
DARK TIME
暗时间
刘未鹏 著
JUST PUB
电子工业出版社

# 暗时间

## 刘未鹏

## 简介

2003 年，刘未鹏在杂志上发表了自己的第一篇文章，并开始写博客。最初的博客较短，也较琐碎，并夹杂着一些翻译的文章。后来渐渐开始有了一些自己的心得和看法。总体上在这 8 年里，作者平均每个月写 1 篇博客或更少，但从未停止。刘未鹏说——写博客这件事情给我最大的体会就是，一件事情如果你能够坚持做 8 年，那么不管效率和频率多低，最终总能取得一些很可观的收益。而另一个体会就是，一件事情只要你坚持得足够久，“坚持” 就会慢慢变成“习惯”。原本需要费力去驱动的事情便成了家常便饭，云淡风轻。这本书便是从刘未鹏 8 年的博客文章中精选出来的，主要关于心智模式、学习方法和时间利用，《暗时间》的书名便来自于此。

# 内容简介

2003 年，刘未鹏在杂志上发表了自己的第一篇文章，并开始写博客。最初的博客较短，也较琐碎，并夹杂着一些翻译的文章。后来渐渐开始有了一些自己的心得和看法。总体上在这 8 年里，作者平均每个月写 1 篇博客或更少，但从未停止。

刘未鹏说——

写博客这件事情给我最大的体会就是，一件事情如果你能够坚持做 8 年，那么不管效率和频率多低，最终总能取得一些很可观的收益。而另一个体会就是，一件事情只要你坚持得足够久，“坚持”就会慢慢变成“习惯”。原本需要费力去驱动的事情便成了家常便饭，云淡风轻。

这本书便是从刘未鹏 8 年的博客文章中精选出来的，主要关于心智模式、学习方法和时间利用，《暗时间》的书名便来自于此。

# 作者简介

刘未鹏：

南京大学计算机系硕士毕业

现就职于微软亚洲研究院创新工程中心

兴趣爱好：计算机科学，人工智能，认知科学

博客名 Mind Hacks 的含义：

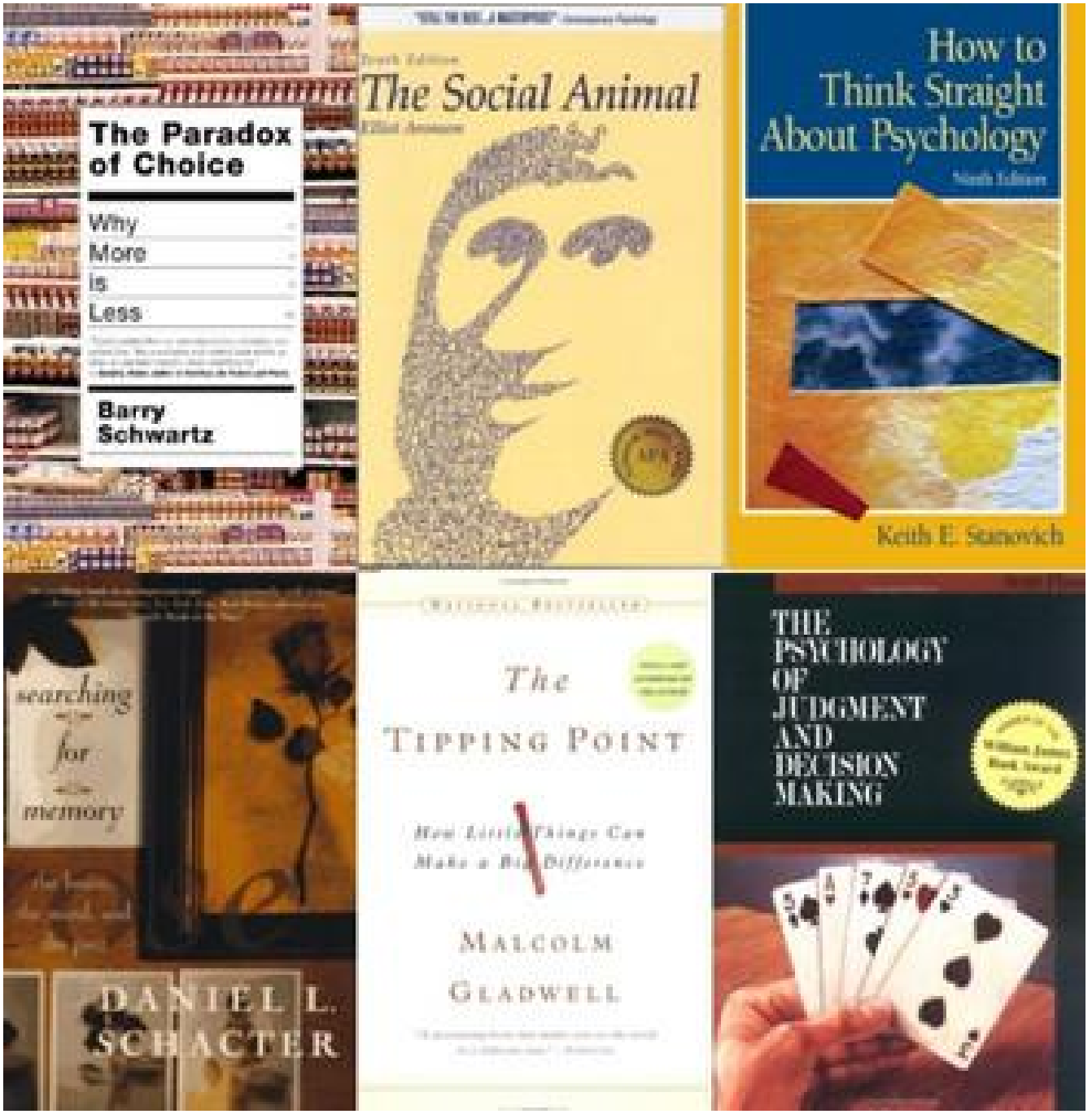
-Mind Hacks 是一本书

-Mind Hacks 是一系列思维工具

-Mind Hacks 有一个漫长的前生——一个有着近 6 年历史的技术博客

-在 CSDN 上有超过 120 万的访问量

# 序言 为什么人人都该学点心理




卷帙浩繁的心理学著作，帮助我们理解世界、了解自己、品味人生

提到心理学，很多人脑海中的第一印象就是“心理问题”、“心理咨询”、“弗洛伊德”，“抑郁症”、“读心术”

这些字眼，总觉得关心心理学的都是一些心理阴暗或有问题的家伙 ,这是对现代心理学典型的误解( 参见 How to Think Straight About Psychology , 中译名《与“众”不同的心理学》)，实际上心理学早就从弗洛伊德那一套走出来，结合现代科学技术手段（如 FMRI）和研究方法（如受控随机实验），跨学科研究人脑思维的特点以及和现实生活的关系。其中最硕果累累的交叉领域就是经济学和心理学的联姻，催生出了一大批优秀的研究者和著作，如 Richard Thaler 的 Nudge，研究如何针对人们思维的固有弊端来创造性地制定一些经济策略，从而为大众的健康、经济和幸福谋福利 :如著名的 Save More Tomorrow（中译名《明天存储更多》）就成功的促使了大量没有存储习惯的美国人开始存储更多；如 Barry Schwartz 的 Paradox of Choice–Why More is Less 用大量生动有趣的研究案例说明了很多时候选择少反而让人感觉舒服得多。再如研究创意点子之所以流行之本质原因的 Made to Stick，其中剖析出的原因，学过一些社会心理学（The Social Animal，中译名《社会性动物》）

和大脑记忆原理（Searching For Memory，中译名《找寻逝去的自我》）的人应该一眼就看出其中的端倪，书中更是开篇就赤裸裸地引用心理学实验。就连这些年在畅销书市场大红大紫的 Malcolm Gladwell 的几本流行全球、傲居榜首的著作（The Tipping Point，Blink，Outliers）其实也是在用大量活生生的案例佐证一些早就不新鲜的心理学原理。

心理学与日常生活息息相关的另一方面就是日常判断与决策（生活中需要进行判断与决策的地方远远多于我们的想象）（Psychology of Judgment and Decision Making，中译名《决策与判断》）。波普尔曾经说过：人生不过是解决问题（All Life is problem-solving）。而判断与决策又是其中最常见的一类问题解决。尽管生活中似乎面临重大决策的时候并不多，但另一方面我们时时刻刻都在进行最重大的决策：决定自己的时间和金钱到底投入到什么地方去。比如：你能想象有人宁可天天花时间剪报纸上的优惠券，却对于房价的 1%的优惠无动于衷吗？（《别做正常的傻瓜》、Predictably Irration

al，中译名《怪诞行为学》）再比如：你知道为什么当手头股票的股价不可抑止地滑向深渊时我们却一边揪着头发，一边愣是不肯撤出吗？（是的，我们适应远古时代的心理机制根本不适应金融市场。）为什么我们会在小摊和超市前为几块钱的交易踌躇半天，却在生活中的重大决策面前浑浑噩噩人云亦云呢？糟糕的判断与决策令我们的生活变得糟糕，然而这还不是最关键的，最关键的是我们很难学会质疑自己的判断，而总是能“找到”其他为自己辩护的理由（Mistakes were Made,but not by Me），中译名《错不在我》）。

现在是一个信息泛滥的时代，这导致我们经常面临另一个问题：如何在海洋中有效筛选有价值的信息，以及避免错误的信息左右我们的大脑（critical thinking）关于以上提到的几点我在豆瓣上有一个豆列（“学会思考”），希望有一天我能够积累出足够多的认识对这个主题展开一些详细介绍。

人类的大脑和思维是目前已知最为复杂的系统，对

这个系统的研究是一件极其迷人的事情，即便对于像我们这样的芸芸众生来说，即便不去做研究，学习一些这方面的科普知识，对于学会正确地思考也有极大的益处。

大脑是我们最重要的工具，要正确利用这个工具，唯一的途径就是去了解它，尤其是了解它的弱点。与很多人的直觉相反，我们的思维有着各种各样的缺陷和陷阱（cognitive bias），我们解决日常问题的思维方式也并不总是最优的（bounded rationality），我们感觉正确的事情有很多是错的，我们习以为常的天经地义的行为也未必就是合乎效益最大化原则的。这里摘抄一段我在豆列上的导言：

我们的思维有很多很多的弱点，我一向认为，正确的思维方式，是一切高效学习的基础。比如参见如下 2 个例子，错误的思维方式得到的结论有大得多的可能性是谬误。

•人总喜欢沿袭以往习得的经验，并通过类比来进行外推。我第一次在一个地铁终点站坐地铁的时候，看着

从远方开来的地铁，我心生疑惑——“这车每节车厢都这么长，待会怎么调头呢（我心里说没看到铁轨终点有一个大大的供调头的 U 形弯啊）？”，当车开始开的时候我终于意识到原来车是可以往两头方向开的。

•人喜欢从关联当中寻找因果，有一次我和老婆去银行取款，到了 ATM 室的自动门口，我开玩笑地拿着手头的饭卡去刷了一下，然后门居然开了。我顿时来了劲，立即得出一个结论：这个刷卡装置不安全，至少不是能够专门识别银联的卡的。我甚至飞快地泛化出了一个更具一般性的理论来解释这个现象：即可能所有带有磁性的卡都可以用来开门。老婆看我得意洋洋，就泼过来一盘冷水：不一定是你的卡刷开的啊，你不刷卡试试看。我不信，说怎么可能呢，心想我刷卡，门就开了，还有比这更明显的因果关系嘛。但出乎我意料的是，我走出门，这次没刷卡，门也开了——原来是感应门——原先这个 ATM 室的确是刷卡门，但后来改成了感应门，刷卡的那个装置只不过没拆掉残留在那里而已。

总的来说

•人类的思维充满着各种各样的捷径，每一条捷径都是一把双刃剑。一方面，它降低了大脑的认知复杂性（笼统的看一个问题要比细致的分析简单得多），有助于迅速做出绝大部分时候都正确的判断；但另一方面，它也常常导致人们把大部分情况下成立的法则当成了放之四海而皆准的。可以说，有多少捷径，就有多少条谬误。

•人类的情绪也在很大程度上影响着人的思考。比如，如果你憎恶一个人，你往往就会反对他的所有立场，反之亦成立。

•人类大脑经过长时间的进化，先天就具备一些特定的“思维定势”，以使得人类能够在面对进化过程中经常出现的适应性问题时能够不假思索的做出迅速的反应。然而，在现代社会，很多这类思维定势已经不适应了。（Mean Genes，中译名《本能》;Influence，中译名《影响力》;Sway，中译名《摇摆》，Evolutionary Psychology，中译名《进化心理学》）

•人类不可避免的受着各种各样的偏见的影响，这些偏见有些是有一定适应价值的“思维定势”（如事后聪明式偏见），而有些则是大脑的认知机制的“缺陷”。

以上，构成了人类思维中的种种谬误。而学会思考，就是学会认识到这些谬误。

对于人类思维的种种谬误之处，在西方有一门历史悠久的学科，叫做批判性思维（critical thinking）（Asking the Right Questions，中译名《学会提问》），早在古希腊时代，亚里士多德就已经对人类语言中的各种各样的谬误有了一定的认识（譬如，“我们无法讨论不存在的东西，所以所有的事物都是真实的”），并对辩论之中存在的各种各样的谬误进行了归类。

这个豆列(http://book.douban.com/doulist/127649/)中的书，有一些是介绍人类思维工作的机制的，认识这些机制是正确思考的大前提；有许多是关于人类推理（Reasoning）过程中存在着形形色色的谬误（Nonsense，Crimes Against Logic），唯有认识到这些谬误，

才能避免它们，唯有避免了思维的谬误，才能进行正确的思考。

相关阅读:

一个最完整的认知偏见（cognitive bias）列表：http://en.wikipedia.org/wiki/List_of_cognitive_biases

一个完整的谬误（fallacies）列表：http://en.wikipedia.org/wiki/Fallacies

维基百科上关于批判性思维（critical thinking）的条目：http://en.wikipedia.org/wiki/Critical_thinking

# 第一篇 暗时间

## 1、暗时间

如果你有一台计算机，你装了一个系统之后就整天把它搁置在那里，你觉得这台计算机被实际使用了吗？没有。因为 CPU 整天运行的就是空闲进程。运行空闲进程也是一天，运行大数据量计算的程序也是一天，对于 CPU 来说同样的一天，价值却是完全不一样的。

大脑也是如此。

善于利用思维时间的人，可以无形中比别人多出很多时间，从而实际意义上能比别人多活很多年。我们经常听说“心理年龄”这个词，思考得多的人，往往心理年龄更大。有人用 10 年才能领悟一个道理，因为他们是被动领悟——只有在现实撞到他脸上的时候才感到疼，疼完了之后还是不记得时时提醒自己，结果很快时过境迁抛之脑后，等到第二次遇到同一个坑的时候早忘了曾经跌过跟头了，像这样的效率，除非天天摔坑里，否则遗忘的效率总是大过吃亏长的记性。善于利用思维时间的人则能够在重要的事情上时时主动提醒自己，将临时的记忆变成硬编码的行为习惯。

每个人的手表都走得一样快，但每个人的生命却不是。衡量一个人生活了多少年，应该用思维时间来计算。举一个极端的例子，如果一个人从生下来开始就呆在一个为他特殊建造的无菌保护室里，没有社会交往，没有知识获取，度过了 18 年，你会不会认为他成年了？

认为时间对每个人是均等的是一个错觉，认为别人有一天，我也有一天，其实根本不是这样。如果你正在学习一门专业，你使用自己所投入的天数来衡量，很容易会产生一种错觉，认为投入了不少时间，然而其实，“投入时间”这个说法本身就是荒唐的，实际投入的是时间和效率的乘积。你可以“投入”很多时间在一件事情上面，却发现毫无进展，因为你没有整天把你要做的事情，要学习的东西常驻在你的大脑中，时刻给予它最高的优先级。你走路的时候吃饭的时候，做梦的时候心心念念想的就是这件事情，你的 CPU 总是分配给它，这个时候你的思维时间就被利用到了极致，你投入的时间就真正等于了实际流逝的时间，因为你的 CPU 是满载的。

如果你有做总结的习惯，你在度过一段时间之后总结自己在某某领域投入了多少时间，建议千万不要粗略地去计算有多少天下班后拿起书来翻看过，因为这样你也许会发现书倒是常翻，但领悟却不见得多深，表面上花的时间不少，收益却不见得那么大。因为看书并记住书中的东西只是记忆，并没有涉及推理，只有靠推理才

能深入理解一个事物，看到别人看不到的地方，这部分推理的过程就是你的思维时间，也是人一生中占据一个显著比例的“暗时间”，你走路、买菜、洗脸洗手、坐公车、逛街、出游、吃饭、睡觉，所有这些时间都可以成为“暗时间”，你可以充分利用这些时间进行思考，反刍和消化平时看和读的东西，让你的认识能够脱离照本宣科的层面。这段时间看起来微不足道，但日积月累将会产生庞大的效应。

能够充分利用暗时间的人将无形中多出一大块生命，你也许会发现这样的人似乎玩得不比你少，看得不比你多，但不知怎么的就是比你走得更远。比如我就经常发现一些国外的牛人们为什么不仅学习牛逼，连“业余”玩儿的东东也都搞得特牛逼，一点都不业余（上次在《How We Decide》上看到斯坦福的一个牛人，理论物理学博士，同时是世界扑克大赛的前六名保持者，迄今累计奖金拿了六百多万刀），你会奇怪，这些家伙到底哪来的时间，居然可以在不止一个领域做到卓越？

程序员们都知道，任务切换需要耗费许多额外的花销，通俗地来讲，首先需要保存当前上下文以便下次能够顺利切换回来，然后要加载目标任务的上下文。如果一个系统不停地在多个任务之间来回倒腾，就会耗费大量的时间在上下文切换上，无形中浪费很多的时间。

相比之下，如果只做一件任务，就不会有此损失。这就是为什么专注的人比不专注的人时间利用效率高得多的原因。任务切换的暗时间看似非常不明显，甚至很多人认为“多任务”是件很好的事情（有时候的确是），但日积月累起来就会发现，消耗在切换上的时间越来越多。

另外，大脑开始一件任务的时候必须要有一定时间来“热身”，这个时间因人而异，并且可以通过练习来改变。举个例子，你看了一会书之后，忽然感到一阵无聊，忍不住打开浏览器，十分钟后你想起来还要继续看书，但要回复到当时理想的状态，却需要一段时间来努力去集中精力，把记忆中相关的知识全都激活起来，从而才

能进入“状态”，因为你上了十分钟网之后这些记忆已经被抑制了。如果这个“热身”状态需要一刻钟，那么看似十分钟的上网闲逛其实就花费了二十五分钟。

如果阅读的例子还不够生动，对于程序员来说其实有更好的例子：你写程序写得正 high，忽然被叫去开了一通会，写到一半的代码搁在那儿。等你开完会回来你需要多久能够重新进入状态？又或者，你正在调试程序，你已经花了二十分钟的时间把与这个 bug 可能相关的代码前前后后都理解了一遍，心中构建了一个大致的地图，就在这时，呃，你又被叫去开了个会(:D)，开完会回来，可想而知，得花上一些时间来回想一下刚刚弄清的东西了。

迅速进入状态的能力是可以锻炼的，根据我个人的经验，至少可以缩短到 3-5 分钟。但要想完全进入状态，却是很难在这么短的时间实现的。所谓完全进入状态，举个例子：你看了 3 个小时的书，或者调试了半个小时的程序之后，往往满脑子都是相关的东西，所有这些知

识都处在活跃状态，换言之你大脑中所有相关的记忆神经网络都被激活了，要达到这样一种忘记时间流逝的“沉浸”状态（心理学上叫做“流体验”），不是三两分钟的事情。而一旦这种状态被破坏，无形间效率就会大打折扣。这也是为什么我总是倾向于创造大块的时间来阅读重要的东西，因为这样有利于“沉浸”进去，使得新知识可以和大脑中与其相关的各种既有的知识充分融合，关联起来，后者对于深刻的记忆非常有帮助。

要充分利用暗时间，不仅要能够迅速进入状态，另一个很重要的习惯就是能够保持状态多久（思维体力）。《The Psychology of Invention in the Mathematical Field》上有一段关于庞加莱的思考习惯的介绍，很有代表性。庞加莱经常在去海边休假或者在路上走的时候在脑海中思索数学问题，很多时候解答就在这些时候忽然闪现。虽然我和庞加莱是没法比的，但是常常也在路上想出答案，这真是一种愉悦的体验。

能够迅速进入专注状态，以及能够长期保持专注状

态，是高效学习的两个最重要习惯。

很多人都有这样的体验( 包括我自己 ),工作了之后，要处理的事情一下多出了很多，不像在校园，环境简单，生活单纯，能够心无旁骛地做一件事情而不被打扰。工作之后的状况就是，首先需要处理的事情变多，导致时不时需要在多个任务之间切换；另一方面，即便能够把任务的优先级分配得比较合理，也难免在做一件事情的时候心中忽然想起另一件事还没做的焦虑来，因为没做完的事情会在大脑中留下一个“隐藏的进程”，时不时地发个消息提醒你一下，中断你正在做的事情。

因此这里就涉及到最后一个高效的习惯：抗干扰。只有具备超强的抗干扰能力，才能有效地利用起前面提到的种种暗时间。抗干扰能力也是可以练习出来的，上本科那会经常坐车，所以我就常常拿着本大部头在车上看，坐着看或者站着看都可，事实证明在有干扰的环境中看书是非常锻炼专注能力的一个办法:D 另外，经常利用各种碎片时间阅读和思考，对迅速集中注意力和保持

注意力都非常有帮助。记得很久以前 TopLanguage 上大伙曾经有次饶有兴趣地讨论“马桶时间”的利用，包括在卫生间放个小书柜。（估计很多同学心有戚戚焉吧:D）

## 2、设计你自己的进度条

设计你自己的进度条

进度条的设计是一个很多人都知道的故事：同样的耗时，如果不给任何进度提示，只是在完成之后才弹出一个完成消息，中间没有任何动态变化，那么整个过程就会让人等得非常焦急，导致一些人干脆把程序关了了事。如果有进度不断更新，那么对整个过程耗时的心理感受就会远低于实际值，用户也不会郁闷到把程序关了。（你有多少次在银行处理手续的时候，看着工作人员把一堆材料不停地倒腾来去，心里多希望他们可以在柜台小窗口上投影一个进度条？）

这里的原因在于，没有进度提示的话，我们无法判断这个等待什么时候才是个尽头。如果有不断增长的进

度条，那么我们对于什么时候会达到 100%就会有一个粗略的估计，这个估计是一剂定心丸，让我们知道这事情总会并且会在不久的将来完成。

做事情也是同样的道理，善于规划的人，会将目标分割成一个个的里程碑，再将里程碑分割成 TODO 列表。前阵子流行的 GTD 方法学，核心的理念就在于，如果你把任务分割了，你就有了进度条，你就知道，事情在不断的进展，你总会完成任务或到达你的目标，你会有一个时间估计。反之如果没有这个分割，整个的任务或目标对你来说就只有两种状态——“完成”和“未完成”，如果不幸是一个比较漫长的目标，那么你会发现你的进度条总是“未完成”，一次又一次的等待未果会耗尽你的耐心，让你下意识的产生“这事什么时候才能完呢？”的疑惑，没有分而治之，你就不知道未来还需要付出多少努力才能达到目的，这就会让你心生怯意，不敢进一步投入时间，免得血本无归。在这样的心理下，不少人就会选择保守策略——退出，以免到头来花了时间还一事无成。

而所谓的规划其实就是针对这种心理弱点的做事方法。如果你对整个目标的几个重大步骤有清晰的界定，能够对每个步骤的耗时作出靠谱的上界估计，你就不会被不确定的未来，不确定的时间投入感到恐惧，就不会被这种不确定感压迫到过早退出。

不要过早退出循环

我们在尝试新的事物的时候，总是会遇到各种各样的困难，不同的人会在碰壁不同的次数之后退出。用程序员喜欢的话来说就是，我们都在 for 循环，区别在于你是什么情况下 break;的。有的人退出阈值高，这是能坚持的一类人，有的人退出阈值低，这类人很可能遇到一些障碍就退出了。

过早退出的原因往往在于对于未来的不确定性，对于投资时间最终无法收到回报的恐惧，感受到的困难越大，这种恐惧越大，因为越大的困难往往暗示着这个任务需要投资的时间越大。所以其实我们都是直觉经济学家，当我们说“畏难”的时候，其实我们畏惧的不是困

难本身，而是困难所暗示的时间经济学意义。

然而，我们的情绪大脑毕竟比较原始，仅根据碰壁的次数或硬度来判断事情的难易并不一定靠谱，如果你遇到困难，不妨用一用互联网，用一用群体的智慧，看看别人当时是怎么想怎么办的，绝大多数情况下你并不孤单，你遇到的问题早就有人遇到过，你踩过的坑里面尽是前人的脚印，不要仅仅因为一时摸不着头绪，找不着出路就退出，这不是 informed decision，问一问自己作出退出的决策是否基于足够的信息，我是否进行了足够的调查，至少，是否去简单用了用搜索引擎。

模仿高德纳先生的名言：过早退出是一切失败的根源。

兴趣遍地都是，专注和持之以恒才是真正稀缺的。

很多人看了书中的故事之后得出这样的结论：兴趣最重要。然而，我觉得区别他们和其他人的，并不是他们拥有超过常人的兴趣，而是他们拥有超过常人的毅力。

其实人天生就对新事物怀有好奇心，难以找出谁没有对任何事物或领域产生过兴趣，然而不同的是，有些人的兴趣只能持续几天，当遇到第一个困难，第一道坎的时候，他们就熄灭了，然而另一些人的兴趣火花会变成火苗，火苗会变成火种，一直稳定的燃烧很多年。区别他们的并不是兴趣的有无，而是他们的性格里面有没有维持兴趣的火种一直燃烧下去的燃料。

