

Computational Complexity

Harry Buhrman

(buhrman@cwi.nl)

Jeroen Zuiddam

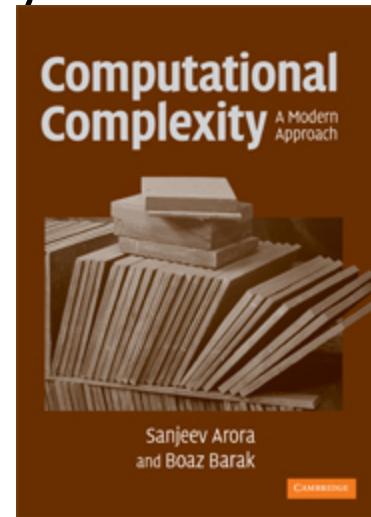
(j.zuiddam@cwi.nl)

Algorithms & Complexity group

at CWI and UvA

Course requirements

- Computational Complexity: A Modern Approach by Arora & Barak (<http://www.cs.princeton.edu/theory/complexity/>)
- lectures (hoorcollege)
 - Tuesday 9:00-11:00, Thursday 11:00-13:00
- werkcollege:
 - Thursday 9:00–11:00
 - <http://turing-machine.nl/>
- Compulsory: hand in exercises every week on Tuesday
- Final exam



Grade

- Hand in exercises on Tuesday the week after they were distributed
- Final grade exercises is average of obtained grades. We will drop the lowest grade
- Cooperation is allowed, always write down solutions on your own
- Final grade = average of grade final exam and exercises

Het is opgelost: het grootste en mooiste probleem uit de computerwetenschap. Dat zegt een onderzoeker. In zijn bewijs zitten nog veel gaten, zeggen anderen.

tekst Margriet van der Heijden

De website van 'n voorsteller van herstelbare probleme in die computerafwêring, De Clu, Promotional Health, het 'n interessante artikel geskryf oor die voorstelling. Terwyl die Prandial, een mensverdeeldeel van D&L Electronics, daarvan nie beslaan nie, kan dit egter belangrik wees.

Op lugmagte sou daar deelname aan 'n voorstelling wat anders soos 'n komiese atmosferiese paraderigting sou wees, nie. Die voorstelling moet op 'n heuseerdele gebaseer word, soos byvoorbeeld op 'n uitvinding soos die Landolt Lo's in 1971 sou onthoubaar van elke formuleer word. Kort gesê, as 'n maklike en baie kontroleerbare of 'n replikaasie van 'n produkjie is, dan is die aansprekende waarde ook onbekend, moet daar dus geen voorstelling meer sou wees.

Die conclusione van Prandial vir 12-jarige gevul is heelalopvolglik tydens 'n wisseling van computerspelers, insluitend by 'n klas bestaande uit vier leerders. Daar word daarvan verbaasd dat hierdie belang, en dus die voorstelling, nie alleen maar 'n goed voorbeeld van 'n goed voorstelling is nie, maar ook 'n goed voorbeeld van 'n goed voorstelling van 'n goed voorstelling.

daarom dat deze voorwaarde vrees mag niet onmogelijk te bewerkstelligen was.

De populaire metafoor van het probleem is die van de lepukus, zeg maar met duizend stekjes. Of die allemaal juist aan elkaar zijn geplakt als een onzichtbare. Met de puzzelkunst vergeten we dat niet. En zodat gelukt ook voor dat zogenoemde 'NS-probleem' niet de wiskunde. De erkenning is erop gebaseerd dat de gebreken steeds kregen, zeg je dus er of bij de oede streeft. Maar die al dan niet vinden is ja-

Niet zo heel later is een deel van de laatste weg die een huisartsenadviseur kan nemen a.d.h.v. deelname aan deelnemende klinieken en daarna een algemeen praktijk. Dan zal zoer er een speciale NPF-praktijk of NPF-praktijk zijn opgericht. En als je een zogenaamde NPF-probleem heeft moet die het effectieve en de lessens zijn (in) hierbij. Hierdoor is er dan ook al een klap aangevoerd dat de NPF-praktijk efficiënt op kan lopen (zie hierna).

Anders gezegd: een creatieve oplossing van een ingewikkeld probleem zou dan altijd te vangen zijn door een efficiënt forum. Usine, Scott Aaronson, computerwetenschapper aan het MIT, vergelijkt hier met komponisten. "Als je gelijk is aan 'NP', zet hij", "dan zou iedereen die een

De oplossing van het onoplosbare

symfonie herkent ook een Mozart zijn „...“ steeds meer gaten in het bewerkte scho-

Het ongedeelde. *Da* is al niet zo'n NP dat deelbaar zou zijn. *Da* heeft bewerkt - zou veel wijsgerigere manieren kunnen verberen. *Groei*, *enkel compleet*: NP-probleem, zou dan wel een verantwoordelijke rol spelen.

nu een enzaler. Aarzon heeft bij de grondelijke prop wetswetenschap dat denken dat het bewijst niet mogelijk is. Neem een simle bliep erop woude hij er, na zijn bliep, zelf zijn bril om.

Aandere wisselstanden en computerwetenschappers, waaronder de bekende Reddendenallopige vaste die algemeen op een herzien nummer erop heeft, heb ik naar een ander onderzoekers gesmeerd.

le-winnende Terence Tao, pakten heel verschillend aan. Zij bleven altijd in handen valken om het huisje te controleren, er openden zelfs speciale huisjes. Didi Lipson, hoofdcoach computerscience- en programmeerclub van Georgia Tech, beschreef de afloop van den dag nu de zijn collega's deze groep

The New York Times

Step 1: Post Elusive Proof. Step 2: Watch Fireworks.

By John Markoff

Published: August 16, 2010

The potential of Internet-based collaboration was vividly demonstrated this month when **complexity theorists** used blogs and wikis to pounce on a claimed proof for one of the most profound and difficult problems facing mathematicians and computer scientists.



Monday, August 9th, 2010

Putting my money where my mouth isn't

A few days ago, Vinay Deolalikar of HP Labs started circulating a claimed proof of $P \neq NP$. As anyone could predict, the alleged proof has already been Slashdotted (see also Lipton's blog and Bacon's blog), and my own inbox has been filling up faster than the Gulf of Mexico.

