DTIME, P, EXP, and of Course NP

Exposition by William Gasarch—U of MD

Which Problems are Hard?

We want to prove that

Which Problems are Hard?

We want to prove that

1. Some problems L have **a fast program to solve them**

Which Problems are Hard?

We want to prove that

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

Which Problems are Hard?

We want to prove that

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

Which Problems are Hard?

We want to prove that

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

Which Problems are Hard?

We want to prove that

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

We first look at some problems of interest.

Problems of Interest

Exposition by William Gasarch—U of MD

Graphs

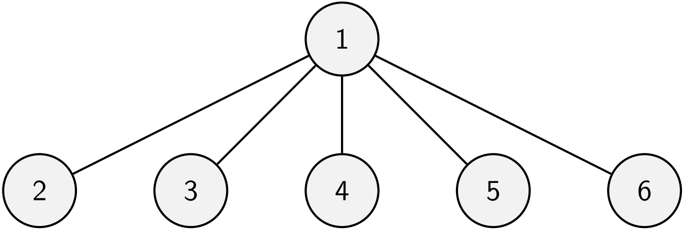
Def A **Graph** $G = (V, E)$ is a set V and a set of unordered pairs from V, called edges. These can easily be drawn.

Graphs

Def A **Graph** $G = (V, E)$ is a set V and a set of unordered pairs from V, called edges. These can easily be drawn.
Example

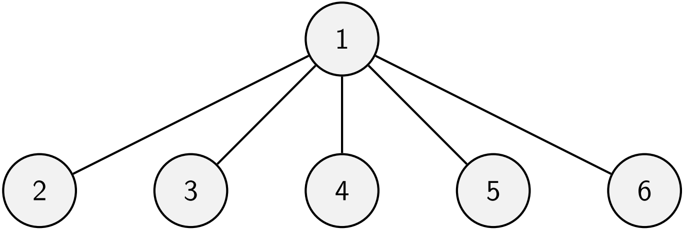
Graphs

Def A **Graph** $G = (V, E)$ is a set V and a set of unordered pairs from V, called edges. These can easily be drawn.
Example



Graphs

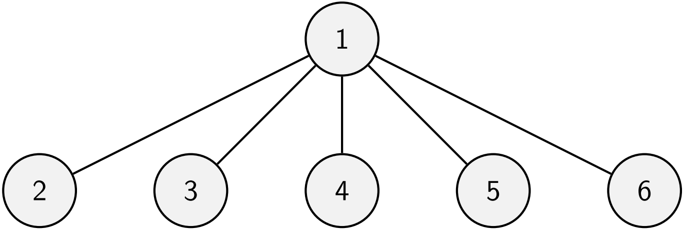
Def A **Graph** $G = (V, E)$ is a set V and a set of unordered pairs from V, called edges. These can easily be drawn.
Example



$V = \{1, 2, 3, 4, 5, 6\}$.

Graphs

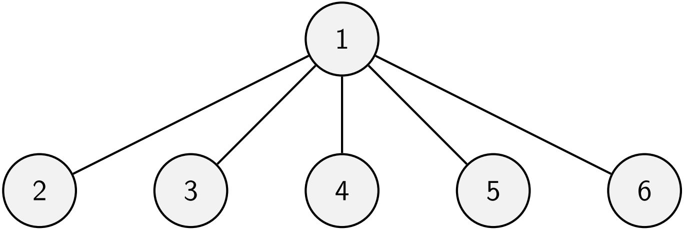
Def A **Graph** $G = (V, E)$ is a set V and a set of unordered pairs from V, called edges. These can easily be drawn.
Example



$V = \{1, 2, 3, 4, 5, 6\}$.
$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{1, 6\}\}$.

Graphs

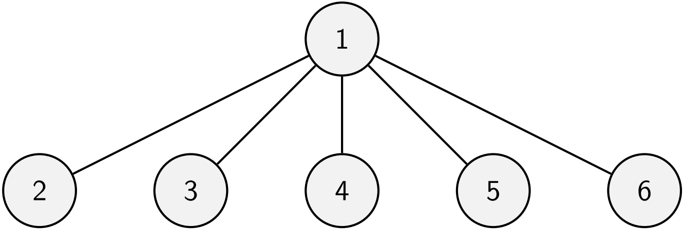
Def A **Graph** $G = (V, E)$ is a set V and a set of unordered pairs from V, called edges. These can easily be drawn.
Example



$V = \{1, 2, 3, 4, 5, 6\}$.
$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{1, 6\}\}$.
Def The **degree (deg)** of a vertex is how many edges use it.