一个人有专注和持之以恒的性格，即便在一个没有多大兴趣的领域也能成为专家（更何况，兴趣的很大一类来源就是“我擅长做这件事情”）；反之就算有兴趣也很快会被一些冷水泼灭。

生活中的选择远比我们想象得要多，细微的选择差异造就了不同的人生

唐雅薇同学的故事中，有这么一个细节吸引了我的注意：当时她正在找工作阶段，对女生在 IT 行业的发展很迷茫，恰逢微软的郭蓓菁女士到他们学校演讲，演讲完了之后她立即就奔上讲台拦住郭女士询问女生与 IT 的

问题。

这是一个细节，但我相信不是所有人都有勇气上去拦住名人问普通问题的，我们会给自己找很多很多的理由和接口，我想最常见的应该是两个原因：1.如果被批评了自尊心会受到打击。2.认为问了也问不出特别的信息。然而事实却是相反：1.自尊心受到打击算不上实质性的损失。2.你想不出能问出什么特别的信息并不代表就真的问不到重要的信息。别把不知道当成没有。

一个小小的思维差异，可能导致很多人在遇到困惑的时候原地打转，冲突不出，而另一些人则取经得到宝贵的经验，站在别人的肩膀上越过了障碍。唐雅薇从郭女士那儿得到了最宝贵的信息：女生在 IT 行业也能有很好的发展。信心，是这样一种奇怪的东西，就算你没有确切的证明未来会更好，你也会坚持下去，你不会过早退出循环；而来源于过来人的信息则是信心最靠谱的保障。

你是不是意识到，在平常的生活中，你所作出的选

择比你想象的要多得多呢？有没有想过有一些看似细小的选择可能会产生巨大的影响？

想想看，试一下，是不是真的没什么损失，还有可能得到巨大的回报呢？

靠专业技能的成功是最具可复制性的

它需要的只是你在一个领域坚持不懈地专注下去，只需要选择一个不算太不靠谱的方向，然后专心致志的专下去，最后必然能成为高手或者绝顶高手。世上有很多成功带有偶然因素和运气成分或出身环境，但至少这一样，被无数人复制了无数遍，否则就不会存在学校和教育了。

反思是让人得以改进自己的最重要的思维品质

很多人在成年之后甚至未及成年，性格就难以再发生大的变化。性格是这样一种自我实现和强化的陷阱：如果你是不容易专注的人，你会发现生活中处处都是分散你注意力的东西，你的思维难以在一个事情上停留半

小时，于是你的时间变得琐碎，你很难在一个领域有长久的积累和深入的思考，这样的现实可能会让你感到沮丧，后者让你更加无法专心，这样的现实可能会让你感到焦虑，为了避开焦虑你又会去寻求其他的刺激，结果是恶性循环。

反思是改变自己的第一步，我们常常容易发现别人的问题，别人的错误，却难以发现自己思维中的问题，因为我们很少会把自己的思维当成目标去思考。

作为程序员，相信没有人不知道能修改自身的程序，而能修改自身的程序的前提就是，首先这个程序必须有法子能够指向自身。

饿死在干草堆之间的驴子

有很多在迷茫期的同学，迷茫都是相似的：面前有两条路，到底选哪一条？“转行还是不转行？”“学 C++还是学 Java。？“做管理还是做程序员？”有些问题其实不是问题：比如“学 C++还是学 Java。”答案是都学而且还不仅学两个。有些问题不是一个泛泛的答案能

够适合的，比如转行还是不转行，需要考虑很多自身因素。

但更重要的是，有人会因为无法作出决定就推迟决定，然而实际上推迟决定是最差的决定，在推迟决定期间，时间悄悄流逝，你却没有任何一条路上的积累，白白浪费了时间。

如果你有一些钱不知道花在 A 还是 B 上，你先不作决定，没问题，因为钱还是你的，但如果你有一些时间，不知道花在 A 上还是 B 上，不行，因为过了这段时间，这段时间就不是你的了。

所以，不管有多纠结，也不要从纠结中逃离，试图推延决定，既然终究是个痛苦的决定，就痛一回，好好思考和调查之后作出一个决定并坚持下去，只要不是太不靠谱的行业（相信也没谁会在纠结了之后却选了一个不靠谱的行业的），经过你的积累总会成为高手。

一生的知识积累，自学的起码占 90%

你会在这本书当中看到的一个重复出现的现象就是自学，大规模的自学，逃课自学，上网找书自学，程序员行业是最适合自学的行业，网络是程序员的天堂，需要的资源、工具，比课堂上的多出何止百倍，如果说还有一个学科，并不需要传统的教育就可以成才，估计非程序员莫属了。作为程序员如果没有查过 wikipedia，没有看过几本原版电子书，没有在国内外主要邮件列表里面提过问题吵过架，没有用技术博客记录学习的独特体会，没有订阅技术牛人们的博客，怎么好意思说身在这个行业呢？

最后，看完了书还是说“说起来容易做起来难”的，怪自己，不怪书。

## 3、如何有效地记忆与学习

你所拥有的知识并不取决于你记得多少，而在于它们能否在恰当的时候被回忆起来。

让我稍微说得更详细一点：学习新知识并将其存放于大脑中，最终的目的是要在恰当的时候能够想得起来

去使用。因此，学习的有效性显然应该这样来衡量：当遇到需要用到学过的知识的时候，相关的知识是否会自动从你脑海中“蹦”出来，最起码——能否通过有意识的搜索将它们提取出来。

这可不像它听上去那么简单，否则就不会有“掉书袋”、“读死书”这种修辞手法了。

为了更深入地说明这一点，以下是几个著名的关于学习与记忆机制的实验：

《找寻逝去的自我》上提到这样一个例子：

假设这样一个任务：给你一个单词（如 brain），要你寻找它的押韵单词（如 train）。一段时间之后问你记不记得当时给你的单词是什么了。你可能会不大记得了。现在，如果当时不是要你寻找押韵单词，而是要你联想该单词的含义或功能（如 brain 的功能），那么你事后回忆起来当时是什么单词的可能性就大大增加了。

对此一个靠谱的解释是：后一种记忆编码方式（称

为精细编码）提供了更多的提取线索。所谓条条大路通罗马，任何一个线索被触发都可能顺藤摸瓜地拎出整条鱼记忆来。

一个非常类似的实验是这样进行的（只记了实验，忘了出处了，顺便请教知道的同学:)Update：感谢 Leeve 指教，这是 Craik&Tulving 于 1975 年做的一个关于记忆的浅层深层加工的经典实验，参考这里(http://www.psychologistworld.com/memory/levels_processing.php)，论文原文可参考这里 http://scholar.google.com/scholar?q=Depth+of+processing+and+the+retention+of+words+in+episodic+memory）：

给出同样一组单词，让一组被试数一数每个单词有多少个音节，让另一组被试阅读单词的含义（或者设想单词可以被使用在哪些场景中），之后让两组被试回忆列表上的单词，猜哪一组能够回忆出更多？

这是一个被广为认可的记忆机制，即：我们在记忆的时候将许多线索（例如当时的场景、问题的背景，甚

至所处的语言环境、空间位置）一并编码进了记忆，事后能否提取出这段记忆严重取决于提取线索是否丰富、以及在回忆的时候是否重现了记忆时的线索。

原则上，在上面提到的两个实验中，两组被试都接触到了同样的单词，都记忆了同样的知识，但取决于在记忆的时候将哪些线索，多少线索和该记忆进行了挂钩，就决定了事后回忆（提取）该记忆的成功与否。

联系我们日常的经验，不难注意到，死板的记忆方式和我们常说的“理解记忆”正对应了不同的编码方式。书呆子记忆就是死记硬背，到最后如果你问他某书某章节讲了什么内容他能倒背如流，问他哪个例题怎么解也能倒背如流，但遇到具体的问题或问题的变种就傻眼了，因为他记忆的时候根本没有深入理解知识，在他眼中的解题过程其实和电话号码簿也没啥区别，也许他唯一编码了的提取线索就是这个答案来自哪一章、哪一节或者哪一个问题。

然而对于理解记忆的人来说，知识中包含了精细的

概念、逻辑、一般的解题原则、通用的解题手法、背景知识、类似的问题等等无数的记忆和提取线索，而不是某段孤立的、任意的文本序列。（当然，众所周知理解记忆的另一个重要特点则是记住一般性原理之后，其他细节即便遗忘了也可以自然推导出来，从而无需费力去记忆。有一个广为流传的《数学牛人们的轶事》（荣耀属于ukim）里面讲了希尔伯特的一个故事：一次在 Hilbert 的讨论班上，一个年轻人报告，其中用了一个很漂亮的定理，Hilbert 说“这真是一个妙不可言（wunderbaschon）的定理呀,是谁发现的？” 那个年轻人茫然的站了很久，对 Hilbert 说：“是你.....”。）

缺乏线索的记忆就像记忆海洋中的孤岛，虽然在那里，但是难以访问。而富含线索的记忆则是罗马，条条大路通罗马。

古希腊（或者古罗马）有一种著名的记忆法就是利用空间位置线索来辅助记忆，我曾经用过类似的手法，在小规模临时记忆任务中似乎相当好用，具体是这样的：

我有一个习惯，经常跑到实体书店看看有没有新书，浏览之后觉得不错的再考虑从网店购买。有时候我会一下看到好几本书，手头又没有带纸笔，怎么有效地记住几本书的书名呢？我发现要记 4 本书以上就比较困难了（具体数目可能因人而异），至少回头要想好久才可能想全。但我发现通过回忆当时手拿这本书翻看时所站的位置、面朝的方向等信息能够有效地帮助将记忆“拉出来”。

事实上，不仅是位置，有研究表明就连当时的环境、味道、声音都被作为提取线索和记忆编码在一块了。例如，考试环境和学习环境不一致可能会影响发挥（记忆的提取）。

科学松鼠会上有一篇科普《气味与记忆——非一般的亲密》中提到：

我们的回忆很多都是和气味连接在一起的。当闻到某一种味道时会突然想起以前的一些事情，比如端起一杯香热的巧克力饮料，想起了最初品尝巧克力的情景，将一块黑褐色的糖放入嘴中，浓浓的滑滑的，有一些甜

蜜和温馨；再比如，夏天在暴雨来临之前，浓郁的泥土和小草味道，会不会让你回忆起小时候因为没有拿伞被大雨淋透的感觉，甚至串联起回家挨揍的记忆，屁股上还有点火辣辣的痛。而当我们想起过年，鼻腔里是不是也会有厚厚的爆竹烟火味道，仿佛立马置身于热闹的大年夜。尤其是在社交活动中，我们经常会因为某一种味道想起一个熟悉的人，甚至是几十年没见的老朋友。

的确如此，我有时候会在看小说的时候放上一段背景音乐，之后每当听到这段音乐就想到当时看到的那段情节。

我们甚至会把语言背景作为线索编码进记忆。一项有趣的研究使用双语询问有多个答案的问题，例如：

“说出一个举起一只手并遥望远方的雕像”（“name a statue of someone standing with a raised arm while looking into the distance”），或者“在一个著名的悲剧爱情故事中，双方因为家庭不同意最终双双殉情，故事的主角是..？”（“In a famous love story,w

hat were the names of two lovers who died because of family disapproval?”），被询问的人皆是熟练的双语使用者。

结果显示提问所用的语言能够影响答案，例如第一个答案可能是“毛泽东雕像”或者“自由女神像”，第二个的答案则可能是“梁山伯和祝英台”或者“罗密欧与朱丽叶”。用英语询问能够导致被试更可能给出英文环境中的答案，中文同理——当然，是相对于基线而言，而非绝对，具体可参考这篇(PDF Paper)

另一方面，在回忆的时候如果不能呈现当初记忆的时候某些关键线索，就可能导致所谓的线索依赖性遗忘(cue-dependent forgetting)，在线索依赖性遗忘中，你的大脑中并非没有存放目标记忆，只是线索不对，无法提取出来而已。

对此有一个生动的比方是：你要到图书馆去找书，如果你不知道索引号、作者名、书名等信息，你是无法找到你要的那本书的，尽管书就大摇大摆地站在图书馆

里的某排书架上（注：严格来说你是可以找到的——一本一本翻看，看到那本书你自然会注意到是你想要的，不过很可惜对于我们的记忆系统来说似乎并没有这么一个方便的线性遍历机制——如果我问你，“对于数学你都记得哪些东西？” 你能有次序地一个不漏地告诉我吗？）。

我有记笔记的习惯，我的电子笔记本里面有大量的文本片段，我按照主题组织他们，方便检索。然而总有那么一些时候，我记得有某段材料，记得它的主题和大致说什么，但是缺乏某个关键字，结果就是遍寻不着，往往只能到处翻，同时提醒自己下次在上面多加几个自己熟悉的关键字，比如用自己的话来概括一下主旨，因为自己的习惯用词和作者的习惯用词往往不一样，在阅读作者的文字的时候，你也许下意识里会用自己的习惯词汇来重新表述这段文本，并存放在记忆中，结果一段时间之后当要寻找的时候大脑中只记得自己的说法，却不记得作者原话了，然而为了检索到原始文本你必须要知道作者是用什么词汇来表述的。为了弥补这个问题，

可以在存放文本的时候加上自己的一段概述，这似乎是一个不错的方案——我们平时在学习和记忆的时候也经常听到类似的提倡：用自己的话复述一遍之后理解得更深刻（实际上是更容易记住和提取出来——知识还是那些知识；此外用自己的话复述也常常触发与自己的知识体系中其他知识的联系，进而编码进更多的记忆提取线索，这也是另一个好处（《书写是为了更好的思考》））。

另一个经典实验则是关于抽象在知识的学习和提取中的作用的：

（其实这个实验我已经在博客里用过两次了）先让被试（皆为大学生）阅读一段军事材料，这个材料是说一小撮军队如何通过同时从几个不同方向小规模攻击来击溃一个防守严实的军事堡垒的。事实上这个例子的本质是对一个点的同时的弱攻击能够集聚成强大的力量。然后被试被要求解决一个问题：一个医生想要用 X 射线杀死一个恶性肿瘤，这个肿瘤只可以通过高强度的 X 射线杀死，然而那样的话就会伤及周围的良好组织。医生

应该怎么办呢？在没有给出先前的军队的例子的被试中只有 10%想到答案，这是控制基线。然后，在先前学习了军队例子的被试中，这个比例也仅仅只增加到 30%，也就是说只有额外 20%的人“自动”地将知识进行了转移（自己就能触类旁通）。最后一组是在提醒之下做的，达到了 75%，即比“自动”转移组增加了 45%之多（需要别人提醒）。

这个实验说明，知识的表象细节会迷惑我们的眼睛，阻碍我们对知识的转移运用（推广），在这个例子中，两个问题领域表面上是不相似的，但本质上是一样的。然而就是因为表面上不相似，而我们的记忆提取又是很大程度上依赖于一些表象上的线索来提取的，因此这些表面不相似性便阻碍了我们在问题之间进行的类比，阻碍了我们将在一个情境下掌握的道理运用到另一个情境下。

这就意味着，我们在从既有经验中总结知识的时候，应利用适当的抽象来得出适用范围更广的知识（而不仅

仅是一个萝卜一个坑）;另一方面，在遇到新问题的时候，同样应该对问题进行抽象，触及其本质，去除不相干因素避免干扰，从而有效提取之前抽象出来的知识。

通俗的来说，这就是举一反三，触类旁通的解释。

前文说的是记忆的机制、为何记忆的质量有高低、什么样的记忆和学习是更有效的。下文是一些具体的实践方法，关于如何更有效地从日常经验中总结知识，以及如何能够真正学以致用——使知识能够在你需要它们的时候自动从大脑中“蹦”出来，而不是搜肠刮肚半天还是没有头绪。

1)养成习惯，经常主动回顾一段时间学到的东西（老生长谈了）：这不仅有利于巩固长时记忆，而且一段时间之后的回顾你可能已经因为新的知识学习从而对原先的认识有了进一步的看法，通过回顾，可以整合新旧知识，得到新的启发。

2)创造回忆的机会：我知道第一条不顶用，没有人（好吧，很少有人）能够真正坚持执行。所以有了第二

条——创造回忆既有知识的机会。具体来说就是通过：

2.1)经常与别人讨论，或者讲给别人听。经常和朋友讨论交流，说说一段时间总结的东西，这样别人也学到新东西，你也从别人那里学到新东西，并且彼此在表达的过程中都强化了自己的记忆和理解，双赢的事情。除了面对面的交流之外，一个好的邮件列表和 BBS 也是不错的途径。（详细解释可以参考《为什么你应该（从现在开始就）写博客》第三节：“教是最好的学“）

2.2)整理笔记：经常整理你的笔记——如果你没有做笔记，现在就开始——整理之前的笔记一来巩固已经淡化的记忆，二来给你重新审视知识的机会。我常常发现对知识的首次记忆往往是有偏颇的，或者只看到了一个方面，或者只关注了一个点，一段时间之后再回来看往往能够和这段时间以来的一些新思考和知识结合起来，得到更多的东西。留心一下你会发现记忆实际上是很脆弱的东西，而且我们对事物的首次理解几乎肯定是不深入的。Tip：我知道你懒（我也是），所以为了更好

地创造整理笔记的机会，你可以使用一个不整理就难以检索的电子笔记软件，这虽然乍看上去是麻烦了一点，但他迫使你对知识隔一段时间就进行重读，并分类——你的记忆同样如此：良好分类的信息更容易提取。

2.3)书写 :将一段时间学习的知识按照一个主题系统地“串”起来大大地丰富了知识之间的关联，平添无数提取线索。我经常做这个事情，这个博客上的文章几乎都是此类文章，例如我始终关注一个主题：学习思维相关的科学（认知科学、心理学、行为经济学等等）如何能帮助我们进行更好的判断、决策、学习、记忆和生活，我将这个大的主题分为一些小的主题，例如“逃出你的肖申克”主要是总结思维中的盲点，以及如何避免这些盲点从而成为更好的独立思考者，作出更好的判断与决策。“BetterExplained”系列则是按照小主题总结一些思维相关的知识，目的仍然是如何成为更好的独立思考者，对事物进行更理性的判断；这些小主题都归结在“思维改变生活”这个大的主题之下。（关于书写的好处，详细解释可参考《书写是更好的思考》）

3)设身处地地“虚拟经历”别人经历过的事情：我们的自传式记忆似乎是有某种单独存储机制的，一个证据是一些因基因上其他缺陷而导致所谓“天才综合症”的家伙具有超强的自传式记忆（注意，不是超强的一般记忆，而是自传式记忆）。另外我们在日常经验中也知道，我们的记忆中关于哪些是自己的性格或做事方式，哪些是我们所了解的朋友的性格或做事方式，我们可是分得清清楚楚的。我们可以在不同场景中非常快速地揣测“某某在这种情况下会怎么想”（这被称为 theory of mind），却不会将其与“我自己会怎么想”混淆起来，证明在我们的记忆中，关于自己的知识和关于别人的知识是泾渭分明的。

对于经验知识的学习来说，光是看着别人做或者听着别人说还不够，往往到了自己就想不起来，结果就是你虽然学到了知识，它却不会在恰当的时候从你大脑中蹦出来，属于“死”的知识。为什么会这样，可能的原因是很多的，其中一个关键的原因也许是“别人的事情”和“自己的事情”在大脑中的加工方式是不一样的，别

人撞墙你也许不仅不疼还会幸灾乐祸，自己擦破皮就龇牙咧嘴了；别人的糟糕事情似乎永远不会发生到自己身上。所以我们总是难以从别人的经验中获得自己的教训。一个弥补的办法在于努力设想自己处于别人的境地，经历别人所经历的事情，感受它们，使它们和你的情绪记忆挂钩（进化赋予我们的情绪是提取的绝佳线索，也是强化记忆的最佳催化剂），虽然仍然不够亲身经历那么深刻，但似乎已经是我们能做到的最好的办法了。由于我们真切地设想了自己处于这些场景中，在我们设想的场景中我们是第一人称视角，所以当以后遇到类似场景的时候就更容易回忆起当时的感受。

当然，另一个经常被号召的方法就是实践，比起刚才提到的“虚拟实践”而言，实际实践的印象自然要深刻得多。不过并不是所有的时候实际实践都是必须或者可能的。例如你并不需要自己去倾家荡产一次才能领会到什么是金融市场中正确的风险控制——你甚至只需要在纸上演算一番就有数了。有证据表明非洲的一种鱼甚至都能使用简单的推理来替代实际经历，例如，如果它

和鱼 B 有过一次冲突并失败了，如果它观察到鱼 B 和鱼 C 有一次冲突，鱼 B 失败了，它就能直接意识到它自己不是鱼 C 的对手，从而避免所谓“直接去经历一下”而可能导致的灾难性后果（这里的进化价值是显而易见的）。

此外，很多时候你也无法真正遍历每条人生路径看看会发生什么，你没有这样的时间资源，取而代之的是你只能通过别人的“替代经验”，自己的“虚拟经历”，来获得尽量多的信息。

4)抽象和推广：如果一件事情就是一件事情，那么我们永远也无法学习到“未来”的知识，结果就是每堵墙都要去自己撞一遍试试硬度。人类大脑最杰出的能力之一就是强大的归纳推理（inductive reasoning），或者我们常说的：泛化、推广、举一反三、抽象。意思都差不多，都是将特例中得到的规律推广到一般情况。前面关于激光杀死肿瘤的那个实验充分说明了抽象的价值所在，不加抽象的话，知识总是会和无关紧要（irreleva

nt）的细节挂钩，被约束在狭窄的一个特定场景中，无法传播，抽象使其在知识树（for non-geeks：设想一颗倒长的树，根在上）上上升一个或多个层次，从而能够被运用到更多的分枝上。同样，在遇到具体问题的时候也别忘了将问题也抽象一下，剔除不相关细节，使问题也从一个特定的分枝往上抽象，从而碰到之前泛化过的结论。

以上这段介绍本身就有点抽象，不妨举一个例子：我们从大量的经济决策中得到一条适用范围很广的规律——经济决策可以抽象为对投入/回报比例的考量。这是知识获取阶段的抽象；而在问题求解阶段，我们遇到决策问题就可以从投入/回报这个维度上来考量，而不是没头苍蝇一样这边看着想想也对，一忽儿又觉得那个选项看看也对。如果不懂得看到问题的本质（如：经济决策），便很难利用之前推广出来的结论（如：投入/回报，风险估计等等），而是会被我们的原始大脑的一些可预期的非理性所控制（例如从众、从权威、甚至最可怕的行为陷阱——“推迟决策”），成为正常的傻瓜。

5)联系/比较自身的经历：将别人的经历或者通过阅读和观察得来的经历和自身的经历进行比较，常常能够得到非常有价值的结论。“观察”和“比较”本身就是获得知识的一个重要途径，例如：我之前做过某件事情，但不知道什么原因失败了；有一天我看到或阅读到某个人做类似或同样的事情，他成功了。我通过比较两人的差别，可以比较靠谱地推测到底是什么导致了我们的成功概率的差异。

值得注意的是：1)样本大小很关键：比较的个体样本越少越容易产生错误归因，最好多多观察，多多比较和总结。2)警惕“沉默的证据”、事后偏见、自利归因：读他人的传记的时候，不管传记是本人写的还是传记作家通过访谈写的，都会有意无意地犯事后偏见，例如最常见的将成功归因为个人能力，忽略机遇因素。将偶然看作必然。

有一天我在书店看巴菲特的那本最近很火的自传《滚雪球》，开篇就提到巴菲特小时候第一次滚雪球的场

景，“1939 午的冬天，9 岁的巴菲特在院子里玩雪。他把少量的积雪铲到一块，揉成一个雪球，然后把它放在地上慢慢滚动，雪球越滚越大..从此，巴菲特再也没有停下脚步，目光投向白雪皑皑的整个世界..”（虽然你可以说这只是一种修辞或衬托，但不可否认的是它背后隐藏的是一种无法抗拒的归因倾向）这种手法读来令人倍感深沉，仿佛冥冥之中有一条强大而确定的因果之线，穿越 60 年光阴，将人一生所有的事情穿在一起。令人肃然起敬。然而这并不是事实，从一个单个个体的观察角度来说，外界的不确定性因素实在太多了，机缘巧合的事情太多了。然而无论是作者本人还是“客观”作家都很难抵抗这种演绎手法的诱惑，可叹的是这已经是我们能够了解他人的经历，拥有“虚拟”的多重经历的唯一途径。

观察、阅读，并别忘带着你的理性去审视（包括本文），弄清娱乐是娱乐，知识是知识，如果你想真正得到一些知识，最好过滤一下你的信息。否则你只是在别人的思考中得意着。

## 4、学习密度与专注力

上次学校里面有一个免费的李阳英语讲座，好奇于是就去听了一下。对一句话印象比较深刻，大意是说许多人学了快 10 年的英语 ,其开口的时间还不如在集训的七天内开口的时间长。也就是说，尽管学习时间很长，但学习密度极低，结果乘起来还是低。其实这种情况不仅存在于英语学习中，而是一种普遍的现象。人太容易为各种各样的事情分心，要集中注意力做一件事情是非常难的，而正因为难，少有人做到，那些做到的，就都变成了牛。

其实，在大学期间，最不缺的就是业余时间，最缺的就是专注精神，非凡的注意力造就非凡的专家(http://blog.csdn.net/g9yuayon/article/details/1436970)。而生活中太多的分散注意力的因素：游戏、篮球、选修课、女朋友...要想集中注意力对一个单一的目标猛下功夫，其实还是相当有难度的。这个难度并非来自自制力，如果一个人要靠自制力去强迫自己不受干扰，那只能说

