

# Data checking and verification

## Randomization and coding

A guide to PCP, an approach towards  
classification of NP problems

# P versus NP

Have you heard about P vs NP problem?

- Millennium prize problems by Clay institute

<http://www.claymath.org/millennium/>

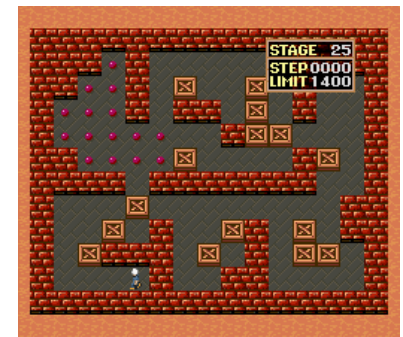
- P versus NP is not only million dollar problem, Its value is trillion dollars.

# Class P and class NP

- Consider a computational problem  $M$  of input size  $n$
- P: class of polynomial time solvable problems in usual model of computation
  - $M$  is in P if there is an algorithm to solve it in polynomial time
    - Example: Sorting, Searching, Graph connectivity
    - Shortest path, text pattern matching
- NP: class of polynomial time solvable problems in nondeterministic model of computation
  - $M$  is in NP if there is a nondeterministic algorithm to solve it in polynomial time (difficult to understand, isn't it?)
- P is a subclass of NP

# More intuitive description

- Verification is often easier than solution
  - Solving an equation is more difficult than verification
- NP: If we know the solution, we can show it such that verification is done in polynomial time
  - 1: Hamilton path
  - 2: Graph coloring,
  - 3: Does this SUDOKU puzzle have a solution?
- A problem (probably) not in NP
  - 1: Will black win this Chess game ?
  - 2: SOKOBAN game
  - 3: #3SAT (number 3SAT)
- Intuitively, a problem that we are convinced with correctness of solution easily if it is shown.



# NP and co-NP

- It is notable that negation of an NP problem might not be in NP
  - Is a given graph  $G$  is colored by using  $k$  colors?
    - We give the actual coloring to verify. Thus NP problem
  - Is a given graph  $G$  has no coloring with  $k$  colors?
    - If so, how to convince it?
- Co-NP problem: negation of an NP problem
- It is believed NP and co-NP are different
- It might be true that  $NP \cap coNP = P$

# P vs NP

- Is it true that verification is easier than solution ?
  - Consider an equation  $f(x) = x^5 - 3x^4 + 3x^2 + 4 = 0$
  - Ask “ is this equation has an integer solution less than n (say, n=100)”.
  - Verification:  $f(2) = 32 - 48 + 12 + 4 = 0$
  - Solution???
- P vs NP: Is there any problem where verification is easy (P-time) but solution is difficult (not in P-time)?
- Philosophical question
  - Is it easy to learn than solve by yourself?
    - Is it true that a difficult problem remains to be difficult even if you are suggested a solution (you must make sure it is true).
  - In real life, we must solve many NP problems(or even more difficult problems). Human can solve them by training (like SUDOK) for most of instances. Why??

# Implication of $P=NP$

- If  $P=NP$ , we can solve many problems
  - Many-body problem in physics
  - Protein folding problem in biology
  - Optimal scheduling in manufacturing
  - Optimal traffic control
  - Many problems in computational chemistry
- If  $P=NP$ , we have serious inconvenience
  - Current cryptology assumes  $P$  is not  $NP$
  - Information security system is destroyed
- Most of researchers believes  $P$  is not  $NP$ , and the above situation only occurs in Scientific Fiction
  - But no one knows the truth
- The biggest mathematical challenge in 21<sup>st</sup> century

# Several approaches towards P vs NP

<http://people.cs.uchicago.edu/~fortnow/papers/pnp-cacm.pdf>

Lance Fortnow's article

- NP- complete theory (S.Cook and R.Karp)
  - Almost all NP problems are either in P or NP-complete( one of the most difficult problems in NP)
- Circuit complexity
  - Show NP needs exponential size of circuits
- Algebraic/group theoretic method
  - Relation to generalized Riemann hypothesis (Mulmuley)
- From mathematical logic
- Relation to randomness
- **Interactive Proof and Checkable Proof**



# Textbooks/papers

- Randomized Algorithms (Motwani-Raghavan)
- Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems (M. Sudan, ACM Distinguished Thesis, 1995)
- Proof Verification and the Hardness of Approximation Problems (S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy), J. ACM, Vol 45(3), 1998, pp. 501-555