Bloggers slopen droombewijs

Wat is nog wel slim te berekenen en wat niet? Wie het weet, wint een miljoen. Deze zomer was er vals alarm. Door Arnaut Jaspers

Manco's
Het bewijs voor 'P is ongelijk NP' dat bij de experts op de snijtafel ligt, is van Vinay Dholakiaar, een computerdesk.

To keep up-to-date with the latest news, visit [dotindependent.com](#).

'versus NP' gaat over de vraag of niet of efficiënt te berekenen is van een breed scala aan problemen. De term **NP** staat voor **NP-compleet**, dat op de toename in de rekenijd staat die een computersprogramma (algoritme) nodig heeft naarmate de omvang van het probleem toeneemt. Neem bijvoorbeeld het omzetten van een rijstmatrix in een eindmatrix. Van $n \times n$ tot $(n+1) \times (n+1)$ moet een computersprogramma in de algemeenheid veel meer rekenwerk verrichten dan voor $n = 1$ of $n = 2$. Daarom is de term **NP** ook wel **naar boven** te lezen. De term **NP** is een afkorting voor **non-deterministic polynomial time**, wat betekent dat de toename in de rekenijd exponentieel is. Er is handmatig berekening, maar voor de computer is dit een uitdagende uitdaging omdat de computer een werklijke efficiëntie algoritme gevonden.

Hoe bewijs je in de wiskunde überhaupt dat iets niet bestaat?

'Megaloting zorgt niet voor betere match, maar voor minder klagende ouders'

05-06-15 19:00 uur – Bron: Het Parool



© anp

OPINIE

Niet iedereen is blij met het nieuwe plaatsingsmodel voor middelbare scholen. Volgens Krijn van Beek is het een onzichtbare megaloting en een ideaal middel om klagende ouders te omzeilen, zo schrijft hij in een opinie-artikel in Het Parool.

Niet één middelbare school kiezen maar wel tien

8.000 kinderen kiezen dezer dagen een middelbare school in Amsterdam. Populaire scholen moesten vorig jaar 518 leerlingen uitloten. Nu is het systeem anders. „De pijn wordt beter verdeeld.“

Mirjam Remie Bram Budel

Door MIRJAM REMIE & BRAM BUDEL 27 FEBRUARI 2015

NU IN HET NIEUWS: GRIEKENLAND KIM JONG UN VLUCHTELINGEN

• 18 juni 2015, 10:42

A'damse ouders boos over uitgelote kinderen: scholenkoepel bedreigd



Harlaeus Gymnasium in Amsterdam, een van de populaire scholen. ANP / Kippa

ENLAND Een commissievergadering in het Amsterdamse stadhuis zit vol met boze ouders, zo [lt AT5](#). Ze willen hun onvrede uiten over het aangevoerde 'matchingsysteem', waarmee 7.500 achtstegroepers via een computer op middelbare school zijn geplaatst. Gisteren bleek dat de helpdesk van scholenkoepel naar een geheime locatie is verhuisd vanwege bedreigingen van ouders aan werkers.

door Mirjam Remie

ers zijn boos dat zij niets meer kunnen doen aan de uitslag van de matching. De meeste zijn geplaatst in de top 5 (99 procent) of top 3 (95 procent) van het lijstje van plaatst cookies om een optimale gebruikservaring te kunnen bieden. De cookies worden ingezet en op analisten. Totdat we de website kunnen verbeteren. Daarnaast plaatsen we cookies die het

P versus NP problem



P versus NP



- One of the seven millennium prize problems
 - “In the case of the P versus NP problem and the Navier-Stokes problem, the SAB will consider the award of the Millennium Prize for deciding the question in either direction.”
-
- P not equal NP \Rightarrow 1 million \$
 - P equal NP \Rightarrow 6 million \$

Main characters: Algorithms

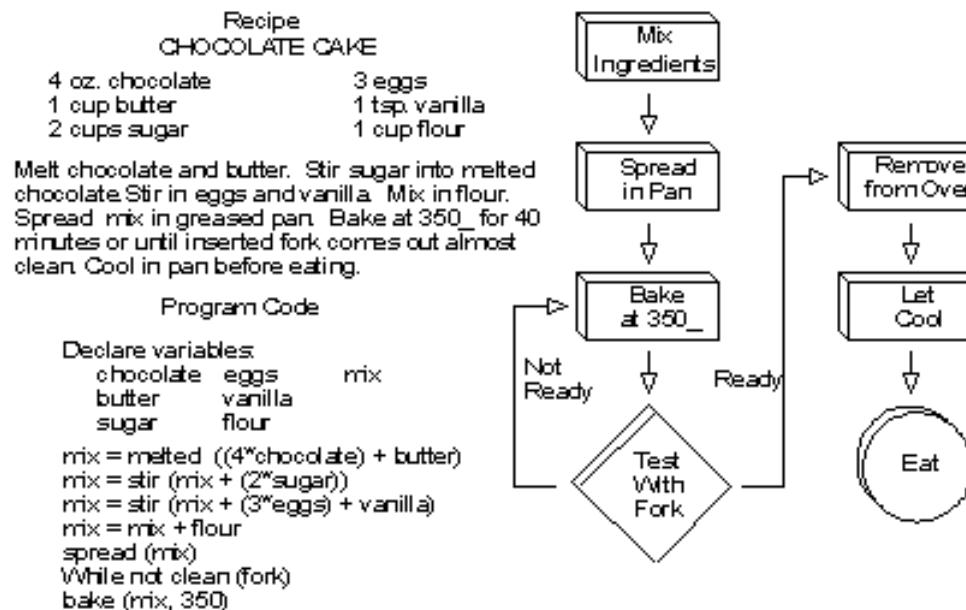
Algorithm

- Algorithm is like a cooking recipe



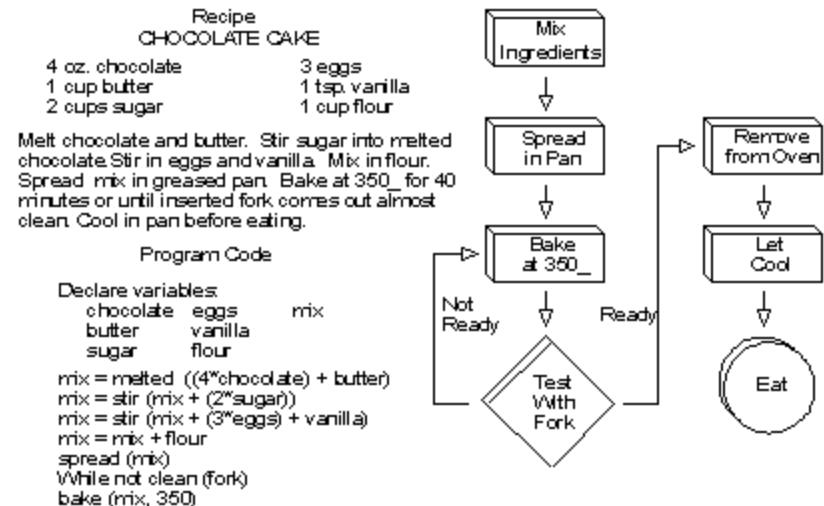
Algorithm

- Algorithm is like a cooking recipe



Algorithm

- algorithm is like a cooking recipe
- input
- computation
 - steps (1 time unit)
- output