还是寻常人(mediocre)。真正的效率源自于内心对一个东西强烈的热忱，也就是我们俗称的追求，这时候从表层意识到深层意识都关注在这件事情上面，脑细胞高度活跃，才能创造最大的效率。为什么作诗的时候要趁着酒兴，就是因为少了这种狂热的专注，效率就低下了，一首诗作个好几天顶多是个平庸之作，跟交家庭作业也差不到哪去了。很多人正是因为缺乏专注，所以虽然也和别人一样过来了大学四年，实质上四年里面利用的时间无形中少了不知多少。

专注力为什么会对学习效率造成这么大的影响。这来源于两个方面，一是专注于一件事情能让表层意识全功率运作，这个是显式的效率。第二点，也是更重要的，它还能够使你的潜意识进入一种专注于这件事情的状态。有过连续几天乃至一周或更长时间思考同一个问题的人想必都有一种感觉，就是在这个思考的期间，有时候虽然表层意识因为种种原因不在思考这个问题了（比如睡觉，比如被其它事情中断），但潜意识层面仍然保留着其“惯性”，也就是说，潜意识层面仍然在做思考的努

力，从而虽然表层意识被其它事情占据，但潜意识仍然将时间无形中利用起来了。这种无形中的时间利用日积月累可以产生宏大的效应。关于后一点，著名的例子有我们熟知的那个睡觉中想出苯的化学分子式的老大。非著名的例子有老爸告诉我的两个事情，一是他在 20 岁左右，组装村子里第一台电视机的时候，装到最后关口，电视机总是不能工作，苦思冥想一整天不得要领，结果睡到半夜突然从梦里醒来，想到了答案，连夜就把电视机装好了。还有一次是妹妹拿一道高中数学题问他，也是想了一天多没答案，结果睡午觉的时候想到了。这些都很好的证明了潜意识能在你觉察不到的情况下产生效率。另一方面，潜意识也能在你觉察不到的情况下干扰你的注意力，我们平常就有这样的经验，一个球迷即使在表层意识专心工作的时候也会不知为何突然想起比赛的事情，一个焦虑某件事情的人即使在做其它事情的时候也会被突然涌上来的焦虑打断。也就是说表层意识在关注一件事情，但潜意识却在关注另一件事情，并且时不时来打扰表层意识，从而影响注意力和效率。所以，

如果表层意识和潜意识都能专注同一件事情，也就是俗称的完全投入，这个时候的效率就能 double。此外这种专注成了一种习惯之后，就容易在很短时间之内把自己的潜意识带入到一种关注的“惯性”中，于是即便表层意识的注意力已经移开了，然而潜意识仍在继续关注原来的问题。比如你可能有这样的经历，学习一首歌曲，一开始的时候并没有完全学会，然后你就去忙别的事情了，一个星期之后想起这首歌曲，居然发现原来难学会的几个地方突然会哼了；或者思考一个问题，一开始的时候总有一个地方没有思考出来，然后你就先放着了，几天之后回想这个问题，突然发现一切都清晰了。这就是潜意识的效率，它能在你不知不觉中把时间利用起来。

了解专注力的作用不够，如何获得专注力才是更重要的问题。跟人身上的其它特质（性格、心态...）一样，专注力也是一种习惯。一个习惯于专注事情的人不管做什么事情都容易并迅速进入一种专注的状态。既然是一种习惯，就能够培养，金出武雄在《像外行一样思考，像专家一样实践——科研成功之道》里面提到“思维体

力”的概念，所谓思维体力就是能够持续集中注意力的时间，注意力造就非凡专家，天才来源于长期的专注的训练。培养你的思维体力，是成为非凡专家的一个必要条件。除了培养专注的习惯之外，还可以通过另一个充分条件来实现专注力，即做自己喜欢做的事。我们从小对自己喜欢做的事情都是极其专注的，当然，即使长大了之后，仍然还是某种程度上保留了这种专注的能力，只不过因为种种外界因素，长久专注的能力反而削弱了，要考虑房子，要考虑业绩，要考虑小孩，要考虑医疗保险...这些让人焦虑的事情会积压在潜意识当中，总是在影响你专注做事，削弱你人生的效率。卡耐基用一整本《人性的优点》来介绍如何克服焦虑，可见焦虑的负面影响有多大。要使自己能像小的时候一样对喜欢的事情投入最大的专注，除了克服焦虑的负面影响之外，还有另一个条件就是不能放弃，今年的奥斯卡独立电影《阳光小美女》上，Frank 和 Dwayne 在码头的那场 Loser 对话，以及 Richard 决定把他老爸的遗体带走时说的：“世界上有两种人，赢家和失败者，两者的区别在于，

赢家从不放弃...”。佛家说逆境是增上缘，课本说天降降大任于斯人必先苦其心智劳其筋骨...宗教里说经受磨难才能到达彼岸，说的都是同样的道理。不过我还是更欣赏 Frank 在码头说的那段话（摘自卓别灵的 blog）：

“(普鲁斯特)是个法国作家。彻底的失败者。一生没工作，情事不断还是个同性恋。花 20 年写了一本没几个人看的小说。但他也许是莎士比亚之后最伟大的作家。晚年回首人生，他发现那些难熬的日子才是一生中最好的时光，因为那些日子造就了他。而快活的日子全是浪费时间，没有任何收获。你想一觉醒来就到 18 岁，觉得这样可以跳过高中时期的痛苦。但高中是你一生中最重要的苦难时光，你不可能经历比这更好的苦难了。”

多好的心态啊。什么是黄金心态，这就是。其实过来人你我都有这样的体会。此外，如何不让生活中其它细节干扰也是一个重要的因素，除了那些你焦虑的事情之外，还有诸如收拾衣服、打扫房间、清理书桌、接孩子回家、瑜伽等等；对此史蒂芬柯维在《高效能人士的

七个习惯》里面提到的第四代时间管理法则，即要事第一（指重要但不紧迫的事，即长远积累会对你今后人生产生重大影响的事）法则就非常有效。关注有两种关注法，主动关注和被动关注，许多人对琐事错误的采取了主动式关注，比如常常回到家就开始想“今天还有哪些事没做完”，实际上，让这些不重要的事情自己来找你就可以了，即中断式被动关注，后者可以防止空转轮询浪费的时间，从而把最集中注意力的时间利用在最重要的事情上。

最后，如何知道你已经获得了专注力。这样的现象太多了，比如本不想洗头却无意识把洗发露倒在手上结果不得不洗头，或者干脆把洗发露当沐浴露了。比如去食堂吃饭指着南瓜说黄瓜（因为南瓜是黄的），或者端了免费汤还拿卡出来打卡。至于像牛顿这样牛到顿的老大能把手感差异如此巨大的表当成鸡蛋煮了的阶段还远没达到，看来人家姓牛也不是白姓的:-)

## 5、一直以来伴随我的一些学习习惯

## （一）学习与思考

1.Google&Wiki（遇到问题做的第一件事情，也是学习某个东西做功课（homework）最先用到的东西。

2.看书挑剔，只看经典。如何选择经典，可以到网上做做功课，看看评价，综合分析一下。

3.做读书笔记。一是将自己阅读的时候的思考（包括闪念）总结下来，二是将书中的好例子摘抄下来。（这个习惯是一年前才养成的，发现受益极大。）有了 google note，笔记可以加上 tag，非常便于回顾，加深理解。我觉得，人与人学习的差距不在资质上，而在花在思考的时间和思考的深度上（后两者常常也是相关的）。

4.提到思考，我有一个小习惯。利用走路和吃饭的时候思考，还有睡觉前必然要弄一个问题放在脑子里面，在思考中迷糊入睡。发现这样一来往往在不知不觉中多出来大量的思考时间。

4a.将思考成为习惯还有一个很大的好处——避免

焦虑。卡耐基用一整本书讲了一个有效的做法来避免焦虑——底线思考。然而实际上还有另一个有效的做法，就是投入地做另一件事情。不去想"喜马拉雅山上的猴子"的方法并不是使劲的告诉自己不去想"喜马拉雅山上的猴子"，因为那样等于就是脑袋里想了那只猴子，正确的做法是真的不去想那猴子，而是想别的。用别的东西充满工作记忆，其他的神经活动自然会被抑制（神经科学基本事实）。所以，感到焦虑的时候不妨思考吧，甚至完全可以去理性分析和思考导致焦虑的问题本身，将其拆解，分析来源，在不知不觉中，大脑的工作重心就从情绪模块转向了推理模块了，而且这思考也可能顺带更有效地解决了导致焦虑的问题呢:)

5.重要的事情优先（详见史蒂芬·柯维的《高效能人士的七个习惯》或《要事优先》）。尽量避免琐事骚扰，不重要的事情能不做就不做。有时候，紧急的事情往往只是当事人觉得必须马上做完才显得紧急或者干脆就是紧他人之急，最糟糕的就是纯属性格上原因觉得每件事情都得第一时间完成，很多看上去紧急的事情实际上并

不是真的"不能再拖了"，有的干脆就并不需要或值得去做。有很多事情都是可以先放一放甚至完全 let go 的，否则的话就整天被所谓"紧急"的事情牵着鼻子走了。

6.重要的事情营造比较大的时间块来完成。比如一本好书，或者一个重要的知识点，最好不要切得太琐碎了看，否则看了后面忘了前面。不利于知识的组织&联系。

7.多看心理学与思维的书，因为它们是跨学科的。知识分两种，一是我们通常所谓的知识，即领域知识。二是关于我们的大脑吸收知识的机制的知识，后者不妨称为元知识。虽说这也是领域知识，但跟其它的领域知识不同的是，它指导着我们学习其它所有的领域知识。

8.学习一项知识，必须问自己三个重要问题：1.它的本质是什么。2.它的第一原则是什么。3.它的知识结构是怎样的。

9.获得的多少并不取决于读了多少，而取决于思考了多少、多深。

10.善于利用小块时间，也就是《奇特的一生》中所说的“时间下脚料”，如何利用前面有几个方法。同时，也善于创造整块时间（如通过要事优先）。

11.关于习惯的养成，必须要说明的：经常看到有些人评论说，说说容易，做起来哪有那么容易啊（另一个无关习惯的“说起来容易做起来难”则是因为纸上谈兵不可能算计到所有现实中的因素，但那是另一个问题）。对此我要说的是，做起来当然不容易，所谓江山易改，本性难移。人的性格和认识事物的框架是长期积累养成的，并且人们非常珍视自己的信念（英语里面表达不相信某个东西叫做“I don’t buy it”）。从进化心理学上这是有依据的，一个经过时间检验的信念往往是更靠谱的。只不过可惜的是靠谱不代表最佳，一个信念能让你活下来并不代表能让你活得最好（详见《Mene Genes》，更多的例子参见《How we know what isn’t so》）。我们评判一个信念的标准是 satisficing 原则（即足够，能行就好，这个术语不是我提的，是大牛 Herbert Simon 提的），并不是 optimizing 原则。话说回来，为什么说

起来容易做起来难，是因为“说”只是理性上承认正确，并没有考虑到我们每个人大脑中居住的那个非理性自我。这个自我以强大的情绪力量为动机，以习惯为己任，每时每刻都驱使着我们的行为。因为它掌握了“情绪”这个武器，所以我们只能时时拿它当大爷。不记得是哪位哲学家说的了，理性是感性的奴隶。那么，是不是就是说无法克服既有习惯了？以我的经验（以及观察到的别人的经验），还是可以的。第一条就是认识到习惯的改变绝不是一天两天的事情，承认它的难度。第二条就是如果你真想改掉习惯，就需要在过程中常常注意观察自己的行为，否则习惯会以一种你根本觉察不到的方式左右你的行为让你功亏一篑。有一个认知技巧也许可以缓解更改习惯过程中的不适：即把居住在内心的那个非理性自我当成你自己的孩子（你要去培养他），或者你的对手（你要去打败他）也行。总之不能当成自己，因为每个人都不想改变自己。这里转一个认知技巧的例子：李笑来老师在《把时间当作朋友》（顺便也推荐这本开放电子书）中提到他一个朋友用另一个认知技巧来克服背单

词的枯燥的：

因为，一共要搞定 20,000 个单词，而因此可能获得的奖学金是每年 40,000 美元左右——并且连续四年没有失业可能（后来的事实是，他直到五年之后才获得了博士学位）。当时的美元兑换人民币的汇率差不多是 8:1，所以，大约应该相当于 320,000 元人民币。而如果一年的税后收入是 320,000 元人民币的话，那么税前就要赚取差不多 400,000 元人民币。那么，每个单词应该大约值 20 元人民币——这还只不过是这算了一年的收入而已。

所以，他终于明白背单词是非常快乐的。他每天都强迫自己背下 200 个单词。而到了晚上验收效果的时候，每在确定记住了的单词前面画上一个勾的时候，他就要想象一下刚刚数过一张 20 元人民币的钞票。每天睡觉的时候总感觉心满意足，因为今天又赚了 4000 块！

注意，这跟自我欺骗不同。一来，我们的情绪系统只能这般对付（《Synaptic Self》中提到，大脑中的新皮

层（neocortex，所谓“理性”居住的地方，尤其是前额叶）在进化历史上是较为新近的年代才进化出来的，跟底层较原始的模块（如主管情绪的杏仁核）之间的神经网络沟通并不是合作无间，这就解释了为什么有些事情我们明明知道是对的，但就是不能说服自己，情绪还是在那里不依不挠的驱使你去做另一样事情）。二来，我们知道在干什么，所以不能算欺骗:P 总之，对于习惯的更改，除了最重要的一日三省，加上一些认知技巧（其实每个人都是自己的心理学家，你可以自己看看能不能想出什么法子）。其实是没有什么速效银弹的。但是，知难而不退嘛，值得做的事情几乎总是如此:)

## （二）时间管理

1.学习和思考的过程中常问自己的几个问题：

你的问题到底是什么？（提醒自己思考不要偏离问题。）

OK，到现在为止，我到底有了什么收获呢？（提醒自己时不时去总结，整理学习的东西）。

设想自己正在将东西讲给别人听（有声思考；能否讲出来是判断是否真正理解的最佳办法）。

3.1 设想需要讲给一个不懂的人听。（迫使自己去挖掘知识背后最本质、往往也是最简单的解释）。

时常反省和注意自己的思维过程。尤其是当遇到无法理解或解决的问题之后，最需要将原先的思维过程回顾一遍，看看到底哪个环节被阻塞住了妨碍了理解。问题到底出在哪里。并分析以后需要加强哪方面的思维习惯，才能够不在同样或类似的时候被绊住。对此，将思维的大致脉络写下来是一个很好的习惯。

养成反驳自己的想法的习惯：在有一个想法的时候，习惯性地去反驳它，问自己“这个难道就一定成立吗？”、“有没有反例或例外？”、“果真如此吗？”之类的问题。（参见 Critical Thinking）

人的思维天生就是极易流于表面来理解事物的（参见《Psychology of Problem Solving》第 11 章）。觉得自己理解了一个问题了么？条件反射性地问自己：你

真的理解了吗？你真的理解了问题的本质了？问题的本质到底是什么？目前我的理解是什么？我对这个理解感到满意吗？这样的理解到底有什么建设性呢？等等。

2.重视知识的本质：对于程序员来说这一点尤其重要，程序员行业的知识芜杂海量，而且总是在增长变化。很多人感叹跟不上新技术。应对这个问题的办法只能是：抓住不变量。大量的新技术其实只是一层皮，背后的支撑技术其实都是十来年不变的东西。底层知识永远都不过时。算法数据结构永远都不过时。基本的程序设计理论永远都不过时。良好的编码习惯永远都不过时。分析问题和解决问题的能力永远都不过时。强大的学习能力和旺盛的求知欲永远都不过时。你大脑的思维方式永远都不过时。

3.重视积累的强大力量，万事提前准备：计划订长一点，自然就可以多获得准备的时间。设想你若干年后会在做什么事情，需要哪些技能，现在就开始准备。一个5年计划便可以让你获得从现在开始的5年准备时间。

5 年中每天腾出半个到一个小时专心于某一件事情，认准一个方向，每次走一点，其实不要说 5 年，两年就会发现会起到宏大的效应。长期订阅我的 Blog 的朋友们也一定注意到我基本上不写东西，一般一个月写上 2 篇就算多的了。但总结一段时间的学习和思考的习惯却一直都没有停止（博客文章对我来说是学习和思考的副产品，我并不为写文章而写文章），所以 5 年下来竟也写了不少东西。所以这就是一个简单的例子。你大致还可以从我的 Blog 看出来我一段时间关注的东西，一般来说，一段比较长的时间（少则半年至一年——譬如对心理学与思维的关注；多则几年——譬如对编程技术的关注），在这段时间内，我的业余时间会被一个主题所充斥。反之，如果不知道目的是什么，就不知道往哪个方向上使劲，就容易产生无用功。

4.抬起头来：人的思维是非常容易只见树木不见森林的（否则这个成语从哪来的呢？）。时不时抬起头来审视一下自己正在做的事情，问一问它（对现在或未来）有什么价值，是不是你真正希望做的。你学到的东西到

底是什么？它们重要吗？你需要在这个时候学习这些吗？（见第 2 条）。你的时间就是你的资源，你投入这些资源来掌握知识，所以到底用来掌握哪些知识是一个很重要的问题。仅仅遵循兴趣是不够的，人会对很多次要的东西产生兴趣，并一头钻进去浪费好多时间。所以判断一个东西值不值得学习是很重要的。

杂项

1.退订 RSS：RSS Reader 是个时间黑洞。就算 mark all as read，在有大量 feed 的情况下，也会无形中消耗掉大量的时间。我们一旦订阅了某个 RSS 之后就会倾向于不肯退订它，心想也许某天有个重要的信息会从那里得到。这其实是源于人不肯"关上一扇门（即便门内的收益概率极小）"的心理（参见《Predictably Irrational》）；而实际上，关上一扇门，有时能够增大收益期望。仔细观察一下 reader 里面的 feeds，有哪些是真正有价值的，把那些没价值的或者价值很小乃至于不值得每天被它骚扰的，全都退订掉。不要舍不得，那些一个星期

都没出现让你眼睛一亮的内容的 feed，很大的可能是永远也不会出现。就算可能，也别担心你会漏掉什么宝贵信息，真正宝贵的信息，在其他来源你也会接触到的。一开始我的 Greader 里面每天都有大量的新内容，每天都是 1000+，但一段时间之后发现除了信息焦虑，实际上有价值的内容不多，现在，我很高兴地发现自己摆脱了这种状况，我持续不断地退订 feeds，留下的内容越来越少，也越来越精，带来的阅读焦虑也越来越少。（顺便推荐一个东西，aideRSS，初步使用，感觉对订阅 reddit 这样的每天更新大量内容的 feed 很有用）。

2.有时间吗？总结总结最近得到的新知识吧。一般来说，我在一段时间内学习的一些东西总是会在这段时间内一直在脑子里打转，一有时间空隙（譬如走路，吃饭）它们就会自己蹦出来，促使我去进一步思考和总结。永远不要认为对一个知识的把握足够深刻，“理解”的感觉很多时候只是假象。学会反问自己对知识到底把握了多少，是很有价值的。（如何反问，前面的总结中有提到）。

3.有时间吗？看本书吧。（传统的）阅读和思考永远优于所谓的在互联网上汲取新知识，后者往往浅表、不系统、乃至根本没价值。

4.制定简要的阅读计划：选出最近认为对你最有价值的书，先总览一下，决定阅读的顺序（哪些章节可以优先阅读）。然后每天看一点。并利用走路、吃饭、乘车或其他不适合带着书和笔的时间来总结看过的内容，建立知识结构，抽取知识本质，与以往的大脑中的知识建立联系。（参见《奇特的一生》）

## （三）阅读方法

这篇主要写一些学习（尤其是阅读）的基本方法。

1.趁着对一件事情有热情的时候，一股脑儿把万事开头那个最难的阶段熬过去。万事开头难，因为从不了解到了解基本的一些事实，是一个新知识暴涨的阶段，这个时候的困难是最大的。有人熬不过去，觉得困难太大就放弃了。不过，狂热的兴趣可以抵消对困难的感觉，所以趁着对一件事情有热情的时候，开一个好头是很重

要的。（当然，这并不是说持之以恒就不重要了）。当然，也许这个是因人而异的，对我来说我会在对一件事情有浓厚兴趣的时候非常专注地学习，把很多 groundworks 做掉。后面就会顺利一些了。

2.根据主题来查阅资料，而不是根据资料来查阅主题。以前读书的时候是一本一本的读，眼里看到的是一本一本的书，现在则是一章、甚至一节一节的读，眼中看到的不是一本一本的书，而是一堆一堆的章节，一个一个的知识主题，按照主题来阅读，你会发现读的时候不再是老老实实地一本书看完看另一本，而是非常频繁地从一本书跳到另一本书，从一处资料跳到另一处资料，从而来获得多个不同的人对同一个主题是如何讲解的。比如最近我发现在看蒙特卡罗算法时就查了十来处资料，其中有三四篇 paper 和六七本书；这是因为即便是经典的书，你也不能指望它对其中每一个主题的介绍都是尽善尽美的，有些书对某个主题（知识点）的介绍比较到位，有些书则对另一些知识点介绍得比较到位。而有时候一篇紧凑的 paper 比一本书上讲得还要好。我硬

盘里面的书按主题分类，每个主题下面都有一堆书，当我需要学习某个主题的知识时（譬如贝叶斯学习或者神经网络），我会把里面涉及这个主题的书都翻开来，索引到相关章节，然后挑讲得好的看。那么，如何判断一个资料是好资料还是坏资料呢？

3.好资料，坏资料。好资料的特点：从问题出发；重点介绍方法背后的理念（rationale），注重直观解释，而不是方法的技术细节；按照方法被发明的时间流程来介绍（先是遇到了什么什么问题，然后怎样分析，推理，最后发现目前所使用的方法）。坏资料的特点是好资料的反面：上来就讲方法细节，仿佛某方法是从天上掉下来的，他们往往这样写“我们定义...我们称...我们进行以下几个步骤...”。根本不讲为什么要用这个方法，人们最初是因为面对什么问题才想到这个方法的，其间又是怎样才想出了这么个方法的，方法背后的直观思想又是什么。实际上一个方法如果将其最终最简洁的形式直接表达出来往往丢失掉了绝大多数信息，这个丢掉的信息就是问题解决背后的思维过程。至于为什么大多数书做不到这

一点，我在这里试着分析过。

4.学习一个东西之前，首先在大脑中积累充分的“疑惑感”。即弄清面临的问题到底是什么，在浏览方法本身之前，最好先使劲问问自己能想到什么方法。一个公认的事实是，你对问题的疑惑越大，在之前做的自己的思考越多，当看到解答之后印象就越深刻。记得大学里面的课本总是瀑布式地把整个知识结构一览无余地放在面前，读的过程倒是挺爽，连连点头，读完了很快又忘掉了，为什么？因为没有带着疑问去学习。

5.有选择地阅读。很多人觉得我读书速度很快，其实我只是有选择地阅读。这里的选择体现在两个地方，一是选择一本书中感兴趣的章节优先阅读。二是对一本书中技术性较弱或信息密度较低的部分快速地略读。一般来说，除了技术性非常强的书之外，大多数书的信息密度很低，有很多废话。一般来说在阅读的时候应该这样来切分内容：1.问题是什么？2.方案是什么？3.例子是什么？如果是需要解释一个现象的（譬如《黑天鹅》），

那么 1.现象是什么？2.解释是什么？3.支撑这个解释的理由是什么？4.例子是什么？一般来说，这一二三四用不了多少字就可以写完了（如果假设只举一到两个精到的例子的话），这样的无废话著作的典型是《合作的进化》；那为什么有些书，明明核心观点就那点东西（顶多加上几个精要的例子罢了）却写得长得要命呢？因为人的思维都有一个“联想”的特点，写着写着就容易旁逸斜出，而且作者自己也往往觉得引申出去挺牛逼，有时候很多与主题无关的废话就掺和进来了；那么，阅读的时候就应该有选择性地滤掉这些不相干的废话；此外还有一种可能性就是大量冗余的例子。一般来说组织得比较好的书会有详细且一目了然的目录和索引，根据目录首先就可以滤掉一部分（比如某个子章节的内容你以前是看过的），然后有时候作者还会举很多冗余的例子，如果你已经觉得印象够深刻了这些例子完全可以不看（一些书就非常厚道地对每个观点只辅以一两个最最经典的例子，譬如《与众不同的心理学——如何正视心理学》，这样的书我最是喜欢）。

6.为什么看不懂？如果看不懂一个知识，一般有如下几个可能的原因：1.你看得不够使劲。对此古人总结过——书读百遍其义自现。虽然这个规律不是任何时候都成立的，但是从认知科学的角度看是完全可以解释的，我们在阅读的时候，注意力往往会有选择性地关注其中的某一些“点”，而忽略了另一些“点”，于是一遍看下来可能因为某一些忽略导致无法理解整体。或者干脆看的时候就没注意其中一些细节但重要的东西。此外，大脑理解一个东西需要一定的处理时间，人脑的处理速度很慢，神经冲动每秒传输速度不过百米，所以不能指望看到哪懂到哪。最后，我们可能因为思维定势的原因会从某个特定的角度去看一句话而忽略了从不同角度去理解的可能性。对于这类情况，仔仔细细地再多读两遍，多试着去理解两遍，往往会“哦！原来这样。”地恍然大悟。2.其中涉及到了你不懂的概念。这是技术性的不理解。这种情况就需要 Cross Reference。如果一句话中用到了你不懂的概念，那就去查，现在很多书都是电子书，直接搜索一下，或者，对于纸书，看一下书后面的

索引就行了。奇怪的是很多人看不懂也不分析一下为什么不懂，就直接放弃了。正如解决问题一样，问题卡住解决不了，第一时间要做的就是分析到底为什么解决不了，而不是直接求救。3.作者讲述的顺序不对，你接着往下看，也许看到后面就明白了前面的了。

