Chapter 34
NP Complete Problems

songyou@buaa.edu.cn

34 NP Complete Problems

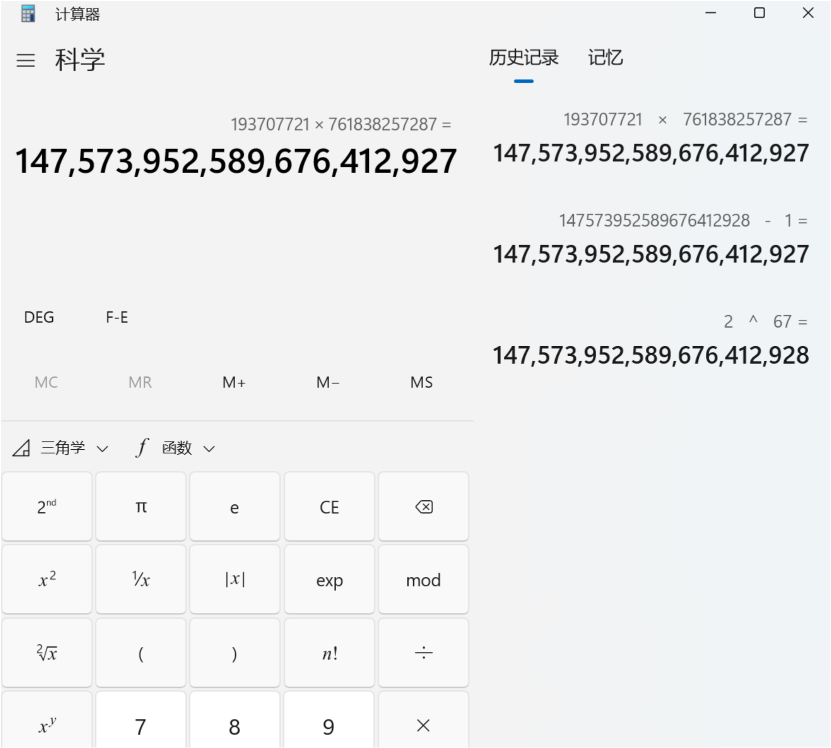
$$21 = 3 \times 7$$

$$2^{67} - 1 = a \times b\ ?$$

34 NP Complete Problems

$2^{67}-1 = a \times b$?

$2^{67}-1 = 193{,}707{,}721 \times 761{,}838{,}257{,}287$



研究困难问题
是否有意义?

应用示例：大数分解的
困难，保证了密码的安全

34 NP Complete Problems

哥德巴赫猜想

1+1：大偶数等于两个素数之和 $M = a + b$

1+2：$M = a + b*c$

陈氏定理：一个大的偶数可以表示为一个素数与不超过两个素数乘积之和，如76=37+13*3。陈景润证明出来。

…

猜想某命题（陈述）是正确的，需要证明之？难！

判断一个命题（陈述）是否正确？难！
若正确，请证明；若不正确，举出反例。

合数的质因子分解定理

合数 m 可唯一分解为如下乘积形式：

$$m = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$$

其中 p_i 为质数（素数），$p_1 < p_2 < \cdots < p_r$ ，且 e_i 为正整数。

34 NP Complete Problems

千僖年数学难题（2000-5-24，美国的克雷(Clay)数学研究所，在巴黎法兰西学院宣布每一个悬赏一百万美元）

一、贝赫(Birch)和斯维讷通 - 戴尔(Swinnerton-Dyer)猜想

二、霍奇(Hodge)猜想

三、纳维叶 - 斯托克斯(Navier-Stokes)方程

四、P (多项式算法可解)问题对NP (“非确定性问题”)

五、庞加莱(Poincare)猜想

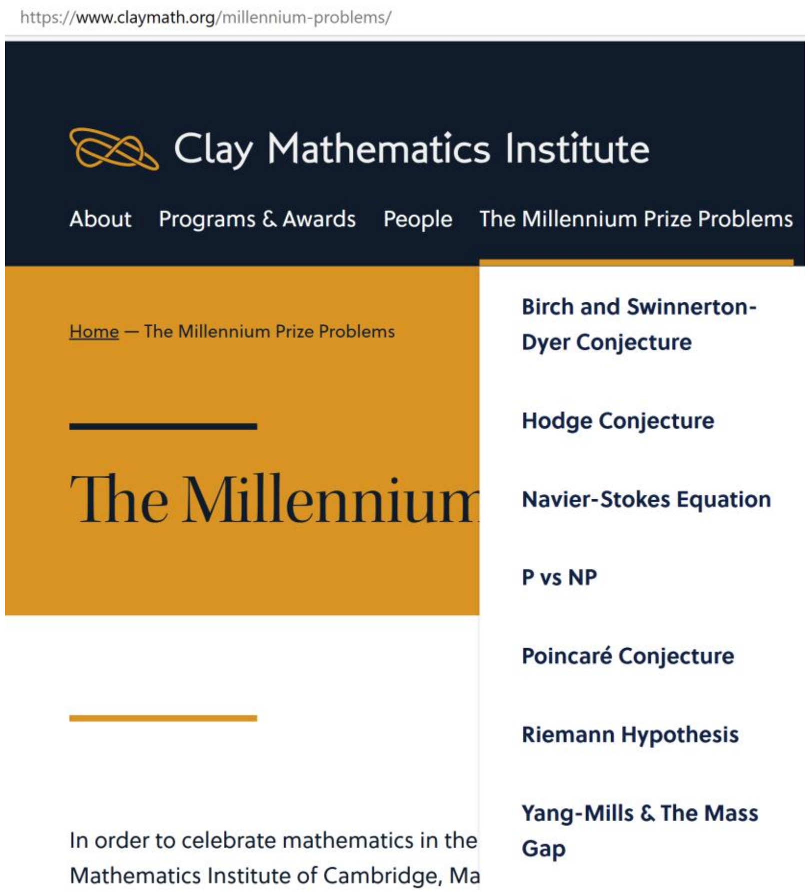
This question turned out to be extraordinarily difficult. Nearly a century passed between its formulation in 1904 by Henri Poincaré and its solution by **Grigoriy Perelman**, announced in preprints posted on ArXiv.org in 2002 and 2003.

俄罗斯数学家格里**戈里•佩雷尔曼**在预印本平台 ArXiv.org 上发布了证明结果。

他拒绝去领奖，“如果我的证明是正确的，这种方式的承认是不必要的。”2006年8月，拒绝了有着数学界诺贝尔奖之称的“菲尔兹奖”。佩雷尔曼得知菲尔兹奖将由西班牙国王颁发时，他说道：“国王又不是数学家，为什么有资格颁奖？”

六、黎曼(Riemann)假设

七、杨 - 米尔斯(Yang-Mills)存在性和质量缺口



34 NP Complete Problems

- 多项式时间算法(n^{1000})，简单!
- 非多项式时间算法，复杂!
- 不知是否存在可解算法的问题，更复杂!

- 旅行商问题(Traveling Salesman Problem, TSP)
  找出一条通过所有城镇并回到原出发点的最短路线
  - $1\rightarrow2\rightarrow3$ $(1\rightarrow3\rightarrow2)\rightarrow$ …… , 这两种走法的距离可能不一样。
  - 可能的路线： $n!$
  - 怎样找出总路程最短(省钱)的一种走法？穷举法！没有最佳的方法。
  - NP完全问题
  - 应用：运输公司配送货物；邮递员；生产线上组装工序；……
  - 穷举法：如果 T(20) = 1 h, T(21) = 21 h, … , T(25) = 728 y

34 NP Complete Problems

问题的复杂性 vs 算法的复杂性

- 算法的复杂性（算法的性质）：解决问题的一个具体的算法的执行时间, For example, sort algorithms
  - Bubble sort, $O(n^2)$
  - Quick sort, $O(n\lg n)$
- 问题的复杂性（问题本身的复杂程度，问题固有的性质）：解决该问题的所有算法中最好算法的复杂性, For example
  - sort problem, $O(n\lg n)$
- 问题的复杂性分析，考虑一类简化问题，即判定问题
  - 问题的复杂性不可能通过枚举各种可能的算法来得到，为了研究的简单，仅考虑一类简单的问题，即判定问题。

Decision Problems （判定问题）

- 判断问题：答案为是或否的问题，如
  - 在数组 A 中，是否存在相同的数
  - 在 G 中，顶点 u 到 v 是否存在小于 k 的一条路径

- 最优化问题很容易简化为判定问题
  - Opti-Prob: Shortest path u to v , $\delta(u, v) = ?$
  - Deci-Prob: If there exists a path u to v, Path(u, v) < 2 ? Path(u, v) < 3 ? … Path(u, v) < k ?
  - If $\delta(u, v) = 3$, then P(u, v) < 2 is NO, … , P(u, v) < 4 is YES

P is the class of decision problems that can be solved in polynomial time ($O(n^k)$, where k is a constant). Intuitively, the problems in P class are easy problems.