Example

Greatest Common Divisor (GCD)

Slow Algorithm

a=21 b=13

step 1 i=13 $13 \nmid 21$

step 2 i=12 $12 \nmid 21$

step 3 i=11 $11 \nmid 21$

step 4 i=10 $10 \nmid 21$

⋮

step 7 i=7 $7 \nmid 13$

⋮

step 13 i=1

output 1

```
slow-gcd(a, b)
  i = min(a,b)
  while i  $\nmid$  a or i  $\nmid$  b
    i := i -1
  output i
```

Analysis of Algorithm

Analysis of alg. is preparation time of recipe

CARL SAGAN'S APPLE PIE

1 universe
1 9" pie shell
6 cups sliced apples
3/4 cup sugar
1/2 cup brown sugar

2 tbsp all-purpose fl
1/2 tsp cinnamon
1/8 tsp nutmeg
1/2 cup all-purpose flour
3 tbsp butter

Preparation time:
12-20 billion years

Servings:
8



Remember -
"If you want
to make an
apple pie
from scratch,
you must first
create the
universe."
-Carl

Preheat oven to **Make the universe as usual.**

Place apples in a large bowl. In a smaller bowl, mix together sugar, 2 tbsp flour, cinnamon, and nutmeg. Sprinkle mixture over apples. Toss until evenly coated. Spoon mixture into pie shell.

In a small bowl mix together 1/2 cup flour and brown sugar. Add butter until mixture is crumbly. Sprinkle mixture over apples. Cover loosely with aluminum foil.

Bake in preheated oven for 25 minutes. Remove foil and bake another 30 minutes, or until golden brown.

Slow Algorithm

a=21 b=13

step 1	i=13	$13 \nmid 21$
step 2	i=12	$12 \nmid 21$
step 3	i=11	$11 \nmid 21$
step 4	i=10	$10 \nmid 21$
	⋮	

step 7	i=7	$7 \nmid 13$
	⋮	

step 13	i=1
---------	-----

```
slow-gcd(a, b)
  i = min(a,b)
  while i ∤ a or i ∤ b
    i := i -1
  output i
```

```
if gcd(a,b)=1 then
  algorithm uses min(a,b) steps
```

output 1

Better Algorithm

Euclidean Algorithm

Greatest Common Divisor (GCD)

step 0 a=21 b=13

step 1 a=13 b= 21 mod 13 = 8

step 2 a=8 b= 13 mod 8 = 5

step 3 a=5 b= 8 mod 5 = 3

step 4 a=3 b= 5 mod 3 = 2

step 5 a=2 b= 3 mod 2 = 1

step 6 a=1 b= 2 mod 1 = 0

 output 1

```
function gcd(a, b)
  while b ≠ 0
    t := b
    b := a mod b
    a := t
  output a
```

Analysis GCD-Algorithm

- worst case number of steps?

Theorem
alg. terminates in
 $2\log(m) + 1$ steps
 $m = \max(a, b)$

Proof:
every second step **a** is
at least halved

```
function gcd(a, b)
  while b ≠ 0
    t := b
    b := a mod b
    a := t
  output a
```

step 0	a=21	b=13
step 1	a=13	b= 21 mod 13 = 8
step 2	a=8	b= 13 mod 8 = 5
step 3	a=5	b= 8 mod 5 = 3
step 4	a=3	b= 5 mod 3 = 2
step 5	a=2	b= 3 mod 2 = 1
step 6	a=1	b= 2 mod 1 = 0

Complexity

- Euclid: $2\log(m) + 1$ $m = \max(a, b)$
- Slow: m' $m' = \min(a, b)$
- length of the input: $\log(a) + \log(b) = n$

Euclid: $O(n)$

Slow: $2^{O(n)}$

- Euclid **exponentially** faster than slow!
- **Complexity of computational problem** is running time of the **best** algorithm

Computation & Complexity

- Computational problem:
 - INPUT $\xrightarrow{\text{computation}}$ OUTPUT
 - Example: a, b output $\text{gcd}(a, b)$
- Complexity:
 - Number of computation steps needed for “best” algorithm
 - function of the input size

Complexity

- Determine the **complexity** of a computational problem:
 - Upper bound: construct algorithm
 - Lower bound: **any** algorithm needs this many steps
- Ideally upper bound = lower bound

↑
↑
functions of the input size



Complexity of gcd problem

- Euclid's algorithm runs in $O(n)$ steps
- Can we devise a faster algorithm?
- Not really: **any** algorithm has to read the whole input: requires n steps
 - Upper Bound: $O(n)$
 - Lower Bound: $\Omega(n)$
- Complexity of gcd is linear.

Complexity Class P

Feasible Problems: P

- Feasible or efficient algorithms run in polynomial time: n^c (some c)
- Complexity Class **P** :
 - All the problems that have feasible algorithms
- Example:
 - Linear Programming
 - Network Flow Problems
 - Shortest Path

For these problems upper bound is “close” to lower bound: at most polynomial far off.