杂项

7.如何在阅读之前就能获得对一本书质量的大致评估。在深入阅读之前能够迅速评估一本书的质量可以节省很多时间。基本上有几个线索：1.看作者。牛作者写的书一般都不错。2.看目录和简介。一份好的目录和简介能够透露这本书质量的相当一部分信息。目录结构是否清晰，是否直白（而不是装神弄鬼），都是衡量的线索。3.看 Amazon 上的评价，这里要注意的是，除了看整体打分之外，更要看打分最低的人是怎么说的，因为小众意见往往有可能来自那些真正懂行的人（除了来踢馆的），如果在打分最低的意见里面看不到真正有价值的反驳意见的话就相当肯定书是不错的了。4.看样章。Amaz

on 上一般都可以随机浏览一些章节的，表达是否清晰，论证是否严谨，内容是否深刻，基本是几页纸就能看出来的。

8.如何搜寻到好书。几个线索：1.同作者的著作。2.Amazon 相关推荐和主题相关的书列（类似豆瓣的豆列）。3.一本好的著作（或一份好的资料——不管是书还是网页）在参考资料里面重点提到的其他著作。4.有时对于一个主题，可以搜索到好心人总结的参考资源导引，那是最好不过的。

## （四）知识结构

自从建立了 TopLanguage 以来，发现在上面待的时间越来越多，与高手讨论问题是个粘性十足的事情，一方面，分享自己的认识是整理不成熟的想法的极好途径，另一方面，互相之间视角不同，所以往往自己忽视的地方会被别人发现。在讨论中不断精化既有的知识体系。以下这段基本上摘抄自（略有整理和添加）在 TopLanguage 上的发言：

抓住不变量

我喜欢把知识分为 essential 的和 non-essential 的。对于前者采取提前深入掌握牢靠的办法，对于后者采取待用到的时刻 RTM(Read the manual)方法（用本）。

如何区分 essential 和 non-essential 的知识想必绝大多数时候大家心里都有数，我举几个例子：对程序员来说，硬件体系结构是 essential 的，操作系统的一些重要的实现机制是 essential 的，主流编程范式（OO、FP）是为了满足什么需求出现的（出现是为了解决什么问题），是怎么解决的，自身又引入了哪些新的问题，从而适用哪些场景）。这些我认为都是 essential 的。我想补充一点的是，并不是说硬件体系结构就要了解到逻辑门、晶体管层面才行（其实要了解到这个层面代价也很小，一两本好书就行了），也并不是说就要通读《Computer Architecture:Quantitative Approach》才行。而是关键要了解那些重要的思想（很长时间不变的东西），而不

是很细的技术细节（易变的东西）。《Computer Systems:A Programmer’s Perspective》就是为此目的，针对程序员的需求总结出那些 essential knowledge 的好书。

再来说一下为什么需要预先牢靠掌握这些essential的知识：

根据 Joel Spolsky 同学的说法（原文 http://www.joelonsoftware.com/articles/LeakyAbstractions.html），编程语言技术是对底层设备的封装，然而封装总是会出现漏洞的，于是程序员被迫下到“下水道”当中去解决问题，一旦往下走，漂亮的 OO、N 层抽象就不复存在了，这时候不具备坚硬的底层知识就会无法解决问题。简而言之就是这些底层知识会无可避免的需要用到，既然肯定会被用到那还是预先掌握的好，否则一来用到的时候再查是来不及的，因为 essential 的知识也往往正是那些需要较长时间消化掌握的东西，不像 Ruby 的 mixin 或 closure 这种翻一下 manual 就能掌握的东西。（英

语也是这样的 essential knowledge——上次在 PyCN 上看到一个招 Python 开发人员的帖子将英语列为必备技能，却并不将自然语言处理列为必备技能，正是因为英语不是可以临阵磨枪的东西，而且作为知识的主要载体，任何时候都少不了它，如果不具备英语能力，这个就会成为个人知识结构的短板或瓶颈，而且由于需要长时间才能获得这项能力，所以这个瓶颈将持续很长时间存在。我们曾经在 TopLanguage 上讨论过如何花最少的时间掌握英语）另一方面，在问题解决当中，如果不具备必要的知识，是根本无从思考的，再好的分析能力也并不是每个问题都能分析出该用哪些知识然后再去查手册的，很多时候是在工具和问题之间比较，联想，试探性的拼凑来解决问题；这就使得一个好的既有知识基变得至关重要。（实际上以上这个是一个较大的话题，希望有一天我能够把它详细展开说清:)）

如果你不知道某个工具的存在，遇到问题的时候是很难想到需要使用这么样一个工具的，essential knowledge 就是使用最为广泛的工具，编程当中遇到某些问题

之后，如果缺乏底层知识，你甚至都不知道需要去补充哪些底层知识才能解决这个问题。

你必须首先熟悉你的工具，才能有效地使用它（须知工具的强是无敌的，但这一切得以“了解你的工具”为前提，甚至得以“了解目前可能有哪些工具适合你的问题”为前提）。一门语言，你必须了解它的适用场景，不适用场景（比如继承能解决你的问题不代表继承就是解决你的问题的最适合的方案，须知问题是一个复杂系统，解决方案总是常常引入新的问题）。你必须了解它支持的主要编程范式，此外你还必须了解它的 traps 和 pitfalls（缺陷和陷阱，如果不知道陷阱的存在，掉进去也不知道怎么掉的。）这些都是 essential knowledge，如果不事先掌握，指望用的时候查 manual，是很浪费时间的，而且正如第 2 点所说，正因为你不知道这些知识（如适用场景），从而用 sub-optimal 的方式使用了一门语言自己可能还不知道（最小白的例子是，如果你不知道语言支持 foreach，那么可能每次都要写一个冗长的循环，较常见的例子是不知道有很方便的库设施可以解决

手头的问题所以傻乎乎的自己写了一堆代码），因为人的评价标准常常是：只要解决了最醒目的问题并且引入的新问题尚能忍受，就行。注意，熟悉并非指熟悉所有细节，而是那些重要的，或者无法在需要用到的时候按需查找的知识。比如上面提到的：适用场景不适用场景，编程范式，主要语言特性，缺陷和陷阱。

当然，以上作为程序员的 essential knowledge 列表并不完备，关键是自己在学习新知识的时候带着第三只眼来敏锐地判断这个知识是否是不变量，或不易变的量，是否完全可以在用的时候查手册即可，还是需要提前掌握（一些判断方法在上文也有所提及）。并且学会在纷繁的知识中抽象出那些重要的，本质的，不变的东西。我在之前的 part 里面也提到我在学习新知识的时候常常问自己三个问题：该知识的（体系或层次）结构是什么、本质是什么、第一原则是什么。

另外还有一些我认为是 essential knowledge 的例子：分析问题解决问题的思维方法（这个东西很难读一

两本书就掌握，需要很长时间的锻炼和反思）、判断与决策的方法（生活中需要进行判断与决策的地方远远多于我们的想象），波普尔曾经说过：All Life is Problem-Solving。而判断与决策又是其中最常见的一类 Problem Solving。尽管生活中面临重大决策的时候并不多，但另一方面我们时时刻刻都在进行最重大的决策：如：决定自己的日常时间到底投入到什么地方去。如：你能想象有人宁可天天花时间剪报纸上的优惠券，却对于房价的 1%的优惠无动于衷吗？（《别做正常的傻瓜》、《Predictably Irrational》）如：你知道为什么当手头股票的股价不可抑止地滑向深渊时我们却一边揪着头发一边愣是不肯撤出吗？（是的，我们适应远古时代的心理机制根本不适应金融市场。）糟糕的判断与决策令我们的生活变得糟糕，这还不是最关键的，最关键的是我们从来不会去质疑自己的判断，而是总是能“找到”其他为自己辩护的理由(《错不在我( Mistakes were made,but not by me )》）又，现在是一个信息泛滥的时代，于是另一个问题也出现：如何在海洋中有效筛选好的信息，以及避

免被不好的信息左右我们的大脑（Critical Thinking）关于以上提到的几点我在豆瓣上有一个专门的豆列（“学会思考”），希望有一天我能够积累出足够多的认识对这个主题展开一些详细介绍。

最后分享一个学习小 Tip：

学习一个小领域的时候，时时把“最终能够写出一篇漂亮的 Survey”放在大脑中提醒自己，就能有助于在阅读和实践的时候有意无意地整理知识的结构、本质和重点，经过整理之后的知识理解更深刻，更不容易忘记，更容易被提取。

杨军（http://hi.baidu.com/yjpro）在 TopLanguage 上也曾分享了三篇非常棒的学习心得的文章，字字珠玑：

[1]有些事情做起来比想象中容易 https://groups.google.com/group/pongba/browse_frm/thread/9a459b6efe94985a/

[2]有关读书方法的一点想法 https://groups.google.com/group/pongba/browse_frm/thread/20a08b6201d88a98/

[3]一件事情如果你没有说清楚，十有八九不能做好 https://groups.google.com/group/pongba/browse_frm/thread/6f6140744ab95c72/

## 6、我在南大的七年

——跨进南大校门的第一天，我知道，我自由了。

父亲是个对新事物有强烈兴趣的人，村里第一台电视机是他自己组装的，当时全村人都跑过去看，电视机只能收到一个台，CCTV。座机电话是第一个装的。大哥大刚出现的时候，他也是第一个买来用的，那个时候的移动电话真是贵得离谱。

父亲告诉我的第二件最重要的事情是：遇到任何问题，找书去就行。他在自己的专业中完全是自学的。在不属于自己的专业中（后来买了电脑之后需要学习如何

架设公司网站，如何网上营销，如何进行电子财务管理，如何使用各种作图软件制图等等）也全都是靠买书自学。

为什么说到这两件事情，因为这是对我一生影响最重大的两个习惯。第一个习惯给了我学习新东西的强烈动机，有了热忱和兴趣，做事情就不觉得累，就自得其乐。第二个习惯则给了我学习任何新东西的方法——不会么？查书去。（当然，学习一门专业并不完全通过看书就行，但这毫无疑问是至关重要的一个途径。）

高三的时候，父亲买了电脑，我立时对这个神奇的事物产生了强烈的兴趣，每期的《电脑爱好者》和《电脑报》都会买来细细看，有时看到各种小工具、技巧还会摘抄下来，回去在自己家里的机器上捣鼓。那个时候我并不知道这样单纯的兴趣会把我引向一条专业的程序员道路。

高三时间变得越来越紧，分配给兴趣的时间越来越少，但兴趣的火花一直都没有熄灭。

跨进南大校门的第一天，我知道，我自由了。

这个自由并不是说我可以做任何事情了，而是我得到了一个重要的决策的自由权，即关于如何利用我的时间。

高考的时候我报了计算机系，但分数差了几分，失之交臂，被调到第二志愿专业——信息与计算科学。当时以为这个专业跟计算机相关的，结果发现是数学系，后来听不少同学提到都上了同样的当。

这里出现了一个歪打正着的事情：我本意并不是上数学系，如果当时知道这个专业是数学系，我可能就不会填报了。但正是因为这个错误，我在数学系好歹也受了一些数学基本功的训练（尽管这个训练的基础是大一上的不多的几节数学分析课，以及每次临考前宿舍哥们例行的“包夜”看书），回过头来看这个基本功在后来还是帮了不少的忙，甚至有一阵子我对数学本身到了很感兴趣的程度。不得不说，这段学习的经历是很锻炼抽象和逻辑思维的。另一方面，困难如数学都学了，对其他学科就不觉得难，不会望而却步。

这是我成长过程中的幸运之一。后面还会提到，还有好几次更大的幸运。

大一上学期很快过去，应该是在大一下学期的时候，学校要开一门 C++课程。我利用假期先把课本基本啃掉了，当时动机也很简单，先啃掉，就不用上课了嘛。

另一件事情是我经常喜欢去逛书店，看到侯捷的《深入浅出 MFC》上面很多人说这本书好，我当时也对 C++有一些基础认识和好感，所以就买下来啃了。一方面侯捷先生写的书的确图文并茂，深入浅出，有意思，另一方面理解一样复杂的东西是个智力挑战。所以看着看着倒是觉得兴致盎然。却不知就这么和 C++结下了不解之缘。

这是另一个歪打正着：为什么说是“歪打”呢？因为 MFC 的设计也并不能说就是 C++的 Best Practice，另一方面若是以用为本的话也未必就要把 MFC 的原理摸个透。所以搞不好现在看来我就不会细看这本书。为什么说是“正着”呢？因为理解一个费解的东西本身需

要长时间投入注意力，无形中练了理解能力和思维体力（专注），另一方面虽然 MFC 不是最佳设计，但理解里面的代码却加强了对 C++本身的认识，这是基本功；也加强了对 C++的兴趣，这是动力，后来这个动力驱使了我去看了大量的系统底层知识，从操作系统代码一直看到硬件体系结构。

大二发生了几件重要的事情：一是我在程序员上发表了第一篇技术文章，是剖析 Boost 源码的。我已经不记得什么时候、通过什么途径知道 Boost 这个库的了，总之是知道了，然后也是由于受到侯捷先生源码剖析的影响，也去看源代码，发现很难，越是难就越是觉得有趣，跟踪代码到临晨四点居然越看越精神了，后来火速写了一篇源码剖析。发给《程序员》杂志的技术主编孟岩先生，孟岩先生给了很大的鼓励，于是我很来劲。后来一鼓作气分析了 N 个库，写了一系列的 Boost 源码剖析的文章，在网上随处可以搜到这个系列。

这是第二个歪打正着，按理来说，研究语言技巧并

不是程序员最佳的时间投入方法。所以现在我可能不会去做这件事情，会认为有更好的时间投入途径。但当时就一头扎了进去。为什么说也是正着呢？因为虽然这也许不是最佳的投入时间的办法，但总归比什么都不专注要强得多，至少这么一深入，对语言的缺陷和陷阱有了更深刻的认识、也锻炼了对代码的亲切感、跟踪调试的耐心（是的，耐心，而不是技巧）。

所以后来我在博客上总结自己学习编程中走过的弯路，孟岩先生说到，是不是弯路，不是那么容易界定的。

的确，也许真的有更好的路，但事前真的很难判断哪条路是最优的，我们能做到的，是把一条路走透了、走深了，只要不是一条太不靠谱的路，深入的过程中总会有很多的收获。只要不是太顽固，善于反省，总有一天也会逐渐意识到越来越靠谱的路。

除了发表第一篇技术文章之外，大二我还用业余时间做了一些技术翻译，寒假里我坐在家里每天晚上翻译半章《Effective C++》，当然，后来我把译稿提交给出

版社的编辑时被告知文笔还显生硬。

同一时间，我继续啃 N 多 C++以及底层知识的书，一段时间我的书架上全是这类书，根本不像数学系的学生。非典那阵子，把饭钱都拿来买了书，为什么买得这么疯，也是因为受父亲的一个影响，他告诉我买书不用心疼，因为是长远投资，收益远远大于这点金钱投入。那段时间我边看边写一些代码玩，有模仿 Windows 核心编程的小程序，也有尝试并失败的小游戏，也有拿来对宿舍玩的游戏文件分析的工具，还有为上机考试写的库，总之玩得不亦乐乎；不像很多知名的程序员在学校里面就写了被广为使用的工具，那个时候我完全没有这个意识，也不知道什么是开源，自己自娱自乐而已，所以没有系统训练编码量和编码素养，比较盲目。

大二下半年还发生了一件重要的事情，我在 CSDN 上开了一个博客，开始写学习 C++和编程的过程中的一些总结。这个博客我一直写到今天，伴随了我整个 7 年的学习和成长，回过头去看就像时光机一样，能够看到

一路过来我都关注了些什么东西，是怎么想的，以及对一些事情的看法是怎么改变的。这些东西如果不记录下来，就会逐渐忘掉，也就无法参照过去的自己，对未来提供更好的借鉴了。所以我一直把记录当做一个很重要的工具。另外我也通过这个博客认识了很多朋友，得到了很多的帮助。

后来，学校提供了转系到软件学院的机会，我立即报名了。后来的两年在软件学院度过。但其实反正我也是自己安排时间，所以无甚区别。

大三大四发生了几件重要的事情：一是荣耀先生邀我合译《Imperfect C++》，我很乐意的接了下来，可没想到这本书比我想象得要密度大得多，六百页，而且排版也很密，我给自己安排了每天 6、7 页纸的量，大概花了半年多译完。中间有一段时间停滞，荣耀先生给我鼓劲，告诉我一个重要的方法：如果觉得做不下去了，就硬着头皮坚持做，然后就类似于麻木了，适应了，那种望而却步的感觉会逐渐自动退去。惊人的简单，但事实

就是如此，硬着头皮，过了那个情绪上最艰难的时候，也就适应了。这本书译完之后，还是有不少的收获，但我总觉得对性格上的磨练才是最有价值的收获。

二是我开始看英文版的书。之前，由于高中不靠谱的英语教育的原因，我恨死了英语，大二的校内四级课程还挂了科，直到大四才补考。但对技术本身的热爱压过了对英语的反感，我还是硬把一整本影印版啃下来了，而且津津有味，这本书就是 Jeffrey Richter 的《Applied.NET Framework Programming》。这个事情的重要性在于，后来我就不再反感和恐惧英语了，这是其一，其二是我开始意识到英文世界的技术资料有多么丰富，所以虽然本身看上去不是一个太起眼的事件，但却是我获取信息方式的一个 Tipping Point，一旦熟练掌握了语言这个平台，背后就是一扇大门，通向一个海量的信息源，后来我的信息获取绝大多数便来自于英文，其中尤数 wikipedia 和英文版的书为多。另外还有一个收益后面会提到。

大四快毕业的时候又发生了一件事情，微软的 Eric Jiang 通过我的博客找到我，推荐我去微软面试，我随随便便就把粗糙的简历给发过去了，差点因为简历太粗糙被 HR 直接过滤掉。远程电话面了两轮，远程 Coding 一轮，然后记得就是飞到北京面试，住在北航招待所。北京的面试又面了好几轮，有考察底层知识的、有考察 C/C++的、.Net 的，还有考察算法的，编码素养的。总之就是公认的基本功考察。最终我还是没能通过面试。个人自己后来总结的结论是算法基本功太差，连什么是动态规划都不知道，编码素养也不够。这部分也是因为本科的学习方法太业余，什么好玩干什么，倒不是说兴趣驱动不好，只是缺乏系统的规划，不清楚也不关心这个领域的蓝图，也弄不清什么是重点。后来在读研的时候恶补了一把算法，好歹弄清了一些基本的概念和思考方法。编码素养的问题也是到了读研的时候才开始思考和学习，现在仍在学习。

另外，在本科阶段，其实我也浪费了很多时间，事实上，是只花了很小一部分时间来学习。之所以还多少

学了点东西，完全是仰赖了专注的习惯。而这个专注的习惯其实又是从小受父亲耳濡目染的，父亲会花一整天揣摩一个问题，父亲跟我说过他以前组装电视机时的故事——一切都似乎组装正确，但电视机就是不工作。他苦思冥想，不得其解，当晚，半夜从睡梦中醒来，想到了问题的症结所在。所以，我在啃一些底层知识时如果弄不懂，也会一遍遍读，然后用走路吃饭坐车的时间在脑子里一遍遍去琢磨。我有很多重要的习惯受到父亲的影响，这些习惯自己一般觉察不到，但却默默影响了平时的一点一滴的时间分配和学习轨迹，这些习惯从纸上很难学到，但耳濡目染却会自然而然地学会。

每当有人觉得我本科就做了不少事情的时候，我就会说其实我本科真的浪费了很多时间，而另一方面，这也说明，要掌握一门专业知识，其实每天一点时间，专注、积累和持之以恒也就够了。后来研究生阶段才算真正开始惜时了，于是经历了两年密度很高的学习和思考，心智才成熟了不少。

大四的时候，和很多人一样，我也考研，因为一来也很茫然，二来也希望能够继续有一个宽松的环境继续沉浸在自己的兴趣中。但四年来我都是自己安排时间，逃掉了无数的课，已经对模式化的做题考试产生了抵触，所以考研的复习也没怎么认真准备，那年考研的数学题又偏难，一下慌了神，结果居然把一整页题压在稿纸下忘了做了，心理准备有多不充分可见一斑。考完数学我很沮丧，那么大分值的题目没做，数学肯定过不了了，接下来的专业课就没去考了。后来想想其实还是应该去考一考，多少能为下一年积攒经验。

后来就工作了，没去成微软，经同学张振推荐，就去了南京西门子。心里的打算还是边工作边考研，为什么考研，动机也简单，我心理还没准备好，本科只顾着埋头学好玩的，也不看路，不知道自己想要什么样的工作，想做什么样的事情。去西门子之后更加觉得如此，觉得效率很低，做的事情也并不是我乐意的，每天还要在班车上浪费两个小时，于是没过多久就辞掉了工作。打算复习考研。那个时候大概还有半年多的时间才到考

研，所以我中途不紧不慢地又翻译了《Exceptional C++Style》，占用了不少时间，到最后时间很紧了，就剩两三个月，我才开始认起真来，回想起来这是糟糕的时间管理。结果我不得不作了最坏的打算：顶多调剂去软件学院读研（我报的是计算机系），考虑到我反正是自己安排时间，差别应该不大。幸运的是，最终一分不差地过了线，算是蹭到了计算机系里。虽然如此，还是觉得这种惊险不要发生的好，以后或者其他事情上就不会有这么幸运了，及早准备总是很重要的。

读研期间的两年半，是我自己觉得心智年龄成长最迅速的一段时间。这里也有几个很幸运的事情。一个事情是我的导师陈家骏先生给了我很大的自主，于是我得以有时间安排一些重要的学习，这段时间对我来说很重要，我学习和思考了很多东西，为个人以后的发展作了很多准备，倒是没帮导师做什么事情。所以，硕士毕业离开的时候是既感激也愧疚。

另一个事情是认识同实验室的师兄陈怀兴，严格来

说是他先来找我聊天，可见那个时候我仍然还是没有意识到与人交流的重要性的，后来，建立了 TopLanguage 讨论组之后越发意识到与他人交流的重要性，也开始主动寻找和参与交流，希望以后自己也能组织交流。陈怀兴对算法很有造诣，也是 TopCoder 上的常客和牛人，那个时候我也正在为以后的工作面试准备一些算法基础，所以经常找他讨论，获益很多。有一句话说：看一个人，只要看他读的书和见的人。还是很有道理的，这两者是一个人成长中最有价值的信息来源。

研一下半年，女朋友找工作的时候需要用到营销方面的知识，于是我去替她找书，偶遇《影响力》这本书，这本书打开了我的视野，让我开始关注一个很有价值的领域：我们如何思考，如何正确地思考。这个领域有很多有意思和有价值的书，我利用近一年的时间，陆陆续续看了近 40 本相关的书( 我把这些书整理了之后以豆列的形式放在豆瓣上 )，对思维的特点和缺陷，以及如何思考有了很多的了解，这些知识后来很大程度上使我更清晰地认识自己，和自己在学习和生活中面临的各种问题。

也是研一下半年，我建立了一个 Google Groups，起名 TopLanguage ,一开始的时候是因为平常没人讨论问题，憋得难受，希望有人能够说两句，无心插柳柳成荫，后来这个讨论组的交流越来越多，如今已经近两年，组内成员超过了 4,000 人，两年里我也从中收益颇多，其中最大的收益有两个：一是和人讨论能够激发自己进一步的思考，也促使自己更清晰地表述自己的观点或问题。倒不是说别人就一定告诉你什么新东西，而是讨论对你自己的思维的刺激。二是交流中认识了不少朋友，后来快毕业的时候也受帮助颇多。我一直把 TopLanguage 的创建看作研究生阶段做得最有意义的事情之一。

此外，我有意识地提前准备了英语，因为我相信如果想要去好的外企 ,口语不过关很可能成为一块短板( 当然，英语作为承载最多技术知识的平台语言还有更大的价值 ), 包括阅读、书写和口语。我想了一个方案，可以不用额外花时间来学习英语：阅读的训练蕴含在平时的英文技术资料的阅读中，尽量读英文的，一来英文资料更一手和全面，二来也顺便练阅读。书写的训练蕴含在

去国外邮件列表发技术贴和自己写的英文博客文章中。口语的训练则蕴含在平时的娱乐中——美剧，有一个暑假我几乎天天开着 Friends 睡觉，另外学校有国外过来的团队演讲我不再错过，而是主动参加，有一次还带他们出去逛南京，说了一天英语，回头在路上听中文都像英文。虽然和外国友人交流的次数不多，但似乎对口语感觉的提高还挺大。后来在微软亚洲研究院的面试最后一轮就是英语的，而且是偏技术的，好在提前准备了，所以毕竟还是顺利地表达出了想表达的意思。

去微软亚洲研究院面试，是因为幸运地认识了微软亚洲研究院技术创新组项目主管邹欣先生。邹欣先生和他组织的团队在那段时间写了《编程之美》，书中有很多很有意思的题目，而我那段时间恰和陈怀兴讨论算法，在讨论组上也组织了专题的讨论，有了一点粗浅的思考，于是和邹欣先生邮件交流，由于对邹欣先生的技术创新组做的事情很有兴趣，所以找工作的时候便向他毛遂自荐。

承蒙邹欣先生推荐，时隔三年，我再次获得了去微软面试的机会。这一次，由于研究生期间作了一些长远准备，所以心里有底了很多，也就比较冷静了，由于当年知识体系的漏洞被我花功夫补了补，所以面试比较顺利。面试的时候邹欣先生更为详细地介绍了技术创新组的工作，我更加感兴趣了，所以尽管已经有另外几个也不错的选择，但心里还是迅速地做了决定。大约一周后，HR 通知 Offer，我毫不犹豫就接受了。

