HW07 Solution

PROMISE-SAT PROBLEM

Let **PROMISE-SAT** be the following problem:

PROMISE-SAT PROBLEM

Let **PROMISE-SAT** be the following problem:

Input A boolean formula $\phi(x_1, \ldots, x_n)$ that you are PROMISED has ≥ 1 satisfying assignment.

PROMISE-SAT PROBLEM

Let **PROMISE-SAT** be the following problem:

Input A boolean formula $\phi(x_1, \ldots, x_n)$ that you are PROMISED has ≥ 1 satisfying assignment.
Output YES if ϕ has ≥ 2 satisfying assignments and NO if it has exactly 1 satisfying assignment. (Because of the PROMISE ϕ cannot have 0 satisfying assignments.)

PROMISE-SAT PROBLEM

Let **PROMISE-SAT** be the following problem:

Input A boolean formula $\phi(x_1, \ldots, x_n)$ that you are PROMISED has ≥ 1 satisfying assignment.
Output YES if ϕ has ≥ 2 satisfying assignments and NO if it has exactly 1 satisfying assignment. (Because of the PROMISE ϕ cannot have 0 satisfying assignments.)

Prove **PROMISE-SAT** in P $\rightarrow$ SAT in P.

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

1. Input $\psi(x_1, \ldots, x_n)$.

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

1. Input $\psi(x_1, \ldots, x_n)$. Plug in $(T, \ldots, T)$.

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

1. Input $\psi(x_1, \ldots, x_n)$. Plug in $(T, \ldots, T)$.
   If returns T then output YES.

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

1. Input $\psi(x_1, \ldots, x_n)$. Plug in $(T, \ldots, T)$.
   If returns T then output YES.
2. (If here then $\psi(T, \ldots, T) = F$).

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

1. Input $\psi(x_1, \ldots, x_n)$. Plug in $(T, \ldots, T)$.
   If returns T then output YES.
2. (If here then $\psi(T, \ldots, T) = F$).
   Create $\phi = \psi(x_1, \ldots, x_n) \vee (x_1 \wedge \cdots \wedge x_n)$.

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

1. Input $\psi(x_1, \ldots, x_n)$. Plug in $(T, \ldots, T)$.
   If returns T then output YES.
2. (If here then $\psi(T, \ldots, T) = F$).
   Create $\phi = \psi(x_1, \ldots, x_n) \vee (x_1 \wedge \cdots \wedge x_n)$.
   ϕ has at least one satisfying assignment, $(T, \ldots, T)$.

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

1. Input $\psi(x_1, \ldots, x_n)$. Plug in $(T, \ldots, T)$.
   If returns T then output YES.
2. (If here then $\psi(T, \ldots, T) = F$).
   Create $\phi = \psi(x_1, \ldots, x_n) \vee (x_1 \wedge \cdots \wedge x_n)$.
   ϕ has at least one satisfying assignment, $(T, \ldots, T)$.
3. Run M on ϕ.

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

1. Input $\psi(x_1, \ldots, x_n)$. Plug in $(T, \ldots, T)$.
   If returns T then output YES.
2. (If here then $\psi(T, \ldots, T) = F$).
   Create $\phi = \psi(x_1, \ldots, x_n) \vee (x_1 \wedge \cdots \wedge x_n)$.
   ϕ has at least one satisfying assignment, $(T, \ldots, T)$.
3. Run M on ϕ.
   - If output is YES then ϕ has ≥ 2 satisfying assignments. One of them is $(T, \ldots, T)$. Other one has to be a SAT assignment for ψ. So output YES.

PROMISE-SAT PROBLEM: SOLUTION

M is poly time program for **Promise-SAT**

1. Input $\psi(x_1, \ldots, x_n)$. Plug in $(T, \ldots, T)$.
   If returns T then output YES.
2. (If here then $\psi(T, \ldots, T) = F$).
   Create $\phi = \psi(x_1, \ldots, x_n) \vee (x_1 \wedge \cdots \wedge x_n)$.
   ϕ has at least one satisfying assignment, $(T, \ldots, T)$.
3. Run M on ϕ.
   - If output is YES then ϕ has ≥ 2 satisfying assignments. One of them is $(T, \ldots, T)$. Other one has to be a SAT assignment for ψ. So output YES.
   - If output is NO then ϕ has Hence ϕ has 1 satisfying assignments. Its. $(T, \ldots, T)$. So $\psi \notin \mathrm{SAT}$. So output NO.

What about Other Variants?

Think About What if we are promised that ϕ has either 0 or 1 satisfying assignments? If **PROMISE-SAT-0-1** problem is in P, then do you get **SAT** in P?

What about Other Variants?

Think About What if we are promised that ϕ has either 0 or 1 satisfying assignments? If **PROMISE-SAT-0-1** problem is in P, then do you get **SAT** in P?

Vote

What are the consequences of **PROMISE-SAT-0-1** in P?

What about Other Variants?

Think About What if we are promised that ϕ has either 0 or 1 satisfying assignments? If **PROMISE-SAT-0-1** problem is in P, then do you get **SAT** in P?

Vote

What are the consequences of **PROMISE-SAT-0-1** in P?

1. **SAT** in P and this is known.

What about Other Variants?

Think About What if we are promised that ϕ has either 0 or 1 satisfying assignments? If **PROMISE-SAT-0-1** problem is in P, then do you get **SAT** in P?

Vote

What are the consequences of **PROMISE-SAT-0-1** in P?

1. **SAT** in P and this is known.
2. **SAT** not in P and this is known.

What about Other Variants?

Think About What if we are promised that ϕ has either 0 or 1 satisfying assignments? If **PROMISE-SAT-0-1** problem is in P, then do you get **SAT** in P?

Vote

What are the consequences of **PROMISE-SAT-0-1** in P?

1. **SAT** in P and this is known.
2. **SAT** not in P and this is known.
3. Some other consequence of interest is known.

What about Other Variants?

Think About What if we are promised that ϕ has either 0 or 1 satisfying assignments? If **PROMISE-SAT-0-1** problem is in P, then do you get **SAT** in P?
Vote
What are the consequences of **PROMISE-SAT-0-1** in P?

1. **SAT** in P and this is known.
2. **SAT** not in P and this is known.
3. Some other consequence of interest is known.
4. No consequences **known to Bill**

What about Other Variants?

Think About What if we are promised that ϕ has either 0 or 1 satisfying assignments? If **PROMISE-SAT-0-1** problem is in P, then do you get **SAT** in P?

Vote

What are the consequences of **PROMISE-SAT-0-1** in P?

1. **SAT** in P and this is known.
2. **SAT** not in P and this is known.
3. Some other consequence of interest is known.
4. No consequences **known to Bill**

Some other consequence of interest is known.

What about Other Variants?

Think About What if we are promised that ϕ has either 0 or 1 satisfying assignments? If **PROMISE-SAT-0-1** problem is in P, then do you get **SAT** in P?