P 问题：多项式时间内可解的判定问题

如：

- 在数组 A 中，是否存在相同的数
- 在图 G 中，顶点 u 到 v 是否存在小于 k 的一条路径

NP is the class of decision problems for which we can verify the correctness of solutions in polynomial time.

NP 问题：多项式时间内可验证的判定问题

- This doesn’t say it is easy to find a solution.
- In fact, it is often hard to find a solution!

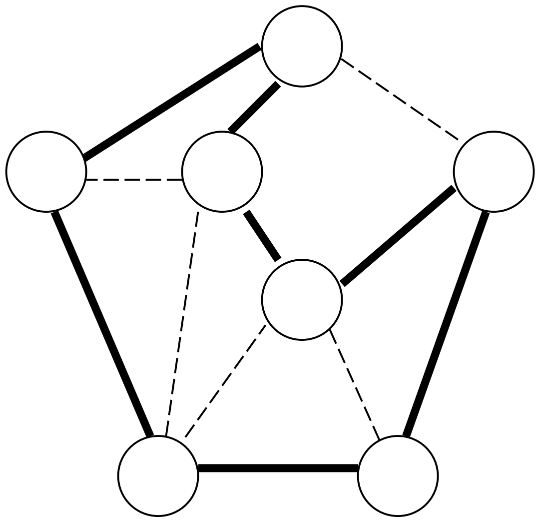
34.2 Problems in NP Class

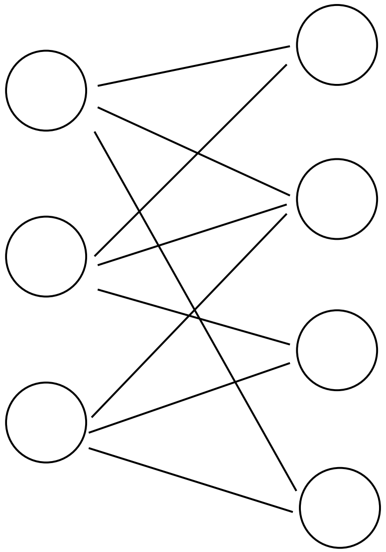
- P ? Polynomial
- NP: N?
  - not Non-Polynomial， but Non-Deterministic
    多项式复杂程度的非确定性问题，即多项式时间内能验证的判定问题。例如 C = A*B，先要猜想 A 和 B（而没有有效的公式能直接求出 A 和 B，猜想是非确定的），再验证 C = A*B 是否成立（验证是多项式的）。
  - 所有的非确定性多项式时间可验证的判定问题构成 NP 类问题。

34.2 Problems in NP Class

Non-Deterministic Problem（没有一个确定的公式（算法），直接地计算出来），e.g.

- Factorization（大合数分解质因数的问题）：$M = a \times b$?
- Hamiltonian cycle Problem: Given a graph *G*, does *G* have a Hamiltonian cycle?
  (A Hamiltonian cycle of a graph *G* is a cycle that visits each vertex of the graph exactly once.)





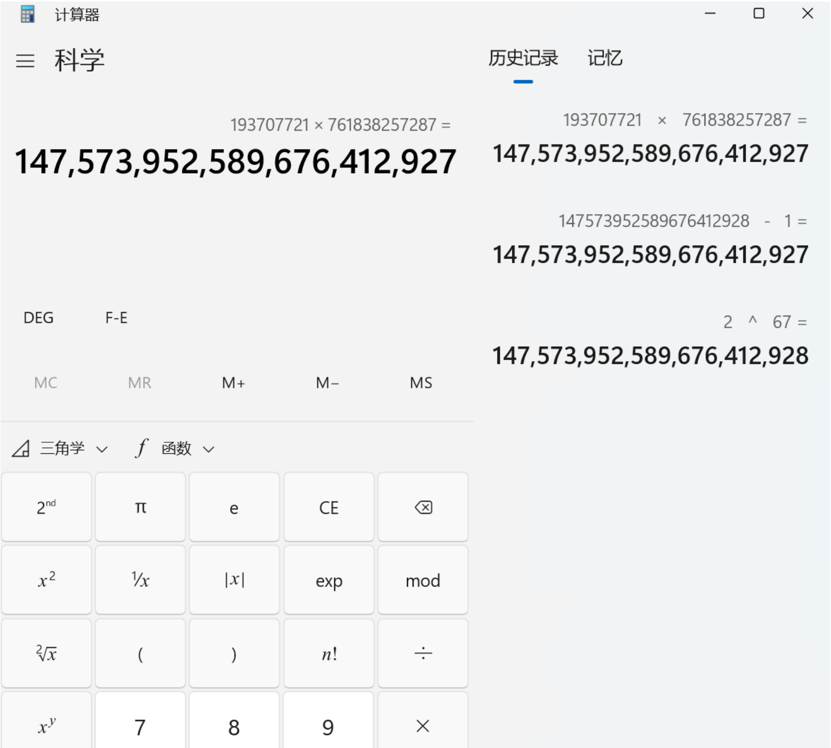
34.3 P and NP

- Problems in P can be solved “quickly”
- Problems in NP can be verified “quickly”
- It is easier to verify a solution than to solve a problem.

In 1903, F. N. Cole factored $2^{67}–1$, i.e.

$$2^{67}-1 \;=\; 193{,}707{,}721 \times 761{,}838{,}257{,}287$$

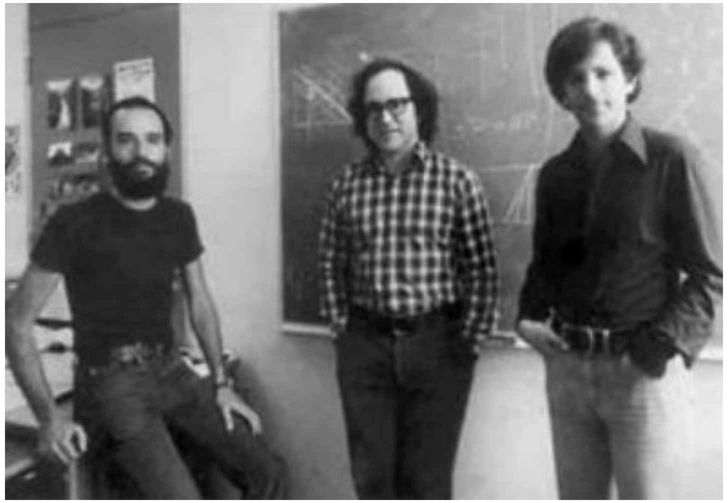
It took Cole 150 days to find the factorization!
but once it was found, we can verify it very quickly!