我想，虽然有很多人本科就明白自己想做什么，我多花了两年多，总还不算太晚。

## 第二篇 思维改变生活

### 7、逃出你的肖申克

#### （一）为什么一定要亲身经历了之后才能明白？

前言 :《逃出你的肖申克》这个题目我早就放在心中，一直想写一写，但一直没有找到恰当的切入点。上次一个偶然的时候，发现可以以对一些人们常常放在嘴边的

俗语进行解释为入口，以一年多来学习的关于思维的知识为基础，展开来说一些也许有用的东西，也刚好是对学过和思考过的东西的总结和整理，如果你在看过上次发的“如何清晰地思考——知识结构图”之后发现要读的书太多，无法下手的话，不妨将这个系列作为一个更详细的引路图，注意文中的各个超链接，他们都指向有价值的资料，引用出去的资料的价值远远大于这篇文章本身。

——每个人心中都有一座肖申克

（１)为什么我们常说很多时候一定要亲身经历了之后才能明白？

1.切身体验。亲身经历一个负性事件带来的情绪记忆要比看着或听说别人遭受一个同样的事件所感受到的强烈得多，形成的负性条件反射也远远更持久。我们一定程度上的确能够感同身受，但心理学实验同样也表明，自己是无法从强度上真正感同身受别人的痛苦的，《Mistakes Were Made(But not by Me)》p192 举了一个极

聪明的实验：

即便两者实际上是一样程度的，我们自己所感受到的痛苦也总是比观望别人的痛苦要强烈得多。一个古老的笑话是这样说的：别人断了条腿没啥大不了的，我们断了根指甲就要大呼小叫了。这个笑话碰巧生动地描述了我们大脑的神经系统的工作方式。英国的神经科学家们曾经做了这么一个实验：将人们配对进行“以牙还牙，以眼还眼”实验，每对被试食指上都夹着一个夹子，实验者通过这个夹子往其中一个人的食指施加一定的压力，然后让他施加同样的压力给他的同伴。结果是没有一个人能够做到公平，尽管他们很努力地试图做到，然而他们总是施加更大的力道给他们的同伴——他们心里认为这正是他们所受到的力道。研究人员认为这一效应是我们的神经处理机制的自然副产品。这个实验有助于解释一个我们常常注意到的现象：两个人你打我一拳，我还你一拳，结果很快拳头的力道就会越来越重，从打闹变成了真正的打架。每一方都认为自己是在公平地还以颜色，而实际上他们却并不是以牙还牙以眼还眼，而

是以眼还牙，结果再弹回来的时候那家伙就想卸你的腿了。（翻译）

好友徐宥最近经历了 7 个小时的肾结石，在博客中写道：

有句话叫感同身受。我没有体验过肾结石的那种痛苦前，只是道义上支持三鹿宝宝的维权；现在，我真心的支持三鹿宝宝维权行动。我甚至很想折腾一下那些往奶粉里面加三聚氰胺的人，那些知情封锁几个月的人和那些不许家长维权的人。有生之年，得给那些害人之人，一人冲杯三聚氰胺奶粉，让他们”感同身受”一下这种绞痛，认识一下自己干的是不是人事。

2.别人口中的故事。别人口中的故事也许只是事情的一个方面，难免受到他们自己观念的影响而产生偏见，我们每个人都带着有色眼镜看待这个世界，客观且全面的描述一个事情极少有人能做到。别人的故事也许只是他们的想法，你自己亲身经历同样的事情也许完全又是另一种想法了。

3.为什么。别人在告诉你一个道理的时候往往只能告诉你怎么(how)做，而难以说清为什么(why)要这么做，遑论“为什么一定(have to)要这么做”了（因为他们自己也不一定能说清）。在没有听到逻辑严密、无法辩驳的证据之前，你很难说服自己 A 选项优于 B 选项，直到最终自己在某一条路上撞了南墙才肯死心。

4.世界是复杂的。更何况，很多时候人们根本无法确切地向你保证 A 选项一定优于 B 选项：比如，好好学习并不一定会有好的前程；不好好学习也并不一定以后就一塌糊涂。吸烟不一定短命，不吸烟也不一定长寿。坚持到底不一定就胜利（甚至有可能万劫不复），而不坚持到底也不一定就失败（学会放弃也是很重要的）。这是一个复杂的世界，各种错综复杂的因素互相影响，用单一因果来解释事件几乎总是不恰当的，唯一能够靠谱地搞清因素 X 和因素 Y 之间的关系的方式就是通过随机控制实验。

5.未来是不确定的。人类天生有一种寻求确定性的

需要，以及控制周遭的小世界的需求。我们总是希望听到“你只要这样这样，以后就一定能够那样那样”这类令人窝心的话。然而与我们的控制错觉相反，这个世界有太多因素是不确定的，除了自己的因素比较可控之外，外界的机遇因素几乎完全不是能够控制或预测的。我们最多只能做好头脑准备，尽量不错失机遇。也正因此，你几乎永远也听不到足够有说服力的证据来告诉你“你只要...，就一定能够...”，因为成功并不是仅取决于个人因素的。个人因素往往只是成功的一个既非充分又非必要的条件，所谓谋事在人，成事在天；但无需悲观，因为毫无疑问，改善个人因素的确能够大大增加成功的几率。

6.别人的道理，自己的事情。我们常常说类似“你说的没错，但是并不适用于我这里的情况”这样的话，自己的事情和别人的事情往往总是有着这样或那样的微小或巨大的差异，如果我们先入为主地不想听取别人的意见，就很容易自己说服（欺骗）自己说“情况不一样，所以道理不适用”（而实际上到底哪些情况不一样，为什

么道理不适用，是不是真的不适用，我们根本就不去深究了）。另外，朋友给出的道理或故事总是跟他当初经历的情境细节有着千丝万缕的联系，你记下了朋友的道理和故事，同时也就将这个道理和他当时经历的情境线索给挂钩起来了，于是当类似的情境发生的时候，你的记忆系统就能够根据情境线索提取出朋友当时说给你听的那些道理（《找寻逝去的自我》）；然而，这种记忆提取机制同时也有他的弱点，那就是当你经历的情境跟朋友当初经历的情境相似性不足（尽管抽象到本质上可能是一回事）的话，你就不会想起他曾经说的那些道理。这就是很多时候我们发现自己道理是听了一堆，结果自己生活中却不会用的原因。而所谓的能够“活学活用”，就是那些善于抓住知识本质，触类旁通，将道理外推到表面不相似但本质一样的问题领域之中的人，对此《Psychology of Problem Solving》的第 11 章举了这样一个例子：

先让被试（皆为大学生）阅读一段军事材料，这个材料是说一小撮军队如何通过同时从几个不同方向小规

模攻击来击溃一个防守严实的军事堡垒的。事实上这个例子的本质是对一个点的同时的弱攻击能够集聚成强大的力量。然后被试被要求解决一个问题：一个医生想要用 X 射线杀死一个恶性肿瘤，这个肿瘤只可以通过高强度的 X 射线杀死，然而那样的话就会伤及周围的良好组织。医生应该怎么办呢？在没有给出先前的军队的例子的被试中只有 10%想到答案，这是控制基线。然后，在先前学习了军队例子的被试中，这个比例也仅仅只增加到 30%，也就是说只有额外 20%的人“自动”地将知识进行了转移（自己就能触类旁通）。最后一组是在提醒之下做的，达到了 75%，即比“自动”转移组增加了 45%之多（需要别人提醒）。这个例子说明，知识的表象细节会迷惑我们的眼睛，阻碍我们对知识的转移运用，在这个例子中，两个问题领域表面上是不相似的，但本质上是一样的。然而就是因为表面上不相似，而我们的记忆提取又是很大程度上依赖于一些表象上的线索来提取的，因此这些表面不相似性便阻碍了我们在问题之间进行的类比，阻碍了我们将在一个情境下掌握的道理运用

到另一个情境下。

我自己就有这样的体会，我在学习专业知识的时候经常使用 Google，遇到知识性问题第一反应就是上 Google，或者上 Wikipedia。然而，在实际生活当中遇到一些生活问题的时候，往往第一时间想到的却不是 Google，而是之前解决生活问题的时候建立起来的习惯（比如询问身边的朋友，或者干脆放一边不管）（虽然我曾经总结并告诉自己说“遇到任何知识性问题，第一时间问 Google”），怎么会这样呢？生活问题难道不也是问题？难道不也应该联想到对待专业问题的方法——Google 之吗？可是我们的记忆系统的特点决定了不是这样的，生活问题就是生活问题，大脑会第一时间将我们之前怎样解决生活问题的方式提取出来，这个优先级要远远高于一个更一般的策略——Google，只有当没有特定策略的时候，大脑才会退而求其次寻求一般性策略。

还有更生动的：有一次在豆瓣上看到某人日记里面提到一个讲座，后面写了一些感想，但讲座的链接没有

给出，于是我第一反应就是留言问他要链接，但是实际上呢？只要把讲座的关键字扔到 Google 上就行了。我可是 Google 的重度使用者啊，怎么会忘了这个呢？！

事实上，我们在解决问题的时候一般有两个思维步骤：一是根据问题情境线索从记忆系统当中提取以往成功过的方案（沿袭类似情境下用过的可行方法，这个也被称为 mental shortcut），然后评估该方案是否已经能够解决当前问题了，如果能够，就中止记忆搜寻（这个也被称为 satisficing 原则），在刚才提到的日记问题中，留言询问作者是我在成为 Google 重度使用者之前建立起来的、针对这类情境的特定习惯，甚至也可以说是我们每个人的第一习惯（如果一个人详细说到某件事情，他肯定知道个中究竟），而且这个方法的确满足可行原则，因此，在这样的思维方式下，我不假思索地就沿袭了旧习惯，而没有成功地将在另一个问题领域建立的更好的方法推广到这个问题中来。如前面所说的，如果这个日记是不允许留言的，并且我不认识这个作者，我可能就会立即想到去 Google 了。

如何解决这个“知识经验跨情境转移失败”的问题？除了多多反省观察自己之外，在面对问题的时候多抽象其本质也是一个有力的办法，因为前面提到，正是表面不相似性阻碍了知识的迁移运用，我们常说有些人善于看到事物的本质，这样的人往往就是那些聪明人，因为他们更能够举一反三，将一个地方领悟的道理推广到另一个看上去很不一样的地方。

7.认知失调与自我辩护。如果我们在听到别人的道理之前已经有了一个心理上的倾向，那么即便别人给出一个有一定说服力的理由，根据认知失调理论，我们也会竭力为自己辩护；又由于世界是复杂的，所以我们几乎总是能够找到辩护的借口——“上次报纸上说一个英国老太太每天必吸一支烟，活了一百多岁呢。”（《Mistakes were made(but not by me)》）

8.失败即成功。有时候，我们的确需要在撞南墙的过程中总结出经验教训（特别是对于尚未有人走过的路），并到达成功的彼岸。从信息收集者的角度来说，世

界上没有成功或失败，失败的事情中揭露出来的信息一点也不比成功事件中的信息少，或许往往还能得到更多的东西。

9.情绪对照。经历了失败之后，我们在做“正确”的事情的时候便会觉得更理直气壮。如果没有经历失败后的糟糕记忆，我们就算理性地认识到目前的做法是更合适的，也很难从情绪上强烈地感受到这么做的“正确感”。

10.天性。我们有很多根植在大脑中的进化选择出来的天性（《Mean Genes》，《进化心理学》，《Predictably Irrational》，《别做正常的傻瓜》，《摇摆(Sway)：难以抗拒的非理性诱惑》，Behavioral Economics）。在判断与决策时这些天性的优先级总是最高的。然而，由于这些天性是在远古社会选择适应的，并不适应短短几百年我们才迈入的现代社会，所以我们总是听到内心两个声音吵架。比如我们的天性是目光短浅，只看到眼前利益（也许这对物质匮乏的远古社会是适应的）。所以即便有

时候别人说服我们应该往长远考虑一些，他自己就曾经吃过只看眼前的亏，然而你的内心一个声音仍然在高叫着“管他呢！”。

如果在你没有很多钱的时候，有人告诉你，钱多的人并不更加幸福；钱与幸福感几乎不相关。你会相信吗？就算他拿出非常严谨、权威、科学的心理学研究结果(《撬动幸福》)，也许你没法反驳，但你内心仍然还会有另一个声音在高喊：“管他的，还是让我先发了财再来担心这个问题吧”，我们似乎有两个大脑，一个理性区域（很可能定位于进化史上较晚近出现的新皮层（neocortex），这个皮层被认为是高级认知推理能力的所在），和一个原始区域。这两个区域并不总是合作无间的，很多时候我们面临两难决策的时候仿佛内心有两个声音在争吵，就是它们在吵架呢——理性的大脑告诉你应该这么做，但是直觉却大喊应该那么做。到底怎么做呢？最终只有一个办法能够弄清楚——实验。但如果别人实验了之后告诉你幸福与钱并不想干，你会怎么看？在无可辩驳的证据面前你的理性大脑是被说服了，但是你的另一个大脑

却根本不买帐，它的工作机制是：没钱就用焦虑来驱动你，让你寝食不安，等你挣到钱了，就给你短暂的满足感，之后让你迅速习惯于这点满足感，迫使你把目光投向更多的钱（进一步用焦虑来驱使你去赚更多的钱）。为什么你自己的大脑会跟你过不去呢？为什么它总是不让你开心呢？很简单，如果你总是感到满足的话，就不会去进取，在一个残酷的优胜劣汰的竞争环境中，你的这种不思进取的基因很快就会被淘汰。经过了漫长的筛选，如今剩下来的基因几乎都是挣钱机器（《Mean Genes》）。

贪婪、嫉妒、短视、投机，这些天性也许在远古社会曾经成功地让我们的祖先占取了生存繁殖优势（并不像某些宗教书籍说的这些是所谓“原罪”，它们只不过是适应于特定社会背景的进化心理机制、判断的与决策的 heuristics 而已），然而现代社会的情境已经改变，分享、合作、交流、长远、诚实，这些才是在现代社会获得成就的方法，但由于我们的天性还没为这个社会准备好（进化是需要时间的，由于人类进入现代社会的时间太短，

才区区数百年，和漫长的几十万年想必只好比一瞬，进化的齿轮——需要经过一代代繁殖淘汰——根本还没来得及跟上，所以我们仍然在使用着适应远古社会的心理和生理机制），因此，我们常常需要用理性的声音去说服内心的原始人。幸运的是，我们可以，前提是我们必须首先了解自身。

11.习惯。我们常说，“说起来容易，做起来难”。习惯的力量远远大于我们的想象，很多时候我们都会有这样的体会：听到一番很有道理的话，但没过几天，发现自己又变回原来的老样子了。甚至于自己在一次困境中领悟出了一些很重要的道理，决定在下次遇到类似情况的时候不再犯以前犯的错误，然而，当真正遇到下一次情况的时候发现自己无意识地又犯了同样的错误，谁说人不会两次踏入同一条河流？

也许，对付我们强大的习惯的最佳办法是将自己认为正确的（不管是自己经过困难或失败而领悟的，还是看到书上或听到别人说的）写下来，并常常拿出来翻看。

事实上，我的经验是，在写下来的时候我们的大脑会进入到理性分析模块，进一步检验和推理那些道理，我们越是对一个道理审视的详细、深入、全面，大脑中留下的印象深刻，从记忆加工的角度来说，这叫深度加工，带来的结果就是该记忆与更多的提取线索相关联，于是便能够在更多的场景下被唤起（而不是被以往的习惯直接覆盖）(《找寻逝去的自我》)。

（2）亲身经历了就一定明白吗？

1.很傻很天真的条件反射。条件反射是一个太傻太天真的系统——我们碰了一鼻子灰之后往往就会选择放弃自己的做法。然而其实撞了南墙并不就一定意味着做法不对，可能只是时运不济。没有得到好的结果并不代表你的过程就错了。有人勤奋学习却发现中学同学撞大运成暴发户了或者找了个好老公，于是得到悲观结论说学习没啥用。可以想见，如果他因此就改变做法，整天等着机遇降临，同样也可能会一败涂地。同样，结果正确也并不代表方法就一定正确。在金融市场里面这样的

情况尤其显著，让大猩猩来选择股票也有运气好的时候（《黑天鹅》），如果大势利好则更是如此。但难道这就代表所用的方法是正确的了吗？客观的做法是：看重过程，而不是看重单次的结果——因为再好的过程也可能会偶尔失利，但从长远来统计，好的过程总体上必然导致更好的结果。（《别做正常的傻瓜》第 12 章："抓住老鼠的一定是好猫吗—结果偏见"对此有介绍。）

2.认知偏差。我们有着各种各样系统的认知偏见：我们经常对事物作出错误的解释和归因（即便自己是亲历者），有时甚至反而是"当局者迷，旁观者清"。李笑来老师曾经讲了他亲身经历的一个有趣的故事：

我的教练臂围是 43 厘米，几乎和常人的大腿一般粗。有一次他告诉我他练习的诀窍——握哑铃的时候，一定要把手掌边缘贴到靠体侧的那一个哑铃片上。这样的话，哑铃的另外一端将自然地向外翻转一个很小的角度，臂屈伸的时候恰好可以使肌肉获得最大的曲张刺激。然后他得意而灿烂地笑着说，"多简单啊！"而我却突然

明白了另外一件事：他的成功其实并不是来自于这个所谓“简单而神秘的技巧”，因为我认识另外一个健身教练臂围 45 厘米，我从来没看到那个 45 厘米臂围的教练用这种方法握哑铃。但他们都成功了。（摘自《心智力量的差异》）

事实上，很多成功者自己的总结都不靠谱，就是因为他们自己也难以对自己成功的原因进行正确的归因，比如我们都有这样一种倾向：将失败归因于外界因素，将成功归因于自己的能耐。（心理学把这个称为自我服务偏差）。此外人类的思维有着林林种种各种各样的认知偏差，不管是成功者还是失败者，只要没有对人类思维和心理机制的基本了解，都难逃认知偏差的影响。作为一个开始：你也可以从《How we know what isn’t so》开始阅读一些经典的思维谬误，或者阅读元凯宁在 TopLanguage 上发的这篇科普：《关于“不了解的领域”，兼谈 Critical Thinking》

3.情绪系统。我们之所以强烈地依赖于需要亲身体

验一个负性事件来学习，是因为我们平常的决策与判断强烈地依赖于情绪系统的输出，如果一个事情“感觉上没错”，我们无论有多好的理由也很难说服自己不去做它，如果一个事情“感觉上不对”，则无论有多好的理由也很难说服自己去做它。这种对情绪系统的强烈依赖使得理性的证据在强烈的情绪面前显得孱弱。事实上，我们的直觉的确有不少时候是很灵的（《Blink》、《Gut Feelings》），但也有不少时候是失灵的（前文已经有例子了），正确的做法不是一概而论地听取直觉的意见或者一概而论地不听取，而是将它当成一个启发式的判断，然后利用自己的理性大脑对其进行进一步的客观的、逻辑的检验（Critical Thinking）。我们是一定程度上能够驾驭情绪系统的，情绪系统毕竟只是我们的进化工具箱中的决策系统之一，而不是全部。另外始终别忘了情绪系统只是一个比较粗糙的判断决策系统，并且它很多时候是为了适应远古社会而非现代社会的（《Mean Genes》）。

（3）不需要经历也能明白——理性的力量

我们对于事物的思考深度常常是不够的，在浅层次的思考了之后，如果我们的情绪系统或者直觉已经给出了倾向，那么很少有人会继续深入地思考，而开始转向着手行动。这种匆忙的态度往往是失败的起源，在碰壁了之后，我们被动地“让事实告诉了我们”某方案是行不通的，让事实替代我们进行了思考和推理，我们从失败当中获得了信息，知道了为什么之前的方法是不恰当的，这就是一些时候我们认为要亲身经历才能明白的原因。然而，这并不意味着任何时候我们都只能“做了之后才发现…”，人类最强大的能力就是社会学习——

普通人从自己的错误中学习，聪明人从别人的错误中学习。

人类最强大的另一个能力则是归纳和推理——

A few lines of reasoning can change the way we see the world.(via http://www.xiaolai.net/index.php/archives/193.html)

我们可以仔细地，理性地思考、权衡各个选择的利

弊，而不仅仅满足于情绪上的判断。假设我们面临两个选择，A 和 B，我们可以结合别人的经历，利用自己的推理能力，去分别推断 A 或 B 选项带来的各种各样可能的利弊，对于其中不确定的因素我们或者可以进一步从别人那里收集更多的信息来使得判断更靠谱，或者可以对风险的上下界进行一些估计，总之，我们尽量去让我们大脑中假想的角色去经历失败——我们通过推理发现某条路行不通，就避免了现实中去碰一鼻子灰。

我们在大脑中走得越远，在现实中就走得越稳。我们在大脑中失败的次数越多，在现实中失败的次数就越少。

直到实在没法在事先知道答案（你所面临的问题是任何前人都没有探索过的）,才必须亲自探险 ,那个时候，我们就不再是在重复别人走过的老路，而是探索者，创新者，因为我们站在了别人的肩膀上。

（4）尾

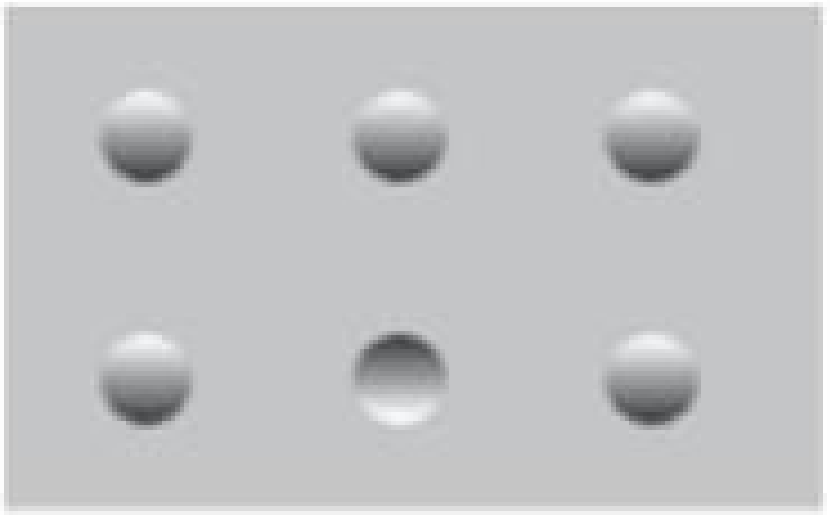
事实上，现代社会人最重要的能力之一就是能否从

别人的错误中学习，往往是这类人能够迅速走在别人的前面，在别人跌倒的地方跳过去。如果我们事必躬亲，那么历史绝对不会进步，我们只会每个人从生下来开始都将别人犯过的错误再犯一遍，将别人趟过的泥潭再趟一遍，阳光底下就真的没有新鲜事了，历史就真的永远重复他自己了。然而历史告诉我们绝非如此，虽然很多人都会甚至需要自己犯一犯某些错误，但同样也有很多人能够在别人的错误中学习。

这是一个信息社会，所有人的经验教训，所有人的知识以前所未有的速度，以互联网为媒介传播开来，不管我们关注什么主题，总能迅速找到一堆书，论坛，网页，然而能否从中获取知识，避免做别人做过的俯卧撑，就看你有没有一双能够辨识的眼睛，和善于思考的心智（见《如何清晰地思考》http://blog.csdn.net/pongba/article/details/3549560），否则在海量的信息面前就永远只能是来打酱油的。

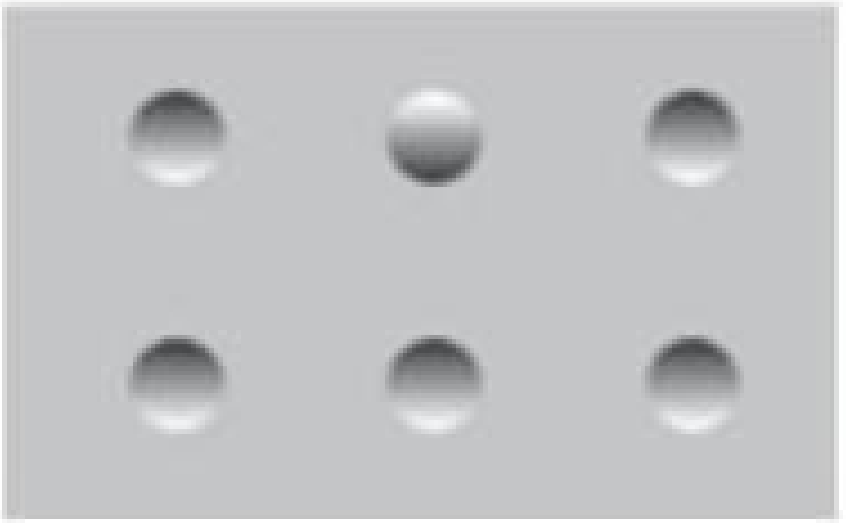
## （二）仁者见仁智者见智？从视觉错觉到偏见

《Making Up the Mind》上讲了这么一个简单但深刻的实验：



我们看到这张图片的第一反应是：5 个凸的按钮，1 个凹的按钮。

现在仅仅将图片上下颠倒一下：



在我们眼中立即就变成了：1 个凸的按钮，5 个凹的按钮。

为什么同一副图片，仅仅是上下颠倒一下，我们就对其作出了完全不同的解释呢？

我们知道，视觉图像要到达大脑，首先要在视网膜上成像（视网膜上密密麻麻地排布着感光细胞），刺激感光细胞形成的神经电冲动然后经过一系列复杂的神经通路到达视觉皮层。但后续的繁杂步骤其实都是对视网膜上成的像的处理。这里，对我们的讨论而言视网膜不妨可以看作一张感光胶片，重点在于视网膜上的像完全是一张二维图片。大脑从图像中提取出来的任何信息都以这张二维图片为原始素材。