Graphs

Def A **Graph** $G = (V, E)$ is a set V and a set of unordered pairs from V, called edges. These can easily be drawn.
Example



$V = \{1, 2, 3, 4, 5, 6\}$.
$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{1, 6\}\}$.
Def The **degree (deg)** of a vertex is how many edges use it.
In the above graph $\mathrm{degh}(1) = 5$ and
$\mathrm{degh}(2) = \mathrm{degh}(3) = \mathrm{degh}(4) = \mathrm{degh}(5) = \mathrm{degh}(6) = 1$.

Weighted Graphs

Def A weighted graph $G = (V, E)$ is a graph together with, for each edge, a natural number.

Weighted Graphs

Def A weighted graph $G = (V, E)$ is a graph together with, for each edge, a natural number.

Example V is the set of cities in America.

Weighted Graphs

Def A weighted graph $G = (V, E)$ is a graph together with, for each edge, a natural number.

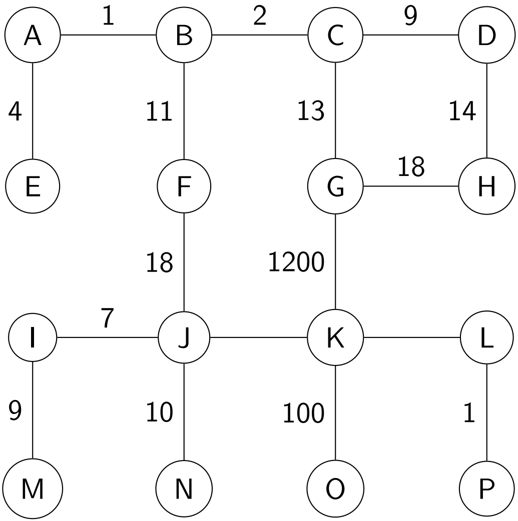
Example V is the set of cities in America.
$E = \{(x, y) : \exists$ a non-stop flight from x to $y\}$.

Weighted Graphs

Def A weighted graph $G = (V, E)$ is a graph together with, for each edge, a natural number.

Example V is the set of cities in America.
$E = \{(x, y) : \exists$ a non-stop flight from x to $y\}$.
Weight of (x, y) is price of the flight. (Cost is symmetric.)

Another Example of a Weighted Graph



Cycles in Graphs

Cycles in Graphs

Def Let $G = (V, E)$ be a graph and $k \in \mathbb{N}$.

Cycles in Graphs

Def Let $G = (V, E)$ be a graph and $k \in \mathbb{N}$.

1. A **Cycle** is a sequence of vertices $v_1, v_2, \ldots, v_m$ such that every adjacent pair has edge, and (v_m, v_1) is an edge.

Cycles in Graphs

Def Let $G = (V, E)$ be a graph and $k \in \mathbb{N}$.

1. A **Cycle** is a sequence of vertices $v_1, v_2, \ldots, v_m$ such that every adjacent pair has edge, and (v_m, v_1) is an edge.
2. An **Eulerian Cycle** uses **every** edge exactly once.

Cycles in Graphs

Def Let $G = (V, E)$ be a graph and $k \in \mathbb{N}$.

1. A **Cycle** is a sequence of vertices $v_1, v_2, \ldots, v_m$ such that every adjacent pair has edge, and (v_m, v_1) is an edge.
2. An **Eulerian Cycle** uses **every** edge exactly once.
3. A **Hamiltonian Cycle** uses **every** vertex exactly once.

Cycles in Graphs

Def Let $G = (V, E)$ be a graph and $k \in \mathbb{N}$.

1. A **Cycle** is a sequence of vertices $v_1, v_2, \ldots, v_m$ such that every adjacent pair has edge, and (v_m, v_1) is an edge.
2. An **Eulerian Cycle** uses **every** edge exactly once.
3. A **Hamiltonian Cycle** uses **every** vertex exactly once.
4. A **Clique of size** k is a set of k vertices such that **every** pair is an edge.