Vote

What are the consequences of **PROMISE-SAT-0-1** in P?

1. **SAT** in P and this is known.
2. **SAT** not in P and this is known.
3. Some other consequence of interest is known.
4. No consequences **known to Bill**

Some other consequence of interest is known.
SAT is in randomized poly time.

What about Other Variants?

Think About What if we are promised that ϕ has either 0 or 1 satisfying assignments? If **PROMISE-SAT-0-1** problem is in P, then do you get **SAT** in P?

Vote

What are the consequences of **PROMISE-SAT-0-1** in P?

1. **SAT** in P and this is known.
2. **SAT** not in P and this is known.
3. Some other consequence of interest is known.
4. No consequences **known to Bill**

Some other consequence of interest is known.
SAT is in randomized poly time.
So there is a fast randomized algorithm for **SAT** with a very small prob of error.

Reductions and P

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Assume $X \leq Y$ via f. Assume f takes $p(n)$ to compute.

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Assume $X \leq Y$ via f. Assume f takes $p(n)$ to compute.

Assume $Y \in P$. Assume the algorithm takes $q(n)$ steps.

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Assume $X \leq Y$ via f. Assume f takes $p(n)$ to compute.

Assume $Y \in P$. Assume the algorithm takes $q(n)$ steps.

Here is the procedure for X:

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Assume $X \leq Y$ via f. Assume f takes $p(n)$ to compute.

Assume $Y \in P$. Assume the algorithm takes $q(n)$ steps.

Here is the procedure for X:

1. Input x

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Assume $X \leq Y$ via f. Assume f takes $p(n)$ to compute.

Assume $Y \in P$. Assume the algorithm takes $q(n)$ steps.

Here is the procedure for X:

1. Input x
2. Compute $y = f(x)$. Time $p(|x|)$. Note $|y| \leq p(|x|)$.

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Assume $X \leq Y$ via f. Assume f takes $p(n)$ to compute.

Assume $Y \in P$. Assume the algorithm takes $q(n)$ steps.

Here is the procedure for X:

1. Input x
2. Compute $y = f(x)$. Time $p(|x|)$. Note $|y| \leq p(|x|)$.
3. Run the Y-algorithm on y. Time $q(p(|x|))$.

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Assume $X \leq Y$ via f. Assume f takes $p(n)$ to compute.

Assume $Y \in P$. Assume the algorithm takes $q(n)$ steps.

Here is the procedure for X:

1. Input x
2. Compute $y = f(x)$. Time $p(|x|)$. Note $|y| \leq p(|x|)$.
3. Run the Y-algorithm on y. Time $q(p(|x|))$.
   3.1 If answer is YES then output YES.

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Assume $X \leq Y$ via f. Assume f takes $p(n)$ to compute.

Assume $Y \in P$. Assume the algorithm takes $q(n)$ steps.

Here is the procedure for X:

1. Input x
2. Compute $y = f(x)$. Time $p(|x|)$. Note $|y| \leq p(|x|)$.
3. Run the Y-algorithm on y. Time $q(p(|x|))$.
   3.1 If answer is YES then output YES.
   3.2 If answer is NO then output NO.

Reductions and P

Question Show that if $X \leq Y$ and $Y \in P$ then $X \in P$.
Answer

Assume $X \leq Y$ via f. Assume f takes $p(n)$ to compute.

Assume $Y \in P$. Assume the algorithm takes $q(n)$ steps.

Here is the procedure for X:

1. Input x
2. Compute $y = f(x)$. Time $p(|x|)$. Note $|y| \leq p(|x|)$.
3. Run the Y-algorithm on y. Time $q(p(|x|))$.
   3.1 If answer is YES then output YES.
   3.2 If answer is NO then output NO.

Algorithm takes time $O((p(|x|) + q(p(|x|))$ which is poly in $|x|$.

Vertex Cover

Vertex Cover

Def Let $G = (V, E)$ be a graph. A **vertex cover for G of size k** is a set $U \subseteq V$ such that

Vertex Cover

Def Let $G = (V, E)$ be a graph. A **vertex cover for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and

Vertex Cover

Def Let $G = (V, E)$ be a graph. A **vertex cover for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and
- For every $(a, b) \in E$ either $a \in U$ or $b \in U$ (or both).

Vertex Cover

Def Let $G = (V, E)$ be a graph. A **vertex cover for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and
- For every $(a, b) \in E$ either $a \in U$ or $b \in U$ (or both).

$\mathrm{VC} = \{(G, k) : G \text{ has a Vertex Cover of size } k\}$.

Vertex Cover

Def Let $G = (V, E)$ be a graph. A **vertex cover for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and
- For every $(a, b) \in E$ either $a \in U$ or $b \in U$ (or both).

$\mathrm{VC} = \{(G, k) : G \text{ has a Vertex Cover of size } k\}$.

(It is known that VC is NP-complete.)

Vertex Cover

Def Let $G = (V, E)$ be a graph. A **vertex cover for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and
- For every $(a, b) \in E$ either $a \in U$ or $b \in U$ (or both).

$\mathrm{VC} = \{(G, k) : G$ has a Vertex Cover of size $k\}$.

(It is known that VC is NP-complete.)

$\mathrm{VC}_{1000} = \{G : G$ has a Vertex Cover of size $1000\}$.

Vertex Cover

Def Let $G = (V, E)$ be a graph. A **vertex cover for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and
- For every $(a, b) \in E$ either $a \in U$ or $b \in U$ (or both).

$\mathrm{VC} = \{(G, k) : G \text{ has a Vertex Cover of size } k\}$.

(It is known that VC is NP-complete.)

$\mathrm{VC}_{1000} = \{G : G \text{ has a Vertex Cover of size } 1000\}$.

Show that VC_{1000} is in P.

VC_{1000} in P

Notation $\binom{V}{1000}$ is the set of 1000-sized subsets of V.

$\mathbf{VC_{1000}}$ in P

Notation $\binom{V}{1000}$ is the set of 1000-sized subsets of V.

1. Input $G = (V, E)$. $|V| = n$.

$\mathbf{VC_{1000}}$ in P

Notation $\binom{V}{1000}$ is the set of 1000-sized subsets of V.

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$

VC_{1000} in P

Notation $\binom{V}{1000}$ is the set of 1000-sized subsets of V.

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   2.1 test if U is a VC.

VC_{1000} in P

Notation $\binom{V}{1000}$ is the set of 1000-sized subsets of V.

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   - 2.1 test if U is a VC.
   - 2.2 If YES then output YES and STOP.

VC_{1000} in P

Notation $\binom{V}{1000}$ is the set of 1000-sized subsets of V.

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   2.1 test if U is a VC.
   2.2 If YES then output YES and STOP.
   2.3 If NO then go back to for loop

VC_{1000} in P

Notation $\binom{V}{1000}$ is the set of 1000-sized subsets of V.

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   2.1 test if U is a VC.
   2.2 If YES then output YES and STOP.
   2.3 If NO then go back to for loop
3. If you got here output NO.

