

Complexity Theory: The P vs NP question

Lecture 28 (December 1, 2009)

The \$1M question

The Clay Mathematics Institute
Millennium Prize Problems

1. Birch and Swinnerton-Dyer Conjecture
2. Hodge Conjecture
3. Navier-Stokes Equations
4. P vs NP
5. Poincaré Conjecture
6. Riemann Hypothesis
7. Yang-Mills Theory

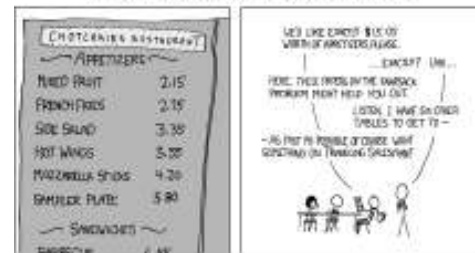
Princeton CS Building



P	1	0	1	0	0	0	0
=	0	1	1	1	1	0	1
N	1	0	0	1	1	1	0
P	1	0	1	0	0	0	0
?	0	1	1	1	1	1	1

xkcd

MY HOBBY:
EMBEDDING NP-COMplete PROBLEMS IN RESTAURANT ORDERS

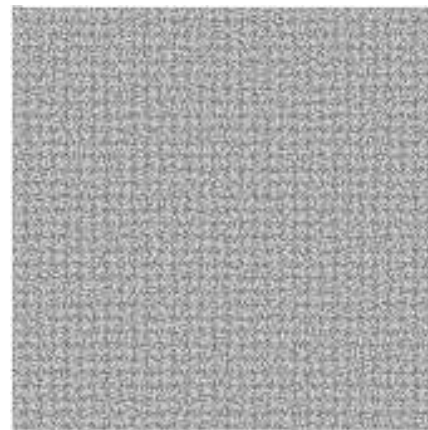


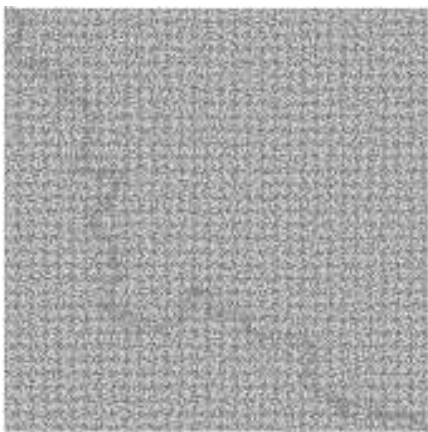
The P versus NP problem

Is perhaps one of the biggest open problems
in computer science (and mathematics!)
today.

(Even featured in the TV show NUMB3RS)

But what is the P-NP problem?





Sudoku

2		3	8		5
	3		4	5	9
	8		9	7	3
6	7		9		
9	8				1
			5	6	9
3	1	9	7		2
	4	6	5	2	8
2		9	3		1

3x3x3

Sudoku

2	9	4	3	7	8	1	5	6
1	7	3	6	4	5	9	8	2
5	6	8	2	1	9	7	3	4
6	5	7	1	9	2	3	4	8
9	8	2	4	3	6	5	1	7
4	3	1	8	5	7	6	2	9
3	1	9	7	8	4	2	6	5
7	4	6	5	2	1	8	9	3
8	2	5	9	6	3	4	7	1

3x3x3

Sudoku

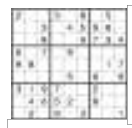
F	2				6		C	B	3
C			4	8	E	A		D	
D	A	8		3	2	7	F		6
6		E	D	F	C		8		7
	9	3	7			A			2
E				6	F	5	B	4	3
C	B	1	3	9	D		D	2	E
D		6		5	E	B	1		
B	6			1	F	3	2		A
			4	A	8		D	0	9
2	A		0	D	5	6	C		F
5				2				A	4
B			4		1	A	2	F	
	0	7		F	3	C	D		2
	5	1		A	B	D	B		D
2	D	A		9				1	4