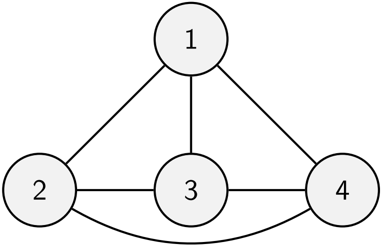
Cycles in Graphs

Def Let $G = (V, E)$ be a graph and $k \in \mathbb{N}$.

1. A **Cycle** is a sequence of vertices $v_1, v_2, \ldots, v_m$ such that every adjacent pair has edge, and (v_m, v_1) is an edge.
2. An **Eulerian Cycle** uses **every** edge exactly once.
3. A **Hamiltonian Cycle** uses **every** vertex exactly once.
4. A **Clique of size** k is a set of k vertices such that **every** pair is an edge.
5. A **Ind. Set of size** k is a set of k vertices such that **n**o pair is an edge.

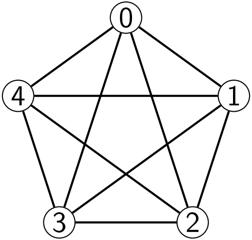
Example of a Clique on 4 Vertices

Example of a Clique on 4 Vertices



An Example of a Clique on 5 Vertices

An Example of a Clique on 5 Vertices



Problems

How hard are the following problems:

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle?

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle?
   Discuss Algorithms To Solve **EUL**.

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle? Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle? Discuss Algorithms To Solve **EUL**.
3. **CLIQ** Given (G, k), does G have a k-clique?

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle?
   Discuss Algorithms To Solve **EUL**.
3. **CLIQ** Given (G, k), does G have a k-clique?
   Discuss Algorithms To Solve **CLIQ**

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle?
   Discuss Algorithms To Solve **EUL**.
3. **CLIQ** Given (G, k), does G have a k-clique?
   Discuss Algorithms To Solve **CLIQ**
4. **IND SET (IS)** Given (G, k), does G have a k-Ind Set?

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle?
   Discuss Algorithms To Solve **EUL**.
3. **CLIQ** Given (G, k), does G have a k-clique?
   Discuss Algorithms To Solve **CLIQ**
4. **IND SET (IS)** Given (G, k), does G have a k-Ind Set?
   Discuss Algorithms To Solve **Ind Set**.

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle?
   Discuss Algorithms To Solve **EUL**.
3. **CLIQ** Given (G, k), does G have a k-clique?
   Discuss Algorithms To Solve **CLIQ**
4. **IND SET (IS)** Given (G, k), does G have a k-Ind Set?
   Discuss Algorithms To Solve **Ind Set**.

   Note $G = (V, E)$ has a k-clique off $(V, \overline{E})$ has a k-ind set.

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle?
   Discuss Algorithms To Solve **EUL**.
3. **CLIQ** Given (G, k), does G have a k-clique?
   Discuss Algorithms To Solve **CLIQ**
4. **IND SET (IS)** Given (G, k), does G have a k-Ind Set?
   Discuss Algorithms To Solve **Ind Set**.

   Note $G = (V, E)$ has a k-clique off $(V, \overline{E})$ has a k-ind set.
   So **CLIQ** has a fast alg iff **Ind Set** has a fast alg.

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle?
   Discuss Algorithms To Solve **EUL**.
3. **CLIQ** Given (G, k), does G have a k-clique?
   Discuss Algorithms To Solve **CLIQ**
4. **IND SET (IS)** Given (G, k), does G have a k-Ind Set?
   Discuss Algorithms To Solve **Ind Set**.

   Note $G = (V, E)$ has a k-clique off $(V, \overline{E})$ has a k-ind set.
   So **CLIQ** has a fast alg iff **Ind Set** has a fast alg.
5. **TSP** Given a weighted graph G and a number k, is there a Ham cycle that costs $\leq k$?

