause.

2) Graph G with $7k$ vertices as follows: For each clause we have 7 vertices. Label them with the 7 ways to set the 3 vars to make the clause satisfiable. For example, for the clause $x \vee y \vee \neg z$, we have 7 vertices: TTT, TTF, TFT, TFF, FTT, FTF, FFF.

$3SAT \le CLIQ$

1) Input $\phi = C_1 \wedge \cdots \wedge C_k$ where each C_i is a 3-clause.

2) Graph G with $7k$ vertices as follows: For each clause we have 7 vertices. Label them with the 7 ways to set the 3 vars to make the clause satisfiable. For example, for the clause $x \vee y \vee \neg z$, we have 7 vertices: TTT, TTF, TFT, TFF, FTT, FTF, FFF.

There are no edges between vertices associated to the same clause. We put an edge between vertices associated with different clauses if the assignments do not conflict. Example:
$(x = T, y = T, z = T)$ has edge to $(w = F, x = T, z = T)$ but not to $(w = F, x = F, z = T)$.

$3SAT \le CLIQ$

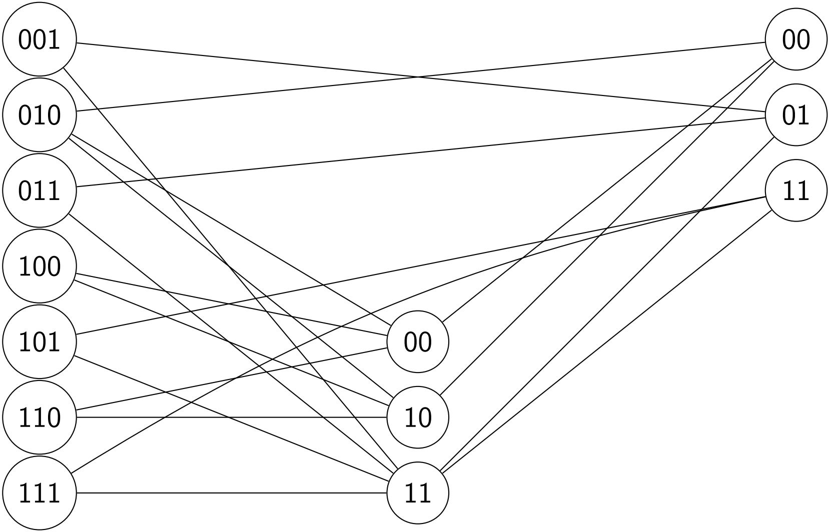
1) Input $\phi = C_1 \wedge \cdots \wedge C_k$ where each C_i is a 3-clause.

2) Graph G with $7k$ vertices as follows: For each clause we have 7 vertices. Label them with the 7 ways to set the 3 vars to make the clause satisfiable. For example, for the clause $x \vee y \vee \neg z$, we have 7 vertices: TTT, TTF, TFT, TFF, FTT, FTF, FFF.

There are no edges between vertices associated to the same clause. We put an edge between vertices associated with different clauses if the assignments do not conflict. Example:
$(x = T, y = T, z = T)$ has edge to $(w = F, x = T, z = T)$ but not to $(w = F, x = F, z = T)$.

3) Example on next slide

$$(x \vee y \vee z) \quad \wedge \quad (w \vee \overline{z}) \quad \wedge \quad (\overline{x} \vee z)$$



So What Do We Know?

So What Do We Know?

1. We **do not know** that $3\mathrm{SAT} \notin \mathrm{P}$.

So What Do We Know?

1. We **do not know** that $3SAT \notin P$.
2. We **do not know** that $CLIQ \notin P$.

So What Do We Know?

1. We **do not know** that $3SAT \notin P$.
2. We **do not know** that $CLIQ \notin P$.
3. We **do know** that $3SAT \in P$ IFF $CLIQ \in P$.

So What Do We Know?

1. We **do not know** that $\mathrm{3SAT} \notin \mathrm{P}$.
2. We **do not know** that $\mathrm{CLIQ} \notin \mathrm{P}$.
3. We **do know** that $\mathrm{3SAT} \in \mathrm{P}$ IFF $\mathrm{CLIQ} \in \mathrm{P}$.
4. We **believe** $\mathrm{3SAT} \notin \mathrm{P}$, hence we **believe** $\mathrm{CLIQ} \notin \mathrm{P}$.

Why Do We Believe $\mathrm{P} \neq \mathrm{NP}$?

Why Do We Believe $\mathrm{P} \neq \mathrm{NP}$?

1. I have done three polls of what theorists think of P vs NP. 88% of the theorists polled think $\mathrm{P} \neq \mathrm{NP}$.

Why Do We Believe $\mathrm{P} \neq \mathrm{NP}$?

