

# 34 NP 完全问题

## 34 NP 完全问题

---

$$21 = 3 \times 7$$

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

## 34 NP 完全问题

---

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

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

## 34 NP 完全问题

哥德巴赫猜想：

1+1：大偶数等于两个素数之和  $M = a + b$

1+2：  $M = a + b * c$

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。

...

已知某命题（陈述）是正确的，需要证明之？难！

判断一个命题（陈述）是否正确？更难！若正确，请证明；若不正确，举出反例。

## 34 NP 完全问题

- 千禧年数学难题（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**）

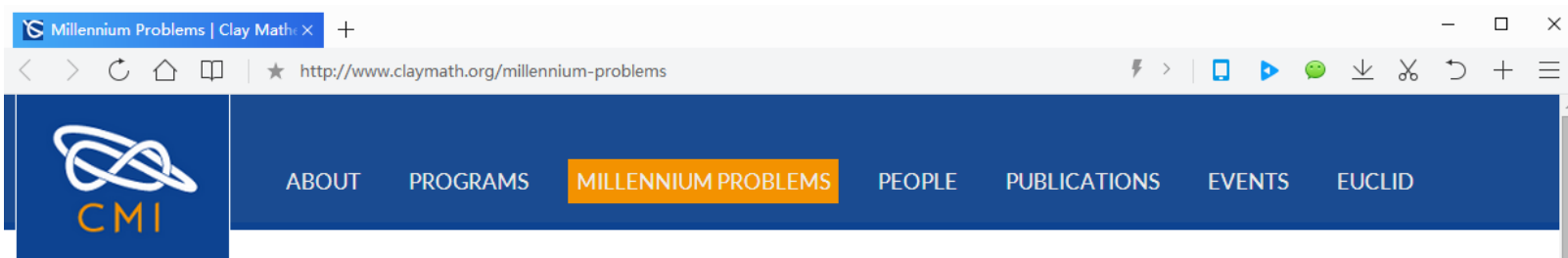
<http://www.claymath.org/millennium-problems>

<http://www.claymath.org/millennium-problems/millennium-prize-problems> (new)

- ◆ 之一： **P (多项式算法可解)问题对NP (“非确定性问题”)**
- ◆ 之二： 霍奇(Hodge)猜想
- ◆ 之三： 庞加莱(Poincare)猜想
- ◆ 之四： 黎曼(Riemann)假设
- ◆ 之五： 杨-米尔斯(Yang-Mills)存在性和质量缺口
- ◆ 之六： 纳维叶-斯托克斯(Navier-Stokes)方程的存在性与光滑性
- ◆ 之七： 贝赫(Birch)和斯维纳通-戴尔(Swinnerton-Dyer)猜想

# 34 NP 完全问题

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



## Millennium Problems

### Yang-Mills and Mass Gap

Experiment and computer simulations suggest the existence of a "mass gap" in the solution to the quantum versions of the Yang-Mills equations; no proof of this property is known.

### Riemann Hypothesis

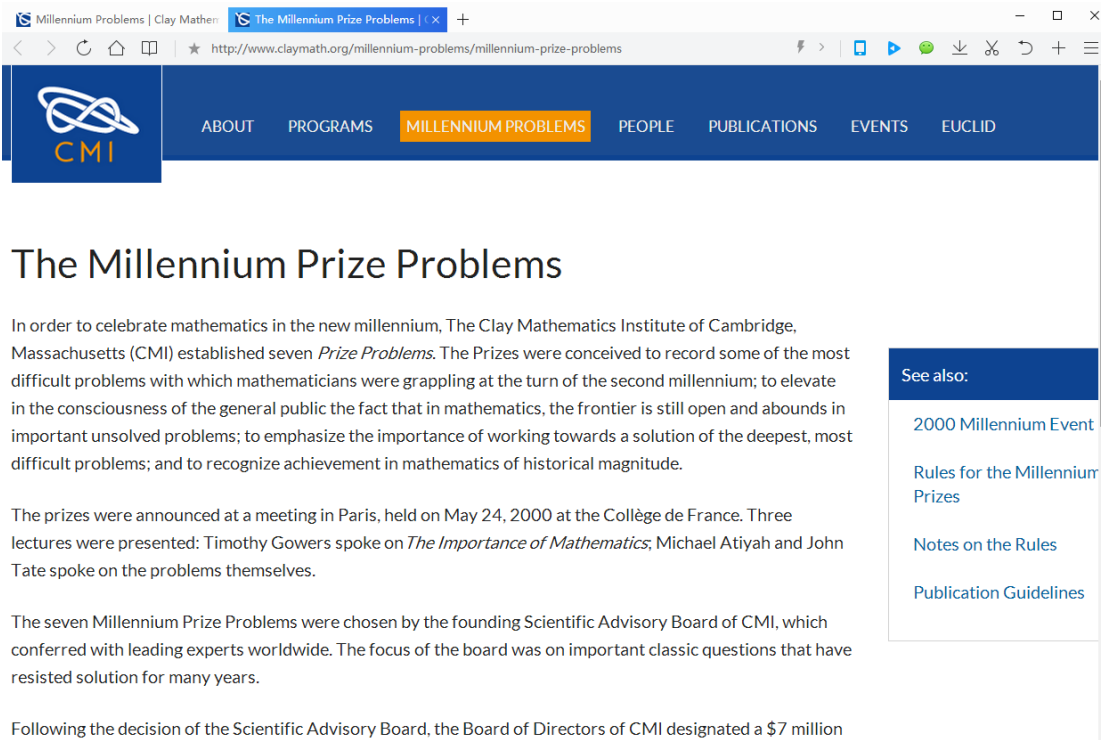
The prime number theorem determines the average distribution of the primes. The Riemann hypothesis tells us about the deviation from the average. Formulated in Riemann's 1859 paper, it asserts that all the 'non-obvious' zeros of the zeta function are complex numbers with real part  $1/2$ .

### P vs NP Problem

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given  $N$  cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

# 34 NP 完全问题

<http://www.claymath.org/millennium-problems/millennium-prize-problems> (new)



The screenshot shows a web browser window with the URL <http://www.claymath.org/millennium-problems/millennium-prize-problems>. The page features the CMI logo and a navigation menu with links to ABOUT, PROGRAMS, MILLENNIUM PROBLEMS (highlighted), PEOPLE, PUBLICATIONS, EVENTS, and EUCLID. The main heading is "The Millennium Prize Problems". The text describes the establishment of the prizes by the Clay Mathematics Institute of Cambridge, Massachusetts, in 2000, to celebrate mathematics in the new millennium. It mentions that the prizes were conceived to record some of the most difficult problems with which mathematicians were grappling at the turn of the second millennium; to elevate in the consciousness of the general public the fact that in mathematics, the frontier is still open and abounds in important unsolved problems; to emphasize the importance of working towards a solution of the deepest, most difficult problems; and to recognize achievement in mathematics of historical magnitude. It also notes that the prizes were announced at a meeting in Paris, held on May 24, 2000 at the Collège de France. Three lectures were presented: Timothy Gowers spoke on *The Importance of Mathematics*; Michael Atiyah and John Tate spoke on the problems themselves. The seven Millennium Prize Problems were chosen by the founding Scientific Advisory Board of CMI, which conferred with leading experts worldwide. The focus of the board was on important classic questions that have resisted solution for many years. Finally, it states that following the decision of the Scientific Advisory Board, the Board of Directors of CMI designated a \$7 million

See also:

- [2000 Millennium Event](#)
- [Rules for the Millennium Prizes](#)
- [Notes on the Rules](#)
- [Publication Guidelines](#)