那么，究竟大脑是怎么从二维图片中看出（推导出）三维的？

其中一个重要的工作就是判定深度。前面的两张图片完全是二维图片，在我们的视网膜上也是二维的。然而大脑却能够从中理解出三维出来，大脑能够判断出一

个按钮是“凹”的还是“凸”的。这是怎么办到的？

很简单，假设环境中有光源，并且光源来自上方，那么凸的物体会使其下部出现阴影，凹陷的物体则会在上部出现阴影。于是，图中按钮的下半部出现阴影就意味着按钮是凸的，按钮的上半部出现阴影则代表按钮是凹的。

然而，别忘了，大脑的这个推理成立必须有一个前提，即光线从上方照下来，如果光线从下方照下来的话，一切就反过来了，凸的物体将会使其上部呈现阴影，凹的物体将会使其下部呈现阴影。因此同样的一副图片如果假设光线从下方照耀的话，原来看成凸的物体就应该看成凹的，原来看成凹的就应该看成凸的。

那么，回到我们的第一副图片，你能够看着第一副图片并假想光线从下方照下来，进而把原来凸的按钮看成凹的吗？事实证明这很难，但我们可以做一个等价的事情——将图片上下颠倒一下：考虑到我们总是假设光线从上方照耀以及按钮的上下对称性，颠倒原图就相当

于对原来的图片而言假设光线从“下方”照上去了。

我们发现（上文第二张图），一旦颠倒图片之后，果然凹凸就换位了。

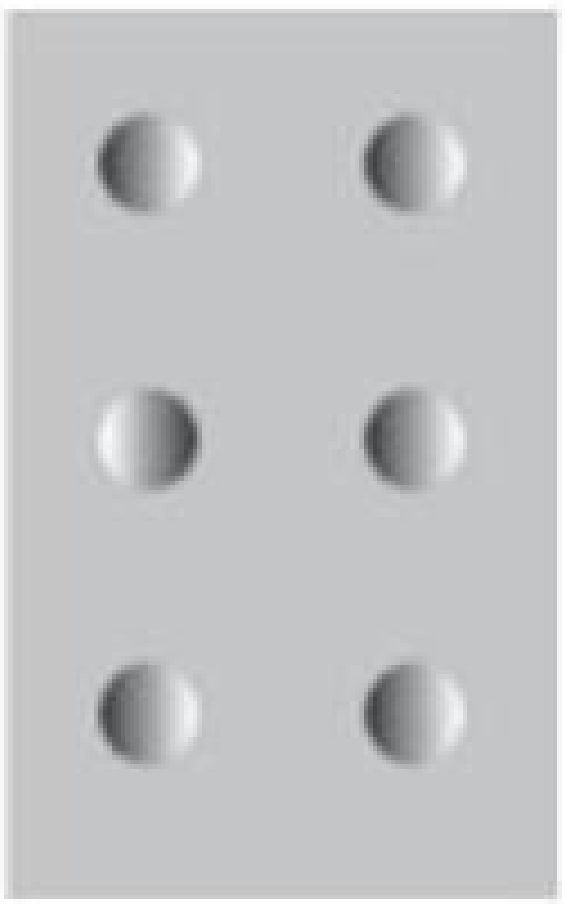
这就是说，同一副图片其实有两种（乃至更多）可能的解释，取决于你的大脑到底假定光照来自下方还是上方。但为什么我们看上面两幅图片却不会出现“二义性”的错觉呢？因为在我们生存的环境中始终就有这么一个巨大的来自上方的光源——太阳，漫长的进化已经在我们的神经回路中刻下了“光源来自上方”这样一个强大的假设，所以虽然第一副图片本该完全有两种解释，我们还是不可避免地只看到其中的一种解释，即假设光线来自上方的解释，即使卯足了劲看也难以将凸的看成凹的，因为难以克服进化印刻在大脑中的“光线来自上方”的假设，因此为了让你看到“当光线来自下方时你会看到什么景象”我不得不将图片颠倒一下，结果你就看到原来凸的变成凹的了。

对于了解贝叶斯方法的同学，这个“光线来自上方”

的假设就是先验（prior）的。

世界在我们眼中其实只是一张二维图片，由于引入了“光照来自上方”这个先验假设，便有了凸凹。否则，文中一开始那张图片中的“按钮”可以是凸的，也可以是凹的，也可以是一张平面的、故意捉弄你的眼睛的画。

最后，我们再来做一个实验，将原图转动 90 度：



是不是发现凸凹感基本消失了？现在图片看上去更

像是透过面板上的一些孔洞看背后的一张黑白条纹纸。前面提到，我们的大脑通过阴影来判断凸凹，在对阴影的“含义”进行推断的时候必须假定光照来自上方，而在这张竖着的图中，假设光照来自上方的话，那些阴影是没有意义的，因为不管凸还是凹，都不会形成这样的阴影，因此我们的大脑便无法判断凸凹了。（注：其实只要稍微把头往某个方向转一下就会看到凸凹了，并且，由于 90 度的偏角远小于上下颠倒，所以可能不少人还是能够在上图中看出凸凹感来的，只要想象光线来自左方或右方即可，比想象光线来自下方容易多了）。

也许这个实验对你来说过于简单，对于我们大脑中的“光线来自上方”的先验假设你还没有强烈的感觉。下面是一个更强的先验假设——人脸。

我们的大脑有一个神经网络模块负责识别人脸，这也是一块硬编码的神经网络，也就是说我们天生就对任何（类似）人脸的图像敏感，所以随处都看到人脸，稍微类似人脸的图像就会被优先解释为人脸（用“手中拿

着锤子，什么东西看上去都想钉子”的话来说，人脸模型就是我们的大脑在图像识别时的一柄黄金大锤）：

当这种对人脸的强大先验假设在与“光线来自上方”假设产生冲突的时候，真正诡异的事情就出现了！

这是卓别林的面具在旋转过程中的四个不同瞬间的截图，左上图是面具正面的正常图像，但右下角是从反面看的情形——这个时侯实际上面具是向内凹陷的面孔，但是我们的大脑欺骗了我们，让我们仍然看到凸的面孔，因为大脑的人脸识别模块对“脸是凸曲面”的先

验假设轻易地打败了“光照来自上方”的假设。大脑的逻辑是这样的：如果假设光照来自上方，那么根据阴影来推断这就应该是一张凹陷的脸。但我们又知道所有的脸都是凸的，因此必须推翻光线来自上方的假定才能符合“事实”——当大脑中的两个假设相冲突的时候，更强硬的那个获胜。如果这不是一张人脸面具，我们便可以轻易地意识到是凹陷的了。

静态图片不够生动，下面是这个视频的动态旋转过程（http://v.youku.com/v_show/id_XODYzODE5NDg=.html）

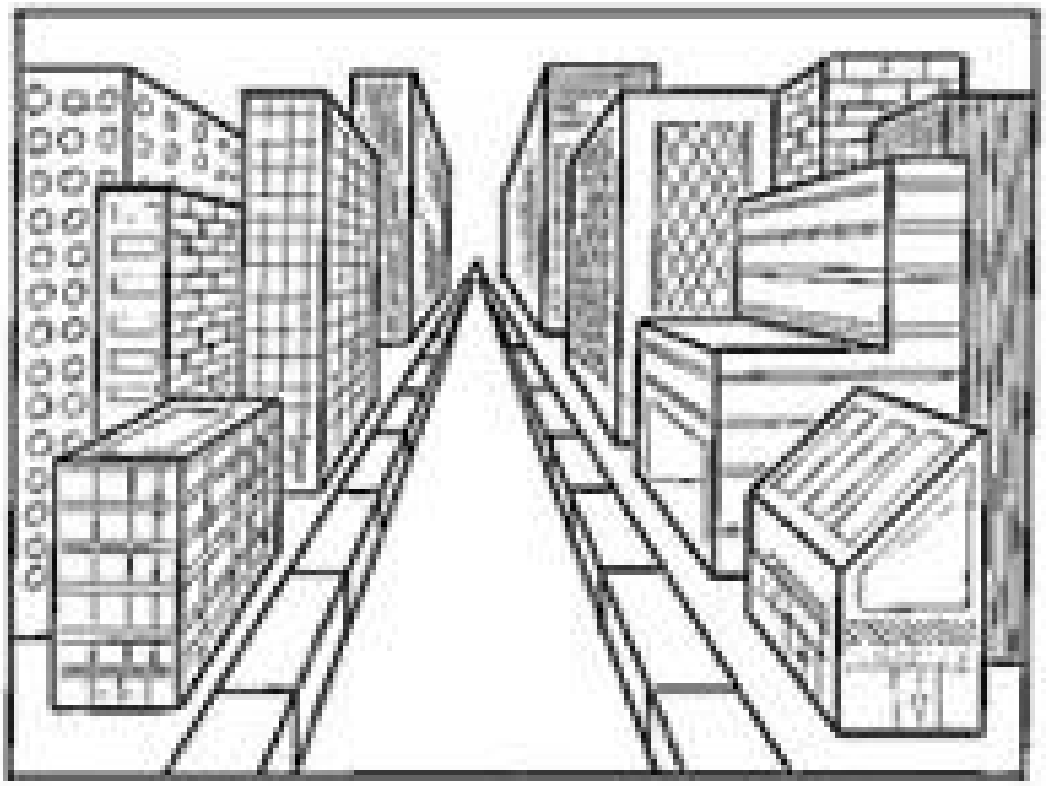
下面这个视频则很好地将上文提到的两个实验结合了起来（http://v.youku.com/v_show/id_XODYzODE4Mjg=.html）

如果你对这种先天印刻在大脑中的先验假设仍然有所怀疑，再来看看著名的诡异的 Ames’ Room 和 Ames’ Window 吧（http://v.youku.com/v_show/id_XODYzODI4MTI=.html）

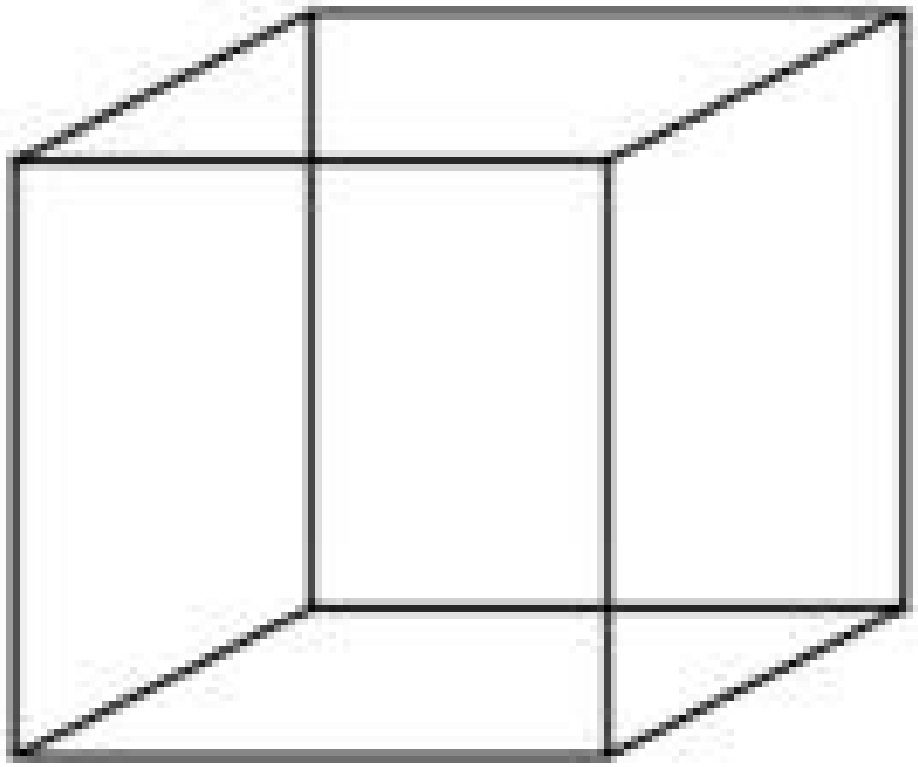
（图片/视频属于 Richard Gregory http://www.richardgregory.org/experiments/index.htm）

Ames’ Room 的构造有点复杂，但 Ames’ Window 是很好造的。

在 Ames’ Window 和 Ames’ Room 中，由于我们假设屋子的框架和窗户的框架是平行的，从而会将视觉上平行的窗户看作是与我们相对平行放置的，而将视觉上扭曲（一头宽一头窄）的窗户看作是与我们相对垂直放置的（因为其一端离我们远去从而变小）。事实上我们在现实中正是通过物体大小的变化来判断远近的，这也正是透视法能够在平面纸张上创造出三维视觉效果的原理：

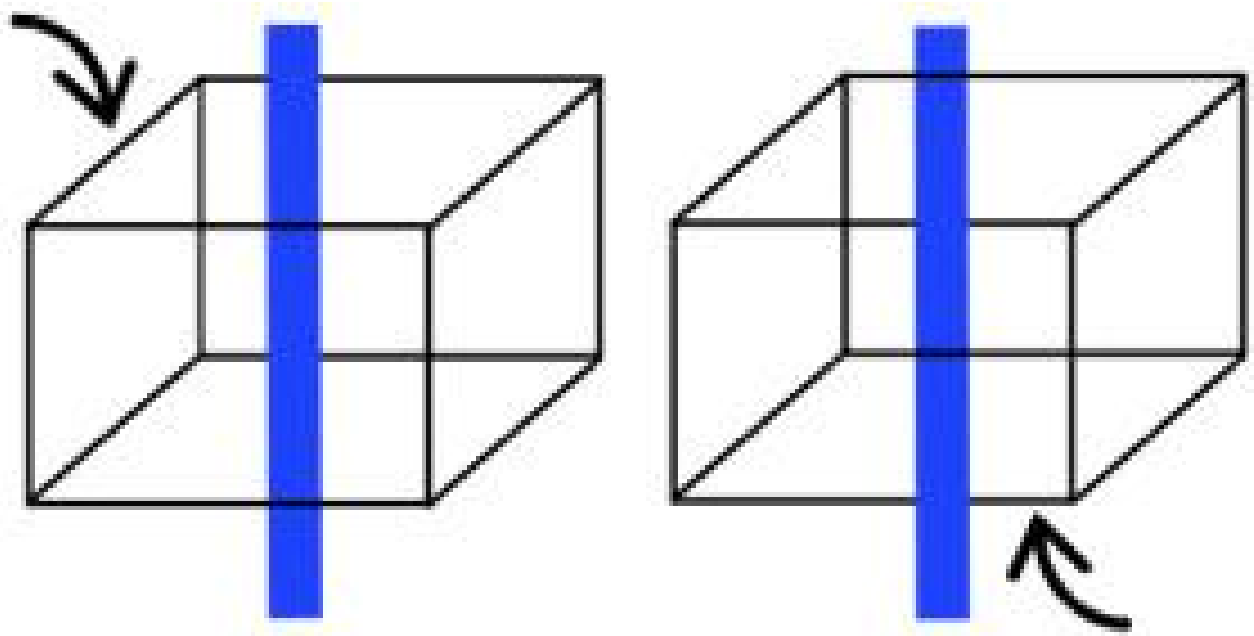


在上文的实验中，我们的大脑由于有“内建”的假设，所以轻而易举地将一些按钮无歧义地解释为凸或者凹（后面我们会看到，虽然先验假设帮助我们消解二义性，但先验知识恰恰也正是偏见的本质来源），我们不免要想：如果缺乏先验知识来消解二义性，会出现什么现象呢？



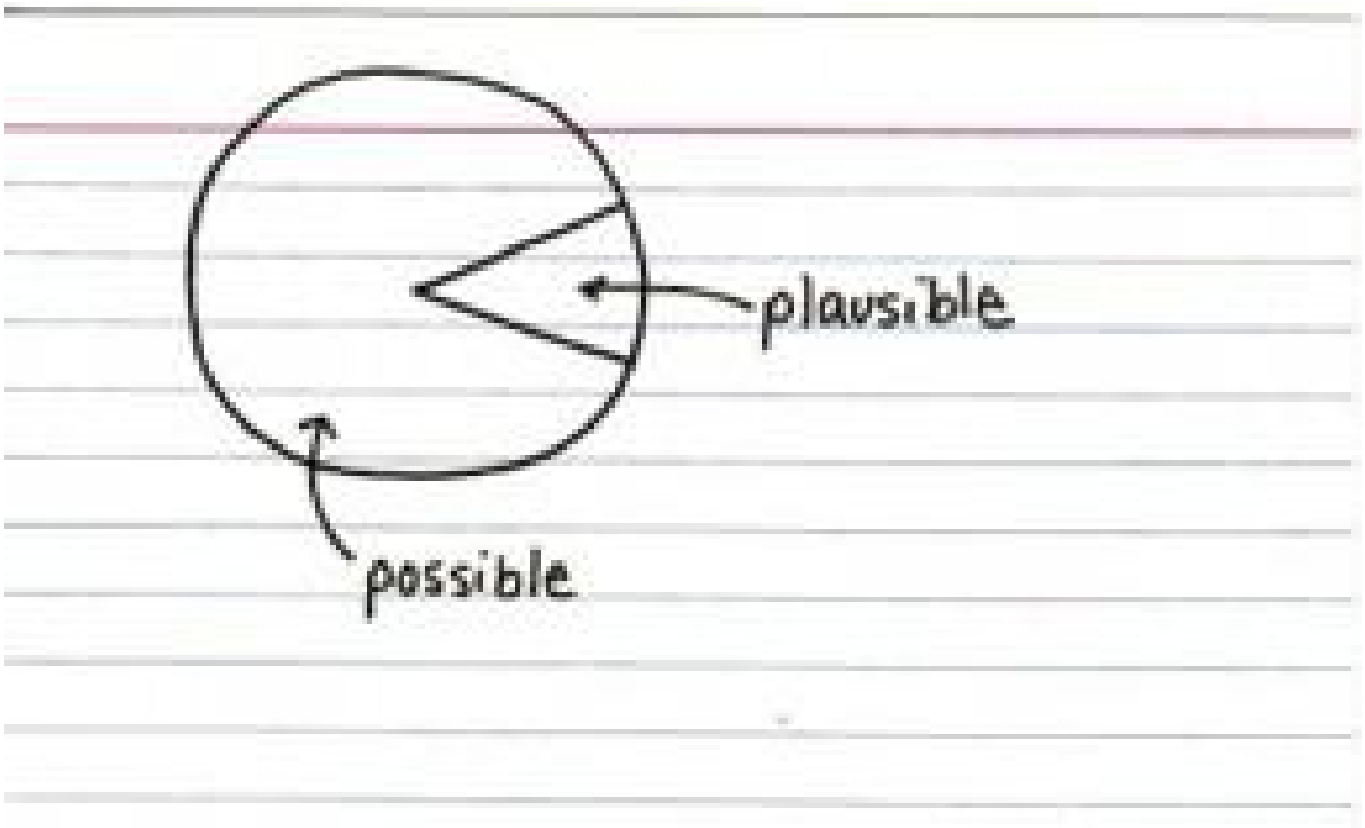
这个是著名的 Necker Cube，对它的三维解释是二义的。也许由于我们对平放的方块更熟悉（对图片来讲这是一个先验知识，因为它并不蕴含在图片本身携带的知识当中），更多的人会看到其中的一种解释（即“一个平放着的方块”），但其实还有一种解释也是完全可能的。如果不引入“现实中平放着的方块更常见”这个先验假设，我们其实是无法在两种假设中选出一种的，两种可能性等同。事实上盯着图片久了之后这两种解释就会随机切换。

要消解二义性其实很简单，引入新的 evidence（了解贝叶斯方法（《数学之美番外篇：平凡而又神奇的贝叶斯方法》）的同学对这个字眼应该很熟悉吧？）：



（图片属于 wikipedia 和 Necker）

我们的大脑会综合图片中所有的 evidence，以及大脑中原本就有的先验假设，给出最可能的解释。但必须注意的是，如果按照统计学派的观点，应该让数据本身说话，不引入先验假设的话，二维图片就是二维图片，每种解释的可能性都是均等的，但如果考虑了先验假设，那么往往只有一种或几种可能性是靠谱的(plausible)：




（图片注：荣耀属于 Indexed）

前一阵子互联网上流行的“看你是左脑还是右脑”的“旋转的女人”图片也是绝佳的例子（注：其实这跟左右脑毫无关系）（类似的多义性视觉错觉参见 Multistable Perception）：

有人看到顺时针有人看到逆时针。但更多的人看到的是顺时针 ,正如 Necker Cube 中更多的人看到的是一个水平放置的方块一样。一种可能的解释是我们对顺时针旋转更为熟悉（先验假设）。

但是如果我们给图片加上一些新的 evidence，就会发现变化出现了：

（图片注：荣耀属于 Nobuyuki Kayahara 和八卦的 wikipedia，以及台大心理系陈建中副教授的解释）

像前面的加上了新的evidence之后的Necker Cube 一样，通过对图中旋转的女人的剪影添加轮廓线索，强烈提示了目前这个瞬间到底是正面还是反面。通过这个提示，大脑正确的对二义性进行了消解。

其实，说到底一张二维图片就是一张二维图片（外界物体反射的光线投射到我们的视网膜上也只是留下二维的剪影），其三维解释有 N 种（甚至无数种），但为什么绝大多数情况下我们的大脑能够一下就锁定在其中的一种可能性解释上呢？皆因我们的大脑对生成这张图片（特别是图片中的阴影）的环境参数有一些先验的假设（如前面提到的“光照来自上方”、“脸是凸曲面——严格来说，鼻子是凸的”）（注意，这些先验假设并不蕴含在图片中，而是我们在长期生活中无意识统计出来的，或者干脆就是漫长的进化过程筛选出来的有价值的先验假设——正如婴儿天生在吃奶期就懂得吮吸一样。）

我们再来回顾一个经典的视觉现象——色彩恒常性（Color Constancy）。我们已经知道，同一个物体，在

不同光照条件下我们知觉到它的颜色是基本不变的。一个青苹果在日挂头顶的中午的白炽光线下看上去是青的，在斜阳西下暗红色的光照下是青的，在清晨淡蓝色的雾霭中还是青的。你可能觉得这很正常，青的本来就是青的，有什么好奇怪的。但问题是如果将我们的眼球换成一个光线接收器，从而客观记录下从苹果表面反射出来的光线的 RGB 值（红、绿、蓝三色的比例），会发现在不同环境光照条件下，实际从苹果表面反射出来的光线差异是很大的；例如环境光只有长波（红色）的话，那么不管苹果表面的反射比如何，反射出来的光也只能是长波，但为什么我们看起来仍还是青色的呢？

如果我们在一个封闭的箱子中放置一张白纸，让观察者透过暗箱上的一个孔洞来观察这张白纸。那么当我们在箱子内用黄光照的时候，观察者会看到黄纸，用红光照的时候会看到红纸。但如果打开箱子，则不管用什么光照，我们看到的还是白纸。

为什么会出现这种现象？目前为止已经有了一个理

论解释框架：尽管同一物体在不同光照条件下反射的光线差异很大，即视网膜接收到的光线差异很大，但视觉皮层对视网膜接收到的光线又进行了一层处理，这层处理就是从视网膜接收到的光线中“抽取”出物体的“真实颜色”。但我们的神经回路如何计算目前还并不最终明确，但有靠谱的逼近算法（被称为 retinex algorithm），其中一个简洁的版本是这样的：假设目标物体周围的邻近环境中存在完全反射光线中的绿光成分的物体、也存在完全反射光线中的红光成分的物体、也存在完全反射光线中的蓝光成分的物体，那么只要将眼睛采集到的环境光线中最强的绿光成分 Gmax，最强的红光成分 Rmax，最强的蓝光成分 Bmax 分别记录下来，然后算一下目标物体所反射的光线的 RGB 对(Rmax,Gmax,Bmax)的比例即可。

这里，再一次，我们的大脑从一个具有多义性的信息源中抽取出了一种最靠谱的解释。从物体表面反射出来的光线并不能唯一确定物体的反射比，一个方程无法解出两个未知数（光照、反射比）。但我们的大脑仍然还