Problems

How hard are the following problems:

1. **HAM** Given a graph G does it have a Ham Cycle?
   Discuss Algorithms To Solve **HAM**.
2. **EUL** Given a graph G does it have a Euler Cycle?
   Discuss Algorithms To Solve **EUL**.
3. **CLIQ** Given (G, k), does G have a k-clique?
   Discuss Algorithms To Solve **CLIQ**
4. **IND SET (IS)** Given (G, k), does G have a k-Ind Set?
   Discuss Algorithms To Solve **Ind Set**.

   Note $G = (V, E)$ has a k-clique off $(V, \overline{E})$ has a k-ind set.
   So **CLIQ** has a fast alg iff **Ind Set** has a fast alg.
5. **TSP** Given a weighted graph G and a number k, is there a Ham cycle that costs $\leq k$?
   Discuss Algorithms To Solve **TSP**.

How Hard Are These Problems?

HAM, EUL, CLIQ, IS, TSP.

How Hard Are These Problems?

HAM, EUL, CLIQ, IS, TSP.

How hard are these problems?

How Hard Are These Problems?

HAM, EUL, CLIQ, IS, TSP.

How hard are these problems?

To even ask this question we need two things:

How Hard Are These Problems?

HAM, EUL, CLIQ, IS, TSP.

How hard are these problems?

To even ask this question we need two things:

1. A way to represent the input.

How Hard Are These Problems?

HAM, EUL, CLIQ, IS, TSP.

How hard are these problems?

To even ask this question we need two things:

1. A way to represent the input. To ask how hard **Given a graph G does it have a HAM Cycle?** is, you have to have a standard way to be **Given a graph**.

How Hard Are These Problems?

HAM, EUL, CLIQ, IS, TSP.

How hard are these problems?

To even ask this question we need two things:

1. A way to represent the input. To ask how hard **Given a graph G does it have a HAM Cycle?** is, you have to have a standard way to be **Given a graph**. Also need a notion of **length of input**.

How Hard Are These Problems?

HAM, EUL, CLIQ, IS, TSP.

How hard are these problems?

To even ask this question we need two things:

1. A way to represent the input. To ask how hard **Given a graph G does it have a HAM Cycle?** is, you have to have a standard way to be **Given a graph**. Also need a notion of **length of input**.
2. A model of Computation.

How Hard Are These Problems?

HAM, EUL, CLIQ, IS, TSP.

How hard are these problems?

To even ask this question we need two things:

1. A way to represent the input. To ask how hard **Given a graph G does it have a HAM Cycle?** is, you have to have a standard way to be **Given a graph**. Also need a notion of **length of input**.
2. A model of Computation. A statement like **EUL can be solved in time $O(n)$** needs to say what device we are computing on.

Formalizing Representation and Computation

Exposition by William Gasarch—U of MD

How to Talk About Speed

Speed for Engineers

How to Talk About Speed

Speed for Engineers

BILL How fast does this program run?

How to Talk About Speed

Speed for Engineers
BILL How fast does this program run?
ENG It usually takes 18 minutes.

How to Talk About Speed

Speed for Engineers
BILL How fast does this program run?
ENG It usually takes 18 minutes.

For the **Real World** this is a fine answer.

How to Talk About Speed

Speed for Engineers
BILL How fast does this program run?
ENG It usually takes 18 minutes.

For the **Real World** this is a fine answer.
However, we seek a more rigorous approach.

How to Talk About Speed

Speed for Engineers

BILL How fast does this program run?

ENG It usually takes 18 minutes.

For the **Real World** this is a fine answer.

However, we seek a more rigorous approach.

BILL How fast does this program run?

How to Talk About Speed

Speed for Engineers
BILL How fast does this program run?
ENG It usually takes 18 minutes.

For the **Real World** this is a fine answer.
However, we seek a more rigorous approach.
BILL How fast does this program run?
TODD On inputs of length n it takes roughly n^2 steps.

How to Talk About Speed

Speed for Engineers
BILL How fast does this program run?
ENG It usually takes 18 minutes.

For the **Real World** this is a fine answer.
However, we seek a more rigorous approach.
BILL How fast does this program run?
TODD On inputs of length n it takes roughly n^2 steps.
BILL What is **the length of the input**? What is *a step?*

How to Talk About Speed

Speed for Engineers
BILL How fast does this program run?
ENG It usually takes 18 minutes.