VC_{1000} in P

Notation $\binom{V}{1000}$ is the set of 1000-sized subsets of V.

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   2.1 test if U is a VC.
   2.2 If YES then output YES and STOP.
   2.3 If NO then go back to for loop
3. If you got here output NO.

Time: Roughly n^{1000}.

VC_{1000} in Time n^{1000}: Can We Do Better?

Vote

VC_{1000} in Time n^{1000}: Can We Do Better?

Vote

1. Bill will show you some way to do VC_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.

VC_{1000} in Time n^{1000}: Can We Do Better?

Vote

1. Bill will show you some way to do VC_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.
2. Bill will tell you about some kind of complexity theory to show that it is likely VC_{1000} requires $\Omega(n^{1000})$ time.

VC_{1000} in Time n^{1000}: Can We Do Better?

Vote

1. Bill will show you some way to do VC_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.
2. Bill will tell you about some kind of complexity theory to show that it is likely VC_{1000} requires $\Omega(n^{1000})$ time.
3. Bill will tell you that the the theory community has no consensus on whether VC_{1000} can be done in $n^{<1000}$.

VC_{1000} in Time n^{1000}: We Can Do Better!

VC_{1000} in Time n^{1000}: We Can Do Better!

Bill will show you some way to do VC_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.

VC_{1000} in Time n^{1000}: We Can Do Better!

Bill will show you some way to do VC_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.

1. The Graph Minor Theorem implies VC_{1000} is in $O(n^3)$ time. The GMT took 20 hard papers to prove.

VC_{1000} in Time n^{1000}: We Can Do Better!

Bill will show you some way to do VC_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.

1. The Graph Minor Theorem implies VC_{1000} is in $O(n^3)$ time. The GMT took 20 hard papers to prove.
2. A very clever algorithm enables you to solve VC_k in time $O(kn + 2^k k^{2k+2})$. **Note** k is not in the exponent of the poly in n.

VC_{1000} in Time n^{1000}: We Can Do Better!

Bill will show you some way to do VC_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.

1. The Graph Minor Theorem implies VC_{1000} is in $O(n^3)$ time. The GMT took 20 hard papers to prove.
2. A very clever algorithm enables you to solve VC_k in time $O(kn + 2^k k^{2k+2})$. **Note** k is not in the exponent of the poly in n.

   Algorithm Sketch Given G, all vertices of degree $\geq k+1$ are in the VC. Remove them to form G'. Now want VC of G' of size $\leq k'$. If G' has a VC of size $\leq k'$ then G' has $\leq kk' \leq k^2$ vertices. Do Brute Force on G'.

Hard Math AND a Clever Algorithm

Bill has said:

Hard Math AND a Clever Algorithm

Bill has said:
To show that, say, $\mathrm{SAT} \notin \mathrm{P}$, *you need to show that neither of the following will happen:*

Hard Math AND a Clever Algorithm

Bill has said:
To show that, say, $\mathrm{SAT} \notin \mathrm{P}$, *you need to show that neither of the following will happen:*

1. *Some very hard math is the key to an algorithm for* $\mathrm{SAT} \in \mathrm{P}$. *The reason it was not found earlier is that it required specialized and new knowledge.*

Hard Math AND a Clever Algorithm

Bill has said:

To show that, say, $\mathrm{SAT} \notin \mathrm{P}$*, you need to show that neither of the following will happen:*

1. *Some very hard math is the key to an algorithm for* $\mathrm{SAT} \in \mathrm{P}$*. The reason it was not found earlier is that it required specialized and new knowledge.*
2. *Some very clever algorithm is the key to an algorithm for* $\mathrm{SAT} \in \mathrm{P}$*. The reason it was not found earlier is that we just missed it.*

Hard Math AND a Clever Algorithm

Bill has said:

To show that, say, $\mathrm{SAT} \notin \mathrm{P}$, you need to show that neither of the following will happen:

1. *Some very hard math is the key to an algorithm for $\mathrm{SAT} \in \mathrm{P}$. The reason it was not found earlier is that it required specialized and new knowledge.*
2. *Some very clever algorithm is the key to an algorithm for $\mathrm{SAT} \in \mathrm{P}$. The reason it was not found earlier is that we just missed it.*

The VC_k problem was solved both ways.

Hard Math AND a Clever Algorithm

Bill has said:

To show that, say, $\mathrm{SAT} \notin \mathrm{P}$*, you need to show that neither of the following will happen:*

1. *Some very hard math is the key to an algorithm for* $\mathrm{SAT} \in \mathrm{P}$*. The reason it was not found earlier is that it required specialized and new knowledge.*
2. *Some very clever algorithm is the key to an algorithm for* $\mathrm{SAT} \in \mathrm{P}$*. The reason it was not found earlier is that we just missed it.*

The VC_k problem was solved both ways.

Bill's Point Bill still thinks $\mathrm{P} \neq \mathrm{NP}$; however, to prove that some hard math won't do it, or a clever algorithm won't do it, is a rather daunting task.

Hard Math AND a Clever Algorithm

Bill has said:
To show that, say, $\mathrm{SAT} \notin \mathrm{P}$, *you need to show that neither of the following will happen:*

1. *Some very hard math is the key to an algorithm for* $\mathrm{SAT} \in \mathrm{P}$. *The reason it was not found earlier is that it required specialized and new knowledge.*
2. *Some very clever algorithm is the key to an algorithm for* $\mathrm{SAT} \in \mathrm{P}$. *The reason it was not found earlier is that we just missed it.*

The VC_k problem was solved both ways.

Bill's Point Bill still thinks $\mathrm{P} \neq \mathrm{NP}$; however, to prove that some hard math won't do it, or a clever algorithm won't do it, is a rather daunting task.

Respect how difficult it will be to prove lower bounds!

Hard Math AND a Clever Algorithm

Bill has said:

To show that, say, $\mathrm{SAT} \notin \mathrm{P}$*, you need to show that neither of the following will happen:*

1. *Some very hard math is the key to an algorithm for* $\mathrm{SAT} \in \mathrm{P}$*. The reason it was not found earlier is that it required specialized and new knowledge.*
2. *Some very clever algorithm is the key to an algorithm for* $\mathrm{SAT} \in \mathrm{P}$*. The reason it was not found earlier is that we just missed it.*

The VC_k problem was solved both ways.

Bill's Point Bill still thinks $\mathrm{P} \neq \mathrm{NP}$; however, to prove that some hard math won't do it, or a clever algorithm won't do it, is a rather daunting task.

Respect how difficult it will be to prove lower bounds!

Abbreviated to **Respect Lower Bounds!**

Dominating Set

Dominating Set

Def Let $G = (V, E)$ be a graph. A **dominating set for G of size k** is a set $U \subseteq V$ such that

Dominating Set

Def Let $G = (V, E)$ be a graph. A **dominating set for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and

Dominating Set

Def Let $G = (V, E)$ be a graph. A **dominating set for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and
- For every $v \in V$ either $v \in U$ or a neighbor of v is in U.