根据科学顾问委员会的决定，CMI理事会为解决这些问题指定了700万美元的奖金基金，每个问题的解决方案分配了100万美元。

## 34 NP 完全问题



- 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）：找出一条通过所有城镇并回到原出发点的最短路线
  - ◆  $1 \rightarrow 2 \rightarrow 3 (1 \rightarrow 3 \rightarrow 2) \rightarrow \dots$ ，这两种走法的距离不一样。
  - ◆ 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  $T(20) = 1 \text{ h}$ ,  $T(21) = 21 \text{ h}$ ,  $\dots$ ,  $T(25) = 728 \text{ y}$



## 34 NP 完全问题

### 问题的复杂性和算法的复杂性

- 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。

## 34 NP 完全问题

### 判定问题

- 判定问题是一个答案为“是”或者“不是”的问题。

#### Examples

- ◆ 判断数组是否包含重复元素。
- ◆ 给定两个序列，判断是否存在长度至少为 $k$ 的公共子序列。
- 优化问题可以很容易地转换为判定问题
  - ◆ **Opti-Prob:** Shortest path  $u$  to  $v$  ?
  - ◆ **Deci-Prob:** If there exists a path  $u$  to  $v$   $< 2$  ?  $\text{Path}(u, v) < 3$  ? ...  
 $\text{Path}(u, v) < k$  ?
  - ◆ If  $S(u, v) = 3$ , then  $P(u, v) < 2$  is NO, ... ,  $P(u, v) < 4$  is YES

## 34.1 P类问题

---

- P是一类可以在多项式( $O(n^k)$ , where  $k$  is a constant)时间内解决的决策问题。直观地, P类问题是简单问题。
- For example
  - ◆ 判断数组是否包含重复元素。
  - ◆ 给定  $n$  个元素, 判断是否存在多数元素。

## 34.2 NP类问题

---

- NP 是一类我们可以在多项式时间内验证解的正确性的决策问题。
  - ◆ 这并不是说解很容易找到。
  - ◆ 事实上，通常很难找到一个解。

## 34.2 NP类问题

- P ? Polynomial

- NP: N?

- ◆ not Non-Polynomial, but Non-Deterministic

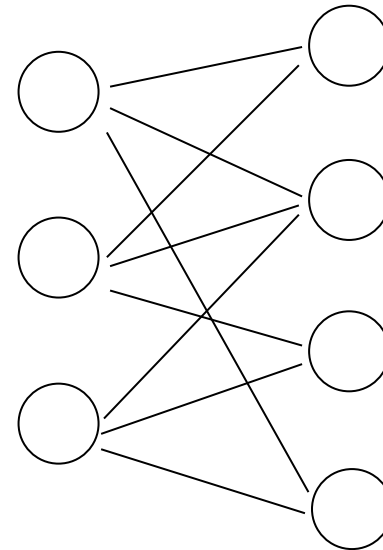
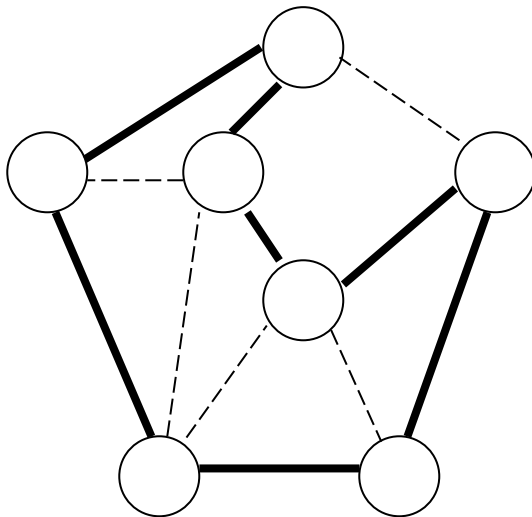
- (多项式复杂程度的非确定性问题，即多项式时间内能验证的判断问题。例如  $C = A * B$ ，先要猜想A和B（而没有公式能直接求出A和B，非确定性的意义也体现在这里），再验证 $C = A * B$ 是否成立。）

- 确定性问题, for example

- ◆ 加减乘除：只要按照公式推导，按部就班一步步来，就可以得到结果。

## 34.2 NP类问题

- 非确定性问题(无法按部就班直接地计算出来, e.g.)
  - ◆ Factorization (大的合数分解质因数的问题) :  $M = ? \times ?$
  - ◆ 图  $G$  的哈密顿回路是一个循环, 正好通过每个顶点一次。
  - ◆ Problem: 给定图  $G$ , 是否  $G$  有一个哈密顿回路?



## 34.3 P and NP

- Problems in P can be solved “quickly”
- Problems in NP can be verified “quickly”
- 验证一个解比求解一个问题更容易。

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

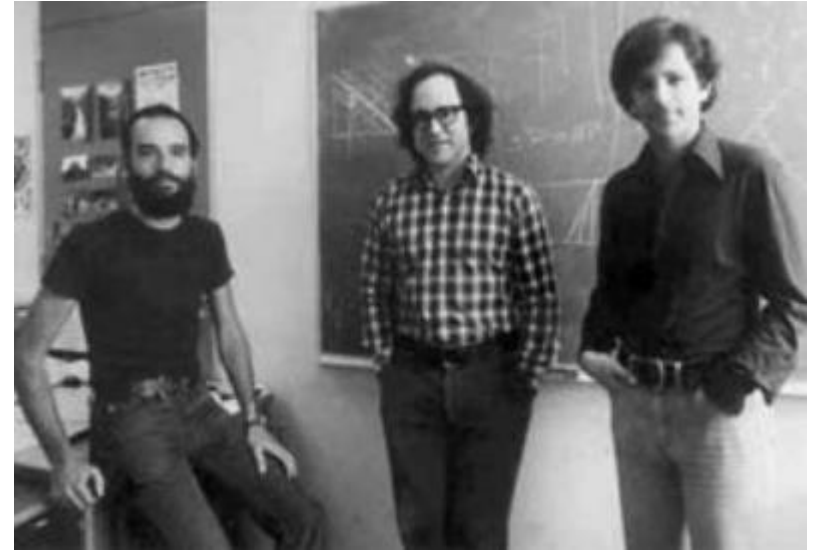
$$2^{67}-1 = 761,838,257,287 \times 193,707,721$$

Cole 花了 **150 天** 来找到一个分解!  
但是一旦解被找到, 我们可以  
快速地**验证**它!

科学					历史记录	内存
147,573,952,589,676,412,927					761838257287 × 193707721 =	
					147,573,952,589,676,	
					412,927	
					2 ^ 67 - 1 =	
					147,573,952,589,676,	
					412,927	
DEG	HYP	F-E				
MC	MR	M+	M-	MS		
$x^2$	$x^y$	sin	cos	tan		
√	10 <sup>x</sup>	log	Exp	Mod		
↑	CE	C	⊞	÷		
π	7	8	9	×		
n!	4	5	6	—		
±	1	2	3	+		

## 34.4 RSA 加密

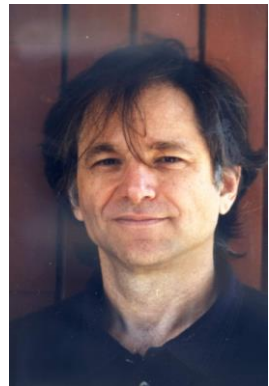
- 公钥密码系统的概念是由Diffie和Hellman在1976年提出的。
- RSA加密系统是由Rivest, Shamir和Adleman提出的, 1977, MIT(图灵奖, 2002)



