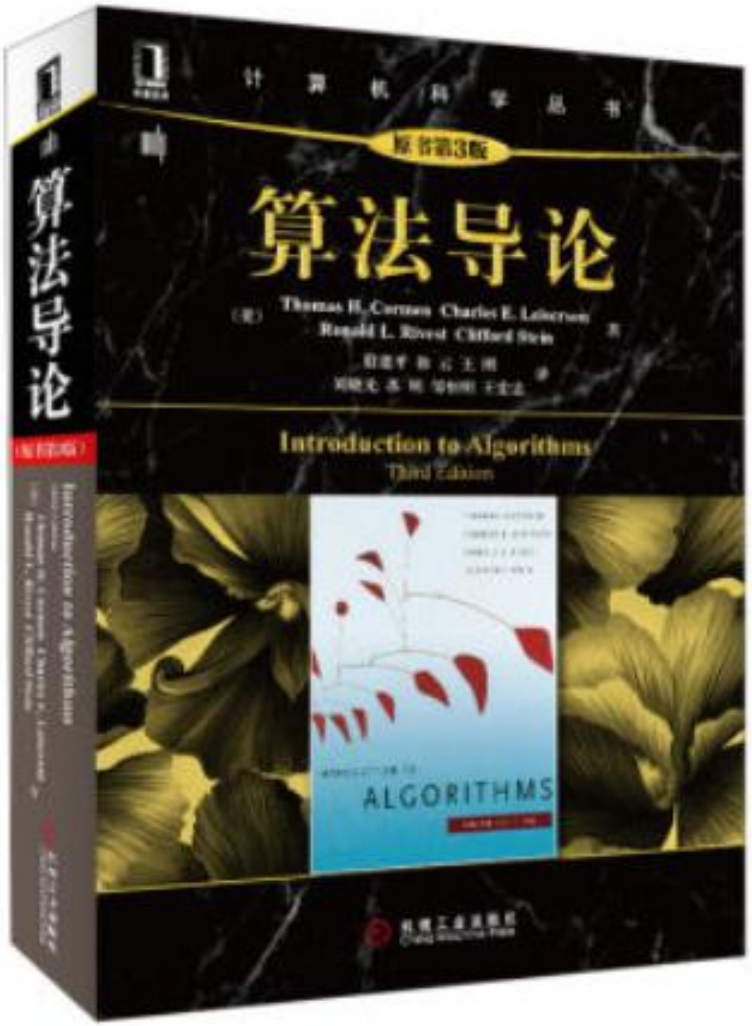
NP问题

P vs. NP：从一则数学家谋杀案说起




我希望他能有所作为
Well,I'd like to see him take a shot at this.

《基本演绎法》

- 美剧《基本演绎法》（也就是美版“福尔摩斯”）第 2 季第 2 集中，两位研究 NP 问题的数学家被谋杀了，凶手是同行，因为被害者即将证明“P=NP 问题”，她为独吞成果而下了毒手。
- 凶手的动机，并不是**千禧年大奖**难题那100万美元的奖金——解决了 P=NP 问题，就能够破译世界上所有的密码系统，这里面的利益比100万美元多多了。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms,” 3rd ed.

千禧年大奖题目

- P/NP问题（P versus NP）
- 霍奇猜想（The Hodge Conjecture）
- **庞加莱猜想**（The Poincaré Conjecture），此猜想已获得证实。
- 黎曼猜想（The Riemann Hypothesis）
- 杨-米尔斯存在性与质量间隙（Yang-Mills Existence and Mass Gap）
- **纳维-斯托克斯存在性与光滑性**（Navier-Stokes existence and smoothness）（可能被解开）
- 贝赫和斯维讷通-戴尔猜想（The Birch and Swinnerton-Dyer Conjecture）

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Clifford Stein, Introduction to "Algorithms," 3rd ed.