Dominating Set

Def Let $G = (V, E)$ be a graph. A **dominating set for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and
- For every $v \in V$ either $v \in U$ or a neighbor of v is in U.

$\mathrm{DS} = \{(G, k) : G \text{ has a Dom Set of size } k\}$.

Dominating Set

Def Let $G = (V, E)$ be a graph. A **dominating set for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and
- For every $v \in V$ either $v \in U$ or a neighbor of v is in U.

$\mathrm{DS} = \{(G, k) : G$ has a Dom Set of size $k\}$.

(It is known that DS is NP-complete.)

Dominating Set

Def Let $G = (V, E)$ be a graph. A **dominating set for G of size k** is a set $U \subseteq V$ such that

- $|U| = k$, and
- For every $v \in V$ either $v \in U$ or a neighbor of v is in U.

$\mathrm{DS} = \{(G, k) : G$ has a Dom Set of size $k\}$.

(It is known that DS is NP-complete.)

$\mathrm{DS}_{1000} = \{G : G$ has a Dom Set of size $1000\}$.
Show that DS_{1000} is in P.

DS_{1000} in P

DS_{1000} in P

1. Input $G = (V, E)$. $|V| = n$.

DS_{1000} in P

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$

DS_{1000} in P

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   2.1 test if U is a Dom Set.

DS_{1000} in P

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   2.1 test if U is a Dom Set.
   2.2 If YES then output YES and STOP.

DS_{1000} in P

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   2.1 test if U is a Dom Set.
   2.2 If YES then output YES and STOP.
   2.3 If NO then go back to for loop

DS_{1000} in P

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   2.1 test if U is a Dom Set.
   2.2 If YES then output YES and STOP.
   2.3 If NO then go back to for loop
3. If you got here output NO.

DS_{1000} in P

1. Input $G = (V, E)$. $|V| = n$.
2. For all $U \in \binom{V}{1000}$
   2.1 test if U is a Dom Set.
   2.2 If YES then output YES and STOP.
   2.3 If NO then go back to for loop
3. If you got here output NO.

Time: Roughly n^{1000}.

DS_{1000} in Time n^{1000}: Can We Do Better?

Vote

DS_{1000} in Time n^{1000}: Can We Do Better?

Vote

1. Bill will show you some way to do DS_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.

DS_{1000} in Time n^{1000}: Can We Do Better?

Vote

1. Bill will show you some way to do DS_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.
2. Bill will tell you about some kind of complexity theory to show that it is likely DS_{1000} requires $\Omega(n^{1000})$ time.

DS_{1000} in Time n^{1000}: Can We Do Better?

Vote

1. Bill will show you some way to do DS_{1000} in time $O(n^3)$ and give his **Fire** and **Brimstone** Sermon about **Lower Bounds**.
2. Bill will tell you about some kind of complexity theory to show that it is likely DS_{1000} requires $\Omega(n^{1000})$ time.
3. Bill will tell you that the the theory community has no consensus on whether DS_{1000} can be done in $n^{<1000}$.

DS_{1000} in Time n^{1000}: We Prob Can't Do Better!

DS_{1000} in Time n^{1000}: We Prob Can't Do Better!

Bill will tell you about some kind of complexity theory to show that it is likely the problem requires $\Omega(n^{1000})$ time.

DS_{1000} in Time n^{1000}: We Prob Can't Do Better!

Bill will tell you about some kind of complexity theory to show that it is likely the problem requires $\Omega(n^{1000})$ time.

1. A problem is **Fixed Parameter Tractable (FPT)** if when you hold a parameter k constant there is a poly time algorithm where the run time does not have k in the exponent.

DS_{1000} in Time n^{1000}: We Prob Can't Do Better!

Bill will tell you about some kind of complexity theory to show that it is likely the problem requires $\Omega(n^{1000})$ time.

1. A problem is **Fixed Parameter Tractable (FPT)** if when you hold a parameter k constant there is a poly time algorithm where the run time does not have k in the exponent.
   Example VC_k is FPT with parameter k.

DS_{1000} in Time n^{1000}: We Prob Can't Do Better!

Bill will tell you about some kind of complexity theory to show that it is likely the problem requires $\Omega(n^{1000})$ time.

1. A problem is **Fixed Parameter Tractable (FPT)** if when you hold a parameter k constant there is a poly time algorithm where the run time does not have k in the exponent.
   Example VC_k is FPT with parameter k.
2. There are problems that are **thought** to NOT be FPT.
   Example SAT_k: Satisfiable with $\le k$ vars are set T.

DS_{1000} in Time n^{1000}: We Prob Can't Do Better!

Bill will tell you about some kind of complexity theory to show that it is likely the problem requires $\Omega(n^{1000})$ time.

1. A problem is **Fixed Parameter Tractable (FPT)** if when you hold a parameter k constant there is a poly time algorithm where the run time does not have k in the exponent.
   Example VC_k is FPT with parameter k.
2. There are problems that are **thought** to NOT be FPT.
   Example SAT_k: Satisfiable with $\le k$ vars are set T.
3. There is notion of reduction $\le_{\mathrm{FPT}}$ such that
   $Y \in \mathrm{FPT}$ and $X \le_{\mathrm{FPT}} Y$ implies $X \in \mathrm{FPT}$.

DS_{1000} in Time n^{1000}: We Prob Can't Do Better!

Bill will tell you about some kind of complexity theory to show that it is likely the problem requires $\Omega(n^{1000})$ time.

1. A problem is **Fixed Parameter Tractable (FPT)** if when you hold a parameter k constant there is a poly time algorithm where the run time does not have k in the exponent.
   Example VC_k is FPT with parameter k.
2. There are problems that are **thought** to NOT be FPT.
   Example SAT_k: Satisfiable with $\le k$ vars are set T.
3. There is notion of reduction $\le_{\mathrm{FPT}}$ such that
   $Y \in \mathrm{FPT}$ and $X \le_{\mathrm{FPT}} Y$ implies $X \in \mathrm{FPT}$.
4. Known that $\mathrm{SAT}_k \le_{\mathrm{FPT}} \mathrm{DS}_k$.

DS_{1000} in Time n^{1000}: We Prob Can't Do Better!

Bill will tell you about some kind of complexity theory to show that it is likely the problem requires $\Omega(n^{1000})$ time.

1. A problem is **Fixed Parameter Tractable (FPT)** if when you hold a parameter k constant there is a poly time algorithm where the run time does not have k in the exponent.
   Example VC_k is FPT with parameter k.
2. There are problems that are **thought** to NOT be FPT.
   Example SAT_k: Satisfiable with $\le k$ vars are set T.
3. There is notion of reduction $\le_{\mathrm{FPT}}$ such that $Y \in \mathrm{FPT}$ and $X \le_{\mathrm{FPT}} Y$ implies $X \in \mathrm{FPT}$.
4. Known that $\mathrm{SAT}_k \le_{\mathrm{FPT}} \mathrm{DS}_k$.
5. Hence we think $\mathrm{DS}_k \notin \mathrm{FPT}$.