是聪明地利用了环境中的 evidence，靠谱地解决了这个问题。

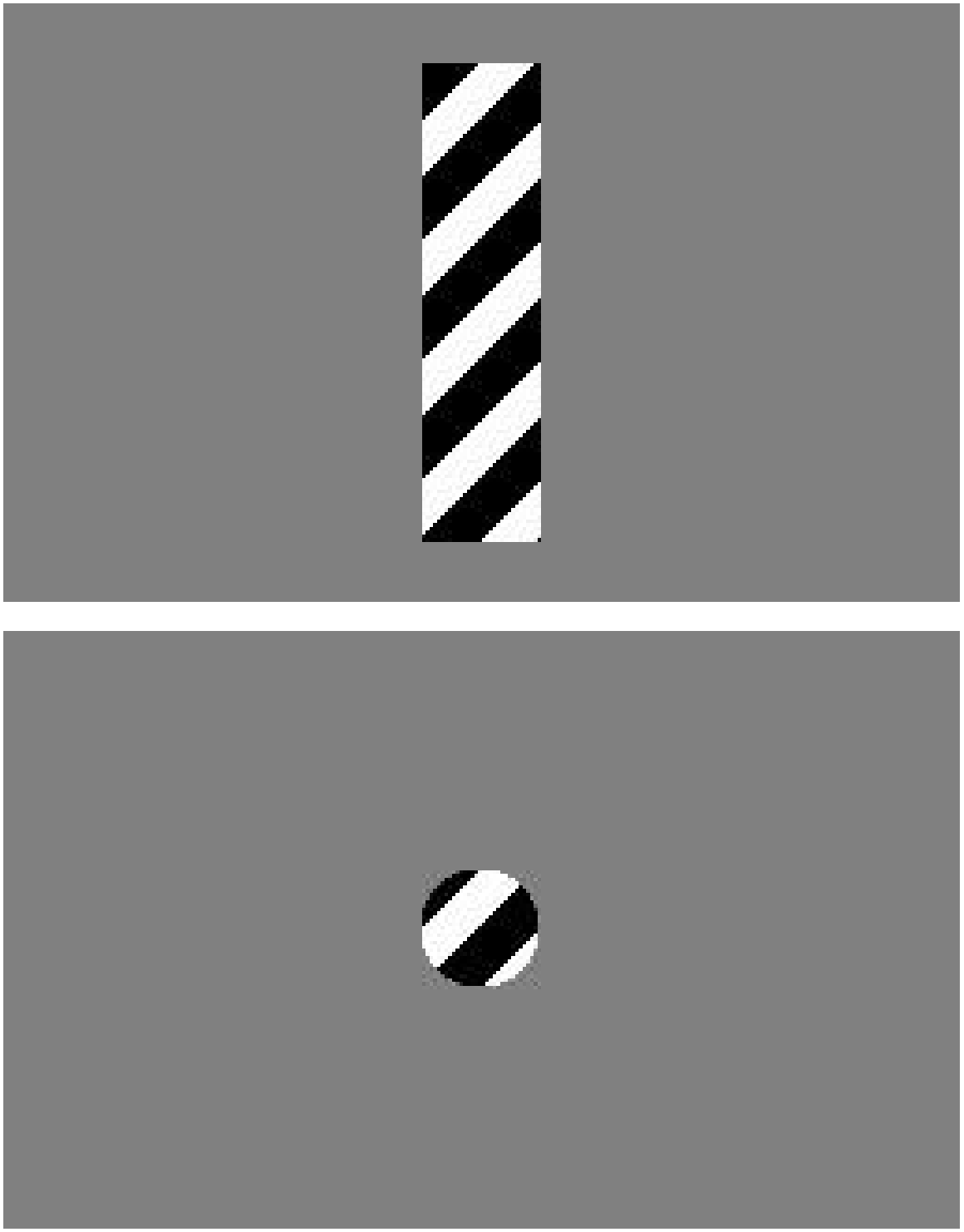
然而，接下来的才是我真正想说的，在刚才提到的算法中，一个先验假设是“目标物体周围的邻近环境中存在完全反射光线中的绿光成分的物体...”，问题是如果这个假设不满足呢？戏剧性的错觉就出现了，见下图：

（图片注：荣耀属于 Mauro Vecchi，这是从一个精美的视觉错觉视频中摘取出来的画面，完整版参见这

里。）

图中白线所指的两个小方块的颜色一样吗？如果你觉得不一样的话，不妨用软件把这两个色块的像素摘取出来对比一下。（注：这里还有一个关于 Color Constancy 的不错的视频：多亏色彩恒常性，多彩世界不混乱，或如下）（注：色彩恒常性只是我们体验到的一系列主观知觉恒常性中的一种）（http://v.youku.com/v_show/id_XODY0NDc2MTY=.html）

以上这些错觉与实际生活多少有点脱离，但我打赌以下这个现象每个人都看到过——只要你去过理发店，因为这个错觉也被叫做（理发店门口的）“旋转彩柱错觉”：



（图片注：荣耀属于 wikipedia）

透过条柱看到的黑白条块在往下移动，而透过孔隙看到的黑白条块则往右下方移动。然而，实际上孔隙背后的黑白条纹纸可能正在往下移动，也可能往左移动，

实际上其移动的角度有无穷多种可能，为什么我们的大脑只看到了一种可能？具体解释可以参照 Barberpole illusion 条目。

我们的视觉系统通过大量的先验假设来解释投射到我们视网膜上的二维图像，从中推断出三维结构，类似的例子还有：Kinetic Effect,Aerial Perspective,Parallax Scrolling,等等。视觉系统感知到的三维图像只是我们的大脑给我们玩的一个小把戏，或者，严格来说，一系列小把戏之一。

在一般人看来，视觉错觉只是拿来哄 MM 开心的小伎俩，是魔术师的小把戏，“不登大雅之堂”，然而在心理学家和认知神经科学家们眼里，视觉错觉是一个迷人的研究领域，是研究人脑如何处理信息的窗口，正如数学家们透过悖论对数学的奥秘一窥端倪，心理学家们也在透过形形色色的错觉现象探索大脑对信息的处理机制。一篇严谨而不失趣味的论文可以参考“Perceiving the Present and a Systematization of Illusions(PD

F)" http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.3729&rep=rep1&type=pdf )

以上这些错觉现象实质上揭示了一个深刻的原理，这个原理不仅适用于视觉现象，同样适用于其他心理现象：我们的大脑从外界接受到的信息其实是满含着歧义的，单单从这些信息本身来看，我们应该感到无所适从才是，然而我们的大脑几乎每次都能够从富含歧义的信息中找出最靠谱的解释，作出无比牛 B 的点估计，这得益于漫长的进化过程，以及我们平常生活中积累的大量先验假设，然而，接下来我们要说到，这些先验假设是双刃剑，其锋刃的另一面就是我们常说的"偏见"。

《Probability Theory,the Logic of Science》上讲了这么一个故事：

一个月黑风高的夜晚，你是一位警察，在一条荒无人烟的街上巡逻，忽然听到入室盗窃自动警报，你转头望去，发现街对面的珠宝店的玻璃窗户破了个大洞，一个蒙面的家伙背着一个鼓鼓囊囊的的包正从窗户中爬出

来，此时，你一定毫不迟疑地断定这个人就是强盗。你是怎么推断的呢？

《疯狂的赛车》里，耿浩到庙里取了骨灰，一出门看到几个黑社会老大模样的家伙，以为是殡仪馆的人，而对方却以为耿浩是杀了泰国佬的地头蛇，并把耿浩手里的骨灰盒当成了藏毒品的工具，还一通佩服，结果一桩阴差阳错的生意就做成了。他们又是怎么推断的呢？

正如以前听过的一句话所说：对于日常生活中的事件，总有一个平凡的解释，和一个疯狂的解释。

例一中的那个背着包的人可能是珠宝店的老板，从假面舞会回来，身上没带钥匙，当他走过自己的珠宝店的时候，一辆卡车呼啸而过，轧飞的石子把他的珠宝店窗玻璃打碎了，为了保护他自己的珠宝，他只能进去把珠宝收起来带走。

至于《疯狂的赛车》，另一种解释不用我说了。

TopLanguage 上的一位朋友 li kai 讲了这么一个故

事：

我有个朋友前些日子刚结完婚，这里有一个故事。他本身并不富裕，因此呢，就跟媳妇商量，咱能不能一切从简，什么三金，(就是金项链、金戒指、金耳环)也就都免了吧，媳妇说这是家里规矩，不能同意，这边老丈人当然就更不同意了，非逼着我这穷哥们买三金，给一万元礼钱。

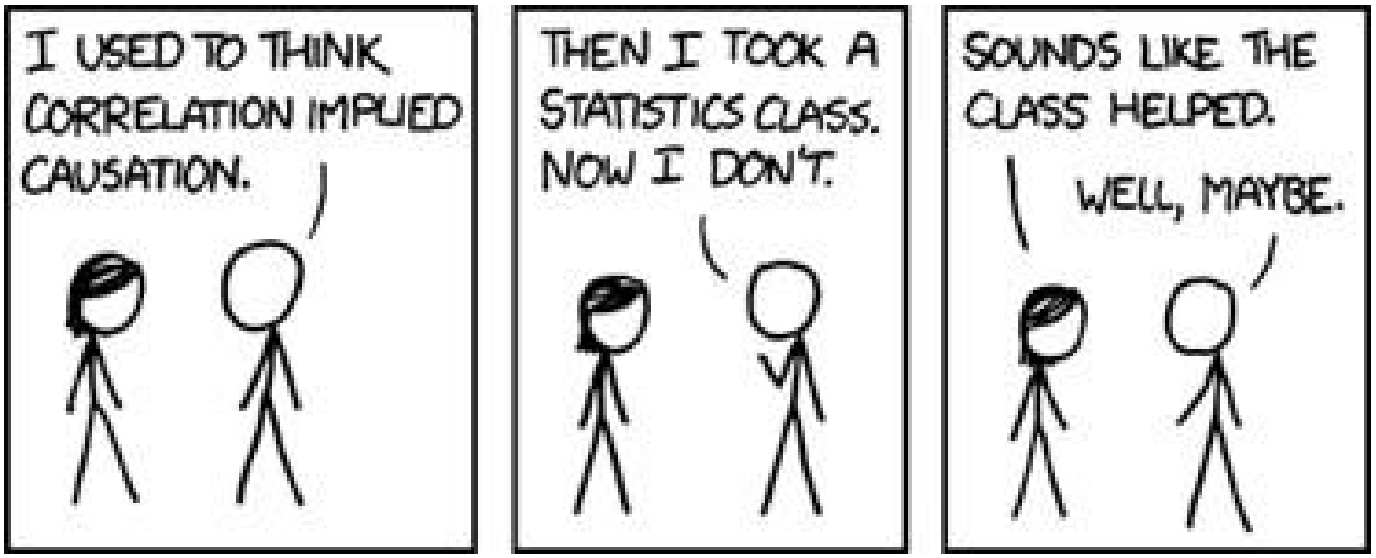
最后，没办法，我这朋友东挪西借把东西弄齐了，婚礼上，老丈人也给新郎一个红包，可我这哥们却始终憋着一股气，接过来之后终于没按耐住，爆发了，你猜怎么？他把红包给撕了扔地上，后来大家就劝他别这样，结婚呢，好不容易安抚下来，有人就说，你把彩礼捡起来吧，看看到底给你多少钱，结果他捡起来一看，是一张存折，上面显示有十万元存款。

原来老丈人并不是想要从男方家捞什么钱，只不过就是认为按照自家风俗这些时必须的，否则女儿嫁的太不风光了。仅此而已。

故事中的这位朋友又是怎么推断的呢？他所得到的信息仅仅是他的丈人坚持要他给礼钱，他并没有得到关于丈人这么做的意图的信息。丈人的意图只是他自己的推断，他对丈人意图的推断只是一种可能性，并非唯一的可能性。然而他仍然还是不可避免地陷入了对自己推断的过分信任的陷阱，一旦脆弱的自尊被触发，接踵而来的便是一连串情绪化的、自动化的行为（仿佛汽车挂上了自动挡）。

想一想生活中有多少误解是由于自以为是地对别人的意图的误读而导致的呢？

我们总是混淆“事实”和“推断”，尤其是当我们对推断的确信度很高的时候，或者某种推断对我们有利的时候，或者当这个推断源于大脑天生的偏见的时候，例如，将关联误当做因果就是我们的认知死穴之一：




（图片注：荣耀属于 xkcd）

松鼠会的新书发布的时候，姬十三发布了一个页面作了简介，我跳转过去浏览了一下，看到介绍的结尾跟着一段话：

作者简介

姬十三

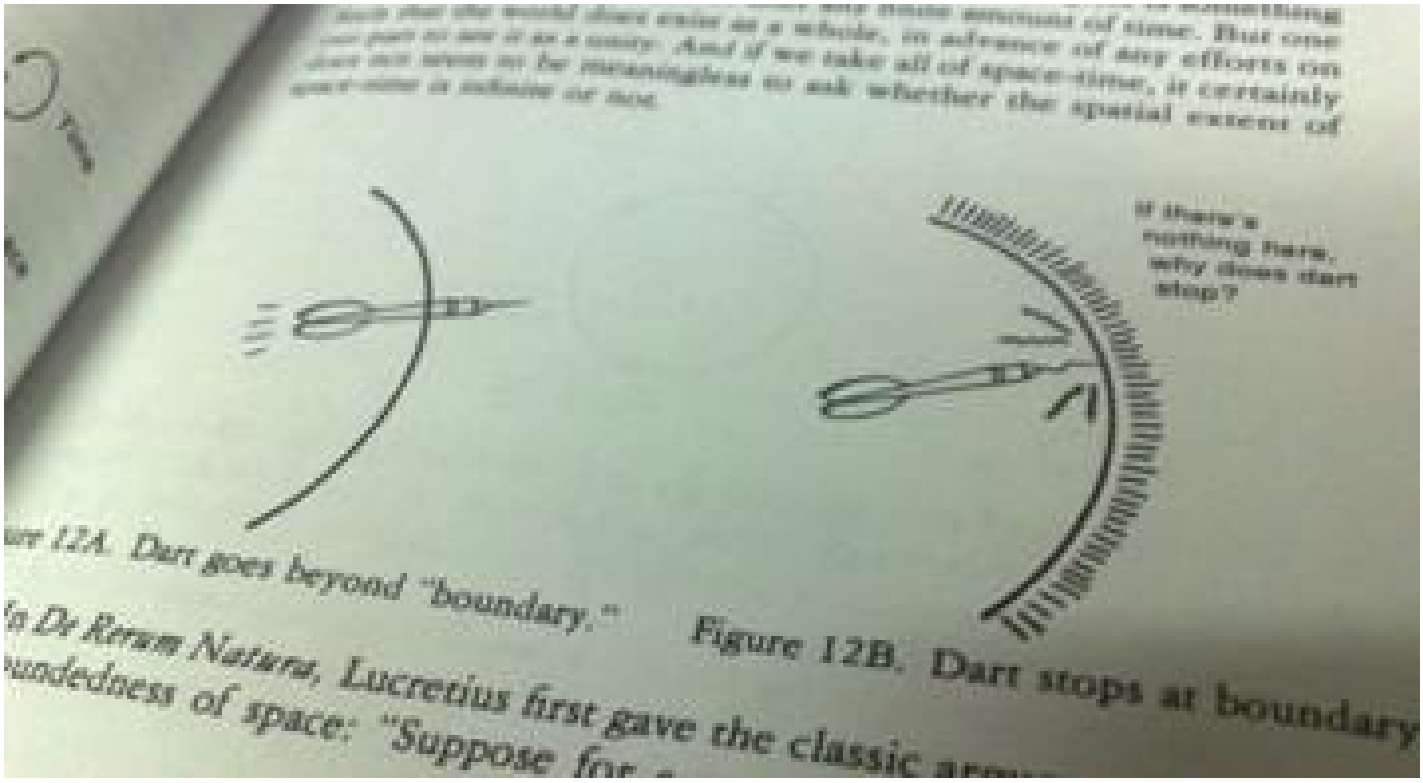
姬十三，神经生物学博士，供职于美商百科网站博闻网( http://www.bowenwang.com.cn )。为《新发现》、《外滩画报》、《时尚健康》等报刊撰写科学专栏。个人博客是：http://jshisan.yculblog.com

我就感到奇怪，《当彩色的声音尝起来是甜的》是松

鼠会诸多作者的作品合集，为什么这里作者简介只写姬十三呢？我想当然地把这里的作者简介当作了是新书的作者简介，我心想：那难道还有什么可能呢？（这同样也是我们的认知偏差之一——把“想不出其他可能”当做“没有其他可能”）。然而老婆又适时地泼来一盘冷水（我为什么要说又呢？）：这里的作者简介难道不可能是这篇博文的作者简介？我一想，也是啊。要证伪我原来的假设很简单，翻开另一篇博文就知道了。于是我随便打开松鼠会网站上的另一篇博文，果然这里的作者简介是博文的作者简介，而非（我原来所猜测的）新书的作者简介。

但是，关键是，原先我并不知道松鼠会的博文有这么一段作者简介，因而就我原来所持有的知识而言，我作出的推断是唯一靠谱的，这也是为什么我感到如此确信的原因之一。而且，由于我不知道松鼠会的博文有这么一段作者简介，因而我根本无法看到或设想另一种可能性。既然想不到另一种可能性，又怎么可能有机会去证伪我的猜测呢？当知识有硬性局限的时候，就算持有

Open Mind 甚至也是不够的：




（图片注：荣耀归于 Matrix67）

古罗马哲学家 Lucretius 认为，宇宙是无限的。让我们来看一看他的经典论证。假设宇宙是有限的。我们往宇宙的边界投掷一根标枪。则我们将看到以下两种情况之一：这根标枪穿过边界飞向远方，这说明宇宙并无边界，它是无限的；或者这根标枪一头装上宇宙边界停了下来，这说明边界外“有东西”挡住了标枪，同样说明宇宙是无界的。（来自 Matrix67）

我想说的是，在我们的知识体系里面还不存在“有限无界”这个概念的时候，上面的推理真的很滑稽吗？我们现在的知识体系和古希腊相比固然得到了极大的进步，但是我们真的变得更“聪明”了吗？要知道推理的引擎（演绎和归纳）几十万年来却并没有变化，我们只是站在巨人（数千年知识的积累）的肩膀上，但这个巨人的高度并不属于我们自己，我们有什么理由五十步笑百步呢？

我们太可能因为受到知识的局限性而对事物的看法产生无法消除的偏见，有时候打破这种偏见的唯一途径就是开阔视野，多积累知识，以及和具有不同知识背景的人讨论，否则就算抱着“我可能是错的”这种信念，你也不知道怎么去证伪自己的一个猜测。

关于我们大脑中的先验假设能够对我们的日常推理和记忆造成多大的影响，有这样一个经典的实验：

1981 年，两位心理学家 Brewer 和 Treyens 作了这么一个实验：

召集一些人，告知他们将会参加一项学术研究计划，实验者先带领他们来到一间办公室，让他们稍加等候，一段时间之后，叫他们出来，并询问他们记得办公室里面有哪些东西。一些人声称看到了书，然而实际上办公室里面根本没有书。

这里的原理是显而易见的，我们期望在一个学术机构的办公室里面看到典型的事物——书。当我们的直接记忆并不深刻或者我们当时等候的时候并没有刻意留心屋子内的摆设和物品时，我们会依靠之前生活中积累出来的先验假设进行推理，填充记忆的模糊或空白。关于虚假记忆的研究也表明，我们的记忆并不像电脑的存储设施那样，忠实记录，然后忠实读取，而是在记录和读取的时候都是相当程度上“构建性”的，而构建所用的“素材”则是我们之前在生活中积累出来的经验。这也是为什么同一个故事经不同的人口口相传之后会出现形形色色的版本的原因。

–以下对了解机器学习的 geeks 插播一段八卦–

对基于统计（特别地，基于贝叶斯）的垃圾邮件过滤的基本机制有所了解的同学应当知道，在判别公式里面有两项分别是 P(S)和 P(H)，分别代表一封邮件是垃圾邮件和非垃圾邮件的（先验）概率，一项统计表明现实世界中这个比例是 8/2，即 80%的邮件是垃圾邮件。这个就是过滤器眼中的世界，“八成的人都是坏人”，这个就是过滤器的“偏见”，或者“先验假设”，来一封邮件不管三七二十一首先作一个最坏的打算。正如机器学习方法的偏见来源于训练数据集，我们头脑中的偏见也来源于我们大脑中神经网络的训练数据集——现实生活。

由于 8/2 的比例并非时间无关的稳定比例，或者其他什么原因（如保守起见），目前大多数贝叶斯垃圾过滤系统实际上将这个比例设为 5/5，表示“无偏见”，不设先验。这就基本上将贝叶斯这个词扔掉了。但我个人觉得这并不能称为“无偏见”，如果现实就是“有偏”的，保持公平也是一种偏见，这让我忍不住想起 P.Norvig 讲的关于人工智能鼻祖 Minsky 的一则轶事：

In the days when Sussman was a novice,Minsky once came to him as he sat hacking at the PDP-6.

"What are you doing?",asked Minsky.

"I am training a randomly wired neural net to play Tic-Tac-Toe,"Sussman replied.

"Why is the net wired randomly?",asked Minsky.

"I do not want it to have any preconceptions of how to play",Sussman said.

Minsky shut his eyes.

"Why do you close your eyes?",Sussman asked his teacher.

"So that the room will be empty."

At that moment,Sussman was enlightened.

根据 P.Norvig 的说法，Minsky 是想告诉 Sussman 一个随机赋值的神经网络也是有模型（或偏见的），只是这很可能是一个极其复杂的模型，我们无法理解。你蒙上眼睛不代表这个屋子不存在，你不知道随机神经网络的模型是什么不代表它不存在。

但我忍不住 YY 了一把另一种解释 :如果现实世界背后的模型本来就是“有偏”的，假装不引入“偏见”本身就是“偏见”。只不过我们所观察到的现实世界纷繁的表象往往只是一个局部有偏样本，导致我们看上去随机抽取的数据其实还是有偏的，如果我们蒙上眼睛骗自己说这就是真正随机的抽样，那么训练出来的模型肯定也是有偏的，为了补偿这种偏差我们有时候宁可扔掉从训练数据中得到的某些概率，这种方法往往导致长期来讲更靠谱（严格来说这里的术语是 robust）的模型，尤其是在金融市场上，小聪明的人从短期趋势数据上自以为得到了靠谱的模型，把太多的赌注放在了一个建立在因在时间维度上没有随机采样而很可能有偏的数据集上得到的模型上，而真正智慧的玩家则会建议普通人最佳投

资方法是无偏见地平均分配资金，避免因模型错误而导致的灾难，这一平均分配的极端形式就是——投资指数。

–八卦结束，回归正文–

最后再来一个例子，生动地说明了我们在平常生活中积累的偏见有多深：

问题：现在有两个孩子，张森和李梅，其中一个孩子有四个卡车玩具，你认为是谁？

问题：莉莉和丁丁谁将来更可能成为护士？

（注：荣耀仍属于 Frith。以上问题演绎自《Making Up the Mind》p168 页的英文版本）

尽管我们只拥有他们的名字，名字本身只是任意的汉字，自身并不携带信息（从一个角度来说），但就连三岁的小孩也能对这两个问题给出“靠谱”的答案。

当然我们也可以说偏见代表着长期积累的生活经验，能够使我们事先就对未知对象的属性进行靠谱的预测，但另一方面，偏见也很可能成为自我实现的预言和

自我妨碍的篱笆：女孩和男孩的数学智商真的有显著差异吗？但受文化影响，女孩认为自己更不擅长数学和理科，这个自我信念从两个角度产生作用，第一，它会让女生倾向于投入更少地时间在理科上，从而导致更低的理科成绩，结果进一步强化了她的“我的数学不好”的信念。这几乎是一个死循环。第二，人们为了维护对自我的信念，会拒绝接受与之相反的信息，如果那一次她数学考得不错，她会寻找外部理由，譬如“只是凑巧罢了”，虽然这种把自己看低的心理过程有点不可思议，但这的确是心理学家们实验证实的结果。

在社会文化方面，人们常用“仁者见仁、智者见智”这个俗语来指代三种现象：

1)偏见：不同的人戴着不同的有色眼镜，对同一现象产生不同的理解或解释。是平凡的解释还是阴谋论的解释？存乎一心。

2)立场：例如对于“生活的意义”没有统一的标准公理，因此每种生活都是合理的，各人可以持有不同的

价值观，优化不同的目标函数。

3)选择性关注：对于同一事物，不同的人关注的点不一样，象有四腿，各摸一条。

与“仁者见仁、智者见智”这个俗语的褒义色彩相反，这里除了第二点是中性的之外，另外两点都不能算是好事，譬如程序员在做项目的时候经常只选择性地关注“使用的技术是否有意思，是否有挑战性，是否好玩，我能否从中得到乐趣，是否能学到新东西”，而并不是关注最应该关注的“如何以最小成本达成项目成功”。至于第一点——偏见——就不用多说了，前文已经说得够多了。如果还觉得不够的，不妨读一读社会心理学经典之作《社会性动物》的冲突和偏见部分。

偏见在我们解决问题、认识世界的过程中都起到了很大的影响，并且很多时候是不好的影响。因此，让我们经常和具有不同信念和知识背景的人讨论，弥补个人经验知识的局限性导致的偏差，并时常使用以下这句话来提醒自己 keep an open mind 吧：

“这只是一种解释（可能），未必是唯一的解释（可能）。（想不出其他解释不代表就不存在其他解释）。”

## （三）遇见 20 万年前的自己

《Synaptic Self》中曾提出一个发人深省的观点：由于人的大脑是经过漫长的进化年代“堆积”起来的，也就是说，从爬行动物到哺乳动物到高级灵长类这些进化阶段，我们的大脑从只有原始的反射模块，到拥有初步的情感区域，一直到神奇的具有 6 层结构的“新皮质”所支撑的高级认知能力，一步步走来。这个过程并非上帝预先编程架构好的，而更像是在既有结构上“叠床架屋”，比如，大脑从内到外基本上是按照进化年代来排序的，比如啮齿类等一些小型哺乳动物的新皮质是光滑的，这是新皮质在进化出高级灵长类之前的样子，后来为了解决大脑中空间不够的问题，进化之手发明了大脑皮层沟回，通过这些褶皱，在不增加太多占用面积体积的前提下使得大脑皮层表面积暴涨，正是这些褶皱使得灵长类进化出独特的高级认知活动，如工作记忆，语言。这

种“堆砌式”的进化有它节省和复用的好处(《Kluge》),然而另一方面也带来了奇特的“进化时滞”效应——进化年代较近的大脑模块和较久远的模块之间要达成完美沟通需要一定的时间，在这之前便会造成多个模块之间面对同一个问题决策不一致的问题。

我们的高级认知模块明明知道有些事情很重要，必须提前准备并持之以恒，然而我们内心的另一个小声音却在万般阻挠我们把屁股从床上挪开或者把眼睛从网页上挪开；我们明明知道赌博，烟酒，犯罪是不对的，然而内心的一个小声音却在喋喋不休地催我们动手去做。我们明明知道高糖高脂肪的食物不宜多吃，但内心的一个小声音却总是怂恿着再吃最后一勺( 不禁让人想起《傀儡人生》)。令人感到遗憾的是，很多时候在这个争端中败下阵来的却是代表更先进生产力的高级认知模块，更令人感到遗憾的是，在大多数时候我们的高级认知模块似乎根本就没有启用，而是凭借着本能或直觉“自动驾驶”自己的身体 (《Gut Feelings》)。