P vs. NP

- 剧中只用了一句话来介绍 P=NP 的意义：“能用电脑快速验证一个解的问题，也能够用电脑快速地求出解”。这句过于简单的话可能让大家一头雾水，今天我们就来讲一讲 P vs. NP。

从排序说起

- 复杂度: $O(n^2)$、$O(n\log n)$ 和 $O(n)$。
- 大家会对多项式级别的算法抱有好感，希望对每一个问题都能找到多项式级别的算法。
- 问题是——每个问题都能找到想要的多项式级别的算法吗？

P问题

- 在一个由问题构成的集合中，如果每个问题都存在多项式级复杂度的算法，这个集合就是 P 类问题（Polynomial）。

- 这意味着，即使面对大规模数据，人们也能相对容易地得到一个解，比如将一组数排序。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

NP问题

- “NP”的全称为“Nondeterministic Polynomial”，而不是“Non-Polynomial”。
- NP 类问题指的是，能在多项式时间内检验一个解是否正确的问题
- 13717421是否可以写成两个较小的数的乘积？验证3607*3803，求解？
- 至于求解本身所花的时间是否是多项式我不管，可能有多项式算法，可能没有，也可能是不知道，这类问题称为NP问题。
- NP概念的奥妙在于，它躲开了求解到底需要多少时间这样的问题，而仅仅只是强调验证需要多少时间

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms,” 3rd ed.

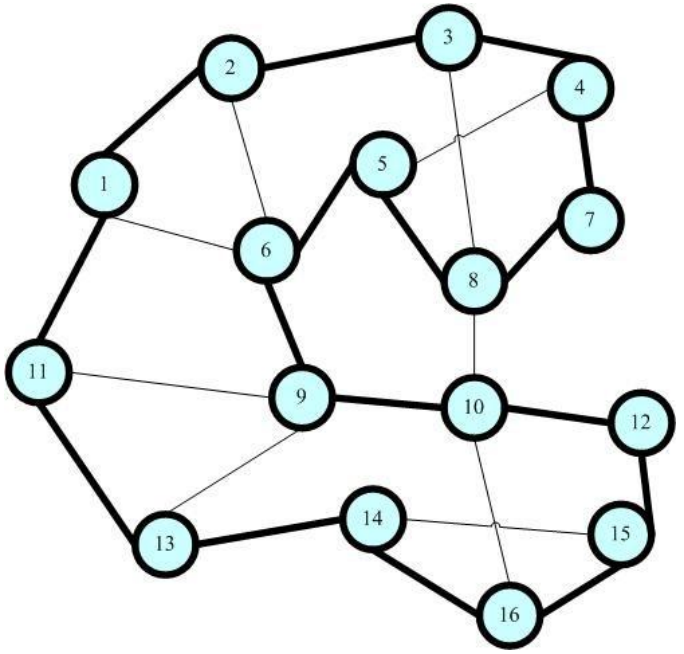
NP问题

- 比如我的机器上存有一个密码文件，于是就能在多项式时间内验证另一个字符串文件是否等于这个密码，所以“破译密码”是一个 NP 类问题。
- NP 类问题也等价为能在多项式时间内猜出一个解的问题。这里的“猜”指的是如果有解，那每次都能在很多种可能的选择中运气极佳地选择正确的一步。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms,” 3rd ed.

NP问题

- 有不是NP问题的问题，如果你不能在多项式的时间里去验证它。
- Hamilton回路——给你一个图，问你能否找到一条经过每个顶点一次且恰好一次（不遗漏也不重复）最后又走回来的路
- NP问题： 存在Hamilton回路
- 非NP问题：不存在Hamilton回路



Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

P=NP？

- NP 问题能在多项式时间内“解决”，只不过需要好运气。显然，P 类问题肯定属于 NP 类问题。

- 所谓“P=NP”，就是问——是不是所有的 NP 问题，都能找到多项式时间的确定性算法？

- 究竟是否有P=NP？通常所谓的“NP问题”，其实就一句话：证明或推翻P=NP。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms,” 3rd ed.

