# 再谈“P vs NP”问题

谨以此文献给“最伟大的计算机科学家”

好几年前曾经写过一篇文章表达对计算机科学里著名的 “P vs NP” 问题的看法。当时正值我人生中第 N 次研究那些东西，由于看透了却不在乎，所以写得特别简略。没想到有人看到后，还以为我没仔细学过复杂度理论，说我信口开河。我一般懒得谈论这种太理论的问题，身边也很少有人关心，所以后来干脆把文章撤了。不是我说的有什么不对，而是我懒得跟人争论。

没想到最近又遇到有人抓住我删掉的文章，乘机拿出来贬损我，尽其羞辱之能力。说王垠你太自以为是了，你成天写那些博客，有什么价值吗？你居然连“P vs NP”都敢批。你知不知道“P vs NP”要是解决了，世界将有天翻地覆的变化，多少的计算难题会被解决，机器学习都没必要了，非对称加密全都被破解…… 跟上课似的头头是道滔滔不绝，几乎把他本科算法课本上的内容给我背了一遍，以为别人不知道一样，却没有显示出任何他自己的思想。

呃，我真是服了某些人背书冒术语的能力，难怪能做国内某大厂的 P10（注：不是我的在职公司）。鉴于很多人对此类问题的一知半解，反倒嘲笑别人不懂，牛逼轰轰打压其他人，我决定事后把这个问题再详细讲一下，免得以后还要为它费口舌。

对于初学者这篇文章有点门槛，需要学习一些东西。“P vs NP” 问题属于计算理论（Theory of Computation）的一部分——复杂度理论。计算理论不止包括复杂度理论（Complexity），还包括可计算性（Computability），也就是“停机问题”一类的内容。

国内大学的计算机教学一般在算法课上对复杂度理论有初级的讲授，但很少人能够真的理解。如果你没有系统的学习过复杂度理论，我建议你研读一下计算理论的专著（而不是普通的算法教材），比如 Michael Sipser 的『Introduction to the Theory of Computation』。

我当年在 Indiana 做研究生计算理论课助教的时候，可算是把这书给看透了…… 被逼的。其中“可计算性理论”在我将来的 PL 研究中起了比较大的启发作用，而复杂度理论的用处一般。我觉得 Sipser 的书写的不够清晰透彻，但很多学校拿它做教材，好像也没有其它特别好的替代品。

计算理论如此晦涩难懂，我认为图灵机是祸首。如果你能理解 lambda calculus，将会大大简化理解计算理论的过程。如果你想用更深刻更容易的方法理解计算理论，可以参考这篇文章的“Lambda 演算与计算理论”一节，里面会提到另一本参考书。从这篇文章你也可以看出来，我丝毫不崇拜图灵。

## “P vs NP” 真的重要吗？

“P vs NP” 这个问题有它的理论价值，它是有趣的问题，里面的有些思路有启发意义，值得花些时间来了解。但计算机科学界长久以来都严重夸大它的重要性，把一个很普通的问题捧上了天，吹得神乎其神。

再加上图灵机模型在计算理论界的广泛使用，使得这门学问显得异常艰深。很多人看到图灵机就晕了，在课程上蒙混过关，考试完了就全忘了，根本无法理解里面的实质内容。正是因为很多人的不明觉厉，使得“P vs NP”登上了它在 CS 界的宝座。

很多人做了一辈子计算机工作，做了很多巧妙的设计构架，写了许许多多的代码，解决了很多性能难题。提到“P vs NP”，虽然一辈子都没用上这个理论，仍然顶礼膜拜。由此可见“不明觉厉”对于人们心理的威力。

很多人认为“P vs NP”是计算机科学最重要的问题。Clay 数学研究所甚至悬赏一百万美元解决这个问题，把它叫做数学界的 7 个千年难题之一，跟黎曼猜想并列其中。