For the **Real World** this is a fine answer.
However, we seek a more rigorous approach.
BILL How fast does this program run?
TODD On inputs of length n it takes roughly n^2 steps.
BILL What is **the length of the input**? What is *a step?*
TODD Why ask me? The answers are on the next few slides that YOUR wrote.

How to Talk About Speed

Speed for Engineers
BILL How fast does this program run?
ENG It usually takes 18 minutes.

For the **Real World** this is a fine answer.
However, we seek a more rigorous approach.
BILL How fast does this program run?
TODD On inputs of length n it takes roughly n^2 steps.
BILL What is **the length of the input**? What is *a step?*
TODD Why ask me? The answers are on the next few slides that YOUR wrote.
BILL Good point!

Representing Elements of Sets

Representing Elements of Sets

1. The **adj matrix** of G is a an $n \times n$ matrix such that the (i, j) entry is 1 if $(i, j) \in E$ and 0 if $(i, j) \notin E$.

Representing Elements of Sets

1. The **adj matrix** of G is a an $n \times n$ matrix such that the (i,j) entry is 1 if $(i,j) \in E$ and 0 if $(i,j) \notin E$.
2. A graph is represented by an adjacency matrix. An n-node graph is an n^2-long string.

Representing Elements of Sets

1. The **adj matrix** of G is a an $n \times n$ matrix such that the (i,j) entry is 1 if $(i,j) \in E$ and 0 if $(i,j) \notin E$.
2. A graph is represented by an adjacency matrix. An n-node graph is an n^2-long string.
3. A **set of graphs** (like HAMC) is a set of strings, all of square length, all interpreted as an adjacency matrix for a graph.

Length of the Input

Def The **length of an input** is simply the length of the string that represents it.

Length of the Input

Def The **length of an input** is simply the length of the string that represents it.
We Sometimes Cheat 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.

Model of Computation: Turing Machines

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

Model of Computation: Turing Machines

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

We are busy people!

Model of Computation: Turing Machines

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 Until we Need to!

Model of Computation: Turing Machines

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 Until we Need to!
In this talk we will not need to!

Model of Computation: Turing Machines

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 Until we Need to!
In this talk we will not need to!

Here is all you need to know:

Model of Computation: Turing Machines

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 Until we Need to!
In this talk we will not need to!

Here is all you need to know:

1. Everything computable is computable by a Turing machine.

Model of Computation: Turing Machines

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 Until we Need to!
In this talk we will not need to!

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.

Polynomial Time and Exp Time

Def Let A be a set of strings.

1. M **decides** A if

Polynomial Time and Exp Time

Def Let A be a set of strings.

1. M **decides** A if
   1.1 If $x \in A$ them $M(x)$ outputs YES.

Polynomial Time and Exp Time

Def Let A be a set of strings.

1. M **decides** A if
   1.1 If $x \in A$ them $M(x)$ outputs YES.
   1.2 If $x \notin A$ them $M(x)$ outputs YES.

Polynomial Time and Exp Time

Def Let A be a set of strings.

1. M **decides** A if
   1.1 If $x \in A$ them $M(x)$ outputs YES.
   1.2 If $x \notin A$ them $M(x)$ outputs YES.
2. $A \in \mathrm{P}$ (Poly time) if there exists a poly p and a TM M such that
   (1) M decides A and,

Polynomial Time and Exp Time

Def Let A be a set of strings.

1. M **decides** A if
   1.1 If $x \in A$ them $M(x)$ outputs YES.
   1.2 If $x \notin A$ them $M(x)$ outputs YES.
2. $A \in \mathrm{P}$ (Poly time) if there exists a poly p and a TM M such that
   (1) M decides A and,
   (2) for all x, $M(x)$ takes $\leq p(|x|)$ steps.

Polynomial Time and Exp Time

Def Let A be a set of strings.

1. M **decides** A if
   1.1 If $x \in A$ them $M(x)$ outputs YES.
   1.2 If $x \notin A$ them $M(x)$ outputs YES.
2. $A \in \mathrm{P}$ (Poly time) if there exists a poly p and a TM M such that
   (1) M decides A and,
   (2) for all x, $M(x)$ takes $\leq p(|x|)$ steps.
   We take Poly Time to be our notion of Fast

Polynomial Time and Exp Time

Def Let A be a set of strings.