# Rajeev Motwani (from wikipedia)

- Motwani joined Stanford soon after U.C. Berkeley. Motwani was one of the co-authors (with [Larry Page](#) and [Sergey Brin](#), and [Terry Winograd](#)) of an influential early paper on the [PageRank algorithm](#), the basis for Google's search techniques. He also co-authored another seminal search paper *What Can You Do With A Web In Your Pocket* with those same authors.<sup>[3]</sup>
- He was also an author of two widely-used theoretical computer science textbooks, *Randomized Algorithms* (Cambridge University Press 1995, [ISBN 978-0521474658](#), with Prabhakar Raghavan) and *Introduction to Automata Theory, Languages, and Computation* (2nd ed., Addison-Wesley, 2000, with [John Hopcroft](#) and [Jeffrey Ullman](#)).
- Prior to his involvement with Google, Motwani founded the Mining Data at Stanford project (MIDAS), an umbrella organization for several groups looking into new and innovative data management concepts. His research included [data privacy](#), [web search](#), [robotics](#), and [computational drug design](#).
- He was an avid [angel investor](#) and had funded a number of successful startups to emerge from Stanford. He sat on the boards of Google, [Kaboodle](#), [Mimosa Systems](#), [Adchemy](#), [Baynote](#), [Vuclip](#), NeoPath Networks (acquired by [Cisco Systems](#) in 2007), [Tapulous](#) and [Stanford Student Enterprises](#) among others. He was also active in the [Business Association of Stanford Entrepreneurial Students](#) (BASES).<sup>[4][5][6]</sup>
- He was a winner of the [Gödel Prize](#) in 2001 for his work on the [PCP theorem](#) and its applications to [hardness of approximation](#).<sup>[7][8]</sup>

# Verification

Algebraic methods to verify  
information

# Verification in real life

- Given two data A and B, we want to verify  $A=B$ 
  - We want to do it without reading A and B explicitly
- If A and B are “persons”
  - How we can “read” data of persons completely?
    - Impossible!
  - Name, blood type, color of hair/eye,.....
  - ID numbers, secret keywords,
  - Finger print, DNA identification
    - If A and B are twins.....

# Fingerprinting

- We recall the data checking problem we discussed some weeks ago
  - Given a set of  $n$  data  $\{a(1), a(2), \dots, a(n)\}$ , and we suspect one (or more) data is modified.
  - How should we check efficiently?
    - The sum of all data
    - Use hash function, and sum of  $h(a(i))$
  - This is an example of “fingerprint”

# Fingerprinting for matrix multiplication

- Consider a prime  $p$
- Given  $n$  by  $n$  matrices  $A$ ,  $B$ , and  $C$ , we want to verify  $AB = C \pmod{p}$ 
  - Computation in the field  $GF(p)$  (or  $\mathbb{Z}_p$ )
- Can you do it in  $O(n^2)$  time?

# Pattern matching via verification

- Given a text  $T$  of length  $n$  (bit string)
- For each query pattern  $P$  of length  $m$ , we want to find location of occurrence of  $P$  in  $T$
- Both  $m$  and  $n$  are long (say,  $m = 100000$ ,  $n = 10000000$ )
- KMP algorithm, BM algorithm: optimal  $O(n+m)$ , but not much practical
- Can we apply verification idea?
- **SOLUTION on blackboard (you can find it in Motwani-Raghavan book)**

# Verifying an identity

- $X = (x_1, x_2, \dots, x_d)$
- We want to verify a polynomial identity  $F_1(X) = F_2(X)$  of degree  $n$

$$\Delta(x_1, \dots, x_n) \equiv \begin{vmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{vmatrix}$$
$$= \prod_{i < j} (x_i - x_j)$$



# Verifying an identity

- $X = (x_1, x_2, \dots, x_d)$
- We want to verify a polynomial identity  $F_1(X) = F_2(X)$  of degree  $n$ 
  - Or more cruel identities for which you cannot probably understand its mathematical proof (like the one below)

**THEOREM 5.1 (Tokuyama [70])** *We have*

$$\sum_{\substack{\mathfrak{T} \in \text{GT}(\lambda + \rho) \\ \mathfrak{T} \text{ strict}}} (t + 1)^{s(\mathfrak{T})} t^{l(\mathfrak{T})} z_1^{d_r} z_2^{d_{r-1} - d_r} \dots z_{r+1}^{d_0 - d_1} =$$
$$\left\{ \prod_{i < j} (z_j + z_i t) \right\} s_\lambda(z_1, \dots, z_{r+1}).$$

# Prover and Proof

Prover is stronger than proof, since we can ask questions instead of reading the proof



# Interactive Proof

- Suppose that you can ask a god (or a powerful supercomputer ) to solve a problem
- If the answer is “YES” (or “NO”), do you believe it blindly?
  - In real life, we blindly believe the weather forecast which a computer software reports
  - In ancient Greek, people believed “oracle of Apollo”.
  - If you are given a program/software , how you believe it?
    - Or how you write a program convincing others its correctness?
  - Even in a university, most of students (and professors) believe Wikipedia blindly ????
- We want to request a proof or an evidence.
  - For a NP problem, we can ask for a proof if the answer is Yes
  - But if answer is NO, can we do something?