好几次有人声称解决了“P vs NP”，上了新闻，闹得舆论沸沸扬扬，小编们吹得好像世界要天翻地覆了一样，把他们追捧为天才苦行僧，后来却又发现他们的结果是错的……

如果你真的理解了“P vs NP”的内涵，就会发现这一切都是闹剧。这个问题即使得到解决，也不能给世界带来很大变化。解决这个问题对于现实的计算，作用是微乎其微的。不管 P 是否等价于 NP，我们遇到的计算问题的难度不会因此有重大改变。

甚至有些数学家认为“P vs NP” 根本没有资格跟黎曼猜想一起并列于“千禧年问题”。我倒是希望有人真的解决了它，这样我们就可以切实的看到这有什么意义。

“P vs NP” 也许不是愚蠢的问题，但计算机科学界几十年以来夸大它的重要性的做法，是非常愚蠢的，让整个领域蒙羞。

真正重要的数学问题被解决，应该对现实世界具有强大的作用。这种作用可以是“潜在的”，它的应用可以发生在很久以后的将来，但这必须能够被预见到。数学家们把这叫做“applicable result”（注意不叫 applied 或者 practical）。否则这个数学问题就只能被叫做“有理论价值”，“有趣”，而不能叫做“重要”。即使所谓“纯数学”，也应该有可以预见的效果。

很多数学家都明白黎曼猜想（Riemann hypothesis）的重要性。大数学家希尔伯特说过："如果我沉睡了三千年醒过来，我的第一句话会是'黎曼猜想被解决了吗？'" 假设希尔伯特还在世，他会对解决"P vs NP"有同样的渴望吗？我觉得不会。实际上，很多数学家都觉得"P vs NP"的重要性根本没法和黎曼猜想相提并论，因为我们预见不到它会产生任何重要的效果。

## 什么是多项式时间？

很多人提到"P vs NP"就会跟你吹嘘，P 如果等于 NP，世界将有天翻地覆的变化。许许多多我们以前没法办到的事情，都将成为现实。非对称加密技术会被破解，生物化学将得到飞跃，机器学习将不再有必要……

这些人都忽略了一个重要的问题：什么是多项式时间。盲目的把"多项式"等同于"容易"和"高效"，导致了对 "P vs NP" 重要性的严重夸大。

$n^{100}$ 是不是多项式？是的。$n^{1000000}$ 也是多项式。$n^{100^{100}}$ 也是多项式，$n^{100^{100^{100}}}$ 也是多项式…… 实际上，只要 n 的指数是常数，它就是一个多项式，而 n 的指数可以是任意大的常数！n 的指数可以是任意大的常数！n 的指数可以是任意大的常数！重要的事情说三遍。

时间复杂度 $n^{100^{100^{100}}}$ 的算法，能用吗？所以即使 P=NP，你需要的计算时间仍然可以是宇宙毁灭 N 次，其中 N 是任意的常数。

说到这里，又会有人跟我说你不懂，当 n 趋近于无穷的时候，非多项式总会在某个时候超越多项式，所以当 n "足够大"的时候，多项式时间的算法总是会更好。很可惜，"无穷"对于现实的问题是没有意义的。任何被叫做"重要"的问题，都应该在合理的时间内得到结果。

我们关心的要点不应该是"足够大"，而是"具体要多大"。精确的量化，找到实际可以用的区间，这才是合格的科学家该有的思路。计算机科学里，大 O 表示法泛滥成灾，只看最高次幂，忽略系数和常数项，也是常见的误区。我也曾经沉迷于如何把 $O(n^3)$ 的算法降低到 $O(n^{2.9})$，现在回头才发现当年是多么的幼稚。

"多项式时间"这个概念太宽泛太笼统。以如此笼统的概念为基础的理论，不可能对现实的计算问题产生意义。我们关心的不应该仅仅是"是否多项式"，而是"具体是什么样的多项式"。$6n^{20} + 26n^7 + 200$，$1000n^3 + 8n^2 + 9$，……每一个多项式的曲线都是很不一样的，在各个区间它们的差别也是不一样的。多项式的幂，系数，常数项，它们的不同都会产生重大的差异。

这就是为什么"P=NP"没有很大意义，因为 P 本身太笼统，其内部的差异可以是天壤之别。与其试图笼统的证明 P 等价于 NP，还不如为具体的问题想出实质意义上高效的算法，精确到幂，系数，常数项。