1. M **decides** A if
   1.1 If $x \in A$ them $M(x)$ outputs YES.
   1.2 If $x \notin A$ them $M(x)$ outputs YES.
2. $A \in \mathrm{P}$ (Poly time) if there exists a poly p and a TM M such that
   (1) M decides A and,
   (2) for all x, $M(x)$ takes $\le p(|x|)$ steps.
   We take Poly Time to be our notion of Fast
3. $A \in \mathrm{EXP}$ (Exponential time) if there exists a poly p and a TM M such that
   (1) M decides A and,
   (2) for all x, $M(x)$ takes $\le 2^{p(|x|)}$ steps.

Polynomial Time and Exp Time

Def Let A be a set of strings.

1. M **decides** A if
   1.1 If $x \in A$ them $M(x)$ outputs YES.
   1.2 If $x \notin A$ them $M(x)$ outputs YES.
2. $A \in \mathrm{P}$ (Poly time) if there exists a poly p and a TM M such that
   (1) M decides A and,
   (2) for all x, $M(x)$ takes $\le p(|x|)$ steps.
   We take Poly Time to be our notion of Fast
3. $A \in \mathrm{EXP}$ (Exponential time) if there exists a poly p and a TM M such that
   (1) M decides A and,
   (2) for all x, $M(x)$ takes $\le 2^{p(|x|)}$ steps.

The algorithms you gave for HAM, etc were EXP.

Polynomial Time and Exp Time

Def Let A be a set of strings.

1. M **decides** A if
   1.1 If $x \in A$ them $M(x)$ outputs YES.
   1.2 If $x \notin A$ them $M(x)$ outputs YES.
2. $A \in \mathrm{P}$ (Poly time) if there exists a poly p and a TM M such that
   (1) M decides A and,
   (2) for all x, $M(x)$ takes $\leq p(|x|)$ steps.
   We take Poly Time to be our notion of Fast
3. $A \in \mathrm{EXP}$ (Exponential time) if there exists a poly p and a TM M such that
   (1) M decides A and,
   (2) for all x, $M(x)$ takes $\leq 2^{p(|x|)}$ steps.

The algorithms you gave for HAM, etc were EXP.
The Question $\mathrm{HAM} \in \mathrm{P}$? The other problems in P?

Why Polynomial Time? Reason I

Why Polynomial Time? Reason I

1. $\mathrm{HAM} \in \mathrm{EXP}$, by brute force.

Why Polynomial Time? Reason I

1. $\mathrm{HAM} \in \mathrm{EXP}$, by brute force.
2. If I had a $(1.618)^n$ algorithm that's **just brute force** with some tricks.

Why Polynomial Time? Reason I

1. $\mathrm{HAM} \in \mathrm{EXP}$, by brute force.
2. If I had a $(1.618)^n$ algorithm that's **just brute force** with some tricks.
3. If I had a n^{1000} algorithm then it's **NOT brute force**. I would have found something **very clever**.

Why Polynomial Time? Reason I

1. $\mathrm{HAM} \in \mathrm{EXP}$, by brute force.
2. If I had a $(1.618)^n$ algorithm that's **just brute force** with some tricks.
3. If I had a 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.

Back to our Problems

Exposition by William Gasarch—U of MD

HAM, EUL, CLIQ All Walk into a Bar

We rewrite HAM, EUL, CLIQ.

HAM, EUL, CLIQ All Walk into a Bar

We rewrite HAM, EUL, CLIQ.

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

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

We rewrite HAM, EUL, CLIQ.

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

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

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

We rewrite HAM, EUL, CLIQ.

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

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

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

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

We rewrite HAM, EUL, CLIQ.

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

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

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

IS and TSP can also be written with a $\exists$ quantifier and something easy-to-check.

HAM, EUL, CLIQ All Walk into a Bar

We rewrite HAM, EUL, CLIQ.

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

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

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

IS and TSP can also be written with a $\exists$ quantifier and something easy-to-check.

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).

HAM, EUL, CLIQ are similar.