3-Col and 4-col

3-Col and 4-col

$$\mathrm{3COL} = \{G : G \text{ is 3-colorable }\}.$$

3-Col and 4-col

$3\mathrm{COL} = \{G : G$ is 3-colorable $\}$.
$4\mathrm{COL} = \{G : G$ is 4-colorable $\}$.

3-Col and 4-col

$\mathrm{3COL} = \{G : G \text{ is 3-colorable }\}$.
$\mathrm{4COL} = \{G : G \text{ is 4-colorable }\}$.
Show that $\mathrm{3COL} \leq \mathrm{4COL}$.

3-Col and 4-col

$3\mathrm{COL} = \{G : G$ is 3-colorable $\}$.
$4\mathrm{COL} = \{G : G$ is 4-colorable $\}$.
Show that $3\mathrm{COL} \leq 4\mathrm{COL}$.

1. Input G

3-Col and 4-col

$3\mathrm{COL} = \{G : G$ is 3-colorable $\}$.
$4\mathrm{COL} = \{G : G$ is 4-colorable $\}$.
Show that $3\mathrm{COL} \le 4\mathrm{COL}$.

1. Input G
2. Create G' which is G with one more vertex v which has an edge to all vertices of G.

3-Col and 4-col

$3\mathrm{COL} = \{G : G$ is 3-colorable $\}$.
$4\mathrm{COL} = \{G : G$ is 4-colorable $\}$.
Show that $3\mathrm{COL} \le 4\mathrm{COL}$.

1. Input G
2. Create G' which is G with one more vertex v which has an edge to all vertices of G.

$G \in 3\mathrm{COL}$ implies $G' \in 4\mathrm{COL}$: Color the vertices of G with 3 colors. Then color the vertex v with a fourth color.

3-Col and 4-col

$\mathrm{3COL} = \{G : G \text{ is 3-colorable }\}$.
$\mathrm{4COL} = \{G : G \text{ is 4-colorable }\}$.
Show that $\mathrm{3COL} \le \mathrm{4COL}$.

1. Input G
2. Create G' which is G with one more vertex v which has an edge to all vertices of G.

$G \in \mathrm{3COL}$ implies $G' \in \mathrm{4COL}$: Color the vertices of G with 3 colors. Then color the vertex v with a fourth color.

$G' \in \mathrm{4COL}$ implies $G \in \mathrm{3COL}$: If $G' \in \mathrm{4COL}$ then coloring must use a coloring of v that is not used on any other vertex. Remove vertex v and you have a 3-coloring of G.

3-Col and 4-col

$3\text{COL} = \{G : G \text{ is 3-colorable }\}$.
$4\text{COL} = \{G : G \text{ is 4-colorable }\}$.
Show that $3\text{COL} \le 4\text{COL}$.

1. Input G
2. Create G' which is G with one more vertex v which has an edge to all vertices of G.

$G \in 3\text{COL}$ implies $G' \in 4\text{COL}$: Color the vertices of G with 3 colors. Then color the vertex v with a fourth color.

$G' \in 4\text{COL}$ implies $G \in 3\text{COL}$: If $G' \in 4\text{COL}$ then coloring must use a coloring of v that is not used on any other vertex. Remove vertex v and you have a 3-coloring of G.
Thus, $G \in 3\text{COL}$ iff $G' \in 4\text{COL}$.

3-Col and 4-col

Think About Is the following true:

$$4\mathrm{COL} \leq 3\mathrm{COL}$$

3-Col and 4-col

Think About Is the following true:

$$4\mathrm{COL} \le 3\mathrm{COL}$$

Vote

3-Col and 4-col

Think About Is the following true:

$$4\mathrm{COL} \le 3\mathrm{COL}$$

Vote

1. Yes, $4\mathrm{COL} \le 3\mathrm{COL}$ but the reduction is **insane**.

3-Col and 4-col

Think About Is the following true:

$$4\mathrm{COL} \le 3\mathrm{COL}$$

Vote

1. Yes, $4\mathrm{COL} \le 3\mathrm{COL}$ but the reduction is **insane**.
2. Yes, $4\mathrm{COL} \le 3\mathrm{COL}$ and the reduction is reasonable.

3-Col and 4-col

Think About Is the following true:

$$4\mathrm{COL} \le 3\mathrm{COL}$$

Vote

1. Yes, $4\mathrm{COL} \le 3\mathrm{COL}$ but the reduction is **insane**.
2. Yes, $4\mathrm{COL} \le 3\mathrm{COL}$ and the reduction is reasonable.
3. If $4\mathrm{COL} \le 3\mathrm{COL}$ then $\mathrm{P} = \mathrm{NP}$.

3-Col and 4-col

Think About Is the following true:

$$4\mathrm{COL} \le 3\mathrm{COL}$$

Vote

1. Yes, $4\mathrm{COL} \le 3\mathrm{COL}$ but the reduction is **insane**.
2. Yes, $4\mathrm{COL} \le 3\mathrm{COL}$ and the reduction is reasonable.
3. If $4\mathrm{COL} \le 3\mathrm{COL}$ then $\mathrm{P} = \mathrm{NP}$.
4. The question of whether $4\mathrm{COL} \le 3\mathrm{COL}$ is **Unknown to Bill**

An Insane Reduction

We show that

$$4\mathrm{COL} \le 3\mathrm{COL} \text{ by an insane reduction.}$$

An Insane Reduction

We show that

$$4\mathrm{COL} \le 3\mathrm{COL} \text{ by an insane reduction.}$$

Cook-Levin: SAT is NP-complete: $(\forall A \in \mathrm{NP})[A \le \mathrm{SAT}]$:

$$4\mathrm{COL} \le \mathrm{SAT}.$$

An Insane Reduction

We show that

$$4\mathrm{COL} \le 3\mathrm{COL} \text{ by an insane reduction.}$$

Cook-Levin: SAT is NP-complete: $(\forall A \in \mathrm{NP})[A \le \mathrm{SAT}]$:

$$4\mathrm{COL} \le \mathrm{SAT}.$$

We proved in class that

$$\mathrm{SAT} \le 3\mathrm{COL}.$$

An Insane Reduction

We show that

$$4\mathrm{COL} \le 3\mathrm{COL} \text{ by an insane reduction.}$$

Cook-Levin: SAT is NP-complete: $(\forall A \in \mathrm{NP})[A \le \mathrm{SAT}]$:

$$4\mathrm{COL} \le \mathrm{SAT}.$$

We proved in class that

$$\mathrm{SAT} \le 3\mathrm{COL}.$$

Hence by transitivity of reductions

$$4\mathrm{COL} \le \mathrm{SAT} \le 3\mathrm{COL}.$$

An Insane Reduction

We show that

$$4\mathrm{COL} \le 3\mathrm{COL} \text{ by an insane reduction.}$$

Cook-Levin: SAT is NP-complete: $(\forall A \in \mathrm{NP})[A \le \mathrm{SAT}]$:

$$4\mathrm{COL} \le \mathrm{SAT}.$$

We proved in class that

$$\mathrm{SAT} \le 3\mathrm{COL}.$$

Hence by transitivity of reductions

$$4\mathrm{COL} \le \mathrm{SAT} \le 3\mathrm{COL}.$$

I call this reduction **insane** since it goes from a graph to a formula (using Turing Machines) and then back to a graph.