更进一步看，这些"复杂度"的数学公式，不管是多项式还是指数，不管你的幂，系数常数项有多精确，终究难以描述现实系统的特性。物理的机器有各种分级的 cache，并行能力，同步开销，传输开销，各种瓶颈…… 最后你发现性能根本无法用数学公式来表达，它根本不是一个数学问题，而是一个物理问题，工程问题。这就像汽车引擎的功率一样，只有放到测试设备（Dyno）上面，通过系统的测量过程才能得到。

有些理论家喜欢小看"工程"，自以为会分析复杂度就高高在上的样子，而其实呢，工程和物理才是真实的。数学只是粗略描述物理和工程的工具。

## P!=NP 有意义吗？

"P vs NP"问题有两种可能性：P=NP（等价），或者 P!=NP（不等价）。以上我说明了 P=NP 的意义不大，那么要是 P!=NP 呢？

很多人会跟你说，要是一个问题是 NP-Hard，然后又有 P!=NP，那么我们就知道这个问题没有多项式时间的算法存在，就避免了为多项式时间算法浪费时间了。这不也有一些价值吗？

我并没有否认 P!=NP 是有那么一点价值：在某些时候它也许避免了浪费时间。但这种价值比较小，而且它具有误导性。

一个常见的 NP-Hard 问题是 SAT。如果 P!=NP，那么大家就应该放弃为它找到高效的算法吗？如果大家都这样想，那么现在的各种高效的 SAT solver 就不存在了。实际上，利用随机算法，我们在大多数时候都能比较快的解决 SAT 问题。

问题在于，"P vs NP"关心的只是"最坏情况"，而最坏情况也许非常罕见。有些问题大部分实际的情况都可以高效的解决，只有少数变态的情况会出现非常高的复杂度。为了这少数情况放弃大多数，这就是"P vs NP"的误导。

如果因为 P!=NP，你认为 NP-Hard 的问题就没有高效的算法，那你也许会误以为你可以利用这些"难题"来做非对称加密。然而 NP-Hard 并不等于没法快速解决，所以要是你因此被误导，也许会设计出有漏洞的加密算法。

即使 P!=NP，我们仍然不能放弃寻找重要的 NP-Hard 问题的高效算法，所以确切的证明 P!=NP 的价值也不是那么重要了。其实你只要知道 P=NP "大概不可能"，就已经能起到"节省时间"的目的了。你没必要证明它。

## 什么是 NP？

这一节我来讲讲“P vs NP”里的“NP”到底是什么。内容比较深，看不懂的人可以跳过。

很多人都没搞明白 NP 是什么就开始夸夸其谈“P vs NP”的价值。 经常出现的错误，是把 NP 等同于“指数时间”。实际上 NP 代表的是“Nondeterministic Polynomial time”，也就是“非确定性图灵机”（nondeterministic Turing machine）能在多项式时间解决的那些问题。

什么是“非确定性图灵机”？如果你把课本上那堆图灵机的定义看明白看透了，然后又理解了程序语言理论，你会发现所谓“非确定性图灵机”可以被很简单的解释。

你可以把我们通常用到的程序看作是“确定性图灵机”（deterministic Turing machine）。它们遇到条件分支，在同一个时刻只能走其中一条路，不能两边同时探索。

那么“非确定性图灵机”呢？你可以把“非确定性图灵机”想象成一个具有“超能力”的计算机，它遇到分支语句的时候，可以同时执行 True 和 False 两个分支。它能够同时遍历任意多的程序分支，这是一台具有超能力的机器！

所以“P vs NP”的含义大概就是这样：请问那些需要非确定性图灵机（超能力计算机）在多项式时间才能解决的问题，能够用确定性图灵机（普通计算机）在多项式时间解决吗？

现在问题来了，具有如此超能力的机器存在吗？答案当然是“No！” 就算是量子计算机做成功了，也不可能具有这样的计算能力。没有人知道如何造出非确定性图灵机，人们没有任何头绪它如何能够存在。