NP

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

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

NP

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

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

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

NP

Def $A \in \mathrm{NP}$ if there exists a set $B \in \mathrm{P}$ and a poly 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 $A \in \mathrm{NP}$ if there exists a set $B \in \mathrm{P}$ and a poly 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 $A \in \mathrm{NP}$ if there exists a set $B \in \mathrm{P}$ and a poly 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 HAM, EUL, CLIQ are all in NP.

All of Our Problems are in NP

HAM, EUL, CLIQ, IS, TSP are in NP.

All of Our Problems are in NP

HAM, EUL, CLIQ, IS, TSP are in NP.

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

All of Our Problems are in NP

HAM, EUL, CLIQ, IS, TSP are 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.

All of Our Problems are in NP

HAM, EUL, CLIQ, IS, TSP are 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.

So Why Is This Important

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 $X \le Y$.

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 (NPC)** if the following hold:

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

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:

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

seemed very hard to meet.

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 $\mathrm{P} = \mathrm{NP}$.

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.
Cook and Levin in the early 1970's showed that SAT, a problem in logic, was NPC. They coded TM's into formulas. We won't do that here.

NP-Complete Problems in Graph Theory

1. CLIQ
2. HAM
3. IS
4. TSP

NP-Complete Problems in Graph Theory

1. CLIQ
2. HAM
3. IS
4. TSP

Hence either

NP-Complete Problems in Graph Theory

1. CLIQ
2. HAM
3. IS
4. TSP

Hence either

1. CLIQ, HAM, IS, TSP are all in Poly time.
2. None of CLIQ, HAM, IS, TSP are in Poly time.

NP-Complete Problems in Graph Theory

1. CLIQ
2. HAM
3. IS
4. TSP

Hence either

1. CLIQ, HAM, IS, TSP are all in Poly time.
2. None of CLIQ, HAM, IS, TSP are in Poly time.

The good money says that None are in Poly Time.

Other Areas that Have NP-Complete Problems

There are NP-complete problems in the following areas:

Other Areas that Have NP-Complete Problems

There are NP-complete problems in the following areas:

1. Scheduling

Other Areas that Have NP-Complete Problems

There are NP-complete problems in the following areas:

1. Scheduling
2. Number Theory

Other Areas that Have NP-Complete Problems

There are NP-complete problems in the following areas:

1. Scheduling
2. Number Theory
3. Logic

Other Areas that Have NP-Complete Problems

There are NP-complete problems in the following areas:

1. Scheduling
2. Number Theory
3. Logic
4. Code Optimization

Other Areas that Have NP-Complete Problems

There are NP-complete problems in the following areas:

1. Scheduling
2. Number Theory
3. Logic
4. Code Optimization
5. Operations Research

Other Areas that Have NP-Complete Problems

There are NP-complete problems in the following areas:

1. Scheduling
2. Number Theory
3. Logic
4. Code Optimization
5. Operations Research
6. Formal Lang Theory

Other Areas that Have NP-Complete Problems

There are NP-complete problems in the following areas:

1. Scheduling
2. Number Theory
3. Logic
4. Code Optimization
5. Operations Research
6. Formal Lang Theory
7. Games and Puzzles

Other Areas that Have NP-Complete Problems

There are NP-complete problems in the following areas:

1. Scheduling
2. Number Theory
3. Logic
4. Code Optimization
5. Operations Research
6. Formal Lang Theory
7. Games and Puzzles
8. Others

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).

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

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

1. 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. 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.
2. Intuitively **coming up with a proof** seems harder than **verifying a proof**.

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

1. 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.
2. Intuitively **coming up with a proof** seems harder than **verifying a proof**.
3. $\mathrm{P} \neq \mathrm{NP}$ has great explanatory power. See next slide.