Rivest



Shamir



Adleman





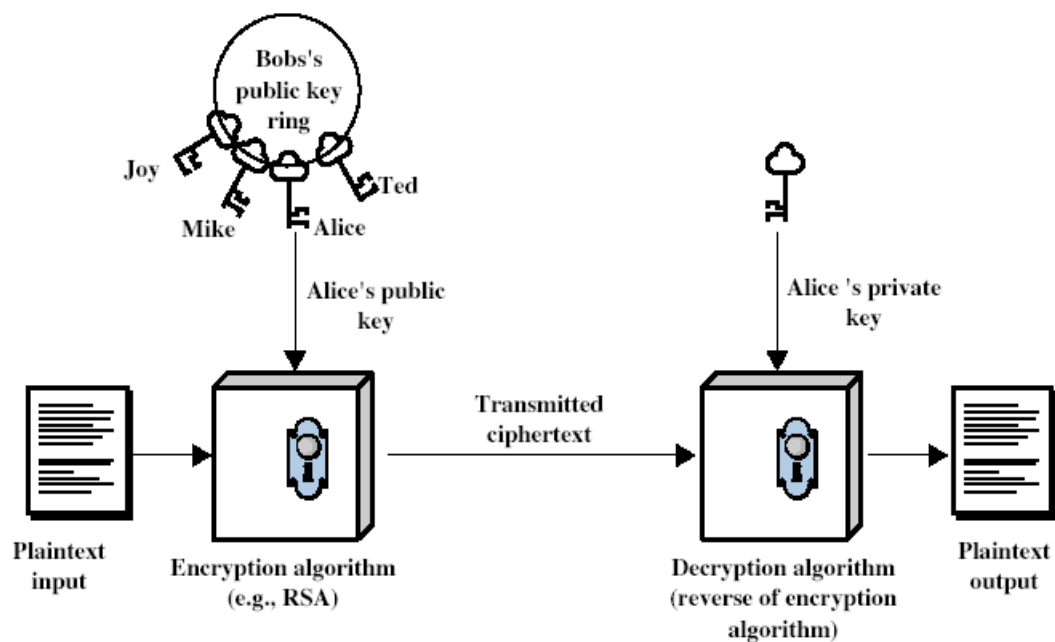
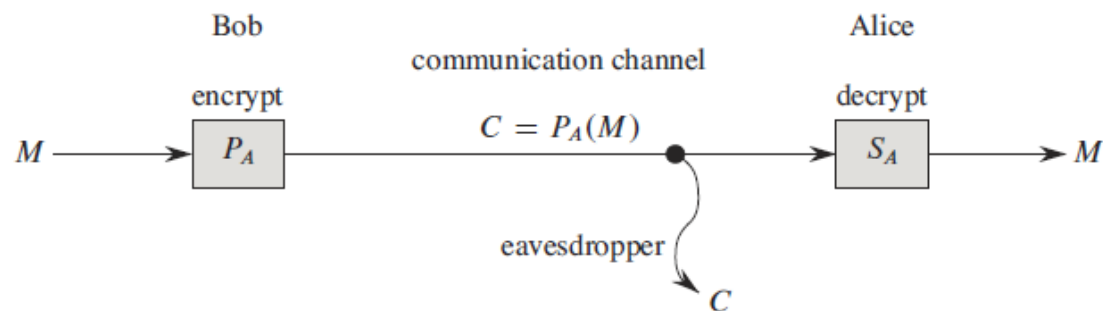
Shamir



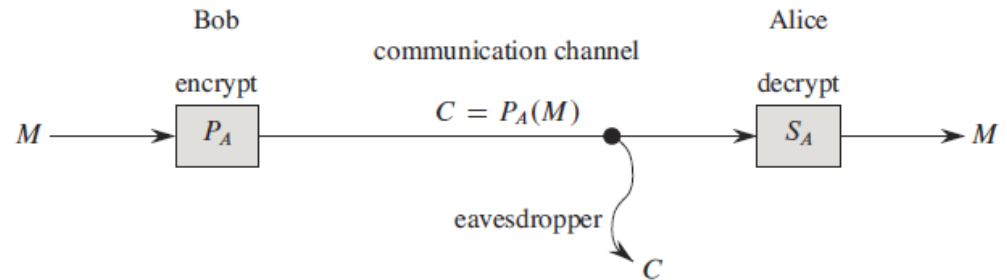
**Turing Award, 图灵奖获得者在北航演讲**

## 34.4 RSA 加密

### 公钥系统中的加密



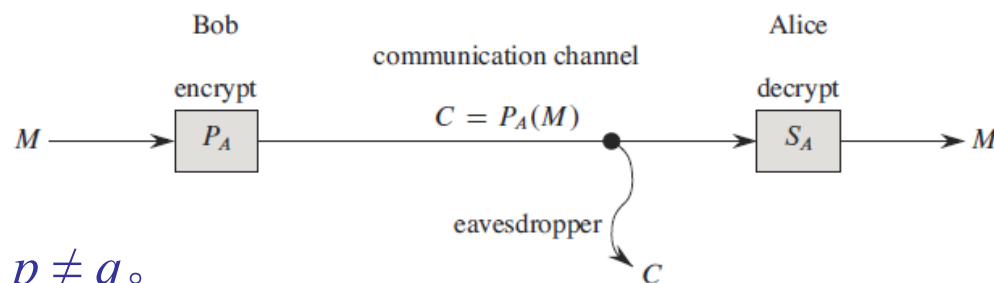
## 34.4 RSA encryption



### ● RSA public-key cryptosystem

1. 随机选择两个大素数  $p$  和  $q$ ，使  $p \neq q$ 。素数  $p$  和  $q$  可以分别为512位、1024位或更多。
2. 用公式  $n = pq$  计算  $n$ 。
3. 令  $\varphi(n) = (p - 1)(q - 1)$ . // 之后, 让  $p$  和  $q$  消失。
4. 选择一个与  $\varphi(n)$  互素的小奇数  $e$ ，即:  $\gcd(e, \varphi(n)) = 1$ 。
5. 计算  $e$  的乘法逆  $d$ ，模  $\varphi(n)$ ，即:  $ed \equiv 1 \pmod{\varphi(n)}$
6. 发布对  $P = (e, n)$  作为他的 **RSA 公钥**。
7. 保密对  $S = (d, n)$  作为他的 **RSA 私钥**。

# 为什么RSA加密是有效的



- **RSA public-key cryptosystem**

1. 随机选择两个大素数  $p$  和  $q$ , 使  $p \neq q$ 。
2. 计算  $n = pq$ .
3. 计算  $\varphi(n) = (p - 1)(q - 1)$ . // Euler's phi function
4. 选一个小奇数  $e$ , s.t.,  $\gcd(e, \varphi(n))=1$ .
5. 计算  $d$ , 通过等式  $ed \equiv 1 \pmod{\varphi(n)}$
6. 公开对  $P = (e, n)$  作为 **RSA public key**.
7. 保密对  $S = (d, n)$  作为 **RSA secret key**.