# Graph non-isomorphism

- You have two graphs  $G=(V,E)$  and  $G'=(V',E')$
- You suspect that they are isomorphic
  - There is a one-to-one map  $f$  of vertices of  $G$  such that  $(x,y)$  is an edge of  $E$  if and only if  $(f(x), f(y))$  is in  $E'$
- You ask your professor who says he is always honest and can solve the problem for any pair of graphs.
  - If he answers “yes”, you can ask him to show the map  $f$ .
  - Can you believe him if he says “NO”?

# An interactive proof

- You (verifier) have  $G$  and  $G'$ , and ask your professor (prover) whether  $G=G'$
- Professor answers “NO”, but you suspect that he tells a lie.
- You ask some more questions to the professor to reveal whether he is honest
  - You can flip a coin, and the random choice is not known to the professor
- **SOLUTION on the blackboard** (you can easily find it in several textbooks)

# How IP is strong

- I will show that #3SAT is solved by using interactive proof system
  - #3SAT: Find the number of solutions of a logical equation (given in a certain form)
- This implies #P is in IP, and co-NP is in IP
  - do not worry about such terminology
- A. Shamir showed that  $IP = PSPACE$ 
  - PSPACE is considered to be larger than NP
  - 2 player's game like GO and Chess are in PSPACE

# 3SAT and #3SAT

- 3SAT: Is a logic equation  $F(X(1), X(2), \dots, X(n)) = 1$  in 3-CNF formula has a solution?
- #3SAT: How many solutions  $F(X)=1$  has?
- 3SAT is an NP-complete problem
  - Any NP problem can be transformed into a 3SAT problem in polynomial time.
  - If 3SAT is in P, then  $NP = P$
  - To show 3SAT is NP-complete
    - SAT is NP-complete (Cook's theorem)
    - SAT is transformed into 3SAT
- #3SAT is more difficult (called #P complete problem)
  - Toda's theory (1989, Goedel award)

# From SAT to 3SAT

- Show that SAT is P-time soluble if 3SAT is.
- Given an instance  $(U, \mathbf{C})$  of SAT, we show a transformation of it into  $(U', \mathbf{C}')$  of 3SAT
  - $U$ : set of variables,  $\mathbf{C}$ : set of clauses
    - $U = \{X(1), X(2), \dots, X(n)\}$ ,  $\mathbf{C} = \{C(1), C(2), \dots, C(m)\}$
    - $c(j) = l_{1,j} \vee l_{2,j} \vee \dots \vee l_{k,j} = \{l_{1,j}, l_{2,j}, \dots, l_{k,j}\}$
- For each clause  $C$  of  $\mathbf{C}$  of length  $k$ , we consider  $k-3$  new variables, and transform it into a set of clauses each of which has length 3
- Thus, we transform into 3SAT input with at most  $nm$  variables and  $mn$  clauses



# Transforming a clause

- $C = \{z(1), z(2), \dots, z(k)\}$ 
  - $z(i)$  is either  $X(i)$  or its negation
- We define new variables  $y(1), y(2), \dots, y(k-3)$ 
  - These variables are only used to transform  $C$
- The clause  $c$  is transformed into
  - $S(c) = \{\{z(1), z(2), y(1)\}, \{\cancel{y(1)}, z(3), y(2)\}, \{\cancel{y(2)}, z(4), y(3)\}, \dots, \{\cancel{y(k-3)}, z(k-1), z(k)\}\}$
  - $c$  is satisfied if and only if all clauses in  $S(c)$  are satisfied