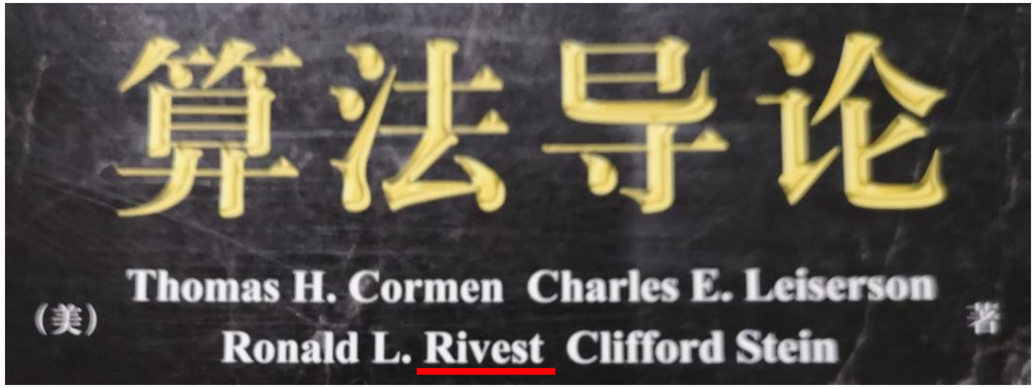


34.4 RSA encryption----an application of NP

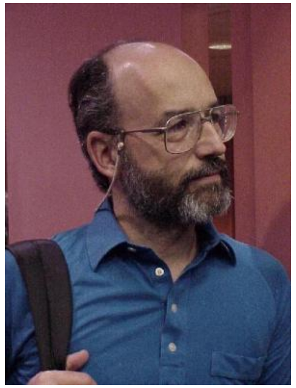
- The concept of a public-key cryptosystem is due to Diffie and Hellman, 1976

- The RSA cryptosystem (1977, MIT) was proposed by Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, (Turing Award, 2002)

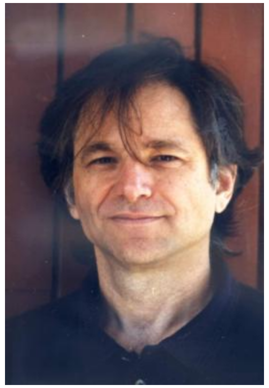




Rivest

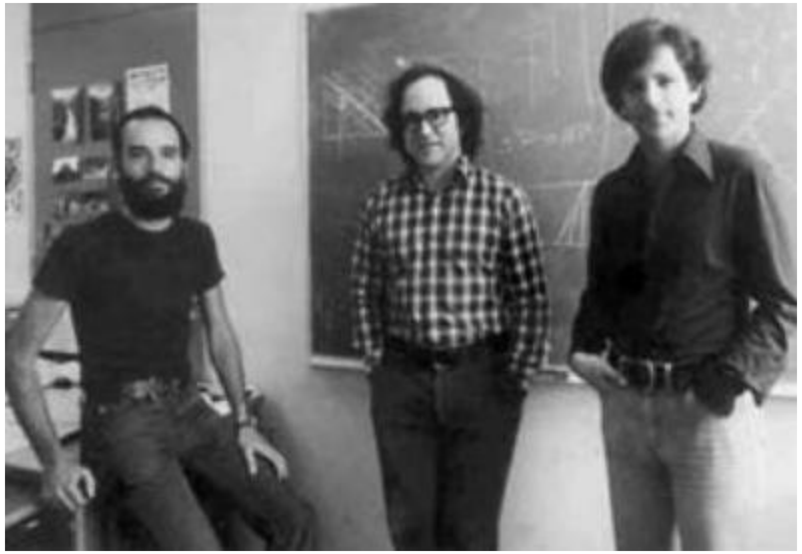


Shamir



Adleman

34.4 RSA encryption



A method for obtaining digital signatures and public-key cryptosystems

RL Rivest, A Shamir, L Adleman - Communications of the ACM, 1978 - dl.acm.org

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:(1) Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intented recipient. Only he can decipher the message, since only he knows the corresponding decryption key.(2) A message can be "signed" using a privately held …

34.4 RSA encryption

news.buaa.edu.cn/info/1005/9613.htm

北京航空航天大学 BEIHANG UNIVERSITY | 新闻网

北航官网 旧版新闻网 ENGLISH IHome

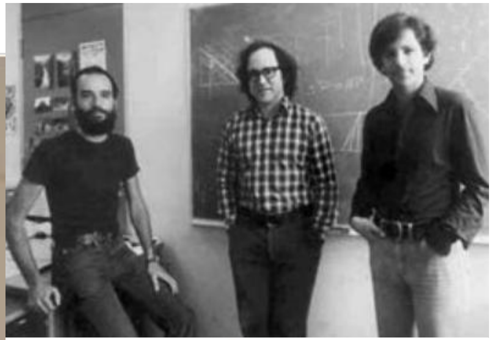
首页 综合新闻 专题新闻 北航人物 校园风采 科教在线 媒体北航 光影北航 视频新闻 文艺园地 信息公告

/ 科教在线

图灵奖获得者Adi Shamir博士来校访问讲学

点击数:4452 | 加入时间:2009-04-15

4月14日，以色列密码学家、图灵奖获得者Adi Shamir博士来校访问，并为北航师生做了题为“密码和安全系统是怎样被破解的”的学术报告。我校校长李未出席。 Adi Shamir博士从数学黑箱模型等基础原理出发，利用图形示范和对比，向大家介绍了密码学领域最重要的基石，也是工业界应用最广泛的系统RSA系统，并演示了如何利用这些新技术破解各种强大的密码系统。会后，Adi Shamir博士还与我校师生进行了互动交流，就相关问题进行解答。 Adi Shamir博士是一位杰出的计算机科学家、密码学家。他与Ronald L. Rivest和Leonard Adelman共同创立了以他们三个人的名字命名的RSA公钥密码系统，他们因此获得了2002年的图灵奖（信息科学、电子科学、计算机科学等领域的最高奖，也称为该领域的“诺贝尔奖”）。他与Uriel Feige 和Amos Fiat共同发明了Feige-Fiat-Shamir认证协议，同时他还是各种密码破译方法的发明者之一，在密码学和计算机科学领域做出了杰出的贡献。 （文/图 张素芳）
编辑：贾爱平 编辑：贾爱平 张素芳



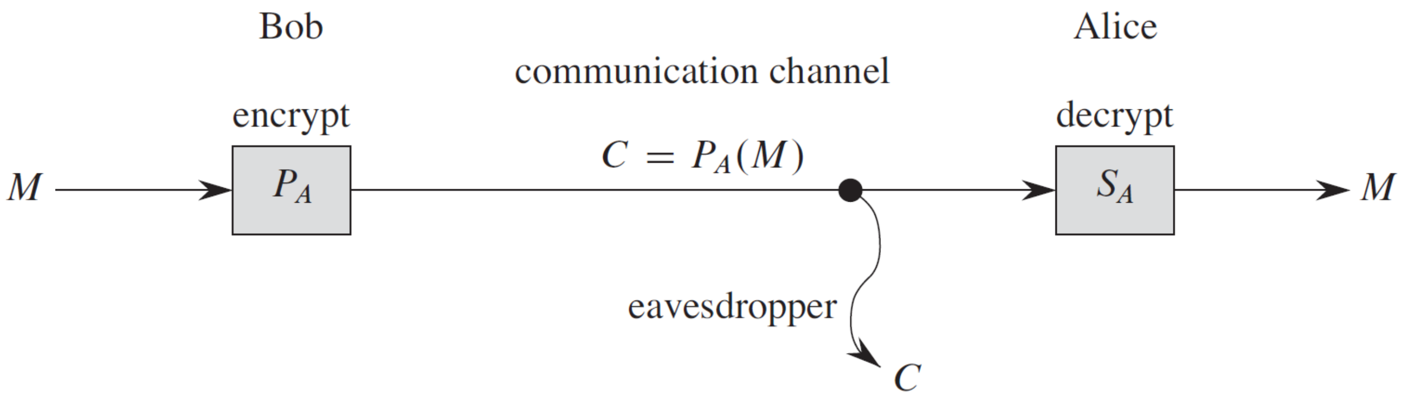
Adi Shamir

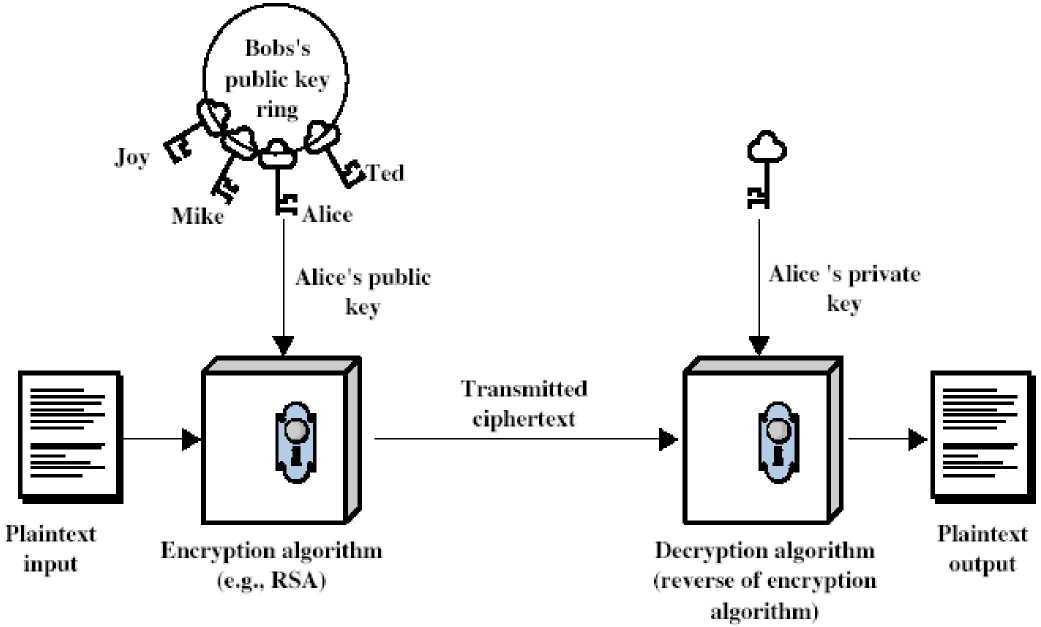
阿迪•沙米尔 (Adi Shamir), 1952 年出生于以色列, 是魏茨曼科学研究所计算机科学与应用数学系的教授，是全球最资深的计算机科学家之一。他是信息加密和解密领域的顶级专家。Shamir 是 RSA 方法的开发者之一，该方法改变了世界计算机通信的面貌，是电子商务和信息安全的基本支柱。