4x4x4

Sudoku

D	F	9	2	A	7	5	1	4	6	E	D	C	B	3	8
7	C	1	3	6	4	8	E	A	B	5	0	2	D	F	9
D	A	8	4	3	3	B	2	7	F	C	1	6	0	5	E
6	5	B	E	D	F	0	C	2	8	9	3	4	A	1	7
4	9	3	5	7	1	C	0	D	A	F	B	8	E	6	2
E	B	7	0	2	A	6	F	5	8	B	4	D	3	C	1
C	B	F	1	3	9	D	4	0	2	E	E	5	7	8	A
A	D	2	6	5	5	E	B	1	1	7	C	3	F	0	4
B	6	4	8	E	B	1	7	F	3	2	5	0	C	A	D
3	7	C	F	4	6	A	8	E	D	0	9	B	1	2	5
2	1	A	B	0	D	3	5	6	C	4	8	7	9	E	F
5	E	0	D	F	C	2	8	9	3	7	1	A	0	4	8
B	3	5	9	E	4	0	1	5	A	2	F	8	1	0	
1	0	E	7	5	8	F	3	C	4	D	6	A	2	9	B
B	4	5	C	1	2	7	A	8	D	B	F	E	6	D	3
F	2	D	A	B	0	9	6	B	E	3	7	1	5	4	0

4x4x4



Sudoku

Suppose it takes you $S(n)$ to solve $n \times n \times n$

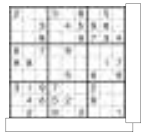
$V(n)$ time to verify the solution

Fact: $V(n) = O(n^2 \times n^2)$

Question: is there some constant such that

$S(n) \leq n^{\text{constant}}$?

•
•
•
 $n \times n \times n$

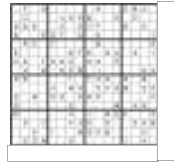


Sudoku

P vs NP problem

=

Does there exist an algorithm for $n \times n \times n$ Sudoku that runs in time $p(n)$ for some polynomial $p()$?



⋮

$n \times n \times n$

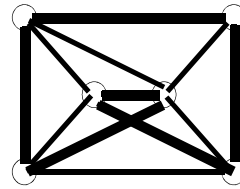
The P versus NP problem (informally)

Is proving a theorem much more difficult than checking the proof of a theorem?

Let's start at the beginning...

Hamilton Cycle

Given a graph $G = (V, E)$, a cycle that visits all the nodes exactly once



The Problem "HAM"

Input: Graph $G = (V, E)$

Output: YES if G has a Hamilton cycle
NO if G has no Hamilton cycle

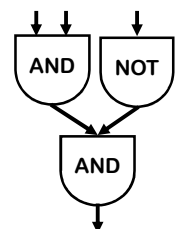
The Set "HAM"

$\text{HAM} = \{ \text{graph } G \mid G \text{ has a Hamilton cycle} \}$

Circuit-Satisfiability

Input: A circuit C with one output

Output: YES if C is satisfiable
NO if C is not satisfiable



The Set "SAT"

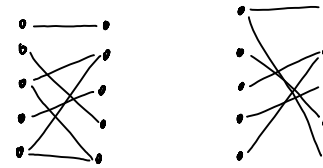
$SAT = \{ \text{all satisfiable circuits } C \}$

Bipartite Matching

Input: A bipartite graph $G = (U, V, E)$

Output: YES if G has a perfect matching

NO if G does not



The Set "BI-MATCH"

$BI-MATCH = \{ \text{all bipartite graphs that have a perfect matching} \}$

Sudoku

Input: $n \times n \times n$ sudoku instance

Output: YES if this sudoku has a solution

NO if it does not

The Set "SUDOKU"

$SUDOKU = \{ \text{All solvable sudoku instances} \}$

Decision Versus Search Problems

Decision Problem
YES/NO answers

Does G have a
Hamilton cycle?

Can G be
3-colored ?