# Interactive proof for #3SAT

## Step 1: Arithmetization

- $F(X(1), X(2), \dots, X(n))$  = a logical function in 3CNF
  - $C(i) = L(i,1) \vee L(i,2) \vee L(i,3)$
  - $F = C(1) \wedge C(2) \wedge \dots \wedge C(m)$
- Transform  $F$  into a real function  $f$
- For each literal  $L = L(i, j)$ , we define  $l(i,j) = 1-x(k)$  if  $L=X(k)$  and  $l(i,j) = x(k)$  if  $L=\overline{X(k)}$
- $c(i) = 1 - l(i,1)l(i,2)l(i,3)$
- Observation:  $C(i)$  is satisfied if and only if  $c(i)=1$
- $F = c(1)c(2)\dots c(m)$ : a polynomial in  $x(1), \dots, x(n)$

# Number of solutions

- If  $F = (X(1) \vee X(2) \vee \cancel{X(3)}) \wedge (\cancel{X(1)} \vee \cancel{X(3)} \vee X(4))$ ,
- $f = (1 - (1 - x(1))(1 - x(2))x(3)) (1 - x(1)x(3)(1 - x(4)))$
- Number of solutions of  $F=1$  is  $7 + 5 = 12$ .
- Define  $\#f = \sum_{x(1)=0}^1 \sum_{x(2)=0}^1 \dots \sum_{x(n)=0}^1 f(x(1), x(2), \dots, x(n))$
- Then,  $\#$  of solutions of  $F=1$  equals  $\#f$
- Computation is difficult, but your professor who says he can compute  $\#f$  for any  $f$ .
- Ask the professor “What is  $\#f$ ”, and he answers that “its value is  $s$ ”. (say, “its value is 2487000”)
- Verify whether he tells a truth.

# Our model

- Define  $\#f = \sum_{x(1)=0}^1 \sum_{x(2)=0}^1 \dots \sum_{x(n)=0}^1 f(x(1), x(2), \dots, x(n))$
- Computation is difficult, but your professor who says he can compute  $\#f$  for any  $f$ .
- Ask the professor “What is  $\#f$ ”, and he answers that “its value is  $s$ ”. (say, “its value is 2487000”)
- Moreover, the professor will answer to any computation request.
  - Thus, you can give any modified question.

# Key idea

$$\# f = \sum_{x(1)=0}^1 \sum_{x(2)=0}^1 \dots \sum_{x(n)=0}^1 f(x(1), x(2), \dots, x(n))$$

$$f_i(x(1), x(2), \dots, x(i)) = \sum_{x(i+1)=0}^1 \sum_{x(i+2)=0}^1 \dots \sum_{x(n)=0}^1 f(x(1), x(2), \dots, x(n))$$

## Lemma

- $f_0 = \#f$

- $f_1(x(1)) = \sum_{x(2)=0}^1 \sum_{x(3)=0}^1 \dots \sum_{x(n)=0}^1 f(x(1), x(2), \dots, x(n))$

- $f_n(x(1), \dots, x(n)) = f(x(1), x(2), \dots, x(n))$

- $f_{j-1}(x(1), \dots, x(j-1)) = f_j(x(1), \dots, x(j-1), 0) + f_j(x(1), \dots, x(j-1), 1)$

# Basic (failing) strategy 1

- Question 1: What is the value of  $f_0$ ?
  - Professor answers : 247800
- Question 2: What are  $f_1(0)$  and  $f_1(1)$ ?
  - Professor answers: 4000 and 243800
  - You check  $f_0 = f_1(0) + f_1(1)$
  - If check fails, professor tells a lie: The end.
  - Else, guess which of two values is wrong.....
- Question 3: What are  $f_2(0,1)$  and  $f_2(0,0)$ ?

# Basic (failing) strategy 2

- Question 1: What is the value of  $f_0$ ?
  - Professor answers : 247800
- Question 2: What is the function  $f_1(z)$ ?
  - Professor answers:  $222800z^6 + 12000 z^5 + 4000$ .
  - You check  $f_0 = f_1(0) + f_1(1)$
  - If fail, professor tells a lie: The end. Else, continue
- Question 3: What is the function  $f_2 (x(1),x(2))$ ?
- Proceed this process

# Successful strategy

- Question 1: What is the value of  $f_0$ ?
  - Professor answers : 247800
- Question 2: What is the function  $f_1(z)$ ?
  - Professor answers:  $g_1(z) = 222800z^6 + 12000 z^5 + 4000$ .
  - You check  $f_0 = g_1(0) + g_1(1)$
  - If fail, professor tells a lie: The end. Else, continue
- Select a random value  $r$ , and compute  $g_1(r)$ 
  - say,  $r = 367$
- Question 3: What is the function  $f_2(367, z)$ 
  - Professor answers:  $g_2(z) = 34800 z^5 + 34900 z^2 + 403000$
  - You check  $g_1(r) = g_2(0) + g_2(1)$
- Next, select another random value  $r'$ , and compute  $g_2(r')$
- CONTINUE



# Analysis

- All functions are considered in  $GF(p)$  for a prime  $p > 2^n$
- What is the probability that  $g_i(z)$  is not  $f_i(z)$  but  $g_i(r) = f_i(r)$  ?
  - In other words, algorithm does not detect the lie in the  $i$ -th step
- Error probability is at most  $3nm/p$ , VERY SMALL
- So if the professor tells a lie, the system detects it with high probability.