阿迪•沙米尔（Adi Shamir）被授予2024年沃尔夫奖（Wolf Prize），以表彰他是一位真正杰出的科学家，并且一直是将密码学转变为一门以数学为基础的科学学科的主导力量。他的基础性发现将数学独创性与一系列分析工具相结合。他们对几个数学领域产生了巨大影响，以无与伦比的方式推动了数学和社会的发展。

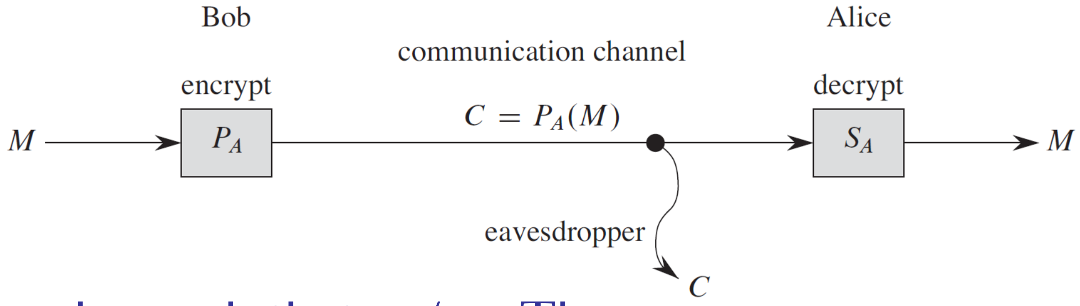
Adi Shamir（2002，图灵奖；2024，沃尔夫奖），
2009年在北航讲学，报告题目“密码和安全系统是怎样被破解的”，
软件学院首任院长孙伟教授主持

Encryption in a public key system 公钥加密系统





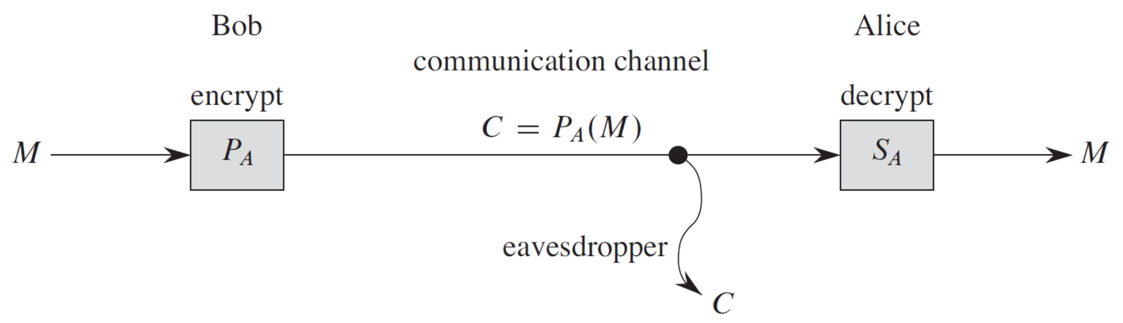
34.4 RSA encryption



RSA public-key cryptosystem

1. Select at random two large prime numbers p and q such that $p \neq q$. The primes p and q might be, say, 512 bits each, 1024, or more.
2. Compute n by the equation $n = pq$.
3. Let $\varphi(n) = (p - 1)(q - 1)$. // after that, let p and q disappear
4. Select a small odd integer e that is relatively prime to $\varphi(n)$, i.e., $\gcd(e, \varphi(n)) = 1$.
5. Compute d as the multiplicative inverse of e, modulo $\varphi(n)$, i.e., $ed \equiv 1 (\text{mod } \varphi(n))$ 【 $ed = 1 + k\varphi(n)$ 】.
6. Publish the pair $P = (e, n)$ as his ***RSA public key.***
7. Keep secret the pair $S = (d, n)$ as his ***RSA secret key.***

Why is RSA encryption effective



RSA public-key cryptosystem

1. Select two large prime numbers p and q, $p \neq q$.
2. Compute $n = pq$.
3. Compute $\varphi(n) = (p - 1)(q - 1)$. // Euler's phi function
4. Select a small odd integer e, s.t., $\gcd(e, \varphi(n))=1$.
5. Compute d, by the equation $ed \equiv 1(\mathrm{mod}\ \varphi(n))$
6. Publish the pair $P = (e, n)$ as his ***RSA public key*** **.**
7. Keep secret the pair $S = (d, n)$ as his ***RSA secret key*** **.**

*(1) RSA is correct

Theorem 31.36:

$$S(P(M)) = S(M^e(\mathrm{mod}\ n)\ = (M^e(\mathrm{mod}\ n)\ )^d(\mathrm{mod}\ n) = M^{ed}(\mathrm{mod}\ n)$$

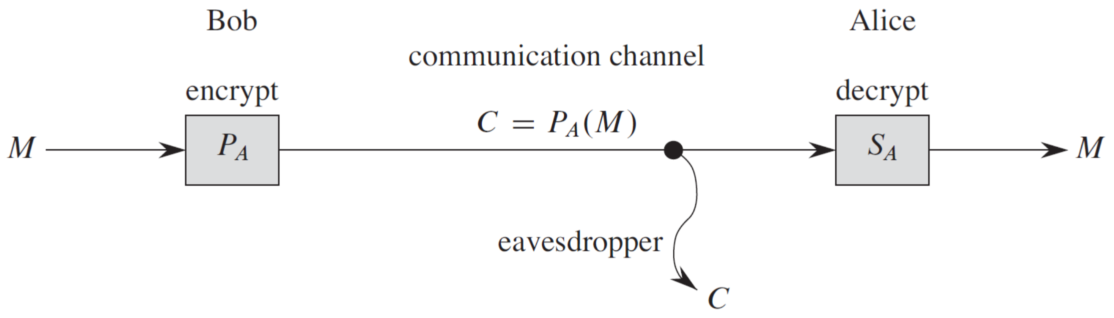
$$= M^{1+k\varphi(n)}(\mathrm{mod}\ n) = MM^{k\varphi(n)}(\mathrm{mod}\ n) = M(M^{(p-1)})^{k(q-1)}(\mathrm{mod}\ n)\ \ldots =M(\mathrm{mod}\ n)$$

(by Theorem31.31, Fermat's Theorem, if p is prime, $M^{(p-1)} = 1(\mathrm{mod}\ n)$)

Why is RSA encryption effective

RSA public-key cryptosystem

1. Select two large prime numbers p and q, $p \neq q$.
2. Compute $n = pq$.
3. Compute $\varphi(n) = (p - 1)(q - 1)$.
4. Select a small odd integer e, s.t., $\gcd(e, \varphi(n))=1$.
5. Compute d, by the equation $ed \equiv 1(\mathrm{mod}\ \varphi(n))$
6. Publish the pair $P = (e, n)$ as his ***RSA public key .***
7. Keep secret the pair $S = (d, n)$ as his ***RSA secret key .***



作业：自行设计一个基于RSA的加密系统（软件）

*(1) RSA is correct: $S(P(M)) = S(M^e(\mathrm{mod}\ n)) = M^{ed}(\mathrm{mod}\ n) = \ldots = M$

*(2) RSA is secure:

- If factoring n is easy, breaking the RSA cryptosystem is easy.
- Conversely, that if factoring large integers is hard, then breaking RSA is hard, is unproven.
- After decades of research, however, no easier method has been found to break the RSA public-key cryptosystem than to factor n.（破解RSA比大数分解更困难）
- In fact, the factoring of large integers is surprisingly difficult.