Search Problem

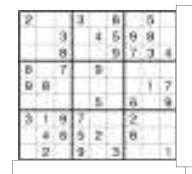
Find a Hamilton cycle
in G if one exists,
else return NO

Find a 3-coloring of
 G if one exists, else
return NO

Reducing Search to Decision

Given an algorithm for decision Sudoku,
devise an algorithm to find a solution

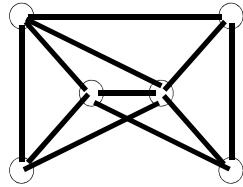
Idea:
Fill in one-by-one and
use decision algorithm



Reducing Search to Decision

Given an algorithm for decision HAM,
devise an algorithm to find a solution

Idea:
Find the edges of the
cycle one by one



Decision/Search Problems

We'll study decision problems because
they are almost the same (asymptotically)
as their search counterparts

Polynomial Time and The Class "P" of Decision Problems

What is an efficient algorithm?

Is an $O(n)$ algorithm efficient?

How about $O(n \log n)$?

$O(n^2)$?

$O(n^{10})$?

$O(n^{\log n})$?

$O(2^n)$?

$O(n!)$?

polynomial time

$O(n^c)$ for some
constant c

non-polynomial
time

Does an algorithm
running in $O(n^{100})$ time
count as efficient?

We consider non-polynomial time
algorithms to be inefficient.

And hence a necessary condition for an
algorithm to be efficient is that it should
run in poly-time.

Asking for a poly-time algorithm for a
problem sets a (very) low bar when asking
for efficient algorithms.

The question is: can we achieve even this
for 3-coloring?

SAT?

Sudoku?

The Class P

We say a subset L of Σ^* is in **P** if there is a program A and a polynomial $p()$

such that for any x in Σ^* ,

$A(x)$ runs for at most $p(|x|)$ time and answers question “is x in L ?” correctly.

The Class P

The class of all sets L that can be recognized in polynomial time.

The class of all decision problems that can be decided in polynomial time.

Why are we looking only at sets $\subseteq \Sigma^*$?

What if we want to work with graphs or boolean formulas?

Languages/Functions in P?

Example 1:

$\text{CONN} = \{\text{graph } G: G \text{ is a connected graph}\}$

Algorithm A_1 :

If G has n nodes, then run depth first search from any node, and count number of distinct node you see. If you see n nodes, $G \in \text{CONN}$, else not.

Time: $p_1(|x|) = \Theta(|x|)$.

Languages/Functions in P?

HAM, SUDOKU, SAT are not known to be in P

$\text{CO-HAM} = \{ G \mid G \text{ does NOT have a Hamilton cycle} \}$

$\text{CO-HAM} \in \text{P}$ if and only if $\text{HAM} \in \text{P}$

Onto the new class, NP

Verifying Membership

Is there a short “proof” I can give you for:

$G \in \text{HAM?}$

$G \in \text{BI-MATCH?}$

$G \in \text{SAT?}$

$G \in \text{CO-HAM?}$

NP

A set $L \in \text{NP}$

if there exists an algorithm A and a polynomial $p()$

For all $x \in L$

there exists y with
 $|y| \leq p(|x|)$

such that $A(x,y) = \text{YES}$

in $p(|x|)$ time

For all $x' \notin L$

For all y' with
 $|y'| \leq p(|x'|)$

we have $A(x',y') = \text{NO}$

in $p(|x|)$ time

Recall the Class P

We say a set $L \subseteq \Sigma^*$ is in **P** if there is a program A and a polynomial $p()$

such that for any x in Σ^* ,

$\left\{ \begin{array}{l} A(x) \text{ runs for at most } p(|x|) \text{ time} \\ \text{and answers question “is } x \text{ in } L?” \text{ correctly.} \end{array} \right.$

can think of A as “proving” that x is in L

NP

A set $L \in \text{NP}$

if there exists an algorithm A and a polynomial $p()$

For all $x \in L$

there exists a y with
 $|y| \leq p(|x|)$

such that $A(x,y) = \text{YES}$

in $p(|x|)$ time

For all $x' \notin L$

For all y' with
 $|y'| \leq p(|x'|)$

we have $A(x',y') = \text{NO}$

in $p(|x|)$ time

The Class NP

The class of sets L for which there exist
“short” proofs of membership
(of polynomial length)
that can “quickly” verified
(in polynomial time).

Recall: A doesn't have to find these proofs y ; it just needs to be able to verify that y is a “correct” proof.