Example of Set Cover

Set Cover Example of the problem

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X : \text{x is the sum of two primes }\}$

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X : \text{x is the sum of two primes }\}$

$S_2 = \{x \in X : x \equiv 0 \pmod 5\}$

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X : \text{x is the sum of two primes }\}$

$S_2 = \{x \in X : x \equiv 0 \pmod 5\}$

$S_3 = \{x \in X : x \text{ is a square}\}$

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X : \text{x is the sum of two primes }\}$

$S_2 = \{x \in X : x \equiv 0 \pmod 5\}$

$S_3 = \{x \in X : x \text{ is a square}\}$

$S_4 = \{x \in X : x \text{ is a Fib Number}\}$

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X : \text{x is the sum of two primes }\}$

$S_2 = \{x \in X : x \equiv 0 \pmod 5)\}$

$S_3 = \{x \in X : x \text{ is a square}\}$

$S_4 = \{x \in X : x \text{ is a Fib Number}\}$

$S_5 = \{0, 1, 2, 3, 4, 5\}$

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X : \text{x is the sum of two primes }\}$

$S_2 = \{x \in X : x \equiv 0 \pmod 5\}$

$S_3 = \{x \in X : x \text{ is a square}\}$

$S_4 = \{x \in X : x \text{ is a Fib Number}\}$

$S_5 = \{0, 1, 2, 3, 4, 5\}$
$S_6 = \{6, 7, 8\}$

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X : \text{x is the sum of two primes }\}$

$S_2 = \{x \in X : x \equiv 0 \pmod 5\}$

$S_3 = \{x \in X : x \text{ is a square}\}$

$S_4 = \{x \in X : x \text{ is a Fib Number}\}$

$S_5 = \{0, 1, 2, 3, 4, 5\}$
$S_6 = \{6, 7, 8\}$
$S_7 = \{7, 8, 9\}$
$\vdots$

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X : \text{x is the sum of two primes }\}$

$S_2 = \{x \in X : x \equiv 0 \pmod 5\}$

$S_3 = \{x \in X : x \text{ is a square}\}$

$S_4 = \{x \in X : x \text{ is a Fib Number}\}$

$S_5 = \{0, 1, 2, 3, 4, 5\}$
$S_6 = \{6, 7, 8\}$
$S_7 = \{7, 8, 9\}$
$\vdots$
$S_{998} = \{998, 999, 1000\}$.

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X : \text{x is the sum of two primes }\}$

$S_2 = \{x \in X : x \equiv 0 \pmod 5\}$

$S_3 = \{x \in X : x \text{ is a square}\}$

$S_4 = \{x \in X : x \text{ is a Fib Number}\}$

$S_5 = \{0, 1, 2, 3, 4, 5\}$
$S_6 = \{6, 7, 8\}$
$S_7 = \{7, 8, 9\}$
$\vdots$
$S_{998} = \{998, 999, 1000\}$.
What is the LEAST number of S_i's whose UNION covers $\{1, \ldots, 1000\}$.

Example of Set Cover

Set Cover Example of the problem

The underlying set is $X = \{1, \ldots, 1000\}$.

$S_1 = \{x \in X :$ x is the sum of two primes $\}$

$S_2 = \{x \in X : x \equiv 0 \pmod 5\}$

$S_3 = \{x \in X : x$ is a square$\}$

$S_4 = \{x \in X : x$ is a Fib Number$\}$

$S_5 = \{0, 1, 2, 3, 4, 5\}$
$S_6 = \{6, 7, 8\}$
$S_7 = \{7, 8, 9\}$
$\vdots$
$S_{998} = \{998, 999, 1000\}$.
What is the LEAST number of S_i's whose UNION covers $\{1, \ldots, 1000\}$.
I actually do not know.

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**. (This was the last in a series of 7 papers, by different authors, that went from 1994 until 2014.)

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**. (This was the last in a series of 7 papers, by different authors, that went from 1994 until 2014.)
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**. (This was the last in a series of 7 papers, by different authors, that went from 1994 until 2014.)
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 $\mathrm{P} \neq \mathrm{NP}$ explains why they cannot be improved.

My Opinions

My opinions

My Opinions

My opinions

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

My 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.

My 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.

What Do Theorists Think of P vs NP?

What Do Theorists Think of P vs NP?

I have done three polls of what theorists think of P vs NP and other issues.

What Do Theorists Think of P vs NP?

I have done three polls of what theorists think of P vs NP and other issues.

	$P \neq NP$	$P = NP$	Ind	DK	other
2002	61 (61%)	9 (9%)	4 (4%)	22 (22%)	7 (7%))
2012	126 (83%)	12 (9%)	5 (3%)	1 (0.66%)	8 (5.1%)
2019	109 (88%)	15 (12%)	0	0	0