Another problem
Satisfiability

Satisfiability

- variables $x_1 \dots x_n$
- Clause $C_1 \dots C_m$ $C_l = (x_i \vee x_j \vee \overline{x_k})$
- formula $\phi(x_1 \dots x_n) = C_1 \wedge \dots \wedge C_m$
- exist $\alpha_1 \dots \alpha_n$ $\alpha_i \in \{T, F\}$
- such that $\phi(x_1 = \alpha_1 \dots x_n = \alpha_n) = T$

Example

$$\begin{array}{ccccccccc} F & F & T & T & T & F & F & T \\ (\overline{x_1} \vee \overline{x_2}) \wedge (x_1 \vee x_3) \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3) \\ \downarrow & \downarrow & \downarrow & & \downarrow & & \downarrow \\ F & \wedge & T & \wedge & T & \wedge & T \\ & & & & \downarrow & & \\ & & & & F & & \end{array}$$

$$x_1 = T$$

$$x_2 = T$$

$$x_3 = T$$

Example

$$(\overline{x_1} \vee \overline{x_2}) \wedge (x_1 \vee x_3) \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3)$$

\downarrow \downarrow \downarrow \downarrow

$$F \quad \wedge \quad T \quad \wedge \quad F \quad \wedge \quad T$$

SATISFIABLE

$x_1 = F$ $x_2 = T$ $x_3 = T$

Satisfiability

- variables $x_1 \dots x_n$
- Clause $C_1 \dots C_m$ $C_l = (x_i \vee x_j \vee \overline{x_k})$
- formula $\phi(x_1 \dots x_n) = C_1 \wedge \dots \wedge C_m$
- exist $\alpha_1 \dots \alpha_n$ $\alpha_i \in \{T, F\}$
- such that $\phi(x_1 = \alpha_1 \dots x_n = \alpha_n) = T$

$$SAT = \{\phi \mid \phi \text{ is satisfiable}\}$$

simple algorithm: try all 2^n assignments

Unknown Complexity

- It is hard to determine the complexity of many problems
- Example:
 - Is this formula satisfiable? **SAT**
 - Traveling Salesman Problem. **TSP**
- Lower Bound: n
- Upper Bound: 2^n



Best Known!

Complexity Class NP

NP

- complexity class NP
 - polynomial time to check solution
- $x \in L$: exists a y : $P(x,y) = 1$ (true)

polynomial time computable in length of x only

SAT in NP

φ is satisfiable
 $\exists \alpha : \varphi(\alpha) = \text{True}$

P & NP

- complexity class NP
 - easy to check solution
 - polynomial time check
 - easy to check assignment is satisfiable

versus

- complexity class P
 - easy to find solution
 - decide in polynomial time
 - compute in polynomial time $\gcd(a,b)$

$$P \subseteq NP$$

SAT

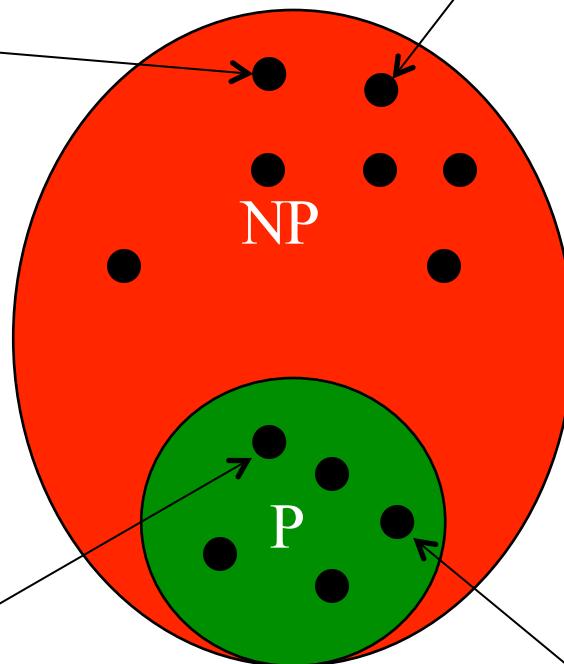
TSP

strictly, gcd is a function and not a set. Will ignore this distinction here

gcd

Many many more problems fall into this classification

Primality



Reductions & Completeness

reduction

$$A \leq_T^p B$$

compute A in poly-time with B as free subroutine

“A is computationally not harder than B”

“if B in P then A in P”

C is NP-complete

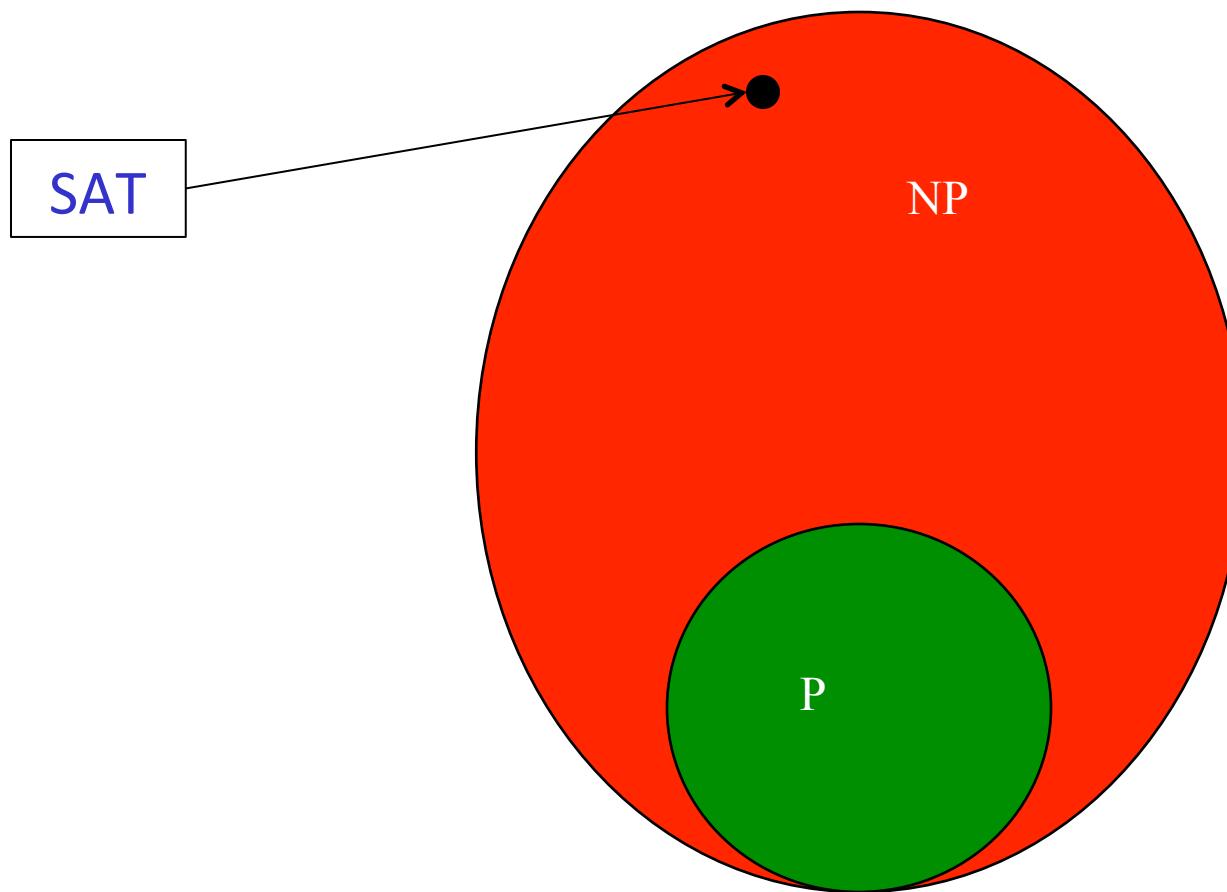
- $C \in NP$
- all $A \in NP: A \leq_T^p C$