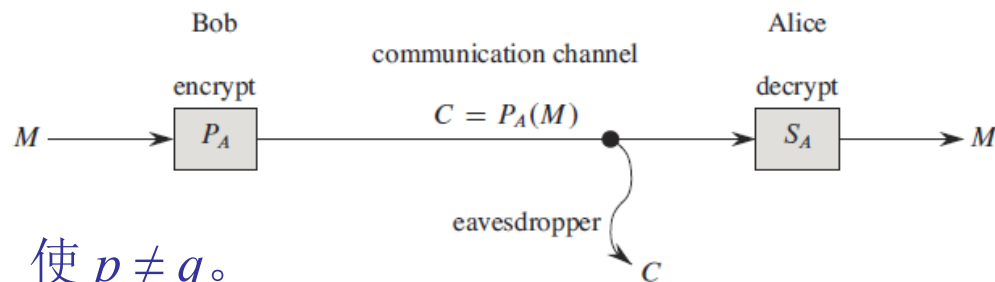
- **RSA 是正确的**

Theorem 31.36:

$$\begin{aligned} S(P(M)) &= S(M^e \pmod n) = (M^e \pmod n)^d \pmod n = M^{ed} \pmod n \\ &= M^{1+k\varphi(n)} \pmod n = M M^{k\varphi(n)} \pmod n = M (M^{p-1})^{k(q-1)} \pmod n \dots = M \pmod n \end{aligned}$$

(by Theorem 31.31, Fermat's Theorem, if  $p$  is prime,  $M^{p-1} = 1 \pmod n$ )

# 为什么RSA加密是有效的



- **RSA public-key cryptosystem**

1. 随机选择两个大素数  $p$  和  $q$ , 使  $p \neq q$ 。
2. 计算  $n = pq$ .
3. 计算  $\varphi(n) = (p - 1)(q - 1)$ .
4. 选一个小奇数  $e$ , s.t.,  $\gcd(e, \varphi(n))=1$ .
5. 计算  $d$ , 通过等式  $ed \equiv 1 \pmod{\varphi(n)}$
6. 公开对  $P = (e, n)$  作为 **RSA public key**.
7. 保密对  $S = (d, n)$  作为 **RSA secret key**.

- **RSA 是安全的**

$$S(P(M)) = S(M^e \pmod n) = M^{ed} \pmod n = \dots = M$$

- ◆ 如果分解 $n$ 很容易, 那么破解RSA密码系统也很容易。
- ◆ 相反, 如果分解大整数是困难的, 那么破解RSA是困难的, 这是未经证明的。
- ◆ 然而, 经过几十年的研究, 还没有找到比分解 $n$ 更容易的方法来破解RSA公钥密码系统。
- ◆ 事实上, 大整数的因式分解异常困难。

# RSA: number-theoretic algorithms (Chapter 31)

## RSA public-key cryptosystem

1. 选择两个大素数  $p$  和  $q$ , 使  $p \neq q$ 。
2. 计算  $n = pq$ . // **Question1: 高精度乘法**
3. 计算  $\varphi(n) = (p - 1)(q - 1)$ . // **Q2: Euler's phi function, 高精度减法、乘法**
4. 选一个小奇数  $e$ , s.t.,  $\gcd(e, \varphi(n)) = 1$ . // **Q3: 高精度除法、辗转相除**
5. 计算  $d$ , 通过等式  $ed \equiv 1 \pmod{\varphi(n)}$  // **Q4: 求方程  $ex \equiv 1$ , 高精度运算**
6. 公开对  $P = (e, n)$  作为 **RSA public key**.
7. 保密对  $S = (d, n)$  作为 **RSA secret key**.

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

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

Chapter 31 .....

# RSA: number-theoretic algorithms (Chapter 31)

## RSA public-key cryptosystem

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

1. 选择两个大素数  $p$  和  $q$ , 使  $p \neq q$ 。
2. 计算  $n = pq$ . // **Question1: 高精度乘法**
3. 计算  $\varphi(n) = (p - 1)(q - 1)$ . // **Q2: Euler's phi function, 高精度减法、乘法**
4. 选一个小奇数  $e$ , s.t.,  $\gcd(e, \varphi(n)) = 1$ . // **Q3: 高精度除法、辗转相除**
5. 计算  $d$ , 通过等式  $ed \equiv 1 \pmod{\varphi(n)}$  // **Q4: 求方程  $ex \equiv 1$ , 高精度运算**
6. 公开对  $P = (e, n)$  作为 **RSA public key**.
7. 保密对  $S = (d, n)$  作为 **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  $a^c \pmod{n}$  as  $c$  is increased by doublings and incrementations from 0 to  $b$ .