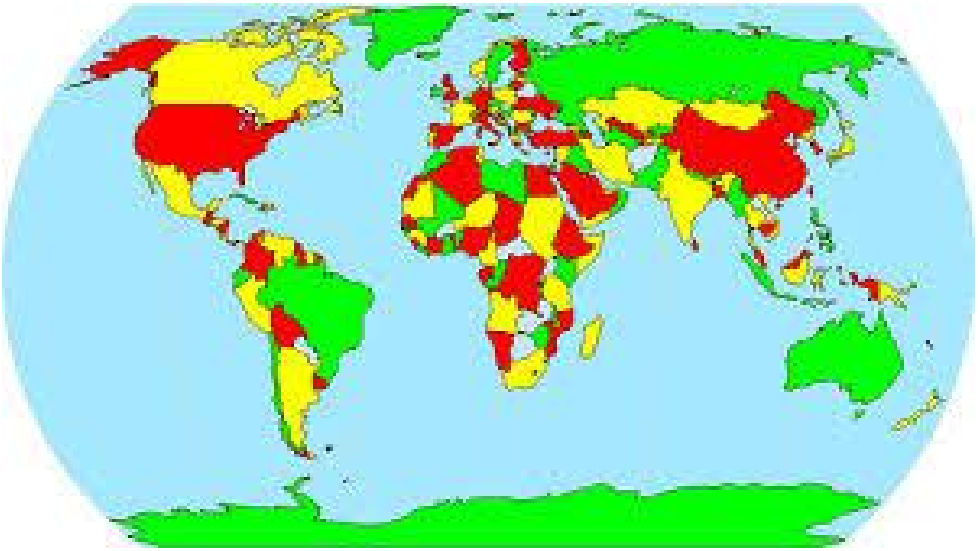
P会不会等于NP？

- 归约(Reducibility)：可以用问题B的解法解决问题A，或者说，问题A可以“变成”问题B。称问题A可以归约为问题B
- 例：求解一个一元一次方程可以规约为求解一个一元二次方程。
- 如果能找到这样一个变化法则，对任意一个程序A的输入，都能按这个法则变换成程序B的输入，使两程序的输出相同，那么我们说，问题A可约化为问题B。
- 存在这样一个NP问题，所有的NP问题都可以约化成它——NPC问题，也就是NP-完全问题。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms,” 3rd ed.

NPC问题

- NPC指的是NP问题中最难的一部分问题，所有的NP问题都能在多项式时间内归约到NPC上。
- **图染色问题：**“任何一张地图只用四种颜色就能使具有共同边界的国家着上不同的颜色。”



Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduc
Algorithms,” 3rd ed.

NPC问题

- **旅行商问题：**即TSP问题（*Traveling Salesman Problem*）。假设有一个旅行商人要拜访n个城市，他必须选择所要走的路径，路径的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市。路径的选择目标是要求得的路径路程为所有路径之中的最小值。

Thomas H. Cormen, Charles E. L

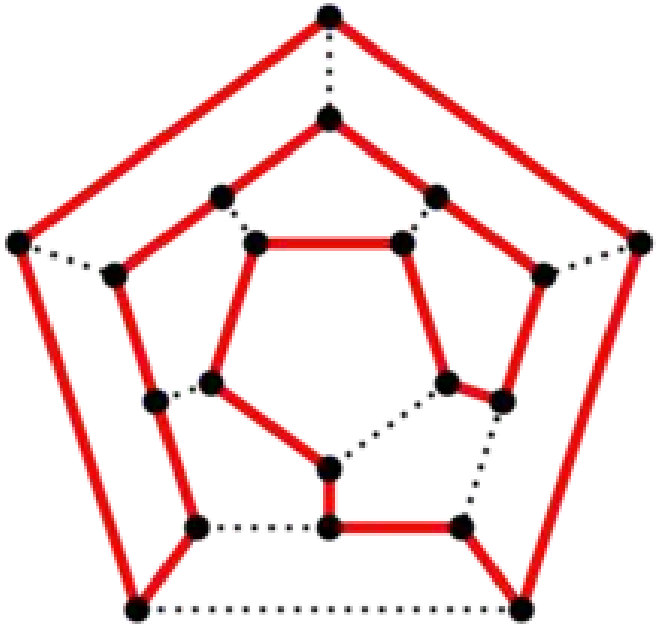
NPC问题

- 华容道问题：通过移动各个棋子，帮助曹操从初始位置移到棋盘最下方中部，从出口逃走。不允许跨越棋子，还要设法用最少的步数把曹操移到出口。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

NPC问题

- 汉密尔顿回路问题：在一个有多个城市的地图网络中，寻找一条从给定的起点到给定的终点沿途恰好经过所有其他城市一次的路径。



Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

P会不会等于NP？

- 目前人们已经发现了成千上万的NPC问题，解决一个，NP=P就得证，可以得千年大奖。
- 图染色、哈密尔顿环都是 NP C问题
- NPC问题目前没有多项式的有效算法，只能用指数级甚至阶乘级复杂度的搜索。
- 正是NPC问题的存在，使人们相信P≠NP