Theorem

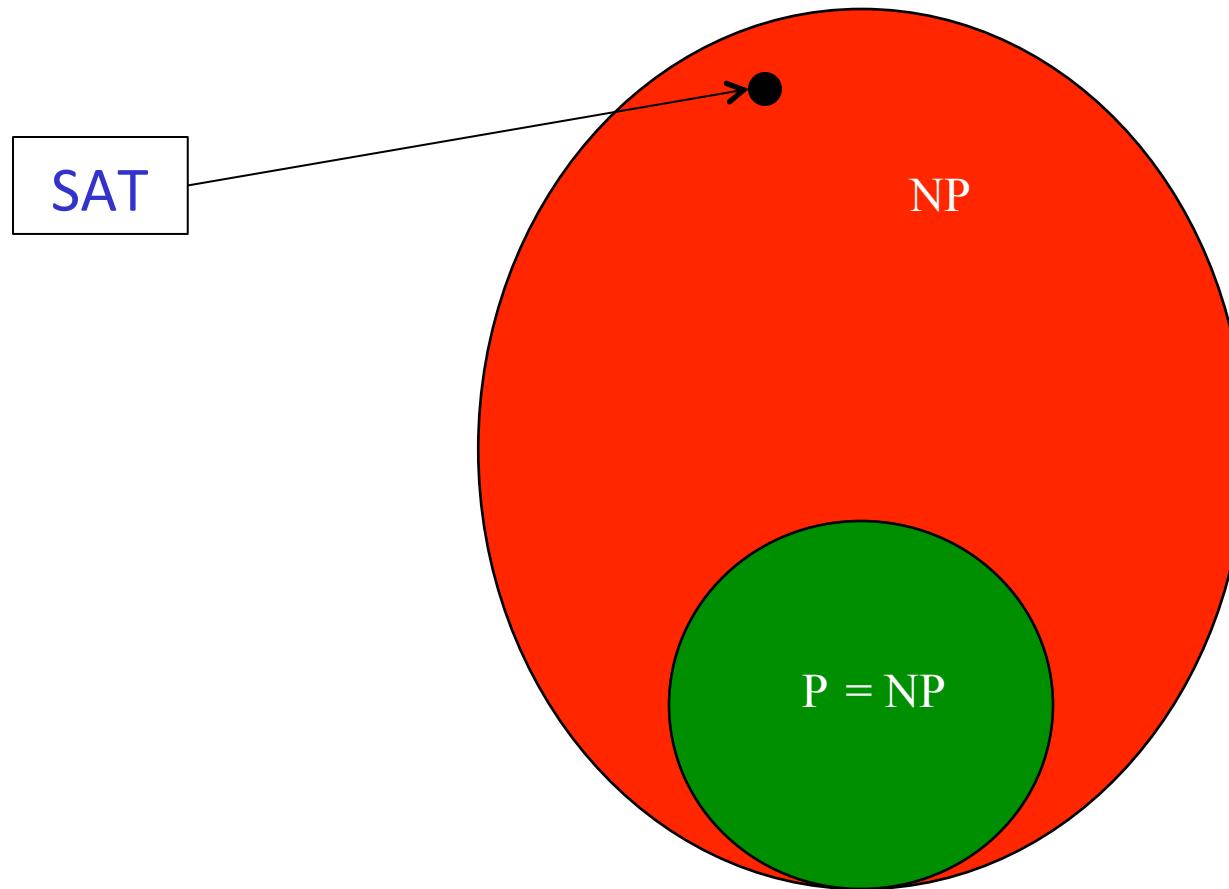
- SAT, TSP, many others NP-complete
- $SAT \in P \Leftrightarrow P=NP$

P versus NP

$P \neq NP$



$P = NP$



P versus NP Question

- $P = NP?$
- widely believed that $P \neq NP$
- how to show this is true?
 - Prove better lower bounds for existing problems like SAT
 - Construct problem in NP with super polynomial lower bound

Lower Bounds

- Construct $D \in NP$
 - no poly-time algorithm solves D
 - for every poly time algorithm M exists a string x such that:
 - $M(x) = 1 \text{ & } x \notin D$ or
 - $M(x) = 0 \text{ & } x \in D$
- $\Rightarrow D \text{ not in } P$

$$D \leq_T^p SAT \Rightarrow SAT \text{ not in } P$$

Diagonalization

How big are the reals ?

- Cantor showed \mathbb{R} not enumerable
- diagonalization
 - given an enumeration of the reals
 - construct real number d not in the enumeration

Diagonalization

reals in some enumeration

	1	2	3	4	5	6	7	8	→
r ₁	0.8	1	0	7	7	4	1	5	
r ₂	0.3	2	1	4	8	6	7	3	
r ₃	0.5	3	9	7	7	9	4	1	
r ₄	0.7	6	9	6	5	7	9	4	
r ₅	0.8	3	6	8	9	5	1	4	
r ₆	0.8	7	9	3	4	6	0	2	
r ₇	0.9	8	5	2	5	3	1	3	
r ₈	0.9	9	3	1	2	3	0	4	

$$d = 0.9 \ 3 \ 0 \ 7 \ 0 \ 7 \ 2 \ 5 \dots$$

ith digit of d is ith entry of diagonal +1

Diagonalizing out of P

Diagonalization (2)

polynomial time algorithms

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	→
M_1	0	1	0	0	0	0	1	1	
M_2	1	1	1	0	0	0	0	1	
M_3	0	1	1	0	0	0	1	1	
M_4	0	0	1	0	1	0	0	0	
M_5	1	0	0	1	1	0	1	0	
M_6	1	1	0	1	1	0	0	1	
M_7	0	1	0	1	0	0	1	1	
M_8	0	1	1	1	0	0	0	1	