PCP

# PCP (probabilistic checkable proof)

- Instead of god, we give a written proof.
- For a NP problem, we have a proof of length  $\text{poly}(n)$ 
  - But, verifier wants to save time to verify
  - You prepare a proof such that verifier can easily verify the correctness
- This is just like database query!
- Like a database, we prepare the proof in a nice structure.
  - We need help of randomness and error correcting code

# Puzzle 1

Captain Cook hid a great treasure, but he needs to escape for a long time to prevent being arrested.

He will send letters to his 20 pirates to inform the location of the treasure, but they are only reliable if they watch each other.

So, he wants to encode the secret key so that it is revealed if and only if 11 or more meet.

How should he do?

# Popular error correcting codes

- Reed solomon code
  - Your CD is encoded by using it
  - Use a polynomial on a field  $F$ 
    - $F = GF(2^q)$  in practical implementation
    - Here, we use  $GF(p)$  for a prime  $p$
- Hadamard code
  - Use randomness
  - We can correct very large error.

# Reed Solomon code

- $a(1)a(2),\dots,a(k)$  :the key we want to send
  - Each  $a(i)$  is a member of  $GF(p)$ , thus a large number.
- Let  $F(x) = a(k)x^k + a(k-1)x^{k-1} + \dots + a(1)x + a(0)$
- We randomly select  $m > k$  values  $x(i)$  and let  $y(i) = F(x(i))$ .
- As for captain Cook,  $k=10$ ,  $m=20$
- Send  $(x(i), y(i))$  to the  $i$ -th pirates for  $i=1,2,\dots,20$

# Puzzle 2

- A team of  $N$  people wear blue or red hats.
  - Hats are distributed randomly
- Each person can see all hats excepts his/her own .
- Each person guess color of his/her hat, and shout simultaneously, but he can say “pass”
- Your team wins if all answer correctly, but you lose if all say “pass”.
- Consider a good strategy

# Try to solve the puzzle

- If  $n=3$ 
  - Solution 1: Everyone randomly calls “B” or “R”
    - Wining probability =  $1/8$
  - Solution 2: Everyone randomly calls “B”, “R”, or “P”
    - Wining probability =  $8/27 - 1/27 = 7/27$
  - Solution 3 : Choose a leader, who calls B or R. Others say “P”.
    - Winning probability =  $1/2$
  - Solution 4: If one sees two “R” (two “B”), then calls “B” (resp. “R”) . If one sees different colors, “P”.
    - Winning probability =  $3/4$



# Try to solve the puzzle

- If  $n=4$
- Solution 1: If one sees more red than blue, then call B, otherwise R
  - Winning probability:  $6/16 = 3/8$
- Solution 2: Force a person “ignored”, and apply 3 person strategy
  - Winning probability =  $3/4$

# Try to solve the puzzle

- General solution for  $n = 2^k - 1$ 
  - Assign each member his ID from 1 to  $n$
  - Each person compute the bitwise sum of ID of red hat persons
- How we call the color
  - If the bitwise sum of ID is his ID, then shout “blue”
  - If the bitwise sum of ID is 0, then shout “red”
- Analysis
  - At least one person gives call
  - The method only fails all red or all blue cases.
  - Winning probability =  $1 - 1/2^k$

# Hamming code

- Given a bit sequence of length  $2^k - 1$ , we append  $k$  check bits to detect an error of the code including the position of the error.
- Check bit is the bitwise sum of the positions of 1 bits
- In practice, we do slightly different implementation.

# Hadamard code

- Given a bit vector  $a = (a(1), a(2), \dots, a(n))$ , we generate  $\sum a(i) r(i)$  for all bit vectors  $r$
- Very large code (size  $2^n$ )
- We can decode  $a$  even if we have many errors.
- We can decode  $a$  by using  $n$  code bits.