$P \subseteq \text{NP}$

For any L in P , we can just take y to be the empty string and satisfy the requirements.

Hence, every language in P is also in NP .

Languages/Functions in NP?

$G \in \text{HAM?}$

$G \in \text{BI-MATCH?}$

$G \in \text{SAT?}$

$G \in \text{CO-HAM?}$

Summary: P versus NP

Set L is in P if membership in L can be decided in poly-time.

Set L is in NP if each x in L has a short “proof of membership” that can be verified in poly-time.

Fact: $P \subseteq NP$

Question: Does $NP \subseteq P$?

Why Care?

NP Contains Lots of Problems We Don't Know to be in P

Classroom Scheduling
 Packing objects into bins
 Scheduling jobs on machines
 Finding cheap tours visiting a subset of cities
 Allocating variables to registers
 Finding good packet routings in networks
 Decryption
 ...

OK, OK, I care...

But where do I begin
 if I want to reason about
 the $P=NP$ problem?

How can we prove that
 $NP \subseteq P$?

I would have to show that
 every set in NP has a
 polynomial time algorithm...

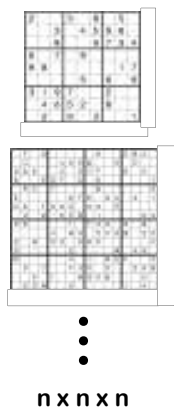
How do I do that?
 It may take a long time!
 Also, what if I forgot one of
 the sets in NP?

We can describe
just one problem L in NP ,
such that
if this problem L is in P ,
then $NP \subseteq P$.

It is a problem that can
capture all other problems
in NP .

The “Hardest” Set in NP

Sudoku



Sudoku has a
polynomial time
algorithm

if and only if

$P = NP$

The “Hardest” Sets in NP

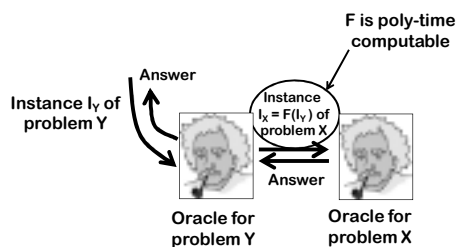
Sudoku	Clique
SAT	Independent-Set
3-Colorability	HAM

These problems are all
“polynomial-time equivalent”.

I.e., each of these can be reduced to any
of the others in poly-time

“Poly-time reducible to each other”

Reducing problem Y to problem X in poly-time



How do you prove these
are the hardest?

Theorem [Cook/Levin]:

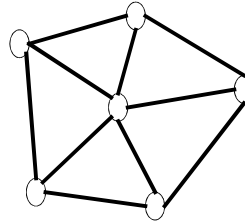
SAT is one language in NP, such that if we can show SAT is in P, then we have shown $NP \subseteq P$.

SAT is a language in NP that can capture all other languages in NP.

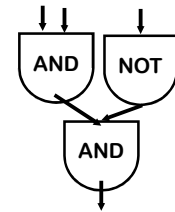
We say SAT is NP-complete.

Last lecture...

3-colorability



Circuit Satisfiability



Last lecture...

SAT and 3COLOR: Two problems that seem quite different, but are substantially the same.

Also substantially the same as CLIQUE and INDEPENDENT SET. (Homework)

If you get a polynomial-time algorithm for one, you get a polynomial-time algorithm for ALL.

Any language in NP

can be reduced
(in polytime to)
an instance of

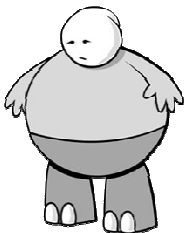
SAT

hence SAT is NP-complete

can be reduced
(in polytime to)
an instance of

3COLOR

hence 3COLOR is NP-complete



Definition of P and NP

Definition of problems
SAT, 3-COLOR, HAM,
SUDOKU, BI-MATCH

SAT, 3-COLOR, HAM, SUDOKU
all essentially equivalent.

Here's What
You Need to
Know...

Solve any one in poly-time
 \Rightarrow solve all of them in poly-time