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

如果P=NP，世界会怎样？

- 假设人类的运气好到 P=NP 是真的，并且找到了复杂度不超过 $O(n^3)$ 的算法。如果到了这一步，我们就会有一个算法，能够很快算出某个帐号的密码。所有的加密系统都会失去效果——应该说，所有会把密码变成数字信息的系统都会失去效果，因为这个数字串很容易被“金钥匙”计算出来。
- 除此之外，我们需要担心或期许的事情还有很多：

1. 一大批耳熟能详的游戏，如扫雷、俄罗斯方块、超级玛丽等，人们将为它们编写出高效的AI，使得电脑玩游戏的水平无人能及。
2. 整数规划、旅行商问题等许多运筹学中的难题会被高效地解决，这个方向的研究将提升到前所未有的高度。
3. 蛋白质的折叠问题也是一个 NPC 问题，新的算法无疑是生物与医学界的一个福音。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms,” 3rd ed.

从哲学的角度看NP=P问题

- 德国哲学家以马内利·康德不是基督徒，不过他曾评论过：作为一个“至高的存在”，上帝“是绝对完美、毫无瑕疵的；这个概念成全了***人类所有智慧的集合***，并成为其王冠。”
- 康德的观点：只有一位永恒的、超验的造物主上帝，才能解释人类知识的丰富性与多样性。没有上帝，这个世界最终毫无意义。
- 如果P=NP，就没有世界的丰富性、多样性，世界毫无意义。

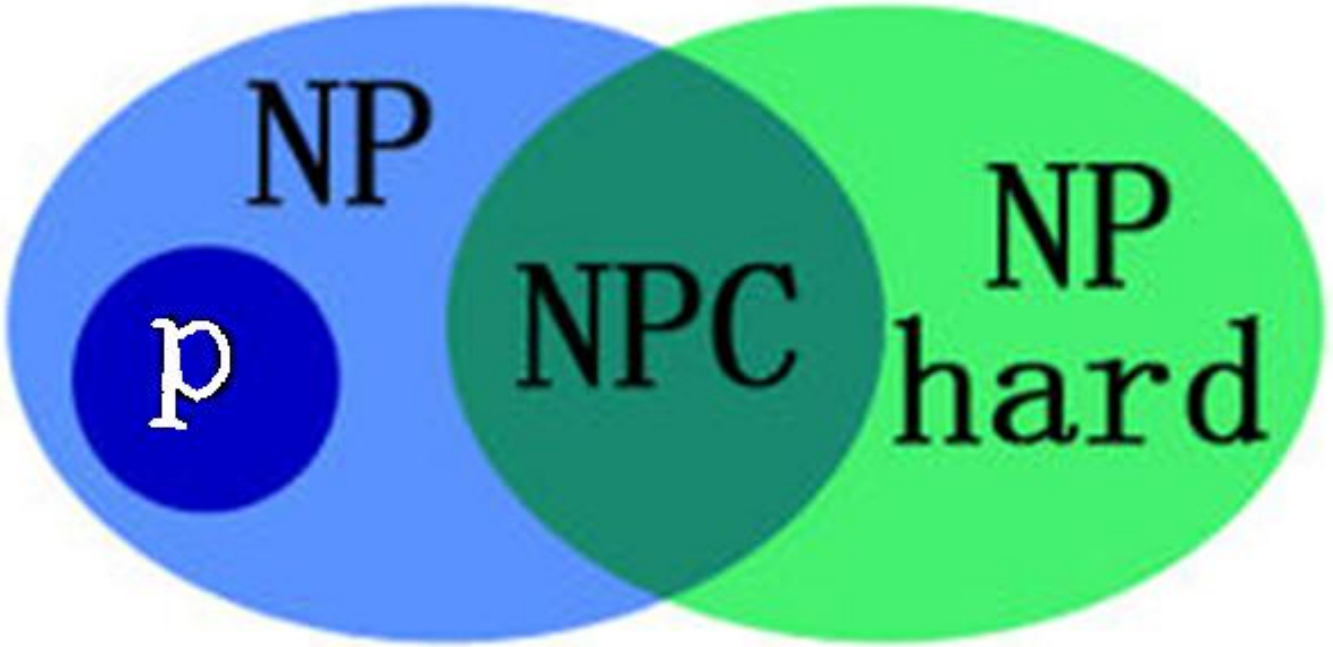
Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms,” 3rd ed.