↓

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	$\in D$	$\notin D$	$\notin D$	$\in D$	$\notin D$	$\in D$	$\notin D$	$\notin D$

x_i in D if and only if $M_i(x_i)=0$

Diagonal Language

$$D = \{x_i \mid M_i(x_i) = 0\}$$

ith poly-time algorithm/machine

D \notin P, every poly-time machine errs on some input

D \in NP ?? probably not, but

D \in time($n^{\log n}$), quasi polynomial time

with more time can compute more

More Bad News

- Relativization (Oracles):
 - Exists oracle A: $P^A = NP^A$
 - (Exists oracle B: $P^B \neq NP^B$)

Proof technique should not relativize



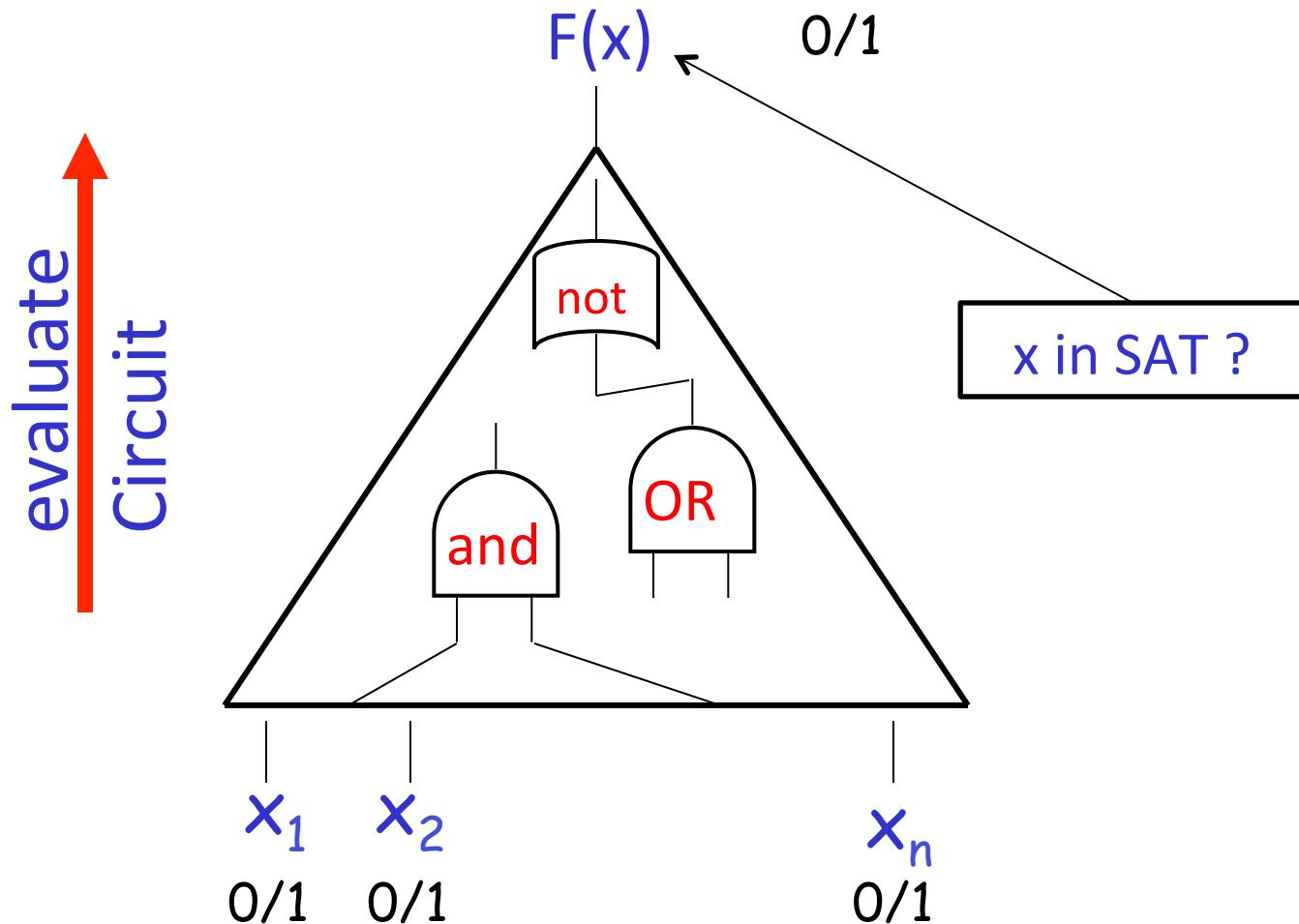
Diagonalization and
most other techniques
we know
relativize

Try something easier

- Study weaker models of computation and develop new lower bound techniques
 - Circuits with small depth
 - Monotone circuits
 - Decision Trees
 - Branching Programs
- The weaker the model the better the lower bounds!