MODULAR-EXPONENTIATION( $a, b, n$ )

```
1   $c = 0$ 
2   $d = 1$ 
3  let  $\langle b_k, b_{k-1}, \dots, b_0 \rangle$  be the binary representation of  $b$ 
4  for  $i = k$  downto 0
5       $c = 2c$ 
6       $d = (d \cdot d) \pmod{n}$ 
7      if  $b_i == 1$ 
8           $c = c + 1$ 
9           $d = (d \cdot a) \pmod{n}$ 
10 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
<a href="#">RSA-100</a>	100	330	US\$1,000 <sup>[4]</sup>	April 1, 1991 <sup>[5]</sup>	<a href="#">Arjen K. Lenstra</a>
<a href="#">RSA-110</a>	110	364	US\$4,429 <sup>[4]</sup>	April 14, 1992 <sup>[5]</sup>	<a href="#">Arjen K. Lenstra</a> and <a href="#">M.S. Manasse</a>
<a href="#">RSA-120</a>	120	397	\$5,898 <sup>[4]</sup>	July 9, 1993 <sup>[6]</sup>	<a href="#">T. Denny</a> <i>et al.</i>
<a href="#">RSA-129</a> <sup>[**]</sup>	129	426	US\$100	April 26, 1994 <sup>[5]</sup>	<a href="#">Arjen K. Lenstra</a> <i>et al.</i>
<a href="#">RSA-130</a>	130	430	US\$14,527 <sup>[4]</sup>	April 10, 1996	<a href="#">Arjen K. Lenstra</a> <i>et al.</i>
<a href="#">RSA-140</a>	140	463	US\$17,226	February 2, 1999	<a href="#">Herman te Riele</a> <i>et al.</i>
<a href="#">RSA-150</a>	150	496		April 16, 2004	<a href="#">Kazumaro Aoki</a> <i>et al.</i>
<a href="#">RSA-155</a>	155	512	\$9,383 <sup>[4]</sup>	August 22, 1999	<a href="#">Herman te Riele</a> <i>et al.</i>
<a href="#">RSA-160</a>	160	530		April 1, 2003	<a href="#">Jens Franke</a> <i>et al.</i> , <a href="#">University of Bonn</a>
<a href="#">RSA-170</a> <sup>[*]</sup>	170	563		December 29, 2009	D. Bonenberger and M. Krone <sup>[***]</sup>
<a href="#">RSA-576</a>	174	576	US\$10,000	December 3, 2003	<a href="#">Jens Franke</a> <i>et al.</i> , <a href="#">University of Bonn</a>
<a href="#">RSA-180</a> <sup>[*]</sup>	180	596		May 8, 2010	S. A. Danilov and I. A. Popovyan, <a href="#">Moscow State University</a> <sup>[7]</sup>
<a href="#">RSA-190</a> <sup>[*]</sup>	190	629		November 8, 2010	A. Timofeev and I. A. Popovyan
<a href="#">RSA-640</a>	193	640	US\$20,000	November 2, 2005	<a href="#">Jens Franke</a> <i>et al.</i> , <a href="#">University of Bonn</a>
<a href="#">RSA-200</a> <sup>[*]</sup> ?	200	663		May 9, 2005	<a href="#">Jens Franke</a> <i>et al.</i> , <a href="#">University of Bonn</a>
<a href="#">RSA-210</a> <sup>[*]</sup>	210	696		September 26, 2013 <sup>[8]</sup>	Ryan Propper
<a href="#">RSA-704</a> <sup>[*]</sup>	212	704	US\$30,000	July 2, 2012	Shi Bai, Emmanuel Thomé and <a href="#">Paul Zimmermann</a>
<a href="#">RSA-220</a> <sup>[*]</sup>	220	729		May 13, 2016	S. Bai, P. Gaudry, A. Kruppa, E. Thomé and P. Zimmermann
<a href="#">RSA-230</a> <sup>[*]</sup>	230	762		August 15, 2018	Samuel S. Gross, <a href="#">Noblis, Inc.</a> <sup>Ⓔ</sup>
<a href="#">RSA-232</a>	232	768			