NP-hard问题

- NP-hard Problem：对于这一类问题，用一句话概括他们的特征就是“at least as hard as the hardest problems in NP Problem”， 就是NP-hard问题至少和NP问题一样难。
- 所有的NP问题都能规约到它，但它不一定是NP问题。
- 存在一些连验证解都不能多项式解决的问题，这些就是NP-hard问题

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms,” 3rd ed.

- 从直觉上说，P<=NP<=NP-Complete<=NP-Hard，问题的难度递增。



Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

停机问题简述

- 通俗地说，图灵当年想要证明希尔伯特的可判定性问题：是否存在一种通用的机械过程，能够判定任何数学命题的真假。
- 图灵就设计了一种假想的机器（图灵机）。他首先证明，图灵机就覆盖了所有的“机械过程”，如果存在一个问题，图灵机判定不了，那么就说明，不存在这种“通用的”过程，这样就证明了原问题。
- 然后，图灵就设计了一个问题，确实是图灵机判定不了的，这个问题就是：对于一个输入，让图灵机判定自己是否能够在有限的时间内停下来。
- 图灵证明，这个问题是图灵机回答不了的，所以原问题得以证否。因为这个问题设计得非常巧妙，所以在历史上留下了名字，叫“停机问题”。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms,” 3rd ed.

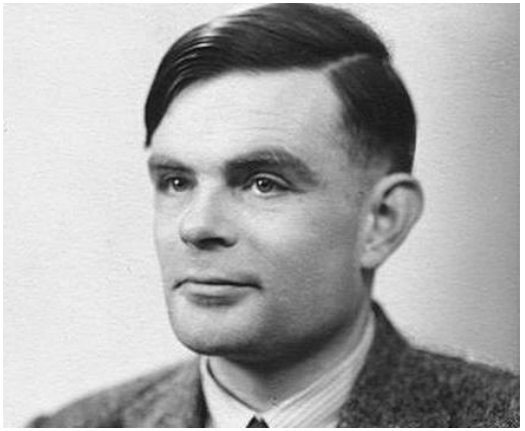
停机问题(1)

- *问题定义：* 给定一个计算机程序和一个输入，判定这个程序是继续工作还是停机

试探解：就用这个给定的输入运行程序呗。如果程序结束，我们就知道程序可以结束，但是如果程序不能在优先时间内而结束，我们也不能下结论说程序不会结束。也可能我们等得不够久？

停机问题(2)

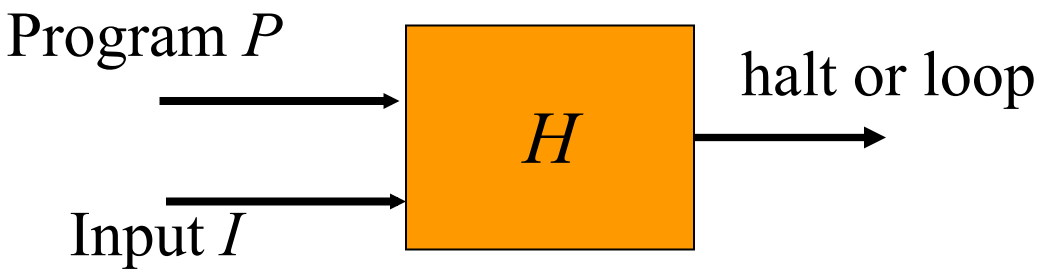
- **Alan Turing 证明了停机问题是不可解的**
- **也就是，不存在一个算法能正确地判定一个任意的程序对于一个给定的输入是否会停机**
- **Turing的证明思想是构造了一个反例：如果这样的算法存在，它就会与自己矛盾，所以这种算法不存在**



Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

停机问题(3)

- 假设H是停机问题的解
- *H* 有两个输入：一个程序 *P* ，还有一个输入 *I* 。
- *H* 产生一个输出：停机，当H认为程序P输入I时会停止；否则输出循环