RSA: number-theoretic algorithms (Chapter 31)

RSA public-key cryptosystem

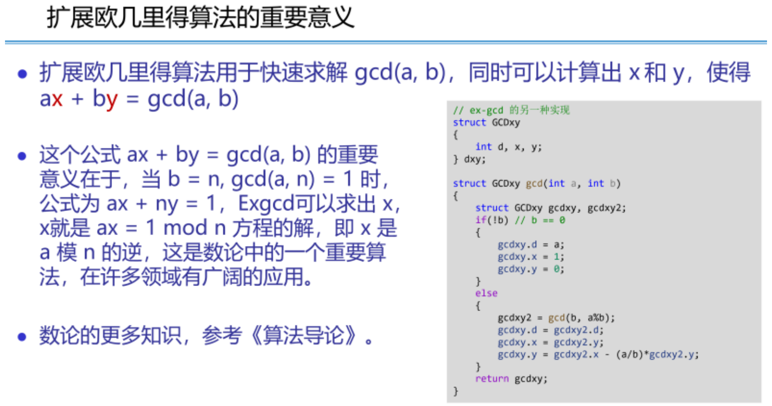
1. Select two large prime numbers p and q, $p \neq q$. // hard
2. Compute $n = pq$. // Question1: 高精度乘法
3. Compute $\varphi(n) = (p - 1)(q - 1)$. // Q2: Euler's phi function, 高精度减法、乘法
4. Select a small odd int e, s.t., $\gcd(e, \varphi(n)) = 1$. // Q3: 高精度除法、辗转相除
5. Compute d, by equation $ed \equiv 1 (\mathrm{mod}\ \varphi(n))$ // Q4: 求方程 $ex \equiv 1$，高精度运算
6. Publish the pair $P = (e, n)$ as his ***RSA public key*** **.**
7. Keep secret the pair $S = (d, n)$ as his ***RSA secret key*** **.**

Q3: $\gcd(e, \varphi(n))$

如果 $e < F_k$ (fib number), 则辗转相除求 $\gcd(e, \varphi(n))$ 的次数少于 k-1 次。

Q4: Q3's $\gcd(e, \varphi(n)) = 1$, 意味着存在 x 和 y，使得 $ex + \varphi(n)y = 1$，因此 $ed \equiv 1 (\mathrm{mod}\ \varphi(n))$，这里 $d = x$.

Chapter 31 ……



扩展欧几里得算法的重要意义

- 扩展欧几里得算法用于快速求解 gcd(a, b)，同时可以计算出 x 和 y，使得 ax + by = gcd(a, b)
- 这个公式 ax + by = gcd(a, b) 的重要意义在于，当 b = n, gcd(a, n) = 1 时，公式为 ax + ny = 1，Exgcd可以求出 x，x就是 ax = 1 mod n 方程的解，即 x 是 a 模 n 的逆，这是数论中的一个重要算法，在许多领域有广阔的应用。
- 数论的更多知识，参考《算法导论》。

```
// ex-gcd 的另一种实现
struct GCDxy
{
    int d, x, y;
} dxy;

struct GCDxy gcd(int a, int b)
{
    struct GCDxy gcdxy, gcdxy2;
    if(!b) // b == 0
    {
        gcdxy.d = a;
        gcdxy.x = 1;
        gcdxy.y = 0;
    }
    else
    {
        gcdxy2 = gcd(b, a%b);
        gcdxy.d = gcdxy2.d;
        gcdxy.x = gcdxy2.y;
        gcdxy.y = gcdxy2.x - (a/b)*gcdxy2.y;
    }
    return gcdxy;
}
```

RSA: number-theoretic algorithms (Chapter 31)

RSA public-key cryptosystem

RSA: $S(P(M)) = S(\,M^e \pmod n\,) = M^{ed} \pmod n = \ldots = M$

1. Select two large prime numbers p and q, $p \neq q$.
2. Compute $n = pq$. // Question1: 高精度乘法
3. Compute $\varphi(n) = (p - 1)(q - 1)$. // Q2: Euler's phi function, 高精度减法、乘法
4. Select a small odd int e, s.t., $\gcd(e, \varphi(n)) = 1$. // Q3: 高精度除法、辗转相除
5. Compute d, by equation $ed \equiv 1 (\mathrm{mod}\ \varphi(n))$ // Q4: 求方程 $ex \equiv 1$，高精度运算
6. Publish the pair $P = (e, n)$ as his ***RSA public key*** **.**
7. Keep secret the pair $S = (d, n)$ as his ***RSA secret key*** **.**

Question: $a^b \pmod n$?

31.6 Powers of an element 957

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

Figure 31.4 The results of MODULAR-EXPONENTIATION when computing $a^b \pmod n$, where $a = 7$, $b = 560 = \langle 1000110000 \rangle$, and $n = 561$. The values are shown after each execution of the **for** loop. The final result is 1.

significant bit.) The following procedure computes $a^c \bmod n$ as c is increased by doublings and incrementations from 0 to b.

```
MODULAR-EXPONENTIATION(a, b, n)
c = 0
d = 1
let ⟨b_k, b_{k-1}, ..., b_0⟩ be the binary representation of b
for i = k downto 0
    c = 2c
    d = (d · d) mod n
    if b_i == 1
        c = c + 1
        d = (d · a) mod n
return d
```

The RSA Challenge Numbers

将一个由两个大质数所乘出来的大数分解回来

e.g., $11 \times 13 \rightarrow 143$ $143 \rightarrow ? \times ?$

Challenge Num	Prize ($US)	Status	Submission Date	Submitter(s)
RSA-576	$10,000	Factored	December 3, 2003	J. Franke et al.
RSA-640	$20,000	Factored	November 2, 2005	F. Bahr et al.
RSA-704	$30,000	Factored	?	?
RSA-768	$50,000	Factored	December 12, 2009	
RSA-896	$75,000	Not Factored ?		
RSA-1024	$100,000	Not Factored ?		
RSA-1536	$150,000	Not Factored ?		
RSA-2048	$200,000	Not Factored ?		

RSA number	Decimal digits	Binary digits	Cash prize offered	Factored on	Factored by
RSA-100	100	330	US$1,000[4]	April 1, 1991[5]	Arjen K. Lenstra
RSA-110	110	364	US$4,429[4]	April 14, 1992[5]	Arjen K. Lenstra and M.S. Manasse
RSA-120	120	397	$5,898[4]	July 9, 1993[6]	T. Denny *et al.*
RSA-129 [**]	129	426	US$100	April 26, 1994[5]	Arjen K. Lenstra *et al.*
RSA-130	130	430	US$14,527[4]	April 10, 1996	Arjen K. Lenstra *et al.*
RSA-140	140	463	US$17,226	February 2, 1999	Herman te Riele *et al.*
RSA-150	150	496		April 16, 2004	Kazumaro Aoki *et al.*
RSA-155	155	512	$9,383[4]	August 22, 1999	Herman te Riele *et al.*
RSA-160	160	530		April 1, 2003	Jens Franke *et al.*, University of Bonn
RSA-170 [*]	170	563		December 29, 2009	D. Bonenberger and M. Krone [***]
RSA-576	174	576	US$10,000	December 3, 2003	Jens Franke *et al.*, University of Bonn
RSA-180 [*]	180	596		May 8, 2010	S. A. Danilov and I. A. Popovyan, Moscow State University[7]
RSA-190 [*]	190	629		November 8, 2010	A. Timofeev and I. A. Popovyan
RSA-640	193	640	US$20,000	November 2, 2005	Jens Franke *et al.*, University of Bonn
RSA-200 [*] ?	200	663		May 9, 2005	Jens Franke *et al.*, University of Bonn
RSA-210 [*]	210	696		September 26, 2013[8]	Ryan Propper
RSA-704 [*]	212	704	US$30,000	July 2, 2012	Shi Bai, Emmanuel Thomé and Paul Zimmermann
RSA-220 [*]	220	729		May 13, 2016	S. Bai, P. Gaudry, A. Kruppa, E. Thomé and P. Zimmermann
RSA-230 [*]	230	762		August 15, 2018	Samuel S. Gross, Noblis, Inc.
RSA-232	232	768			
RSA-768 [*]	232	768	US$50,000	December 12, 2009	Thorsten Kleinjung *et al.*
RSA-240	240	795			
RSA-250	250	829			
RSA-260	260	862			
RSA-270	270	895			
RSA-896	270	896	US$75,000		