Simple model:
Circuits

Circuit Model of Computation

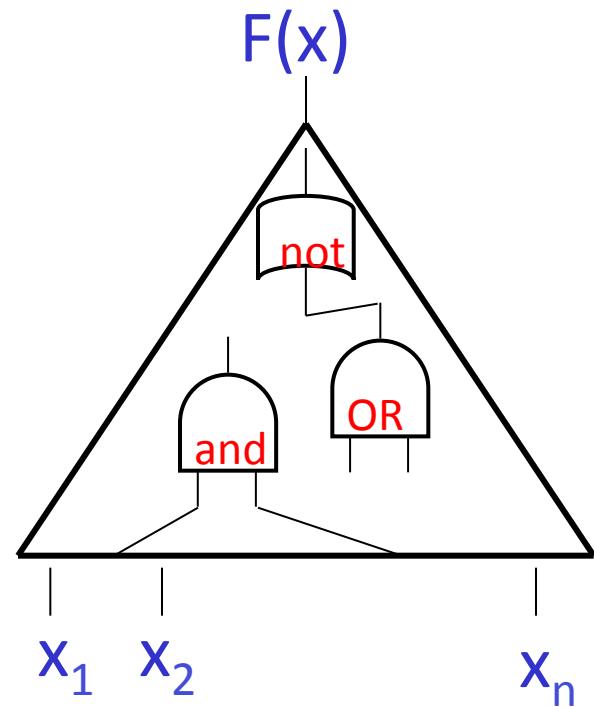


Size of the Circuit

1. most important:
number of gates

2. Depth of the circuit

Parallel time of computation



Constant Depth

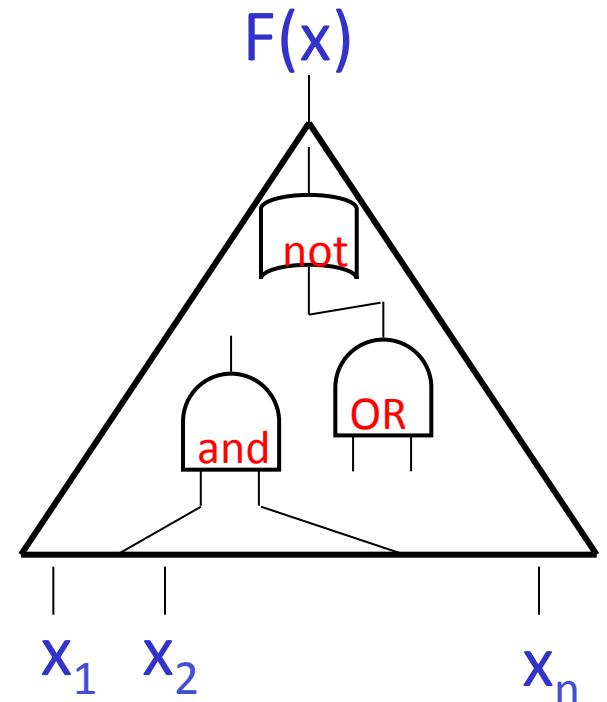
depth is constant
size is polynomial

AC^0

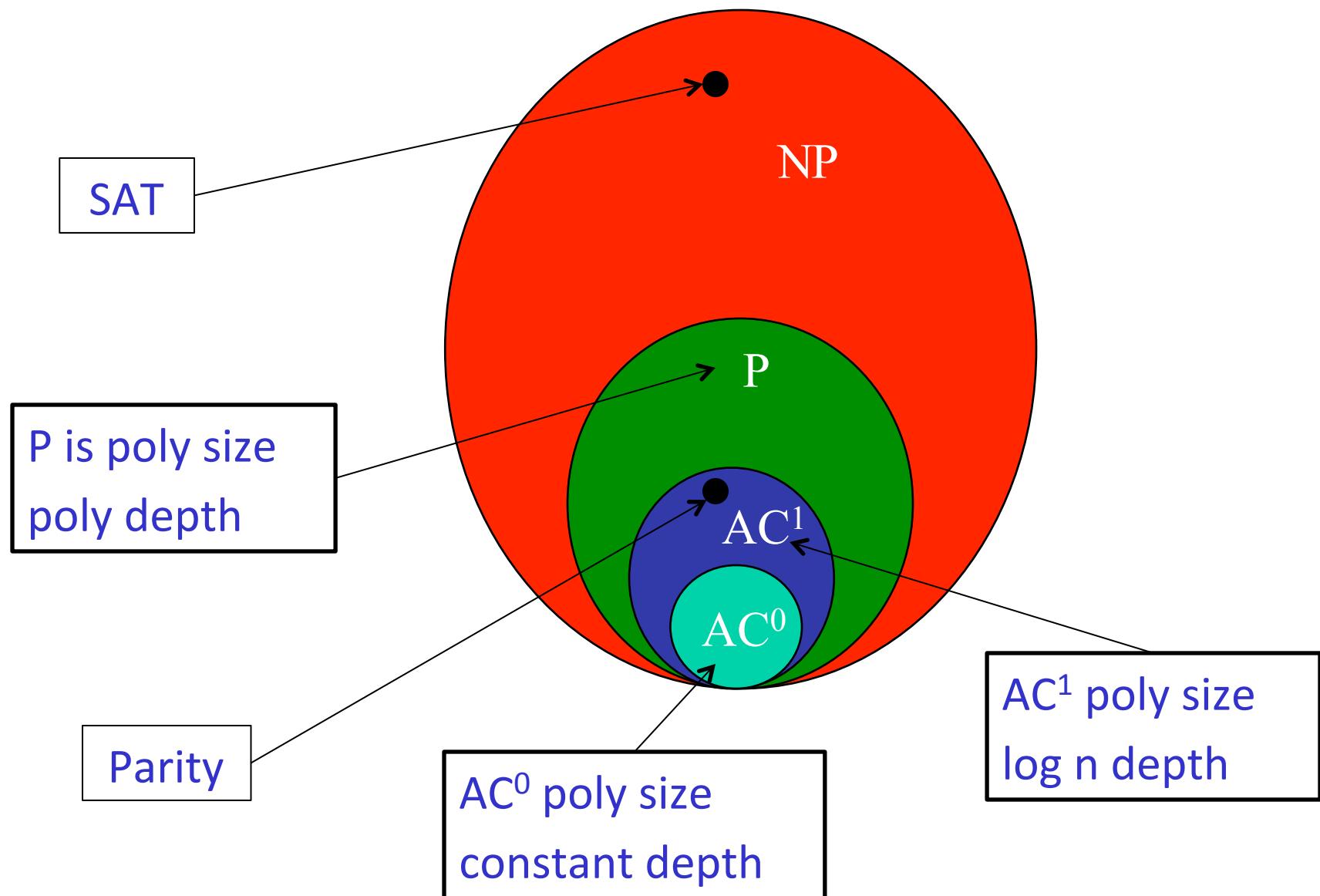
compute parity:
 $F(x) = x_1 + x_2 + \dots + x_n \bmod 2$

Theorem
parity requires $2^{n^{1/d}}$ size circuits
of depth d

Note: $d = \log n$ bound is meaningless



NP = AC¹ ?



natural proofs another hurdle?