<a href="#">RSA-768</a> <sup>[*]</sup>	232	768	US\$50,000	December 12, 2009	<a href="#">Thorsten Kleinjung</a> <i>et al.</i>
<a href="#">RSA-240</a>	240	795			
<a href="#">RSA-250</a>	250	829			
<a href="#">RSA-260</a>	260	862			
<a href="#">RSA-270</a>	270	895			
<a href="#">RSA-896</a>	270	896	US\$75,000		

\* The number was factored after the challenge became inactive

# The RSA Challenge Numbers

- RSA-576 (Factored )                      { Decimal Digits: 174 }

18819881292060796383869723946165043980716356337941  
73827007633564229888597152346654853190606065047430  
453173880113033967161996923212057340318795506569962  
21305168759307650257059

=

398075086424064937397125500550386491199064362342526  
708406385189575946388957261768583317

×

47277214610743530253622307197304822463291469530209  
7116459852171130520711256363590397527

# The RSA Challenge Numbers

- RSA-768 (Factored )

123018668453011775513049495838496272077285356959533479219732  
245215172640050726365751874520219978646938995647494277406384  
592519255732630345373154826850791702612214291346167042921431  
1602221240479274737794080665351419597459856902143413

=

334780716989568987860441698482126908177047949837137685689124  
31388982883793878002287614711652531743087737814467999489

×

367460436667995904282446337996279526322791581643430876426760  
32283815739666511279233373417143396810270092798736308917

# The RSA Challenge Numbers

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

251959084756578934940271832400483985714292821262040320277771378  
360436620207075955562640185258807844069182906412495150821892985  
591491761845028084891200728449926873928072877767359714183472702  
618963750149718246911650776133798590957000973304597488084284017  
974291006424586918171951187461215151726546322822168699875491824  
224336372590851418654620435767984233871847744479207399342365848  
238242811981638150106748104516603773060562016196762561338441436  
038339044149526344321901146575444541784240209246165157233507787  
077498171257724679629263863563732899121548314381678998850404453  
64023527381951378636564391212010397122822120720357

= ? × ?

# The RSA Challenge Numbers

---

Before 2007, if you need money.

[www.rsa.com/rsalabs/node.asp?id=2093](http://www.rsa.com/rsalabs/node.asp?id=2093) ?

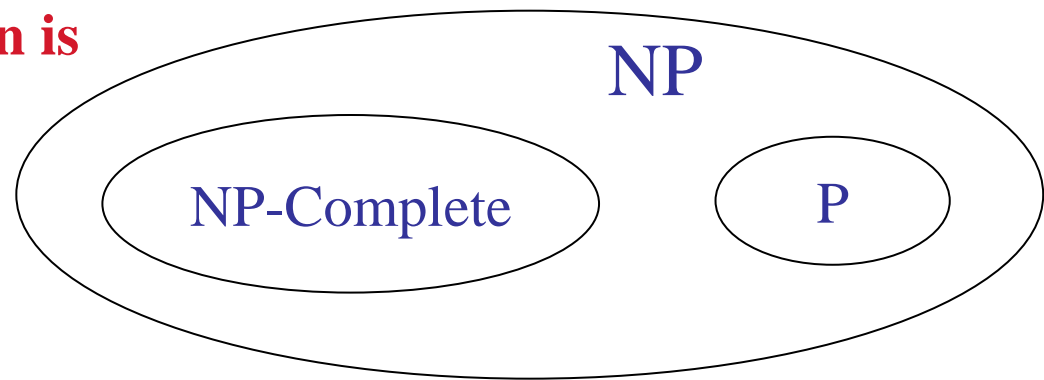
The Challenge is **no longer active** since 2007.

**To know RSA more, Baidu or Wiki.**

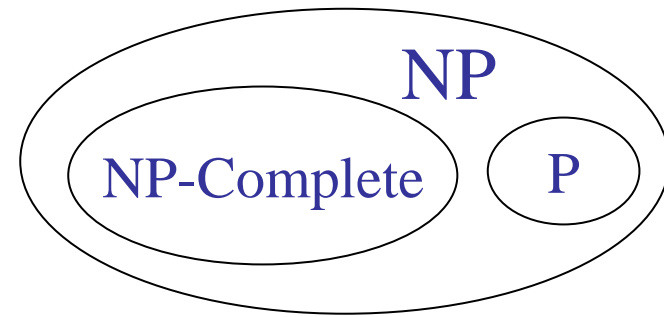
## 34.5 Does $P = NP$ ?

- 显然 $P$ 中的所有问题都在 $NP$ 中, i.e.  $P \subseteq NP$ .
- Does  $P = NP$ ?
- 这是计算机科学中最大的未解问题, 也是最具挑战性的问题!

The current opinion is



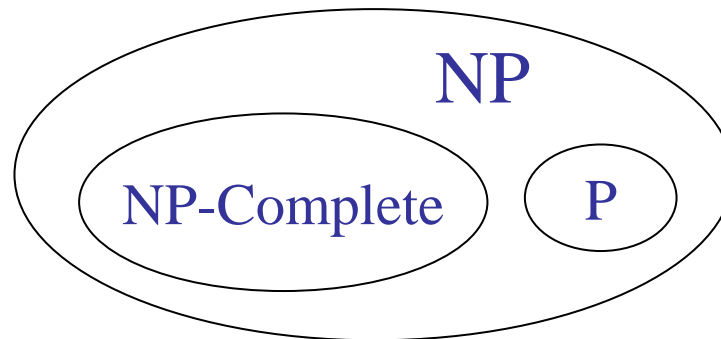
## 34.6 NP-Complete Class



- 显然P中的所有问题都在NP中, i. e.  $P \subseteq NP$ .
- 直观来说, NP- complete是NP中“最难”的一类问题。