The RSA Challenge Numbers

RSA-576 (Factored) 〔**Decimal Digits: 174**〕

188198812920607963838697239461650439807163563379417382700763356422988859715234665485319060606504743045317388011303396716199692321205734031879550656996221305168759307650257059

=

A

×

B

The RSA Challenge Numbers

RSA-576 (Factored) 〔**Decimal Digits: 174**〕

1881988129206079638386972394616504398071635633794173827007633564229888
5971523466548531906060650474304531738801130339671619969232120573403187
9550656996221305168759307650257059

=

3980750864240649373971255005503864911990643623425267084063851895759463
88957261768583317

×

4727721461074353025362230719730482246329146953020971164598521711305207
11256363590397527

The RSA Challenge Numbers

RSA-768 (Factored)

1230186684530117755130494958384962720772853569595334792197322452151726400507263
6575187452021997864693899564749427740638459251925573263034537315482685079170261
22142913461670429214311602221240479274737794080665351419597459856902143413

=

A

×

B

The RSA Challenge Numbers

RSA-768 (Factored)

1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479274737794080665351419597459856902143413

=

33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489

×

36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917

…>高精度计算器

The RSA Challenge Numbers

RSA-2048 ($2^{2048} \approx 10^{2048/3}$)

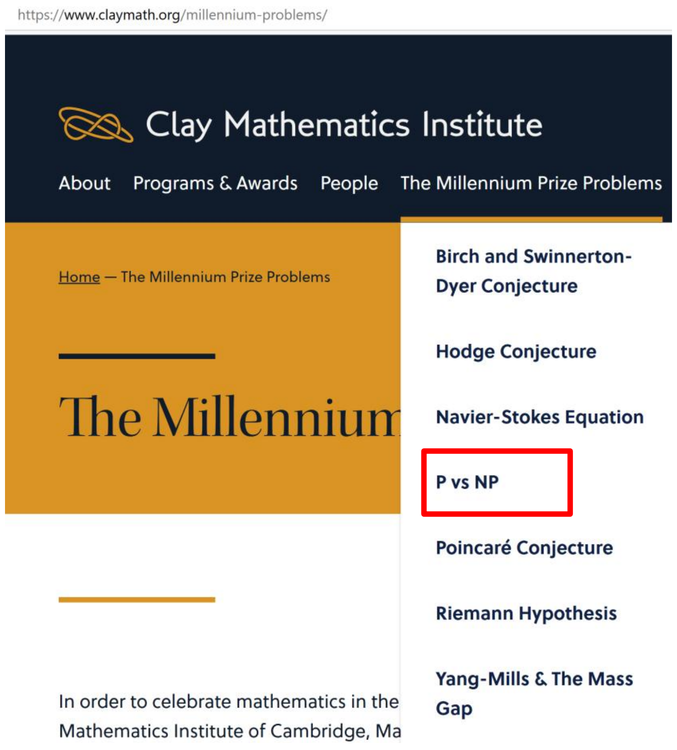
2519590847565789349402718324004839857142928212620403202777
7137836043662020707595556264018525880784406918290641249515
0821892985591491761845028084891200728449926873928072877767
3597141834727026189637501497182469116507761337985909570009
7330459748808428401797429100642458691817195118746121515172
6546322822168699875491824224336372590851418654620435767984
2338718477444792073993423658482382428119816381501067481045
1660377306056201619676256133844143603833904414952634432190
1146575444541784240209246165157233507787077498171257724679
6292638635637328991215483143816789988504044536402352738195
137863656439121201039712282212072035 7

$= \; ? \times ?$

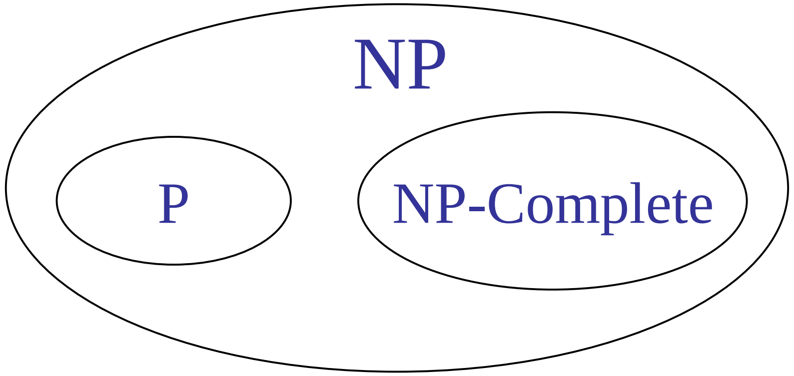
Exercise：大数分解的进展?

34.5 Does P = NP ?

- Clearly all problems in P are in NP, i.e. $P \subseteq NP$.
- Does P = NP?
- This is the biggest unsolved and the most challenging problem in computer science!

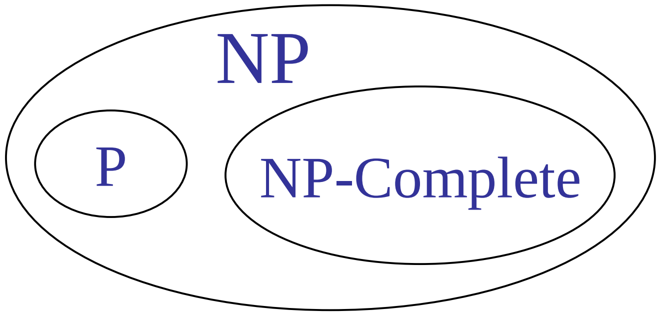


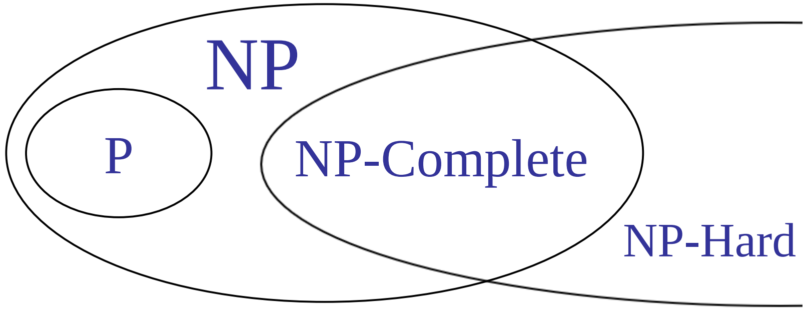
The current opinion is



34.6 NP-Complete Class

- Intuitively, NP-Complete is the class of the “most difficult” problems in NP.
- All NP-Complete problems appear to be difficult.
- No polynomial-time algorithm has been found for any NP-Complete problem.（对任何 NPC，尚没有找到多项式算法） For example,
  - Hamilitonian cycle problem
  - Boolean Satisfiability（布尔可满足性问题）
- NPC 问题：如果 NP 问题的所有可能答案都可以在多项式时间内进行正确与否的验算，就叫 NPC 问题（完全多项式非确定问题）。
- 一个可判定性问题 C 是 NP 完全（NPC）的，如果：
  1. 这个问题是 NP 问题。
  2. 所有其他的 NPC 问题可以归约为 C 问题。

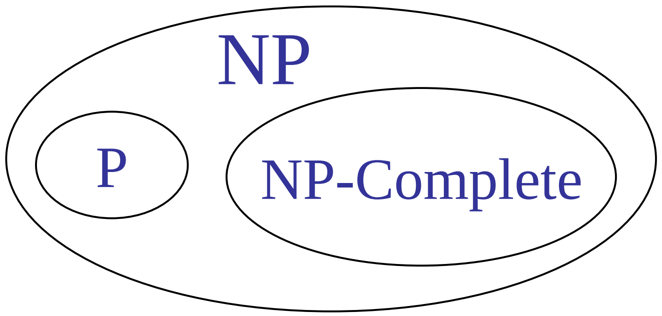




NP-hard: 对于判定问题 A，若 A 满足，所有的 NP 问题都可以约化到它。（NP-Hard 问题比 NPC 问题的范围广）NPC 和 NP-hard 的主要区别在于：验证一个问题 A 是否为 NP-hard 问题，无需判断 A 是否属于 NP。

34.6 NP-Complete Class

- 所有的 NPC 都可以在转换为 Boolean Satisfiability
- Cook 于1971证明了 Sat 是 NPC，现在发现的 NPC 已经超过3000个?
- 如果任一 NPC 问题多项式可解，则所有 NPC 都是多项式可解!