Is there a Sane Reduction?

Is there a Sane Reduction?

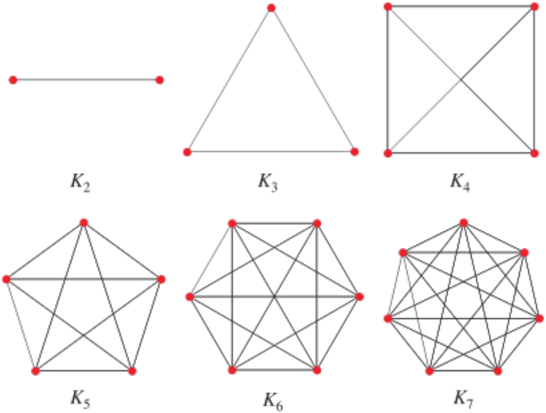
In 2014 my students one of my students was depressed at how insane the reduction was. SO I came up with a **sane** reduction. Its on arXiv here: `https://arxiv.org/pdf/1407.5128.pdf`

What if Graph is Planar?

Def A graph is **planar** if it can be drawn in the plane without crossing.

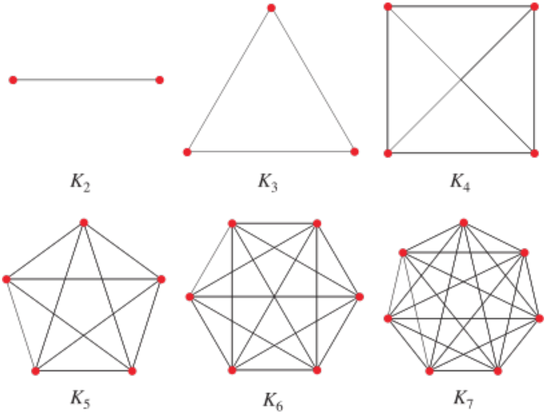
What if Graph is Planar?

Def A graph is **planar** if it can be drawn in the plane without crossing.



What if Graph is Planar?

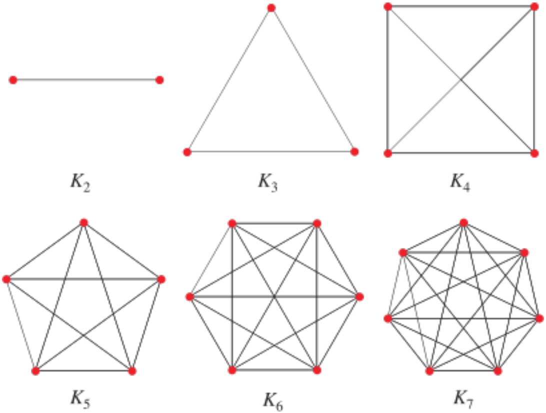
Def A graph is **planar** if it can be drawn in the plane without crossing.



K_1, K_2, K_3, K_4 are planar.

What if Graph is Planar?

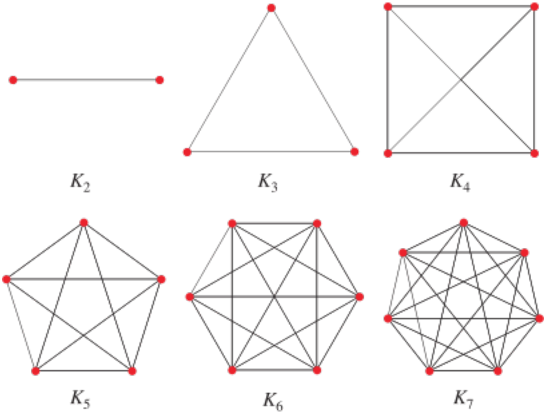
Def A graph is **planar** if it can be drawn in the plane without crossing.



K_1, K_2, K_3, K_4 are planar. $(\forall n \geq 5)$ K_n is not planar.

What if Graph is Planar?

Def A graph is **planar** if it can be drawn in the plane without crossing.



K_1, K_2, K_3, K_4 are planar. $(\forall n \geq 5)$ K_n is not planar.

It is known that testing if a graph is planar is in P.

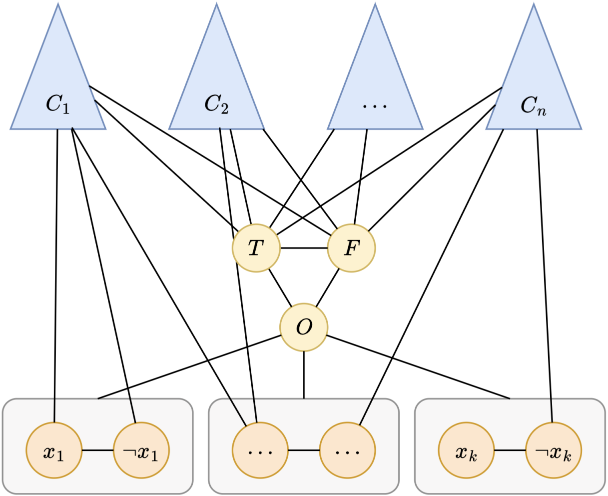
What if Graph is Planar? (cont)

What if Graph is Planar? (cont)

In the proof that $\mathrm{SAT} \leq 3\mathrm{COL}$ we used the following gadget:

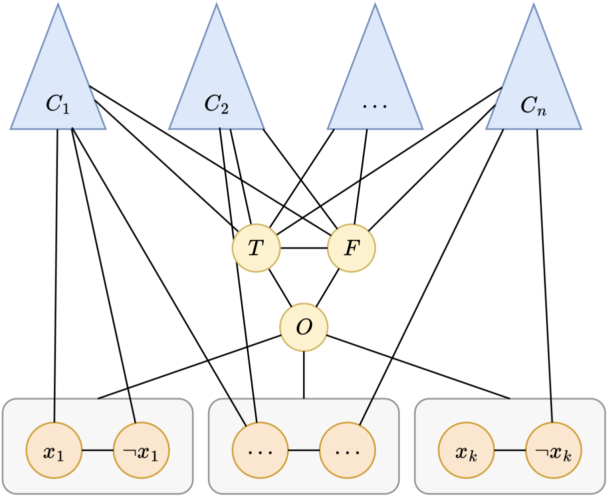
What if Graph is Planar? (cont)

In the proof that $\mathrm{SAT} \leq 3\mathrm{COL}$ we used the following gadget:



What if Graph is Planar? (cont)

In the proof that $\mathrm{SAT} \leq 3\mathrm{COL}$ we used the following gadget:



Note that the gadget is not planar.