- 所有NP完全问题看起来都很困难。
- 对任何NPC, 尚没有找到多项式算法。 For example,
  - ◆ 哈密顿回路问题
  - ◆ 布尔可满足性问题
- 如果NP问题的所有可能答案, 都是可以在多项式时间内进行正确与否的验算的话, 就叫完全多项式非确定问题 (NPC) 。
- 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NP问题可以归约为C问题。

## 34.6 NP-Complete Class

- 所有的NPC都可以转换为 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！
  - ◆ Complete（封闭的、完全的）





## 34.7 布尔可满足性

- **逻辑(布尔)变量**是可以赋值为 *true* (T or 1) 或 *false* (F or 0) 的变量, e.g.,
  - ◆  $p, q, r$  和  $s$  是布尔变量.
- **A literal (文字)** 是逻辑变量或者逻辑变量的否定, 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)** 是文字的析取, e.g.  $(p \vee q \vee s)$  and  $(\neg q \vee r)$  are clauses.
  - ◆ If  $l=T$ , then  $l \vee l_1 \vee \dots \vee l_k = T$
  - ◆ If  $l=F$ , then  $l \vee l_1 \vee \dots \vee l_k = l_1 \vee \dots \vee l_k$

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

- 一个逻辑 (布尔) 公式是**合取范式**的形式如果它是子句的合取。
  - ◆ If  $c=T$ , then  $c \wedge c_1 \wedge \dots \wedge c_k = c_1 \wedge \dots \wedge c_k$
  - ◆ If  $c=F$ , then  $c \wedge c_1 \wedge \dots \wedge c_k = F$
- 以下公式是合取范式形式:
  - ◆  $(p \vee q \vee s) \wedge (\neg q \vee r) \wedge (\neg p \vee r) \wedge (\neg r \vee s)$

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

- 确定CNF公式是否有解(i.e. 使公式为真的一组赋值)
  - ◆ 可满足公式:答案为“**Yes**”, 所有从句必须评估为真(或称为被满足)
  - ◆ 不可满足公式:答案是“**No**”
- Examples
  - ◆ 一个可满足的公式
    - $p=T, q=F, r=T$  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.
  - ◆ 一个无法满足的公式
    - $(p \vee q) \wedge (\neg p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee \neg q)$

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

- 穷举搜索是一个通过尝试各种可能性来寻找解决方案的搜索方法。
- 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( $\Gamma \in \text{CNF}$ )**

**begin**

**for every candidate solution S do**

**if S satisfies  $\Gamma$  then return(Yes);**

**return(No);**

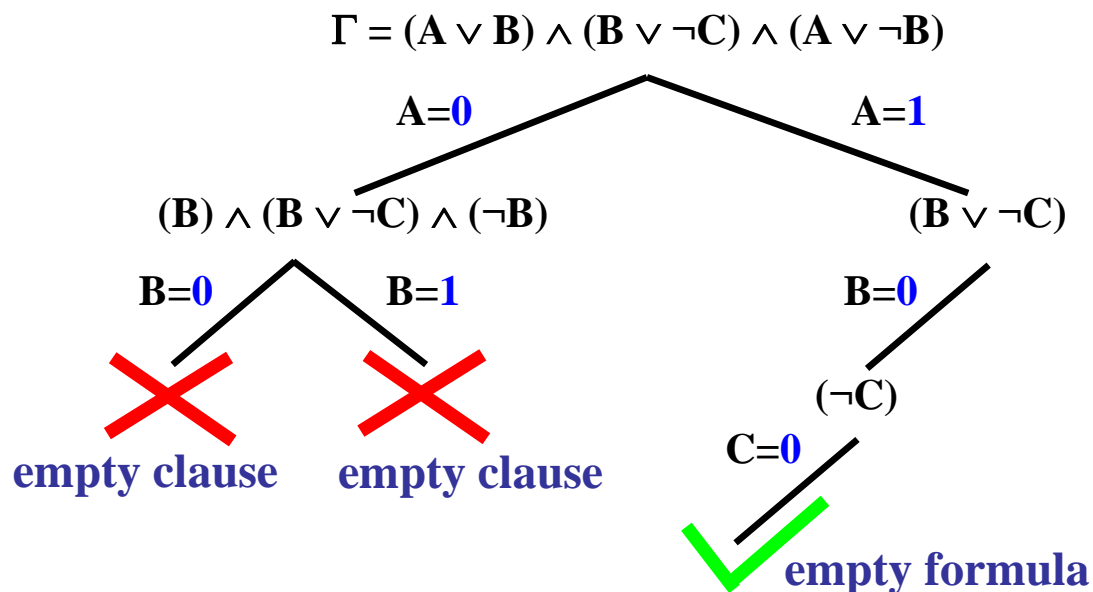
**end**

**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:**  $p=1, q=1$ .

上面的候选解都不满足 $\Gamma$ ，所以 $\Gamma$ 是不满足的。

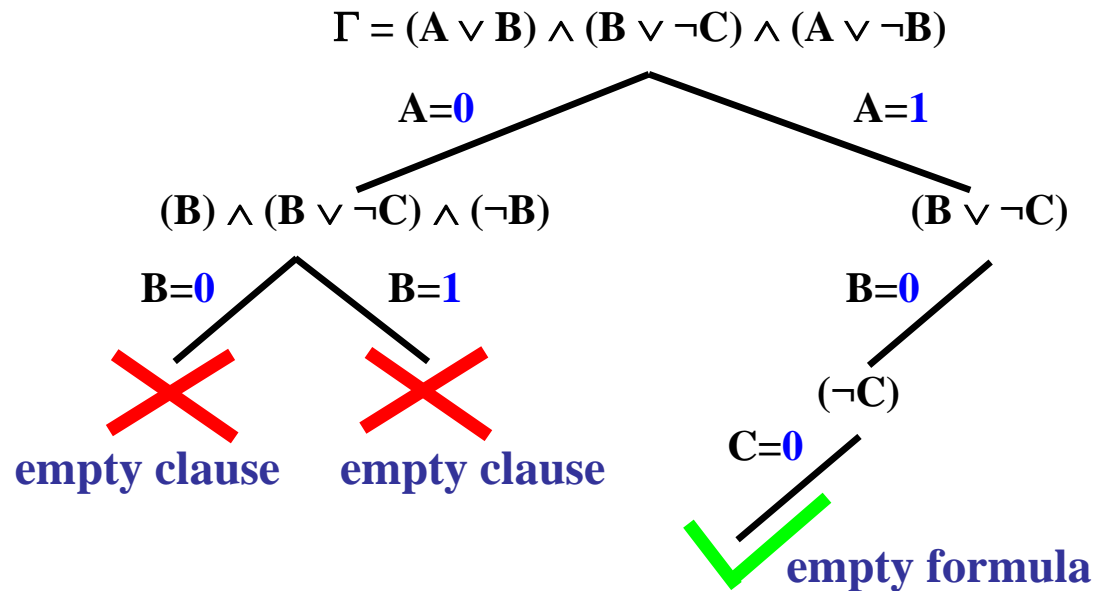
## Search Method 2: Backtrack Search (回溯搜索)



### Basic idea

- 回溯是一种搜索方法，它系统地通过搜索树尝试所有的可能性。
- 搜索从根节点开始，尽可能深入树，直到找到解决方案或到达“死胡同”。如果找到解决方案，则返回“Yes”。如果它到达一个死胡同，它会返回到最近的节点并尝试下一个替代路径。
- 通常比穷举搜索好得多。

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



**BtSAT( $\Gamma \in \text{CNF}$ )**

**begin**

if  $\Gamma$  is an empty formula, then return(Yes);

if  $\Gamma$  contains empty clauses, then return(No);

Select a variable  $x$  from  $\Gamma$

if BtSAT( $\Gamma[x/1]$ )=Yes then return(Yes); // Substitute  $x=1$  into  $\Gamma$

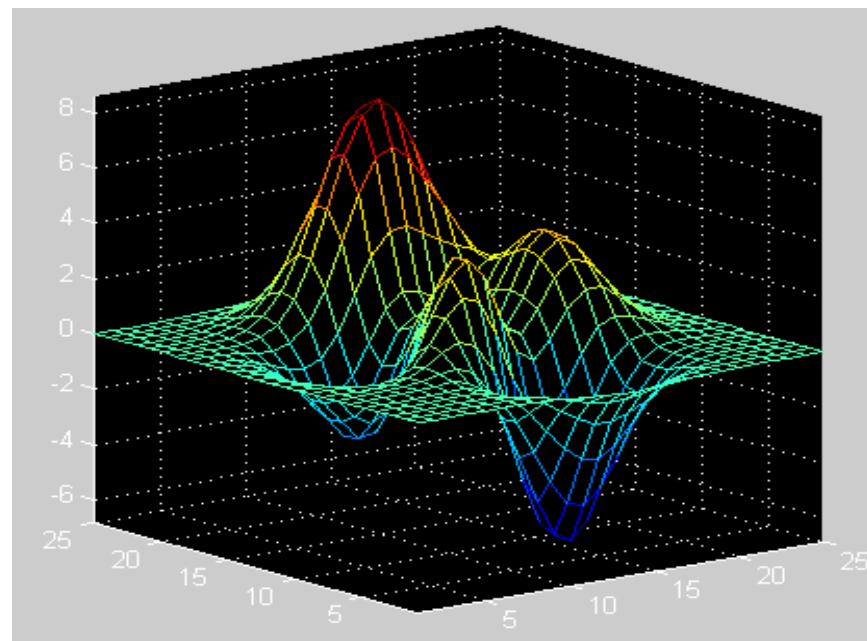
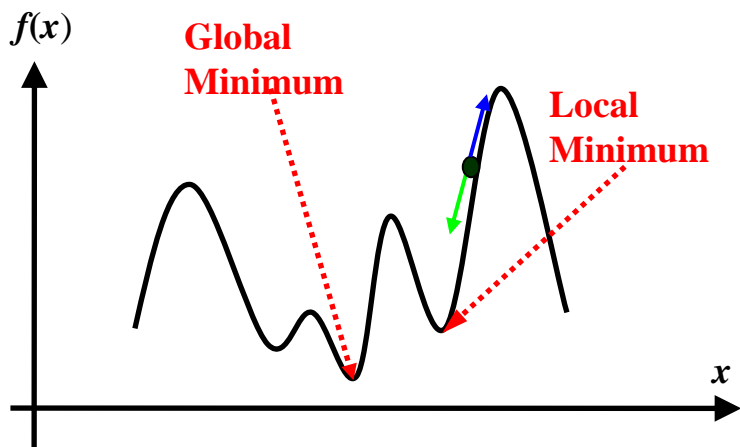
if BtSAT( $\Gamma[x/0]$ )=Yes then return(Yes); // Substitute  $x=0$  into  $\Gamma$

return(No);

**end**

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