*34.7 Boolean Satisfiability

- A logical (Boolean) variable is a variable that may be assigned the value *true* (T or 1) or *false* (F or 0), e.g. ,
  - p, q, r and s are Boolean variables.
- A literal (文字) is a logical variable or the negation of a logical variable, e.g. p and $\neg q$ are literals.
  - If p = T, then $\neg p$ = F. If p = F, then $\neg p$ = T
- A clause (子句) is a disjunction (析取) of literals, e.g. $(p \vee q \vee s)$ and $(\neg q \vee r)$ are clauses.
  - If l = T, then $l \vee l_1 \vee \ldots \vee l_k$ = T
  - If l = F, then $l \vee l_1 \vee \ldots \vee l_k = l_1 \vee \ldots \vee l_k$

*34.7.1 Conjunctive Normal Form (CNF, 合取范式)

- A logical (Boolean) variable is a variable that may be assigned the value *true* (T or 1) or *false* (F or 0), e.g. ,
- A literal (文字) is a logical variable or the negation of a logical variable, e.g. p and $\neg q$ are literals.
- A clause (子句) is a disjunction (析取) of literals, e.g. $(p \vee q \vee s)$ and $(\neg q \vee r)$ are clauses.

- A logical (Boolean) formula is in Conjunctive Normal Form if it is a onjunction (合取) of clauses.
  - If c = T, then $c \wedge c_1 \wedge \ldots \wedge c_k = c_1 \wedge \ldots \wedge c_k$
  - If c = F, then $c \wedge c_1 \wedge \ldots \wedge c_k$ = F

- The following formula is in conjunctive normal form:
  - $(p \vee q \vee s) \wedge (\neg q \vee r) \wedge (\neg p \vee r) \wedge (\neg r \vee s)$

*34.7.2 The Satisfiability (SAT) problem (可满足性问题)

- Determine if a CNF formula has a solution (i.e. a truth assignment which makes the formula true)
  - Satisfiable(可满足的) formula: the answer is “yes”, all clauses must evaluate to true (or called satisfied)
  - Unsatisfiable formula: the answer is “no”
- Examples
  - A satisfiable formula
    - p = T, q = F, r = T and s = T is a solution for

      $(p \vee q \vee s) \wedge (\neg q \vee r) \wedge (\neg p \vee r) \wedge (\neg r \vee s)$
    - Each clause is satisfied.
  - An unsatisfiable formula
    - $(p \vee q) \wedge (\neg p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee \neg q)$

*Search Method 1: Exhaustive Search (穷举搜索)

- Exhaustive search is a method that searches for a solution by trying every possibility.
- For example
  - For a CNF formula of n variables, we have 2^n candidate solutions
  - $(p\vee q\vee r)\wedge(\neg q\vee r)\wedge(\neg p\vee r)\wedge(\neg p\vee q)$
- Conclusion
  - Certainly correct and complete (正确而且完全)
  - Impractical except for very small problems（仅对小问题适用)

p	q	r
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

*Search Method 1: Exhaustive Search (穷举搜索)

ExhSAT($\Gamma \in$CNF)

```
for every candidate solution S do
    if S satisfies Γ return(Yes);
return(No);
```

Example: $\Gamma = (p \vee q) \wedge (\neg p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee \neg q)$

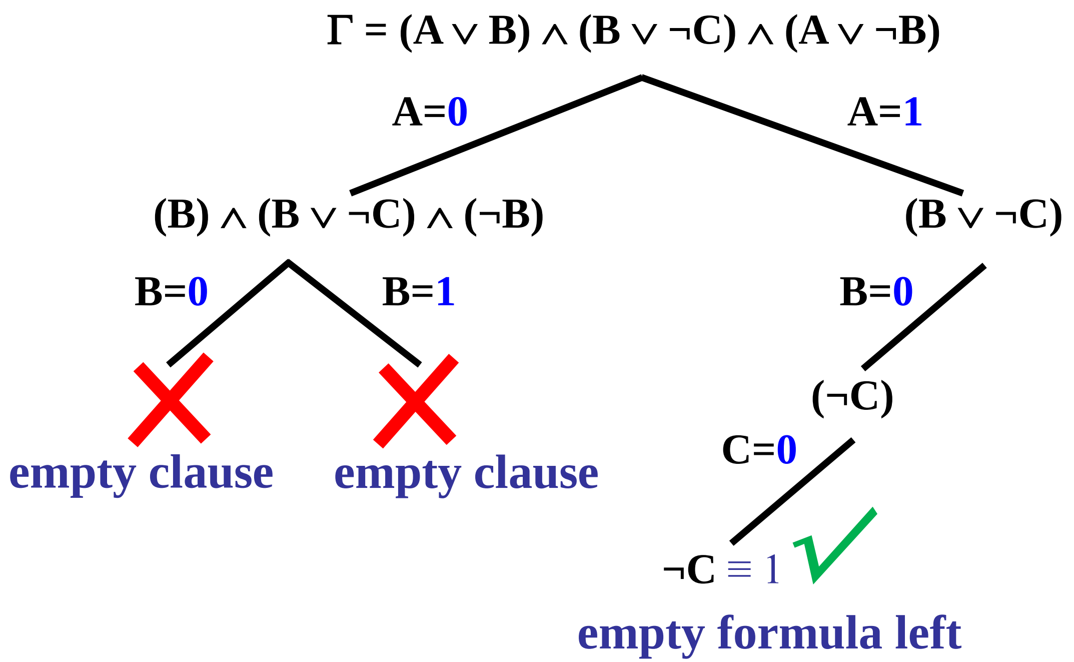
S1: $p = 0$, $q = 0$; S2: $p = 1$, $q = 0$; S3: $p = 0$, $q = 1$; S4: $p = 1$, $q = 1$.

None of the above candidate solutions satisfies Γ . So Γ is unsatisfiable.

*Search Method 2: Backtrack Search (回溯搜索)

BtSAT(Γ∈CNF)

```
if Γ is an empty formula, return(Yes);
    if Γ contains empty clauses, return(No);
    Select a variable x from Γ
    if BtSAT(Γ[x/1]) = Yes return(Yes); // Substitute x = 1 into Γ
    if BtSAT(Γ[x/0]) = Yes return(Yes); // Substitute x = 0 into Γ
    return(No);
```