# PCP (Probabilistic Checkable Proof)

- PCP consists of all languages (i.e. problems)  $L$  that has a randomized polynomial time verifier  $V$  such that for any input  $x$ ,
  - If  $x \in L$  (i.e., the answer of  $L$  for  $x$  is yes), then
    - There exists a proof  $\Pi$ , such that  $\Pr(V(x, \Pi), \text{accept}) = 1$
  - Otherwise,
    - For any (fake) proofs  $\Pi$ ,  $\Pr(V(x, \Pi), \text{accept}) < 1/2$
- By definition, NP is in PCP
- PCP should contain IP
- Known: PCP = NEXP (non-deterministic exponential time)
  - Thus, a very wide class
- Thus, we refine PCP by how many bits of proofs we see, and also how many random bits we use.

# PCP( $r(n)$ , $q(n)$ )

- The number of random bits to use:  $O(r(n))$
- The number of bits of the proof we read:  $O(q(n))$ 
  - The size of the proof is at most  $2^{O(r(n)q(n))}$
- The famous result: (Arora, Lund, Motwani, Sudan, Szegedy 1992)
  - $NP = PCP(\log n, 1)$
- We show an easier but key result:
  - $NP \subseteq PCP(\text{poly}(n), 1)$

# We show that 3SAT is in PCP(poly(n),1)

- Input: 3SAT formula
- Professor (prover) wants to persuade you (verifier) that 3SAT has a satisfying assignment
- Idea: You ask the professor to write the proof by using Hadamard code.
  - Also use idea of Reed-Solomon code
- You randomly check small number of bits of the proof to verify both
  - Proof is written by using Hadamard code correctly
  - The proof code a satisfying assignment of 3SAT.

# 3SAT into degree 3 polynomial

- 3SAT:  $\{C_1, C_2, \dots, C_m\}$  set of clauses
- We showed IP (interactive proof) for #3SAT can be given using high-degree polynomial.
- Here, we use degree 3 polynomial instead.
- For a clause  $C_i = X(1) \vee \cancel{X(2)} \vee X(3)$  we define  $c_i(x) = (1-x(1))x(2)(1-x(3))$
- If A satisfies 3SAT then  $c_i(A) = 0$  for all i
- $h(x(1), x(2), \dots, x(n)) = \sum c_i(x)$
- If A satisfies 3SAT then  $h(A) = 0$ 
  - But, converse is false
  - How to resolve it? Use Hadamard code



# 3SAT into degree 3 polynomial

If  $A$  satisfies 3SAT then  $c_i(A) = 0$  for all  $i$

- Random bit sequence  $r(1), r(2), \dots, r(m)$
- $f(X) = f(r, X) = f(x(1), x(2), \dots, x(n)) = \sum c_i(x) r(i)$
- If  $A$  satisfies 3SAT then  $f(A) = 0$

If  $A$  does not satisfy 3SAT then

$$\Pr(f(A) = 0) = \frac{1}{2}$$

- You may ask professor the table of  $f(r, A)$  for all  $r$ 
  - Professor can easily lie by showing the 0 table
  - You need a clever method to compute  $f(r, A)$  using the proof.

# The proof

The proof consists of three tables

- Sum  $A(S)$  of  $a(i)$  for  $i \in S$  for all index set  $S$ .
  - $G_A(z(1), \dots, z(n)) = \sum a(i)z(i)$
- Sum  $B(S')$  of  $a(i)a(j)$  for  $(i,j) \in S'$  for all possible set  $S'$  of pairs of indices.
  - $G_B(z(1,1), \dots, z(n,n)) = \sum b(i,j) z(i,j)$
  - $b(i,j) = a(i)a(j)$
- Sum  $C(S'')$  of  $a(i)a(j)a(k)$  for  $(i,j,k) \in S''$  for all possible set  $S''$  of triples of indices.
  - $G_C(z(1,1,1), \dots, z(n,n,n)) = \sum c(i,j,k) z(i,j,k)$
  - $c(i,j,k) = b(i,j)a(k) = a(i)a(j)a(k)$

# How to compute $f(A)$ from the proof

The proof consists of three tables

- Sum  $A(S)$  of  $a(i)$  for  $i \in S$  for all index set  $S$ .
  - $G_A(z(1), \dots, z(n)) = \sum a(i)z(i)$
- Sum  $B(S')$  of  $b(i,j)=a(i)a(j)$  for  $(i,j) \in S'$ 
  - $G_B(z(1,1), \dots, z(n,n)) = \sum b(i,j) z(i,j)$
- Sum  $C(S'')$  of  $c(i,j,k)= a(i)a(j)a(k)$  for  $(i,j,k) \in S''$  for
  - $G_C(z(1,1,1), \dots, z(n,n,n)) = \sum c(i,j,k) z(i,j,k)$

$$f(X)=f(r,X)= f(x(1),x(2),\dots,x(n)) = \sum c_i(x) r(i)$$

$c_i(x)$  : degree 3, e.g.  $(1-x(1))x(2)(1-x(3))$

$$f(X) = \alpha + \sum_{i \in S_1} x_i + \sum_{(i,j) \in S_2} x_i x_j + \sum_{(i,j,k) \in S_3} x_i x_j x_k$$