1. I have done three polls of what theorists think of P vs NP. 88% of the theorists polled think $\mathrm{P} \neq \mathrm{NP}$. Some of those who voted $\mathrm{P} = \mathrm{NP}$ emailed me privately that it was a protest vote—They think $\mathrm{P} \neq \mathrm{NP}$ but they also think people should be more open minded.

Why Do We Believe $\mathrm{P} \neq \mathrm{NP}$?

1. I have done three polls of what theorists think of P vs NP. 88% of the theorists polled think $\mathrm{P} \neq \mathrm{NP}$. Some of those who voted $\mathrm{P} = \mathrm{NP}$ emailed me privately that it was a protest vote—They think $\mathrm{P} \neq \mathrm{NP}$ but they also think people should be more open minded.
2. The NP-complete problems have been worked on for a long time (many predating the definition of P and NP) and none have been shown to be in P.

Why Do We Believe $\mathrm{P} \neq \mathrm{NP}$?

1. I have done three polls of what theorists think of P vs NP. 88% of the theorists polled think $\mathrm{P} \neq \mathrm{NP}$. Some of those who voted $\mathrm{P} = \mathrm{NP}$ emailed me privately that it was a protest vote—They think $\mathrm{P} \neq \mathrm{NP}$ but they also think people should be more open minded.
2. The NP-complete problems have been worked on for a long time (many predating the definition of P and NP) and none have been shown to be in P.
3. Intuitively **coming up with a proof** seems harder than **verifying a proof**.

Why Do We Believe $\mathrm{P} \neq \mathrm{NP}$?

1. I have done three polls of what theorists think of P vs NP. 88% of the theorists polled think $\mathrm{P} \neq \mathrm{NP}$. Some of those who voted $\mathrm{P} = \mathrm{NP}$ emailed me privately that it was a protest vote—They think $\mathrm{P} \neq \mathrm{NP}$ but they also think people should be more open minded.
2. The NP-complete problems have been worked on for a long time (many predating the definition of P and NP) and none have been shown to be in P.
3. Intuitively **coming up with a proof** seems harder than **verifying a proof**.
4. $\mathrm{P} \neq \mathrm{NP}$ has great explanatory power. See next slide.

Approximating Set Cover

Set Cover Given n and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets S_i's that **cover** $\{1, \ldots, n\}$.

Approximating Set Cover

Set Cover Given n and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets S_i's that **cover** $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for **Set Cover** that will return $(\ln n) \times \mathrm{OPTIMAL}$.

Approximating Set Cover

Set Cover Given n and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets S_i's that **cover** $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for **Set Cover** that will return $(\ln n) \times \mathrm{OPTIMAL}$.
2. Dinur and Steurer in 2013 showed that, assuming $\mathrm{P} \neq \mathrm{NP}$, for all ϵ there is no $(1-\epsilon) \ln n \times \mathrm{OPTIMAL}$ approx alg for **Set Cover**.

Approximating Set Cover

Set Cover Given n and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets S_i's that **cover** $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for **Set Cover** that will return $(\ln n) \times \mathrm{OPTIMAL}$.
2. Dinur and Steurer in 2013 showed that, assuming $\mathrm{P} \neq \mathrm{NP}$, for all ϵ there is no $(1 - \epsilon) \ln n \times \mathrm{OPTIMAL}$ approx alg for **Set Cover**.
3. These two proofs have nothing to do with each other yet give matching upper and lower bounds.

Approximating Set Cover

Set Cover Given n and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets S_i's that **cover** $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for **Set Cover** that will return $(\ln n) \times \mathrm{OPTIMAL}$.
2. Dinur and Steurer in 2013 showed that, assuming $\mathrm{P} \neq \mathrm{NP}$, for all ϵ there is no $(1 - \epsilon) \ln n \times \mathrm{OPTIMAL}$ approx alg for **Set Cover**.
3. These two proofs have nothing to do with each other yet give matching upper and lower bounds.
4. There are many other approx problems where $\mathrm{P} = \mathrm{NP}$ explains why they cannot be improved.

End with some Opinions

My opinions

End with some Opinions

My opinions

1. 1.1 IF $\mathrm{P} = \mathrm{NP}$ that might be proven in the next decade.

End with some Opinions

My opinions

1. 1.1 IF $\mathrm{P} = \mathrm{NP}$ that might be proven in the next decade.
   1.2 IF $\mathrm{P} \neq \mathrm{NP}$ this will not be proven until the year 2525.

End with some Opinions

My opinions

1. 1.1 IF $\mathrm{P} = \mathrm{NP}$ that might be proven in the next decade.
   1.2 IF $\mathrm{P} \neq \mathrm{NP}$ this will not be proven until the year 2525.
2. $\mathrm{P} \neq \mathrm{NP}$. In fact, SAT requires $2^{\Omega(n)}$ time.

BILL, STOP RECORDING LECTURE!!!!

BILL STOP RECORDING LECTURE!!!