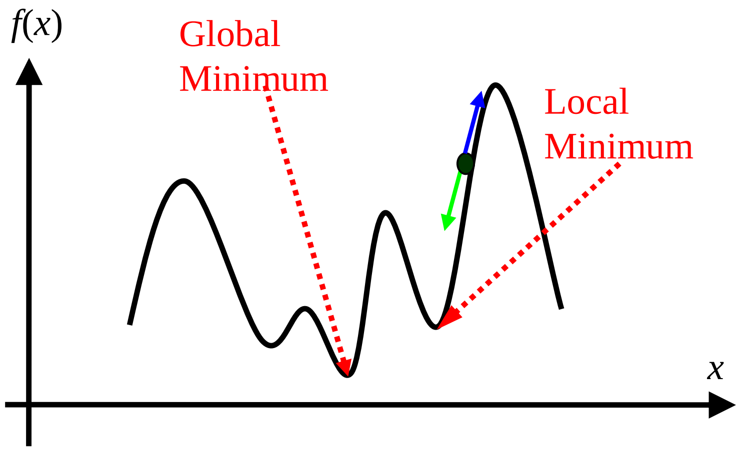


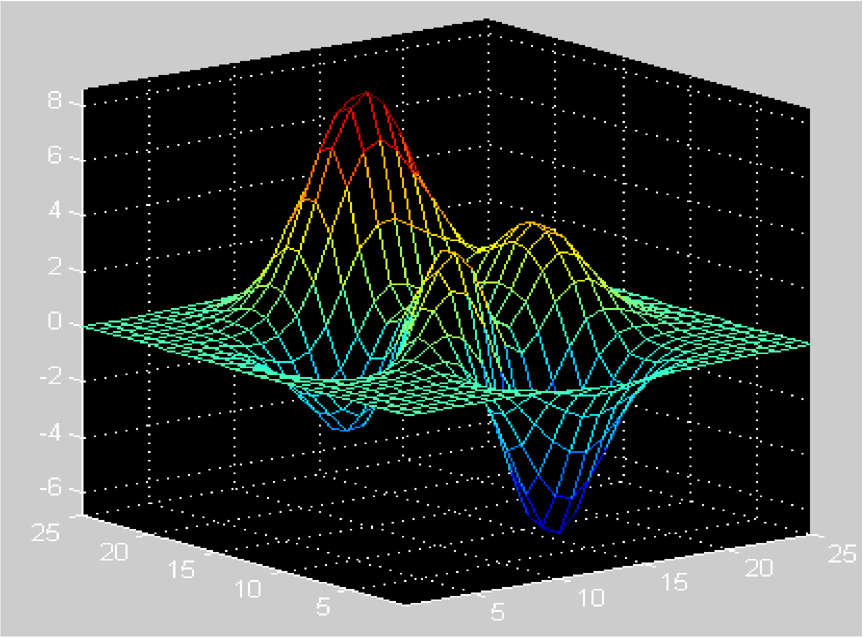
Basic idea

- Backtracking is a search method for systematically (系统地) trying all possibilities through a search tree.
- Search begins at the root node and goes down the tree as deep as possible until it either finds a solution or reaches a "dead end". If it finds a solution, returns "Yes". If it reaches a dead end, it backtracks to the most recent node and tries the next alternative path.
- Generally much better than exhaustive search.

*Search Method 3: Local Search (局部搜索)

- Local search is a simple search method that repeatedly moves from a candidate solution to the best candidate solution nearby.
- Incomplete but generally very efficient.





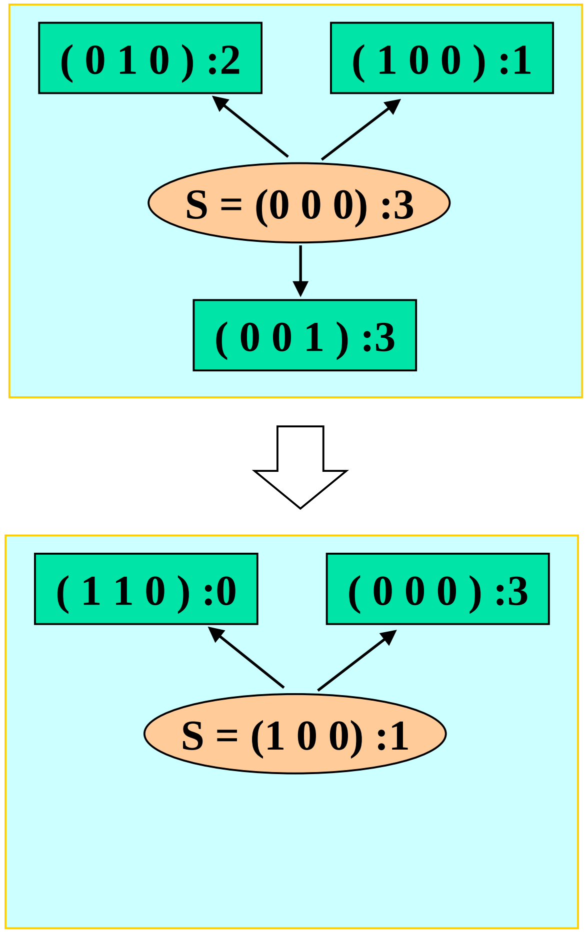
*Search Method 3: Local Search (局部搜索)

Example: Determine if the following formula is satisfiable.

$$\Gamma = (u1 \vee u2) \wedge (u1 \vee \neg u2) \wedge (u1 \vee u3) \wedge (u2 \vee u3) \wedge (u1 \vee \neg u3) \wedge (u2 \vee \neg u3)$$

		u1	u2	u3	
begin:	S =	0	0	0	unsatisfied clauses is 3
flip u1:		1	0	0	unsatisfied clauses is 1
flip u2:		0	1	0	unsatisfied clauses is 2
flip u3:		0	0	1	unsatisfied clauses is 3
	S =	1	0	0	
flip u1:		0	0	0	unsatisfied clauses is 3
flip u2:		1	1	0	unsatisfied clauses is 0

So Γ is satisfiable.

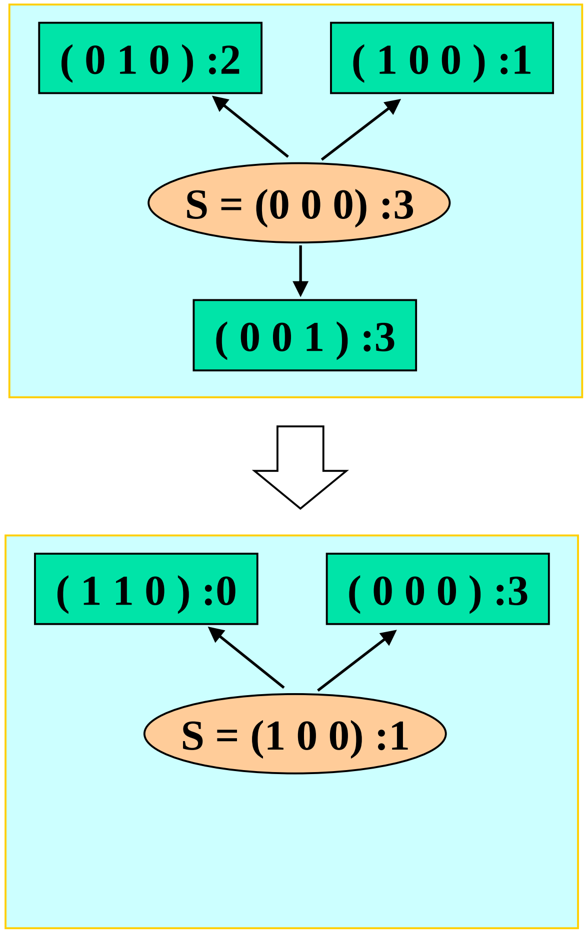


*Search Method 3: Local Search (局部搜索)

Example:

$\Gamma = (u1 \vee u2) \wedge (u1 \vee \neg u2) \wedge (u1 \vee u3) \wedge (u2 \vee u3) \wedge (u1 \vee \neg u3) \wedge (u2 \vee \neg u3)$

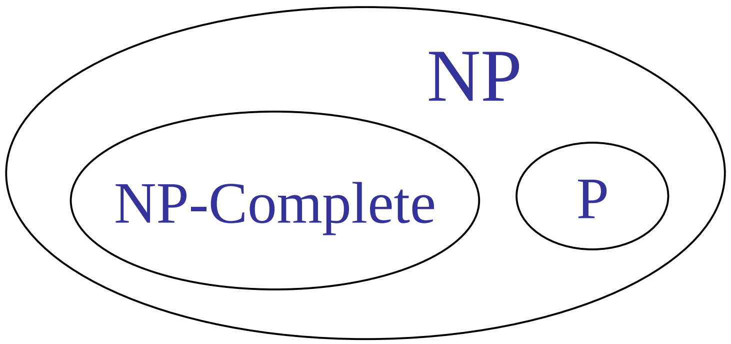
```
GSAT(Γ∈CNF)            // Las Vegas algorithm
for i = 1 to Max-tries
    S = random(Γ);         // random assignment
    for j = 1 to Max-moves
        if S satisfies Γ return(Yes);
        else S = S with some variable flipped to minimize the
                    number of unsatisfied clauses;
return(No satisfying truth assignment found);
```



*34.8 How to solute NP-Complete Class

- No polynomial-time algorithm has been found for any NP-Complete problem.
  对任何 NPC，尚没有找到多项式算法

- 特殊情况
- 概率分析
- 近似算法
- 启发式算法



*Exercises

Implement GSAT on computers.

GSAT($\Gamma \in$CNF) // Las Vegas algorithm
for i = 1 **to** Max-tries
    S = random(Γ); // random assignment
    for j = 1 **to** Max-moves
        if S satisfies Γ **return**(Yes);
        else S = S with some variable flipped to minimize the number of unsatisfied clauses;
return(No satisfying truth assignment found);