- proof technique that shows parity not in AC^0 likely won't work to separate P from NP
- these proofs fit in a framework called **natural proofs**

Theorem

if one-way functions exist then

natural proofs can't separate P and NP

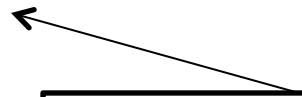
Approaches

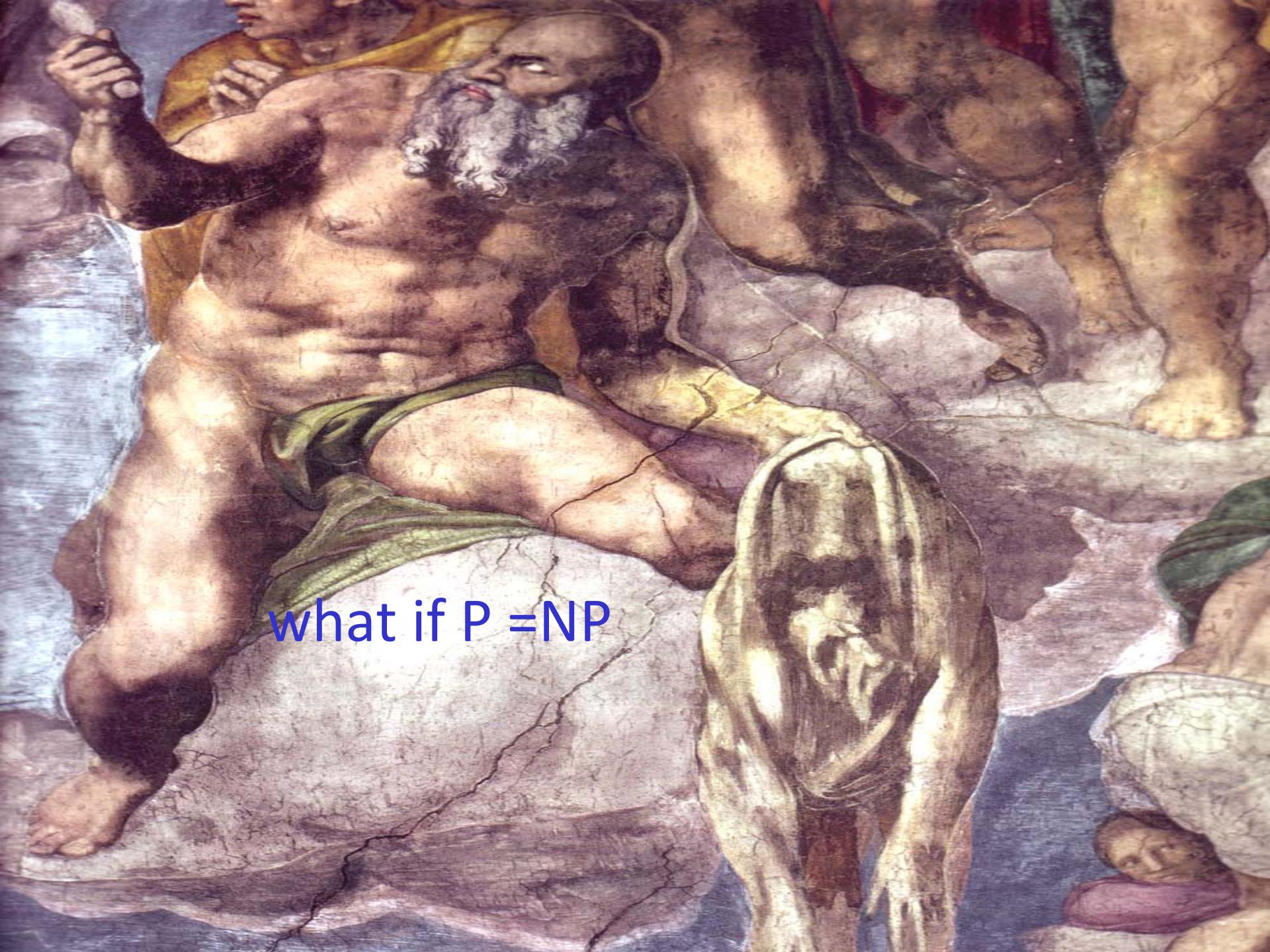
- Structural approach using eg. autoreducibility
- Combinatorial approach
- Algebraic, degrees of multivariate polynomials
- Geometric Complexity
 - algebraic geometry
 - representation theory
- Communication complexity

P vs NP & Cryptography

- computational **hardness** guarantees **security** of cryptographic protocols
 - factoring, discrete logarithm
 - lattice problems
 - learning problems
- one-way functions
 - compute $f(x)$ **quickly**
 - **hard** to invert
- if $P=NP$ then no cryptography

efficient on
quantum computer



A detail from Michelangelo's 'The Last Judgment' fresco. It depicts a bearded man with a long white beard and hair, wearing a green cloth wrapped around his waist. He is being pulled by a large, coiled snake. The snake's head is near the man's neck, and its body is wrapped around his torso. The man has a look of distress on his face. In the background, other figures are visible, some appearing to be in torment. The style is characteristic of Renaissance religious art, with detailed anatomical rendering and dramatic lighting.

what if $P = NP$

P=NP

- P=NP, **but** the proof does not give us an algorithm
- P=NP, **but** algorithm for SAT runs in time $n^{1000000}$
- P=NP, **but** algorithm for SAT runs in time $2^{100}n$
- P=NP, **and** algorithm for SAT runs in time n^2

n^2 algorithm for SAT

- Wonderful!!!
 - computing ground states of Hamiltonians
 - protein folding problem solved
 - artificial Intelligence takes really off
 - optimal scheduling
 - computational learning theory
 - weather prediction improves

n^2 algorithm for SAT

- For mathematics
 - can find proofs to theorems, provided they have short proofs
 - can simply ask computer whether theorem/conjecture is true/false
 - mathematics will change dramatically
 - quickly solve the other 5 remaining Clay problems

Summary

- P versus NP central, not just in mathematics and computer science but also in physics, biology, chemistry, cryptography etc.
- Not clear how to attack it, several obstacles: relativization, natural proofs, algebraization
- Much simpler questions are still way out of reach
- If $P=NP$, the world would drastically change, with lots of fantastic application, but no privacy (cryptography).

Schedule

- 2) P, NP, reductions, co-NP
- 3) Cook-Levin Thm: 3-SAT is NP-complete, Decision vs Search
- 4) Diagonalization, time hierarchies
- 5) Relativization
- 6) Space complexity, PSPACE, L, NL
- 7) The polynomial hierarchy
- 8) Circuit complexity, the Karp-Lipton Theorem
- 9) Parity not on AC^0
- 10) Probabilistic algorithms
- 11) BPP, circuits and polynomial hierarchy
- 12) Interactive proofs, Graph-Isomorphism problem
- 13) IP = PSPACE
- 14) Derandomization/Catalytic Space