- 局部搜索是一种简单的搜索方法，它重复地从一个候选解移动到附近的最佳候选解。
- 不完整，但通常非常有效。





## 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=3

flip u1: 1 0 0 unsatisfied clauses=1

flip u2: 0 1 0 unsatisfied clauses=2

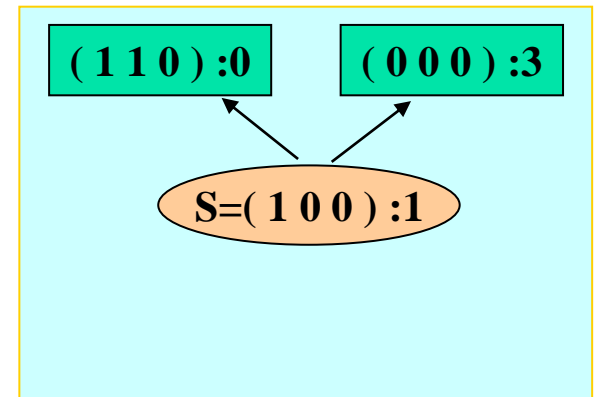
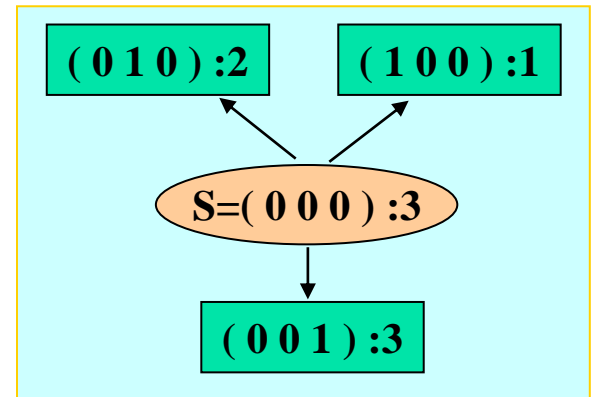
flip u3: 0 0 1 unsatisfied clauses=3

S= 1 0 0

flip u1: 0 0 0 unsatisfied clauses=3

flip u2: 1 1 0 unsatisfied clauses=0

So  $\Gamma$  is satisfiable.

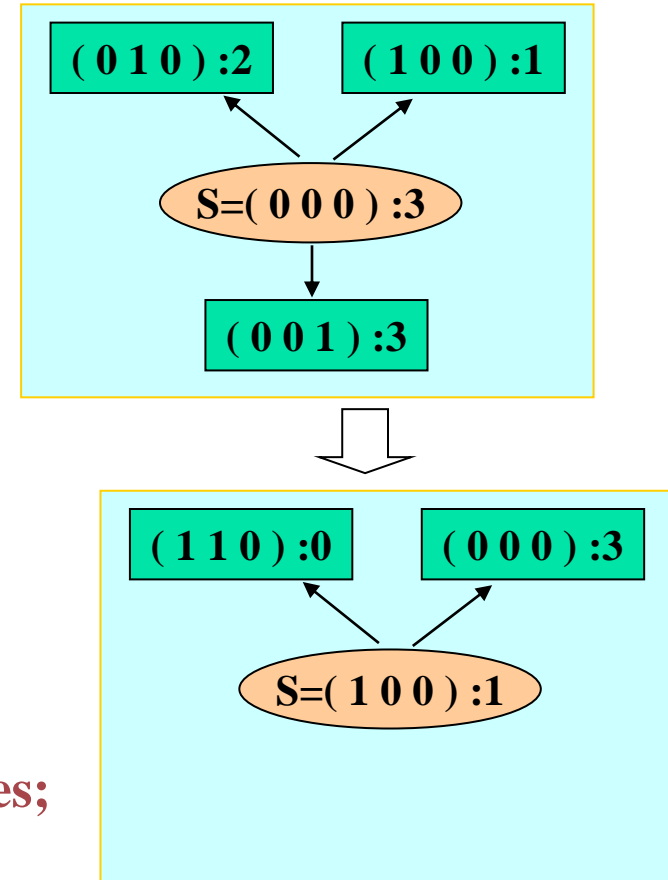


## Search Method 3: Local Search (局部搜索)

**Example:**

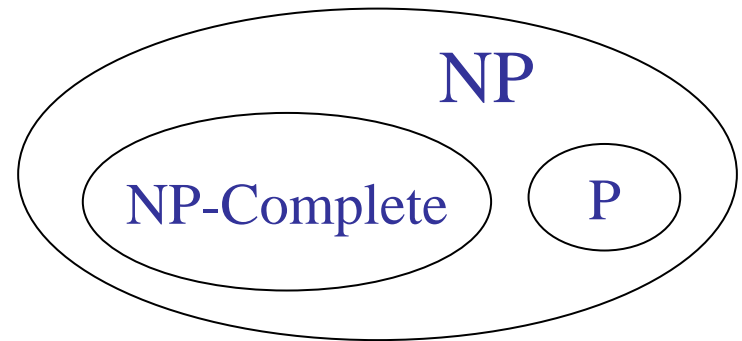
$$\Gamma = (u1 \vee u2) \wedge (u1 \vee \neg u2) \wedge (u1 \vee u3) \\ \wedge (u2 \vee u3) \wedge (u1 \vee \neg u3) \wedge (u2 \vee \neg u3)$$

**GSAT( $\Gamma \in \text{CNF}$ )      // Las Vegas algorithm**  
**begin**  
1 **for**  $i=1$  to Max-tries  
2     $S = \text{random}(\Gamma)$ ;      // random assignment  
3    **for**  $j=1$  to Max-moves  
4     **if**  $S$  satisfies  $\Gamma$  then **return**(Yes);  
5     **else**  $S = S$  with some variable flipped to  
     minimize the number of unsatisfied clauses;  
6    **end**  
7 **end**  
8 **return**(No satisfying truth assignment found);  
**end**



## 34.8 How to solve 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



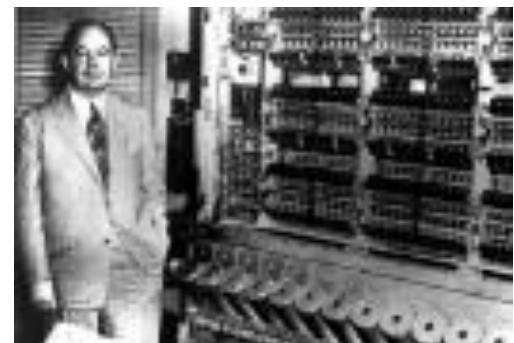
## 在计算机上实现GSAT

# Some Stories



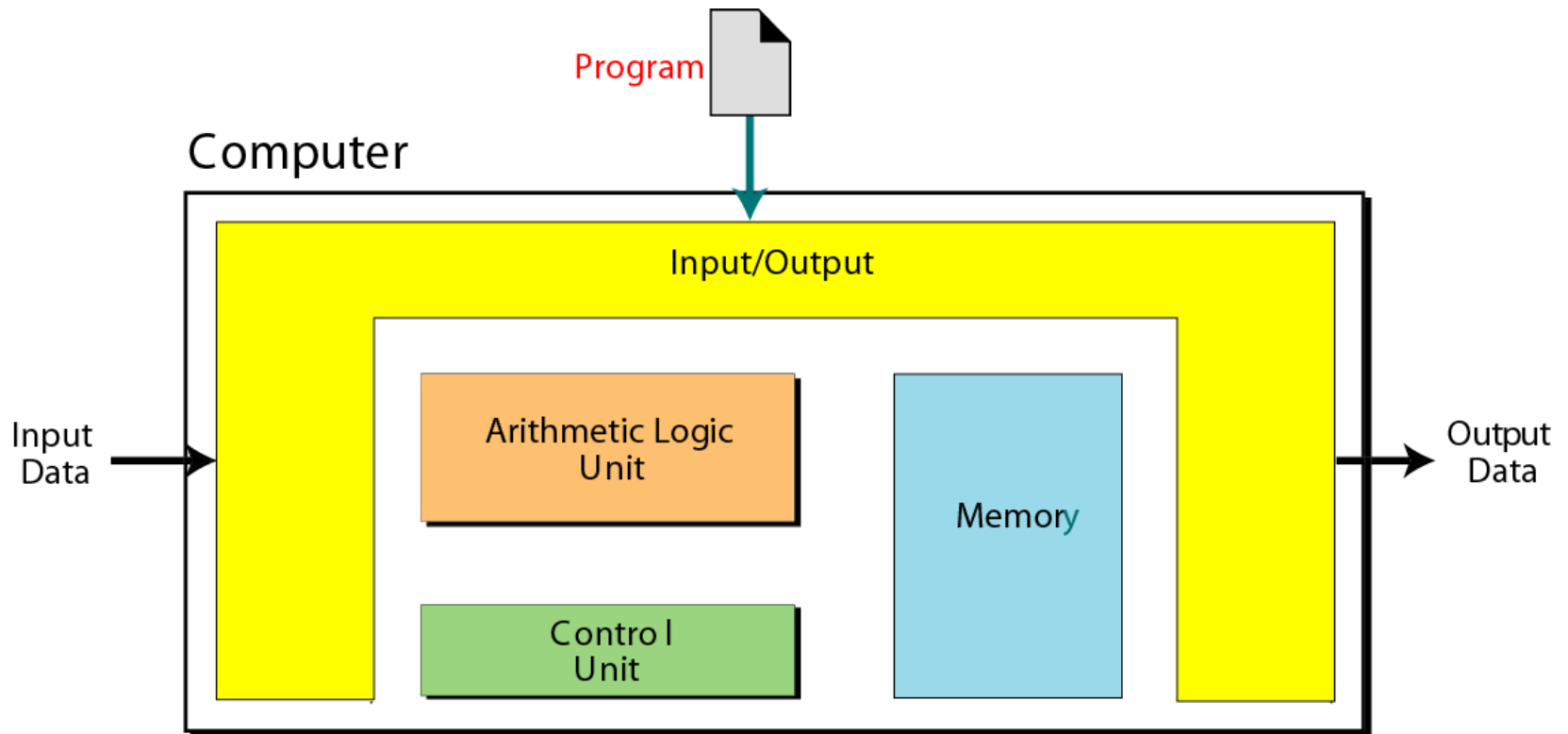
- **John Louis von Neumann**

- ◆ Born 28 December 1903, Budapest, Hungary;
- ◆ Died 8 February 1957, Washington DC;
- ◆ *Brilliant mathematician, synthesizer, and promoter of the stored program concept, whose logical design of the IAS became the prototype of most of its successors - the von Neumann Architecture.*



# Some Stories

- John Louis von Neumann
- von Neumann model



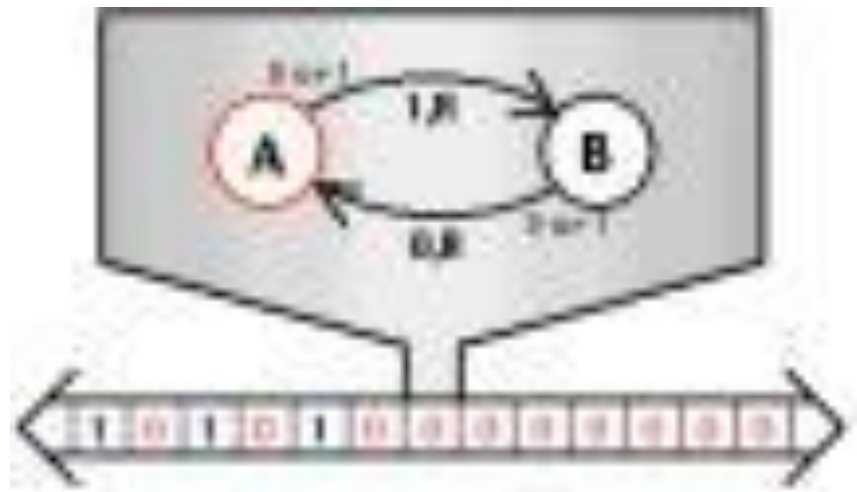
# Some Stories

---

- von Neumann model
- 四个主要部分
  - ◆ Memory
  - ◆ Arithmetic Logic Unit (ALU, 运算及逻辑单元)
  - ◆ Control Unit (CU, 控制单元)
  - ◆ Input/Output (I/O)
- 存储程序概念 (stored program concept)
- 程序执行指令 (sequential execution of instructions)

# Some Stories

- Turing Machine
- Alan Turing 在二次大战时设计，破解了德国的 Enigma 密码





# Some Stories

---

- **Turing Award: 计算机科学的诺贝尔奖**
- **ACM最负盛名的技术奖，并有100万美元的奖金。**  
它被授予为计算机社区做出技术性贡献的个人。  
这些贡献应该对计算机领域具有持久和重大的技术重要性。图灵奖由英特尔公司提供资金支持。
- **1966年开始**

# Some Stories

---

- **Turing Award**

- ◆ **Ronald L. Rivest (dob 1947, Turing 2002)**

<http://www.mit.edu/>

- ◆ **Donald Knuth (dob 1938, Turing 1974)**

<http://www-cs-faculty.stanford.edu/~knuth/>

- ◆ **Stephen A. Cook (dob 1939, Turing 1982)**

<http://www.cs.toronto.edu/~sacook/>

- ◆ **姚期智院士 (dob 1946, Turing 2000)**

# Some Stories

## ◆ Donald Knuth (dob 1938, Turing 1974)

<http://www-cs-faculty.stanford.edu/~knuth/>