Vote on if Planar 3COL is NP-complete

$$\mathrm{PL3COL} = \{G : G \text{ is Planar and } G \in \mathrm{3COL}\}.$$

Vote on if Planar 3COL is NP-complete

$$\mathrm{PL3COL} = \{G : G \text{ is Planar and } G \in \mathrm{3COL}\}.$$

Vote

Vote on if Planar 3COL is NP-complete

$$\mathrm{PL3COL} = \{G : G \text{ is Planar and } G \in \mathrm{3COL}\}.$$

Vote

1. $\mathrm{PL3COL}$ is NP-complete.

Vote on if Planar 3COL is NP-complete

$$\mathrm{PL3COL} = \{G : G \text{ is Planar and } G \in \mathrm{3COL}\}.$$

Vote

1. $\mathrm{PL3COL}$ is NP-complete.
2. $\mathrm{PL3COL}$ is in Polynomial Time.
   Fire and **Brimstone** Speech on lower bounds to follow.

Planar 3COL is NP-complete

$$3\mathrm{COL} \leq \mathrm{PL3COL}$$

Planar 3COL is NP-complete

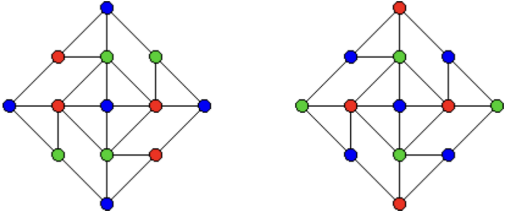
$$3\mathrm{COL} \leq \mathrm{PL3COL}$$

Replace all crossings with this gadget:

Planar 3COL is NP-complete

$$\mathrm{3COL} \leq \mathrm{PL3COL}$$

Replace all crossings with this gadget:



What About Planar 4COL?

What about 4-coloring?

What About Planar 4COL?

What about 4-coloring?

$$\mathrm{PL4COL} = \{G : G \text{ is Planar and } G \in \mathrm{4COL}\}.$$

What About Planar 4COL?

What about 4-coloring?

$$\mathrm{PL4COL} = \{G : G \text{ is Planar and } G \in \mathrm{4COL}\}.$$

Vote

What About Planar 4COL?

What about 4-coloring?

$$\mathrm{PL4COL} = \{G : G \text{ is Planar and } G \in \mathrm{4COL}\}.$$

Vote

1. $\mathrm{PL4COL}$ is NP-complete.

What About Planar 4COL?

What about 4-coloring?

$$\mathrm{PL4COL} = \{G : G \text{ is Planar and } G \in \mathrm{4COL}\}.$$

Vote

1. $\mathrm{PL4COL}$ is NP-complete.
2. **Fire** and **Brimstone** Speech on lower bounds to follow.

Planar 4COL is in P

Planar 4COL is in P

1. In 1852 Francis Guthrie asked *is every map 4-colorable*. This is equivalent to *is every planar graph 4-colorable*.

Planar 4COL is in P

1. In 1852 Francis Guthrie asked *is every map 4-colorable*. This is equivalent to *is every planar graph 4-colorable*.
2. Guthrie told DeMorgan who told Hamilton about the problem. So it got some attention.

Planar 4COL is in P

1. In 1852 Francis Guthrie asked *is every map 4-colorable*. This is equivalent to *is every planar graph 4-colorable*.
2. Guthrie told DeMorgan who told Hamilton about the problem. So it got some attention.
3. Many people worked on the problem, no progress.

Planar 4COL is in P

1. In 1852 Francis Guthrie asked *is every map 4-colorable*. This is equivalent to *is every planar graph 4-colorable*.
2. Guthrie told DeMorgan who told Hamilton about the problem. So it got some attention.
3. Many people worked on the problem, no progress.
4. 1976: Appel-Haken-Koch solved it. Used a comp. search.

Planar 4COL is in P

1. In 1852 Francis Guthrie asked *is every map 4-colorable*. This is equivalent to *is every planar graph 4-colorable*.
2. Guthrie told DeMorgan who told Hamilton about the problem. So it got some attention.
3. Many people worked on the problem, no progress.
4. 1976: Appel-Haken-Koch solved it. Used a comp. search.
5. 1996 Robertson-Sanders-Seymour-Thomas. Simpler but a comp. search.

Planar 4COL is in P

1. In 1852 Francis Guthrie asked *is every map 4-colorable*. This is equivalent to *is every planar graph 4-colorable*.
2. Guthrie told DeMorgan who told Hamilton about the problem. So it got some attention.
3. Many people worked on the problem, no progress.
4. 1976: Appel-Haken-Koch solved it. Used a comp. search.
5. 1996 Robertson-Sanders-Seymour-Thomas. Simpler but a comp. search.

Since every planar graph is 4-colorable

Planar 4COL is in P

1. In 1852 Francis Guthrie asked *is every map 4-colorable*. This is equivalent to *is every planar graph 4-colorable*.
2. Guthrie told DeMorgan who told Hamilton about the problem. So it got some attention.
3. Many people worked on the problem, no progress.
4. 1976: Appel-Haken-Koch solved it. Used a comp. search.
5. 1996 Robertson-Sanders-Seymour-Thomas. Simpler but a comp. search.

Since every planar graph is 4-colorable

$$\{G : G \text{ is Planar and } G \in \mathrm{4COL}\} = \{G : G \text{ is Planar}\}$$

Planar 4COL is in P

1. In 1852 Francis Guthrie asked *is every map 4-colorable*. This is equivalent to *is every planar graph 4-colorable*.
2. Guthrie told DeMorgan who told Hamilton about the problem. So it got some attention.
3. Many people worked on the problem, no progress.
4. 1976: Appel-Haken-Koch solved it. Used a comp. search.
5. 1996 Robertson-Sanders-Seymour-Thomas. Simpler but a comp. search.

Since every planar graph is 4-colorable

$$\{G : G \text{ is Planar and } G \in \mathrm{4COL}\} = \{G : G \text{ is Planar}\}$$

and hence is in P.

Planar 4COL is in P

1. In 1852 Francis Guthrie asked *is every map 4-colorable*. This is equivalent to *is every planar graph 4-colorable*.
2. Guthrie told DeMorgan who told Hamilton about the problem. So it got some attention.
3. Many people worked on the problem, no progress.
4. 1976: Appel-Haken-Koch solved it. Used a comp. search.
5. 1996 Robertson-Sanders-Seymour-Thomas. Simpler but a comp. search.

Since every planar graph is 4-colorable

$$\{G : G \text{ is Planar and } G \in 4\mathrm{COL}\} = \{G : G \text{ is Planar}\}$$

and hence is in P.

Fire and **Brimstone** Speech on next slides.