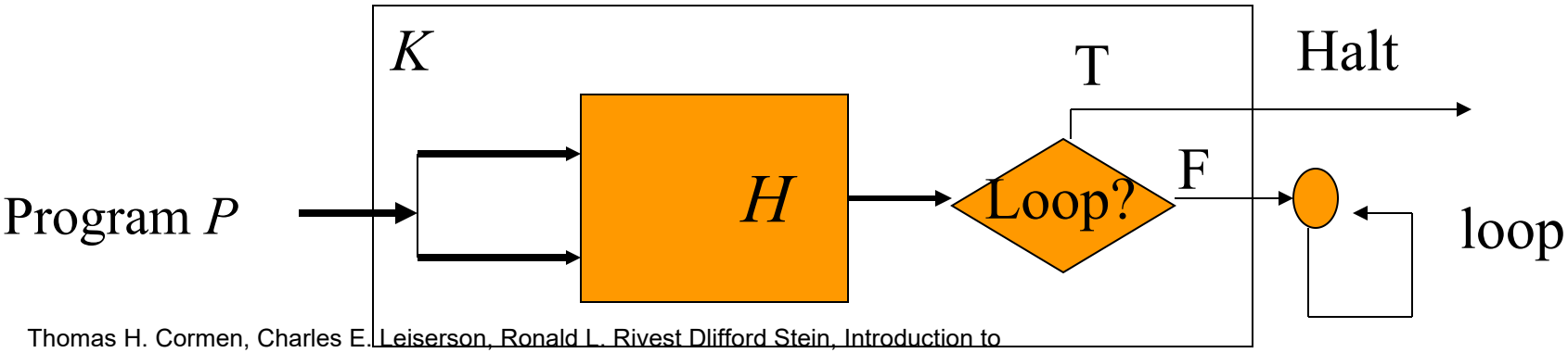


Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

停机问题(4)

- 为了简单，我们把P认为是输入和程序，再设计另一个程序K
  - 如果 *H* 输出循环，则 *K* 停机.
  - 如果*H* 输出停机，则 *K* 永久输出循环
  - 也就是， *K* 总是和 *H*的输出相反.

```
function K( )
    if H( ) == "loop"  return "Halt";
    else  while(true); // loop forever
```



Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

来看两个代码（1）

```
bool God_algo(char * program, char * input)
{
    if(<program> halts on <input>)
        return true;
    else
        return false;
}
```

这里假设if的判断语句是人类天才思考的结晶，它能像上帝一样洞察所有程序的宿命，

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

来看两个代码（2）

```
bool Satan_algo(char * program)
{
   if(God_algo(program, program))
   {
      while(true); // loop forever!
      return false; // can never get here!
   }
   else
      return true;
}
```

- 和这个程序的名字一样，它太邪恶了。
- 当这个算法运用到自身时：

Satan_algo(Satan_algo);

它肯定和所有的程序一样，要么停止，要么永不结束。

Satan_algo(Satan_algo);

■

来看两个代码（3）

- 那我们先假设这个程序能停机，那上图代码块中的if条件判断肯定为真（因为God_algo(Satan_algo, Satan_algo)这个函数返回true），从而程序进入那个包含while(true);语句的分支，那我们就可以得出这个程序不能停机。
- 我们再假设这个程序不能停机，类似的，我们可以得出这个程序能停机。
- 总之，我们有：
- Satan_algo(Satan_algo)能停机 => 它不能停机
- Satan_algo(Satan_algo)不能停机 => 它能停机
- 那么，我们得出了一个悖论。从而我们可以推翻我们最初的假设：**不存在一个程序（或算法），它能够计算任何程序在给定输入上是否会结束（停机）。**