# How to compute $f(A)$ from the proof

The proof consists of three tables

- Sum  $A(S)$  of  $a(i)$  for  $i \in S$  for all index set  $S$ .
  - $G_A(z(1), \dots, z(n)) = \sum a(i)z(i)$
- Sum  $B(S')$  of  $b(i,j)=a(i)a(j)$  for  $(i,j) \in S'$ 
  - $G_B(z(1,1), \dots, z(n,n)) = \sum b(i,j) z(i,j)$
- Sum  $C(S'')$  of  $c(i,j,k)=a(i)a(j)a(k)$  for  $(i,j,k) \in S''$  for
  - $G_C(z(1,1,1), \dots, z(n,n,n)) = \sum c(i,j,k) z(i,j,k)$

$$f(A) = \alpha + \sum_{i \in S_1} a_i + \sum_{(i,j) \in S_2} a_i a_j + \sum_{(i,j,k) \in S_3} a_i a_j a_k$$

$$f(A) = \alpha + A(S_1) + B(S_2) + C(S_3)$$

$$f(A) = \alpha + G_A(\chi(S_1)) + G_B(\chi(S_2)) + G_C(S(\chi_3))$$

# How to compute $f(A)$ from the proof

The proof consists of three tables

- Sum  $A(S)$  of  $a(i)$  for  $i \in S$  for all index set  $S$ .
  - $G_A(z(1), \dots, z(n)) = \sum a(i)z(i)$
- Sum  $B(S')$  of  $b(i,j)=a(i)a(j)$  for  $(i,j) \in S'$ 
  - $G_B(z(1), \dots, z(n)) = \sum b(i,j) z(i)z(j)$
- Sum  $C(S'')$  of  $c(i,j,k)=a(i)a(j)a(k)$  for  $(i,j,k) \in S''$  for
  - $G_C(z(1), \dots, z(n)) = \sum c(i,j,k) z(i)z(j)z(k)$

$$f(A) = \alpha + \sum_{i \in S_1} a_i + \sum_{(i,j) \in S_2} a_i a_j + \sum_{(i,j,k) \in S_3} a_i a_j a_k$$

$$f(A) = \alpha + A(S_1) + B(S_2) + C(S_3)$$

Thus,  $f(A) = 0$  can be certified by reading three entries of the table

# How to compute $f(A)$ from the proof

The proof consists of three tables

- Sum  $A(S)$  of  $a(i)$  for  $i \in S$  for all index set  $S$ .
  - $G_A(z(1), \dots, z(n)) = \sum a(i)z(i)$
- Sum  $B(S')$  of  $b(i,j)=a(i)a(j)$  for  $(i,j) \in S'$ 
  - $G_B(z(1,1), \dots, z(n,n)) = \sum b(i,j) z(i,j)$
- Sum  $C(S'')$  of  $c(i,j,k)=a(i)a(j)a(k)$  for  $(i,j,k) \in S''$  for
  - $G_C(z(111), \dots, z(nnn)) = \sum c(i,j,k) z(i,j,k)$

$$f(A) = \alpha + \sum_{i \in S_1} a_i + \sum_{(i,j) \in S_2} a_i a_j + \sum_{(i,j,k) \in S_3} a_i a_j a_k$$

$$f(A) = \alpha + A(S_1) + B(S_2) + C(S_3)$$

Can professor cheat? Note that  $\alpha=0$  or  $1$  with even probability if original SAT is false

# How to reveal false proof

The proof consists of three tables

- Sum  $A(S)$  of  $a(i)$  for  $i \in S$  for all index set  $S$ .
  - $G_A(z(1), \dots, z(n)) = \sum a(i)z(i)$
- Sum  $B(S')$  of  $b(i,j)=a(i)a(j)$  for  $(i,j) \in S'$ 
  - $G_B(z(1,1), \dots, z(n,n)) = \sum b(i,j) z(i,j)$
- Sum  $C(S'')$  of  $c(i,j,k)=a(i)a(j)a(k)$  for  $(i,j,k) \in S''$  for
  - $G_C(z(111), \dots, z(nnn)) = \sum c(i,j,k) z(i,j,k)$