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

1. Small NFA for $\{a^n \mid n \neq 1000\}$. Clever.

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

1. Small NFA for $\{a^n \mid n \neq 1000\}$. Clever.
2. $\{w : \#_{ab}(w) = \#_{ba}(w)\}$. Clever.

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

1. Small NFA for $\{a^n \mid n \neq 1000\}$. Clever.
2. $\{w : \#_{ab}(w) = \#_{ba}(w)\}$. Clever.
3. Small CFG for $\{w : |w| = n \wedge \#_a(w) = n/2\}$. Clever.

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

1. Small NFA for $\{a^n \mid n \neq 1000\}$. Clever.
2. $\{w : \#_{ab}(w) = \#_{ba}(w)\}$. Clever.
3. Small CFG for $\{w : |w| = n \wedge \#_a(w) = n/2\}$. Clever.
4. VC_k via Graph Minor Theorem. Hard Math or Clever.

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

1. Small NFA for $\{a^n \mid n \neq 1000\}$. Clever.
2. $\{w : \#_{ab}(w) = \#_{ba}(w)\}$. Clever.
3. Small CFG for $\{w : |w| = n \wedge \#_a(w) = n/2\}$. Clever.
4. VC_k via Graph Minor Theorem. Hard Math or Clever.

$\mathrm{PL4COL} \in \mathrm{P}$ is something new.

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

1. Small NFA for $\{a^n \mid n \neq 1000\}$. Clever.
2. $\{w : \#_{ab}(w) = \#_{ba}(w)\}$. Clever.
3. Small CFG for $\{w : |w| = n \wedge \#_a(w) = n/2\}$. Clever.
4. VC_k via Graph Minor Theorem. Hard Math or Clever.

$\mathrm{PL4COL} \in \mathrm{P}$ is something new.

Comp. prog. to prove **all** planar graphs are 4-colorable.

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

1. Small NFA for $\{a^n \mid n \ne 1000\}$. Clever.
2. $\{w : \#_{ab}(w) = \#_{ba}(w)\}$. Clever.
3. Small CFG for $\{w : |w| = n \wedge \#_a(w) = n/2\}$. Clever.
4. VC_k via Graph Minor Theorem. Hard Math or Clever.

$\mathrm{PL4COL} \in \mathrm{P}$ is something new.

Comp. prog. to prove **all** planar graphs are 4-colorable.

The proof used some math but not that hard.

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

1. Small NFA for $\{a^n \mid n \ne 1000\}$. Clever.
2. $\{w : \#_{ab}(w) = \#_{ba}(w)\}$. Clever.
3. Small CFG for $\{w : |w| = n \wedge \#_a(w) = n/2\}$. Clever.
4. VC_k via Graph Minor Theorem. Hard Math or Clever.

$\mathrm{PL4COL} \in \mathrm{P}$ is something new.

Comp. prog. to prove **all** planar graphs are 4-colorable.

The proof used some math but not that hard.

The proof used some cleverness but no that clever.

A New Variant of Fire and Brimstone

My past **Fire** and **Brimstone** sermons:

1. Small NFA for $\{a^n \mid n \neq 1000\}$. Clever.
2. $\{w : \#_{ab}(w) = \#_{ba}(w)\}$. Clever.
3. Small CFG for $\{w : |w| = n \wedge \#_a(w) = n/2\}$. Clever.
4. VC_k via Graph Minor Theorem. Hard Math or Clever.

$\mathrm{PL4COL} \in \mathrm{P}$ is something new.

Comp. prog. to prove **all** planar graphs are 4-colorable.

The proof used some math but not that hard.

The proof used some cleverness but no that clever.

The bulk of the proof was the program.

Fire and Brimstone

To prove $\mathrm{SAT} \notin \mathrm{P}$ we have to rule out that any of the following, or a combination of them, will be used to get an poly time algorithm for SAT:

Fire and Brimstone

To prove $\mathrm{SAT} \notin \mathrm{P}$ we have to rule out that any of the following, or a combination of them, will be used to get an poly time algorithm for SAT:

1. Cleverness

Fire and Brimstone

To prove $\mathrm{SAT} \notin \mathrm{P}$ we have to rule out that any of the following, or a combination of them, will be used to get an poly time algorithm for SAT:

1. Cleverness
2. Hard Math

Fire and Brimstone

To prove $\mathrm{SAT} \notin \mathrm{P}$ we have to rule out that any of the following, or a combination of them, will be used to get an poly time algorithm for SAT:

1. Cleverness
2. Hard Math
3. A Computer Program (perhaps to find some parameters).

Fire and Brimstone

To prove $\mathrm{SAT} \notin \mathrm{P}$ we have to rule out that any of the following, or a combination of them, will be used to get an poly time algorithm for SAT:

1. Cleverness
2. Hard Math
3. A Computer Program (perhaps to find some parameters).

So proving $\mathrm{SAT} \notin \mathrm{P}$ is going to be hard.

Fire and Brimstone

To prove $\mathrm{SAT} \notin \mathrm{P}$ we have to rule out that any of the following, or a combination of them, will be used to get an poly time algorithm for SAT:

1. Cleverness
2. Hard Math
3. A Computer Program (perhaps to find some parameters).

So proving $\mathrm{SAT} \notin \mathrm{P}$ is going to be hard.

It is my hope that Adam-Isaac-Sam show $\mathrm{SAT} \notin \mathrm{P}$ before the final and hence prove me a fool.

Fire and Brimstone

To prove $\mathrm{SAT} \notin \mathrm{P}$ we have to rule out that any of the following, or a combination of them, will be used to get an poly time algorithm for SAT:

1. Cleverness
2. Hard Math
3. A Computer Program (perhaps to find some parameters).

So proving $\mathrm{SAT} \notin \mathrm{P}$ is going to be hard.

It is my hope that Adam-Isaac-Sam show $\mathrm{SAT} \notin \mathrm{P}$ before the final and hence prove me a fool.
If they do then

Fire and Brimstone

To prove $\mathrm{SAT} \notin \mathrm{P}$ we have to rule out that any of the following, or a combination of them, will be used to get an poly time algorithm for SAT:

1. Cleverness
2. Hard Math
3. A Computer Program (perhaps to find some parameters).

So proving $\mathrm{SAT} \notin \mathrm{P}$ is going to be hard.

It is my hope that Adam-Isaac-Sam show $\mathrm{SAT} \notin \mathrm{P}$ before the final and hence prove me a fool.
If they do then

1. The don't have to grade the final.

Fire and Brimstone

To prove $\mathrm{SAT} \notin \mathrm{P}$ we have to rule out that any of the following, or a combination of them, will be used to get an poly time algorithm for SAT:

1. Cleverness
2. Hard Math
3. A Computer Program (perhaps to find some parameters).

So proving $\mathrm{SAT} \notin \mathrm{P}$ is going to be hard.

It is my hope that Adam-Isaac-Sam show $\mathrm{SAT} \notin \mathrm{P}$ before the final and hence prove me a fool.
If they do then

1. The don't have to grade the final.
2. They owe me 1000,000 free lunches.