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

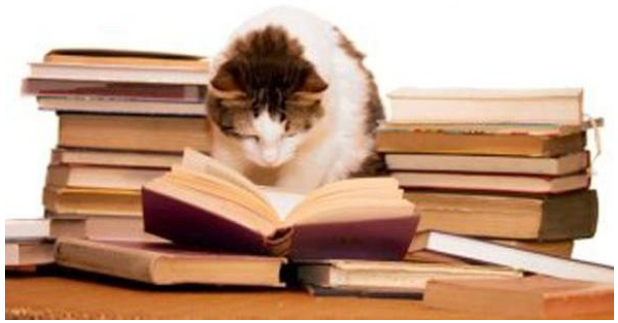
停机问题是NP-hard

- NP-Hard和NP-Complete有什么不同？.如果所有NP问题都可以多项式归约到问题A，那么问题A就是 NP-Hard；如果问题A既是NP-Hard又是NP，那么它就是NP-Complete。NP-Hard问题类包含了NP- Complete类。
- .停机问题是NP-Hard：
  - 停机问题是不可判的，那它当然也不是NP问题。
  - 但对于NP-Complete问题，却可以多项式归约到停机问题——构造程序A，该程序对输入的公式穷举其变量的所有赋值，如果存在赋值使其为真，则停机，否则进入无限循环。这样，判断公式是否可满足便转化为判断以公式为输入的程序A是否停机。所以，停机问题是NP-Hard而不是NP-Complete。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

NPC问题：一个故事 (1)

- 你老板给你了一个NPC的问题，让你给出一个有效的解，这对你的公式很重要
- 最初你和你老板都不知道这是NPC问题
- 你花了大量时间，不眠不休，还是没找到答案

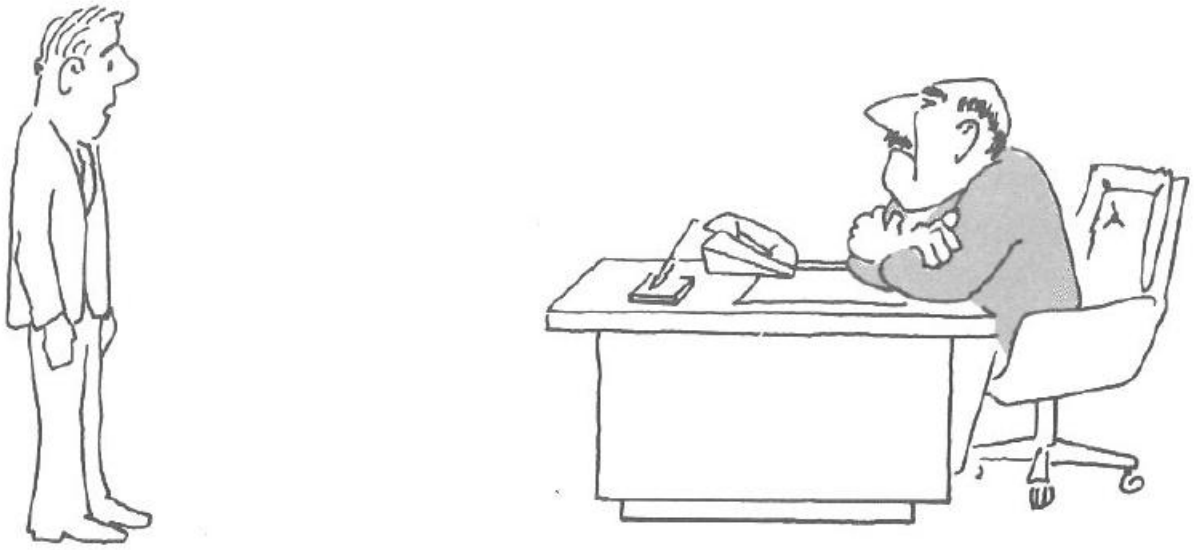


- 你怎么办?

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

NPC问题：一个故事(2)

- 选择1：鼓起勇气告诉老板这事搞不定



"I can't find an efficient algorithm, I guess I'm just too dumb."

- 你完了，老板对你失去了信任...

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

NPC问题：一个故事(3)

- 选择2：你向老板证明这个问题是不可解的

"I can't find an efficient algorithm, because no such algorithm is possible!"

- 杯具了！证明这个问题不可解就如同解这个问题一样难！

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

NPC问题：一个故事(4)

- 选择3：你向老板证明这个问题是一个NPC问题，那么多牛人都解不出NPC问题，我解不出来正常啊！

"I can't find an efficient algorithm, but neither can all these famous people."

- 证明一个问题是NPC问题，相对比较容易。

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.

怎么证明一个问题是NPC问题？

- 如果你有足够的兴趣和时间，请参看课本第34章，加油啊！

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Dlifford Stein, Introduction to Algorithms," 3rd ed.