We should examine the following:

1.  $G_A(Z)$  is a linear function
2.  $B = A \times A$  :  $x$  is the outer product
3.  $C = A \times B$

# How to reveal false proof (Step1)

The proof consists of three tables

- Sum  $A(S)$  of  $a(i)$  for  $i \in S$  for all index set  $S$ .
  - $G_A(z(1), \dots, z(n)) = \sum a(i)z(i)$

First step: ( $\delta < 1/3$  is small)

1.  $G(z) = G_A(z)$  is a **NEAR** linear function, that is there is a linear function  $H$  such that  $G(x) = H(x)$  with probability at least  $1 - \delta$
- Theorem: If  $G(x) + G(y) = G(x+y)$  with probability  $1 - \delta/2$ , then  $G(x)$  is near linear

If  $G(z)$  is near linear, we can do error correction by  $G(z) \leftarrow G(x) + G(z-x)$  for taking random vector  $x$



# How to reveal false proof (STEP2)

The proof consists of three tables

- $G_A(z(1), \dots, z(n)) = \sum a(i)z(i)$
- Sum  $B(S')$  of  $b(i,j)=a(i)a(j)$  for  $(i,j) \in S'$ 
  - $G_B(z(1,1), \dots, z(n,n)) = \sum b(i,j) z(i,j)$

If  $B = A \times A$  is cheating, for random  $r$  and  $s$ ,

$$\text{Prob}(r^t (A \times A)s \neq r^t B s) \geq 1/4$$

$$\text{Prob}(G_A(r)G_A(s) \neq G_B(r \times s)) \geq 1/4$$

# How to reveal false proof (STEP2)

The proof consists of three tables

- $G_A(z(1), \dots, z(n)) = \sum a(i)z(i)$
- Sum  $B(S')$  of  $b(i,j)=a(i)a(j)$  for  $(i,j) \in S'$ 
  - $G_B(z(1,1), \dots, z(n,n)) = \sum b(i,j) z(i,j)$
- Sum  $C(S'')$  of  $c(i,j,k)=a(i)a(j)a(k)$  for  $(i,j,k) \in S''$  for
  - $G_C(z(1,1,1), \dots, z(n,n,n)) = \sum c(i,j,k) z(i,j,k)$

If  $C = A \times B$  is cheating,

$$\text{Prob}(r^t (A \times B)s \neq r^t Cs) \geq 1/4$$

$$\text{Prob}(G_A(r)G_B(s) \neq G_C(r \times s)) \geq 1/4$$

# Overall

- We can check that  $GA(z)$  is near linear by checking constant number of bits of the proof.
- We can check the other two conditions by checking constant number of bits
- Thus, we can reveal cheating of the proof with high probability by checking constant number of bits.
- Thus, 3SAT is in  $PCP(\text{poly}(n), 1)$ .

# Further (and important) improvement

- NP is PCP(  $\log n$ , 1)
  - Reducing the number of bits to  $O(\log n)$
- Thus, we need to  $2^{O(\log n)} < p(n)$  possible choices of query, and each query asks  $c$  bits.
- Thus, the (effective) proof of length is  $c p(n)$
- Thus, the possibility of query set is  $(c p(n))^c$ 
  - This is polynomial size in  $n$

# A surprising implication

- NP is PCP(  $\log n$ , 1)
- Polynomial number of possible choice of queries (without knowing A)
- The probability of “yes” queries is formulated as a polynomial size optimization problem (CSP) considering A as unknown vector X
- This probability is 1 if and only if 3SAT formula has an assignment.
- This probability is less than  $\frac{1}{2}$  if and only if 3SAT formula has no assignment
- This means that if CSP can be solved with approximation ratio  $\frac{3}{4}$ , then, we can solve 3SAT

# Classification of NPO problems

- If P is not NP, then NP optimization problems (NPO) are classified into as follows:
  1. P (polynomial time soluble)
  2. PTAS (a problem with polynomial time approximation scheme)
    - Knapsack problem, Subset-sum problem, Euclidean traveling salesman problem, Max Independent set in planar graph
  3. **APX complete** (a problem with a polynomial time constant approximation solution, but not in PTAS)
    - MAX 3-SAT, MAX 2-SAT, Vertex Cover
  4. Problems with no constant approximation polynomial time algorithm (if P is not NP)
    - CLIQUE, Graph coloring, set covering,