所以 “P vs NP” 这个 问题的定义，是基于一个完全假想的机器——非确定性图灵机。既然是假象的机器，为什么一定要是“非确定图灵机”呢？为什么不可以是其它具有超能力的东西？

仔细想想吧，“非确定性图灵机”对于现实的意义，就跟 Hogwarts 魔法学校和哈利波特对于现实的意义一样。我们为什么不研究“P vs HP”呢，其中 H 代表 Harry Potter。HP 定义为：哈利波特能够在多项式时间解决的问题。

> “P vs NP”问题：请问那些需要非确定性图灵机（超能力计算机）在多项式时间才能解决的问题，能够用确定性图灵机（普通计算机）在多项式时间解决吗？
>
> “P vs HP”问题：请问那些需要哈利波特在多项式时间才能解决的问题，能够用确定性图灵机（普通计算机）在多项式时间解决吗？

我不是开玩笑，仔细回味一下 “P vs NP” 和 “P vs HP” 的相似性吧。也许你会跟我一样意识到 NP 这个概念本身就是虚无的。我不明白“一个不存在的机器能在多项式时间解决的问题”，这样的说法有何意义，基于它的理论又有什么科学价值。

非确定性图灵机存在的意义，也许只是因为它可以被证明等价于其它一些常见的问题，比如 SAT。计算理论书籍一般在证明 SAT 与 非确定性图灵机等价性之后，就完全抛掉了非确定性图灵机，之后的等价性证明都是通过 SAT 来进行。

我觉得 NP 这个概念其实是在故弄玄虚。我们完全可以从 SAT 本身出发去发展这个理论，而不需要设想一个具有超能力的机器。我们可以有一个问题叫做“P vs SAT”，而不出现 NP 这个概念。

（有点扯远了）

## 其它质疑 P vs NP 价值的人

有人认为我质疑 P vs NP 的价值是一知半解信口开河，然而我并不是第一个质疑它的人。很多人对 P vs NP 都有类似的疑惑，但因为这个问题的地位如此之高，没人敢站出来。只要你开口，一群人就会居高临下指责你基础课程没学好，说你眼界太窄…… 再加上那一堆纷繁复杂基于图灵机的证明，让你有苦说不出。

由于这个原因，我从来没敢公开表达我的观点，直到我发现 Doron Zeilberger 的这篇文章。Zeilberger 是个数学家，Rutgers 大学的数学系教授。在那之前他开了个玩笑，戏称自己证明了 P=NP，还写了篇像模像样的论文。在文章里他告诫大家：不要爱上你的模型（Don’t Fall In Love With Your Model）。他这句话说到了我心里。

你还能在网络上找到其它人对“P vs NP”的质疑，比如这篇来自于一位专门研究计算理论的学者：

Is P=NP an Ill Posed Problem?

我觉得他讲的也很在理。正是在这些人的鼓舞之下，我随手写出了之前对“P vs NP” 的质疑。只言片语里面，融入了我多年的深入学习，研究和思考。

## 总结

看这篇文章很累吧？我写着也累。对于我来说这一切都已经那么明了，真的不想费口舌。但是既然之前已经说出来了，为了避免误解，我仍然决定把这些东西写下来摆在这里。如果你暂时看不懂可以先放在一边，等到了需要深入研

究计算理论，想得头痛的时候再来看。你也许会感谢我。

我希望严谨的计算机科学工作者能够理解我在说什么，反思一下对“P vs NP”的理解。计算机专业的学生应该理解“P vs NP”理论，但不必沉迷其中。这并不是一个值得付出毕生精力去解决的问题。计算机科学里面还有其它许多有趣而重要的问题需要你们去探索。如果你觉得计算机科学都不过瘾，你可以去证明黎曼猜想啊 :)

当然所有这些都是我的个人观点，我没有强求任何人接受它们。强迫别人接受自己的观点是不可以的，但想阻止别人表达对此类问题的质疑，也是不可以的，因为我们生活在自由的世界。

没人想抢走你们的玩具，但不要忘了，它只是玩具。