这里的原因是明显的：设想一下，人类的高级认知模块是在相对较近的进化年代出现的，然而原始的情感和条件反射模块却在千万年的进化长河中忠实地保护着我们在自然环境中生存下来并努力繁衍后代（《Mean Genes》），这些模块似乎“理应”拥有更强大的力量，然而工业时代的到来将人类生存的环境极大的改变了，我们大脑的原始模块适应的是远古时期的生活，以采集狩猎为主题的社会构成，这跟现代工业文明相差颇大，举个例子，我们的社会交往本能令我们非常顾及自己的面子（面子不仅是一个东方文化中的东西，在全球都存在），面子可以与很多东西关联——与异性的交往成败，在对手面前是否挺身而出，在困难的任务面前是否完成得很好，这里的逻辑是很明显的：一次糟糕的社会性事件会降低我们的声誉，在远古社会，聚居群体较小，成员之间依赖性很高，糟糕的声誉会导致被赖以生存的群体排挤出去，危机自身的一切，所以我们往往有着不顾一切捍卫自己的面子原始冲动，《Bounded Rationality》里面曾经提到这样的一个例子：两个男人因为酒吧里面的

一点小争执最终大打出手乃至一方杀了另一方。在类似酒吧这样的一个众目睽睽的社会场所，人们往往会为面子而恼羞成怒，作出过激行为，而我们的大脑同时又会认为我们仍然处在没有法制的远古社会，所以杀人便有了可能，事实上我们不难想象在远古社会杀人可以转化为力量和能力的声誉，由于进化的钝刀还没有来得及磨平我们远古时期的“棱角”，所以只有少部分“理性大脑”强大的人才能够作出适应现代工业社会的行为；《Mean Genes》上面说了这么个例子：在远古社会我们在向姑娘求爱的时候会小心谨慎，因为一次洋相会很快被传递开来从而使得我们变成整个群落的笑柄，然而在现代社会，尤其是人口流动剧烈，人际关系变动频繁的大城市，社交失败的成本近乎于 0，所以正确而理性的做法却是永远都勇敢地迈出第一步。

因为我们的大脑中同时存在着远古的自我，和现代的自我，并且两者并没有完美协调，所以才会出现“如著作等身的教授，聪明的数学家，艺术家同样有可能成为性引诱的牺牲品，同样可能犯七宗罪，同样可能成为

焦虑和忧郁症的患者”(《Synaptic Self》)(不禁让人想起前阵子著名的泰格伍兹事件)。

然而除了这些极端情况之外，普通人也常常受到困扰——明知正确的事情就是没法去做。仔细想想这简直是一句类似悖论的话：既然你的大脑认可某种做法是正确的，而同样又是你的大脑主管你的行为，那为什么偏偏你没法执行呢？这就说明主管行为的并不仅仅是你的“认知”模块，认知模块发完话之后自会有更强大的情绪模块藐视“上级”的决定。所以我们常常哀叹“说起来容易做起来难”。世界上最痛苦的事情不是和别人作斗争，而是和自己作斗争。

《Phantoms in the Brain》提到这么一个有趣的例子：我们看到老朋友时会自然微笑，然而站在摄影师面前我们却经常“挤”出难看而别扭的微笑。我们常常说第一种微笑是发自内心的，第二种笑则是有意做出来的。事实上这两种微笑的确涉及到两种不同的机制，只不过不是心脏和面部肌肉，而是两个不同的大脑模块。自然、

下意识的微笑来自于大脑中位于进化年代较古老的丘脑和进化年代较新近的大脑皮层之间的一个叫做“基底核”（basal ganglia）的结构，而有意识的笑则是由大脑中的动作皮层控制的。而这两者只有前者具有正版的微笑神经回路，当我们看到朋友的时候，朋友的脸庞的视觉映像通过视觉神经传导到我们的情绪模块（边缘系统），并进而被转发到基底核，后者的微笑回路负责调动面部肌肉生成一个真正自然的微笑，整个过程在不到一秒内完成，在这个时间里你的高级认知模块根本还没来得及活动呢。

有些时候一些人会因为中风而导致一侧运动皮层受损，如果你叫他对着摄影师笑，你会发现他挤出的是半边脸的山寨微笑，另一边脸（对应受损的运动皮层的那一边）则不笑。然而神奇的是如果他见到老朋友，则会发出两边对称的、自然的微笑，因为控制自然微笑的基底核没有受到损伤。

而在极少数情况下，则会出现一种恰好相反的情况：

中风损伤了一个人的基底核，影响了半边脸的微笑回路，一开始这个人自己也注意不到，直到他对妻子自然微笑，妻子会惊讶地发现他脸上只有“一半”微笑。然而，如果摄影师叫他微笑，他却能够做出对称的微笑，虽然是不自然的那种，因为控制有意识微笑的运动皮层并没有损伤。

就像两种微笑一样，我们的大脑在同一个决策上经常有不同模块的参与，有的人更偏向使用直觉进行决策（《Predictably Irrational》），有的人则更偏向于理性分析，而事实上这两者并没有孰优孰劣之分，只是在不同的场合适用，无法驾驭这两种决策引擎的人要么一方压倒另一方，要么就是陷入纠结。

我们在选择职业的时候“听从内心的召唤”，因为我们对事物的热情来自于我们的情感系统，没有这个系统的支持，我们很难在一件事情上持之以恒的专注投入，emotion（情绪）和 motion（动力）本就是同根生，说明人们很早就了解到情绪和动力的关系。对此有这么一

个真实事件（记不得在哪本心理学书上看到的了，记得的朋友请留言。），一个事业顺利的中年男人，原本过着典型的美国中产阶级的生活，但有一天不幸遭遇了车祸，车祸损伤了他的头部，他的情绪大脑遭到了损坏，后来虽然他健康方面痊愈了，然而却从此对任何事情都无动于衷，再没有动力去发展他的事业，照顾他的家庭，对他来说发生什么都是无所谓的。事实上，我们所谓的生活的意义便来源于情感系统。

我们在面对道德问题的时候听从“良知的召唤”，因为漫长的进化给了我们一套非常优秀的天生道德判断神经回路（《社会性动物》），只要听从良知的召唤我们便能在道德的平衡木上走得稳稳当当。

我们对于很多事情的决策判断都刻画在天性里面，然而同样也正是这些天性在很多时候会让我们陷入困境，我们“能存储能量就尽量存储能量”的食物摄取天性虽然适合远古社会，然而在能量充裕的现代社会却导致大量人的超重。《How We Decide》上有这么一个有

意思的实验：让一群人走过一个屋子，屋子的中间放着一个桌子，上面有诱人的巧克力蛋糕，也有水果沙拉，让其中一部分人默记某个 7 位数字序列，另一部分人则只默记 2 位数（当然，实验者会为默记数字的原因编造一个谎言，例如测试记忆能力），实验结果是，那些默记 7 位数字的人更可能选择巧克力蛋糕，因为记忆数字过载了我们的高级认知模块，使得它无暇和原始大脑的决定作抗争。

我们对于未来的惩罚和收益都估计不足，倾向于就眼下的损益进行决策。这就导致我们天性在决策方面目光短浅，一个典型的例子是我们会在大学阶段花费大量的时间去进行学习之外的娱乐，这些娱乐都有一个典型的特点，就是能够立即获得愉悦，并且并不会导致立即的损失。毕业看似遥遥无期，我们很难提前几年就设想几年后的危机，毕竟，未来的事情谁说得清呢？也许我们的远古生活告诉我们的真理就是，几年后的潜在收益跟眼下唾手可得的好处无法相比。

我们的原始大脑同样也分不清什么是虚拟什么是真实，在获得社会成就和声望的动机的驱使下，即便是虚拟的网络游戏社会，我们也会投入大量精力，某种意义上我们的大脑并没有错，我们的确应该去获得声望，只不过它还没有聪明到跟得上工业文明，它并不明白虚拟世界里面的生存能力和地位并没有办法转化为现实世界的生存能力和地位。

如你所见，很多时候我们只是生活在信息社会的远古人，如果上帝要为我们目前生出的时代设计人类，我们将会是身体上适合长期久坐不见阳光，眼睛尤其适合长久盯着 30 厘米以内的物体看 ,我们情绪上会偏好有节制而健康的饮食，我们的身体发育将不会在肌肉上浪费太多不必要的能量，青少年也不会在青春期强烈渴望冒险和建立小部落内的声望并为之做出各种危险或可怕的事情，而在执行力上我们则会偏好于执行能够积累知识和技能的长远计划 ,正如 geeks 们所崇尚的一句话所言：smart is the new sexy。

然而人类进入工业文明才短短数百年，英特网的历史则更是短的几十年，和漫长的进化长河相比仿佛一瞬，我们匆匆忙忙把自己推入了一个完全不一样的世界，而进化的齿轮转动得却慢了很多拍，于是我们都成了进化时滞的牺牲品，我们用远古的双眼打量着这个世界，关在笼子里的老虎完全不必害怕，但我们还是会汗毛倒竖，汽车酿成的交通事故每天无数，而我们过马路却置若罔闻。一句话：我们的情绪大脑仍然停留在 20 万年前，而 20 万年前是没有汽车这种物种的。

然而，我们毕竟拥有所有动物中神经元和突触数量最多，结构最复杂的新皮层。我们拥有神奇的认识自身的能力，这种能力使我们能够利用情绪系统本身的特点来克服它自身的缺点。

我们做事情难以持之以恒地专注，因为任何一个新鲜刺激的外部信号都足以激活我们强大的情绪大脑，情绪大脑一旦被激活，其神经信号往往轻而易举地抢占我们的注意力，结果就是我们发现在这个纷繁的世界里很

难维护内心的宁静和专注，于是我们发明各种隔绝干扰的方法来保护我们脆弱的理性大脑，从而能够让自己做应该做的事情。

我们的大脑喜欢事情往积极方向发展，有这样一个实验：研究者让被试将手放在冰水中一段时间，有两个选项可供选择，一是将手放在非常冷的冰水中 60 秒并取出，另一是将手放在非常冷的冰水中 60 秒，然后再在逐渐变得不那么冰的冷水中放 30 秒再取出。绝大多数人认为第二种选择更为不那么痛苦。然而从经历的“客观”痛苦上讲，很明显第二种情况下人要受更多的罪。但是那种“情形正在变得越来越好”可以带来明显的正面情绪，于是第二种情况下的主观痛苦要小于第一种情况。而 GTD 的原理正是如此：通过提供不断的进展，让执行者意识到事情正在朝向完成不断迈进，这种正面趋势所带来的积极情感能够进一步激励个体把事情执行到底。

总之我们发明各种认知方法来“诱使”或“要挟”我们的情绪大脑同意去执行一件事情：我们向朋友承诺

我们要做的事情，于是我们的情绪大脑会迫于信誉受损的压力而去主动完成这件事情。而加入互助学习小组则本质上是利用大脑的从众本能和同侪压力。《Nudge》上提到很多这样的例子，比如“一百美元的论文催缴金”：为了“逼迫”自己在计划时间内完成论文，戴维将三张 100 美元的支票预先交给泰勒，戴维每延迟一个月完成论文，泰勒便可以提取其中一张支票并将钱用于开聚会( 而且还不邀请戴维参加:) )，显然，戴维的情绪大脑很难容忍这样吃亏的冤枉事，所以为了避免它，便忠实地督促戴维把论文按时搞定了。此外还有“减肥违约金”，“夏令时”，“圣诞节省钱俱乐部”以及著名的“Save more tomorrow”等很多有意思的例子。

最后，经常动用理性思考也能够锻炼理性大脑的“实力”，在更多的决策场合获得压倒性优势。神经科学显示，大脑的模块的确遵循用进废退的原则（《The Brain That Changes Itself》），一个经典的证据是钢琴家的大脑中对应手指的神经回路占用面积要比正常人大很多。另一个有意思的证据是，如果一个人失明了，那么负责接受

视觉信号的神经回路往往会被听觉所侵占（人们常说瞎子的听觉格外灵敏难道便是这个原因？）

上帝给了我们一个过了时的原始大脑，但同样也更新了我们的新皮层，能否不被 20 万年前的自己绑架，取决于你是否认识到关于大脑的进化历史，和能否正确使用你的理性大脑。

## （四）理智与情感

医学上，对于一些罹患严重癫痫症的病人，一种万不得已但颇为有效的方法是切断其大脑的胼胝体。胼胝体是两个脑半球之间的信息高速通道，含有 2 亿多条神经纤维，一旦切断之后，两脑半球之间的沟通也就相当于从信息时代回到了石器时代。经过这类手术的不幸的病人被称为“裂脑人”。对裂脑人的研究发现了关于人类大脑的一些非常重要的性质，例如《改变心理学的 40 项研究》第一章“一个脑还是两个脑”里面提到的研究揭示出左右脑在空间感，视觉触觉，语言方面的一些深刻而有趣的差异。

《How We Know What Isn’t So》里面则提到另一则非常有趣的有关裂脑人的研究：我们知道，语言能力主要在左脑。对于裂脑人，我们将两幅不同的图画分别呈现给裂脑人的左脑和右脑，呈现给左脑的图画上面是一只鸡爪，而呈现给右脑的则是覆盖着皑皑白雪的牧场。这之后，让他从一堆图片中寻找能跟他看到的图片相匹配的图片。结果非常耐人寻味：裂脑人的左手（右脑控制）会去选择一把铁锹（铁锹铲除牧场上的白雪），而其右手（左脑控制）则会去选择一只小鸡（小鸡和鸡爪配对）。两个脑半球分别根据自己所掌握的信息选择了最匹配的图片。然而，最有趣的地方在于当实验者询问被试为什么选择这两幅图片的时候。裂脑人会说：“哦，很简单啊，小鸡有鸡爪，而铁锹用来铲鸡屎”。

我们不妨停下来，想一想这个实验所揭示的一个令人深思的事实：铁锹之所以被选择，其实际的原因是为了和雪场匹配——铲雪。然而，铁锹和雪场的这个联系仅仅只有右脑清楚，而右脑是没有语言能力的。语言能力在左脑。当被试被要求对他的选择用语言进行解释的

时候，是左脑的 Broca 区域在说话，而左脑没法和右脑沟通，所以不知道铁锹实际是用来铲雪的，但是，左脑仍然还是不假思索的给出了解释，而且病人真心相信这个解释“很显然”，“很简单”。可见我们的理性大脑非常善于对自己的行为作出立即的，看上去合理的解释。

类似的一则关于超市购物的研究表明，人们倾向于购买货架上靠最右侧的袜子，尽管在实验中袜子是一样的。当被询问到选择的原因时人们会给出各种看似合理的解释，颜色质地什么的。可关键是，袜子是一样的！

以下这则经典实验( Maier 的双绳实验( two-cord experiment )) 则被很多心理学书籍所引用，包括著名的《心理学与生活》:

实验者将两根绳子拴在房间的顶上，两根绳子相隔较远，一个人站在中间往两边伸出双臂是没法子同时够到两根绳子的，同样也别指望拽着其中一根去够另一根。而受试者的任务就是要将这两根绳子的末端拴在一起。房间里有钳子，镊子，杆子，加长绳子这些工具。

许多被试很快想出用加长绳子的方法，但是 Maier 要求他们继续想更好的办法。绝大多数被试在 Maier 的要求之下都会最终黔驴技穷一筹莫展。就在他们一筹莫展几分钟之后，Maier 会在房间里走动并偷偷貌似“无意”地碰到其中一根绳子，使得绳子晃荡起来。在这之后，一般 45 秒钟之后便会有受试者表示想到办法了——他们迅速地将一个重物绑到其中一根绳子上，让它像秋千一样荡起来，然后跑到另一根绳子那，抓住它，然后等那根绳子荡过来。Maier 在他们“想出”解决方案之后立即就询问他们怎么想到的，超过 2/3 的被试给出了各种解释：“就是一瞬间，答案出现在脑海里了”，“这是唯一的可能了，要不还能怎样呢？”，“我忽然意识到可以把东西拴在绳子上让它荡起来”...其中一位心理学教授的答案最具创意：“在排除了所有可能之后，把绳子荡起来是剩下的唯一可能了，我脑子里想到通过绳子荡过河的场面，还有猴子从一棵树荡到另一棵。这个意象和答案在我脑中一同闪现。”

剩下的不到 1/3 的被试表示是看到 Maier 把绳子碰

动了才得到启发的。然而，即便是这 1/3 被试，他们真的是因为能够感知到大脑解决问题过程中的认知过程所以才这么说的么？为此 Maier 做了另一个实验，唯一的区别在于，在原来触动绳子的暗示之前，Maier 会先给另一个暗示（当然，被试不知道这是 Maier 的提示）：将一个重物绑在绳子末端。结果是令人吃惊的：这个暗示毫无效果，被试依然一筹莫展。接着 Maier 才给出原先那个暗示。并对解决问题后的被试作采访，这次，他发现所有被试都认为是 Maier 绑重物那个举动提醒了他们，并且他们都不认为是原先真正有用的那个暗示的功劳。可见这 1/3 的被试也并不知道在他们大脑中到底发生了什么。我们知道答案，却往往不知道真正的求解思维过程。

如果有人问你，你最好的朋友叫什么名字？一瞬间，他/她的名字便会“蹦入”你的脑海，毫不费力，你脱口而出。可是如果我问你，你是怎么得到这个答案的？你的回答无非只能是：“很简单，我记得啊”。可是你又是怎么记起来的呢？从你的耳蜗感受到“你最好的朋友叫

什么名字”这句话所产生的空气振动，并将其转化为神经电信号传入你大脑的听觉中枢，经过 Wernicke 区域，直到你的大脑从茫茫记忆中检索出这个唯一正确的答案，中间发生了何等复杂的过程，又岂是简简单单的“蹦出来了”可以解释的？我们的意识就像是一个等着老师给出答案的小学生，只能眼巴巴的在那等着，至于老师是怎么想出答案的，老师不告诉你。

我们的生活中面临很多或小或大的决策：和谁吃饭，什么时候去超市，买这款衣服还是那款，要不要去看那部电影，晚上是学习呢还是玩游戏，该不该主动接下一个有挑战性的工作，买房还是不买，抛掉还是持有，出国还是留下，等等等等。很多人可能会认为自己的决策是有充分的理性考虑的，是有充分的“理由”的，如果你事后问他/她，他/她会告诉你很多很多听上去很有道理的理由：“我今天需要休息一下，所以玩会游戏”，“政府不会让房价下跌的，所以我打算买了”，“现在神马股份还在亏钱，所以不能卖”。如果你问另一些人，他们也同样会告诉另一些听上去同样很有道理的道理：“玩游戏

没意思，还是学点东西好”，“不想拿父母多年的积蓄一下花掉，感觉自己在啃老”，“神马股份还不知道会跌到哪呢，还是撤了吧”。

这些听起来很有道理的道理，真的是驱使我们内心作出决定的理由么？很遗憾的是，很多时候答案是否定的。说要休息一下才玩游戏，其实真正原因也许是受到游戏中那些在现实中找不到的成就感的驱使。说玩游戏没意思，其实真正原因也许是最近现实生活中受了打击想要天天向上一把。说政府不会让房价下跌所以买房，也许真正原因是周围的人买了房而女朋友也在嚷嚷。说不想啃老所以不买房，真正理由可能是厌恶风险。说股票还在亏所以不能卖，真正理由可能是侥幸加贪婪心理，说股票还不知道跌到哪呢赶紧卖，真正理由可能是损失规避心理。

真正的理由有时候往往隐藏在意识触及不到的地方，由我们的情绪大脑所掌控，当它引导我们的情绪大脑作出决定之后，才发个红头文件通知我们的理性大脑，

我们的理性大脑于是便像文章开头提到的实验中描述的那样，迅速而果断地给出各种听上去很合理的解释，让我们的决定和行为看上去无懈可击。

晚上是玩游戏呢还是看书呢？你的情绪大脑果断给出答复——玩游戏。你其实理智上希望自己能够看看书，但在你强大的情绪大脑面前，你的理性大脑只能屈服，但你又不能让自己处于天人交战的纠结状态，所以你的理性大脑便用各种理由来搪塞自己："就玩一小会"，"人也要有休息嘛"，"今天玩，明天一定加倍学习补偿今天的时间"。

为什么投降的一方反而是代表着更高级认知能力的理性大脑？从进化角度来说，我们原始的情绪大脑早在远古的远古就已经存在并且为物种的生存繁衍作出卓越的贡献了（虽然大脑中的这一部分系统只能进行很简单的判断和条件反射，但他们无疑把守了对物种的持续存在而言最为基本且重要的一些功能——食物，性，自然环境中的危险，社会交互行为，道德感等等(《欲望之源》，

《进化心理学》)。它们的进化年代要比理性大脑深远的多，它们就像漫长岁月中伴随着生物一路进化走过来的老功臣，拥有强大的权力和力量，却没有意识到世界已经在最近的 5 百年发生了迅速和巨大的变化，这种变化对于几十上百万年的漫漫进化路来说只仿佛一瞬，然而就在这一瞬间，整个世界完全不一样了，可老功臣还没有来得及退休，还在掌管着我们的大脑，引导着我们作出各种跟不上时代的决策。(《逃出你的肖申克（三）: 遇见 20 万年前的自己》)

人的大脑并不是一个一蹴而就的整体设计，而是随着漫长的岁月在进化中被不断地添添补补，就像一台 7 拼 8 凑攒出来的电脑，CPU 是新的，主板却是老的，老的主板不能很好的兼容新的 CPU，结果 CPU 的性能便不能很好的发挥出来。可怜我们在进化上比较新近的新皮质（neocortex），拥有强大的计划能力和认知能力，但在一些原始诱惑面前却总是无法做到淡定。因为大脑中的这些原始模块还没有很好的和新模块兼容。如果你面前有一条毒牙被拔掉了的小蛇，你敢用手去抓么，抓

的时候你能不汗毛倒竖么？漫长的进化在我们的基因中刻下了一些“硬编码”的特性，比如对滑不溜丢游来游去的东西感到本能的害怕，因为自然环境中这些动物往往是有剧毒的，它们是生活在野外的先祖们主要的生命威胁之一。虽然你的理性大脑的强大认知能力让你确信你面前的这条蛇是没有毒的，完全不用担心，但是你的原始大脑却根本不理会。你的理性大脑能够运用语言，能够理解公式，而你的原始大脑却只懂条件反射，如果你多接触这些无毒的蛇，你多抓他们几次，久而久之你就会不怕了，这叫去敏感化，去敏感化是你的原始大脑所懂得的语言，只有这种方式才能较容易地说服你的原始大脑。同样的道理，我们常说不到黄河心不死，不见棺材不落泪。别人说破了嘴皮子的道理也没用，非要吃个大亏载个大跟头，跌得头破血流，才印象深刻从此不敢越雷池一步，为什么？因为你的原始大脑根本不懂那么多道理，它就是要遇到吃亏之后巨大的负面反馈才能习得一个条件反射。

只要我们的情绪大脑首先认定了一件事情，我们那

点可怜的理性思维便很容易屈从于情绪大脑发下的命令——把事情往利于自己的方向解释。

《谁会认错》里面便引用了一位社会心理学家在宣扬末日论组织里面“卧底”的见闻，该组织的成员为了能够在末日到来的时候被 UFO 接走，抛家弃子，放弃财富。而当那个被预言的时刻到来的时候，一切没有发生，心理学家发现，预言的破灭却并没有影响信徒们的信念一分一毫，他们的领头人解释道：因为那些相信者的诚意打动了上帝，所以毁灭没有发生。那些原本相信的，反而变得更相信，他们相信是他们的诚意避免了灾难，并努力地劝说更多的人加入。也有那些原本就不大信的，自然是更不相信。

类似的，《决策与判断》上提到过一个有趣的真实故事：1980 年的某一天，美国空战司令部的计算机突然发出警报——苏联的一枚核弹正在向美国本土飞来。司令部立即调兵遣将，迅速为一场核战做好了准备，然而 3 分钟之后，工程人员发现是计算机的一个小零部件故障

造成的。然而，这场虚惊之后，大众的反应才是真正有意思的：原先支持核武装的，认为现在感觉更加安全了（因为“事实证明这类的故障是完全可克服的”）；而原先反对核武装的则认为更不安全了（因为“这类错误信号可能导致苏联过度反应，引发真正的核战”）。类似的情况也发生在三里岛核泄露事件之后，同样的，反对者认为（“这表明管理部门没有办法安全管理核能”），支持者认为（“这正表明这样的危险没有想像得那么严重，是可克服的”）。

只要一件事情尚存在对自己有利的解释，我们的大脑便会毫不犹豫地掩耳盗铃地认为那就是唯一的解释。

慕容雪村在《中国，少了一味药》当中记录了自己在传销窝中待了近 1 个月的见闻。令人匪夷所思的是，传销者真正相信他们正在做的是一项国家暗中支持的事业。根据书中描述，典型论据有三：一，要不是国家暗中支持，怎么会有内部通话免费的集团号码。二，要不是国家支持，他们通过银行转账的操作怎么不被国家查

封。三，要不是国家支持，当地怎么那么多住房租给他们。再加上亲友说服，从众压力等各种手段之下，他们逐渐相信这是件大好事。在外界对传销的印象和事实的差别之中，也许最大的差别就是，和人们想象的不同，传销者并不挟持人生自由和财物，相反，他们想尽办法鼓励你去自己思考和判断！例如以上关于“国家支持”的论证，对于有常识的人来说漏洞很多，对于没有常识的人来说，只要有警惕心理，肯去调查，总能发现另一种解释。但是，对于内心希望这一切都是“国家支持”的所以自己就能真的赚几百万大钱的人来说，这些“有利”的证据便不会被过分深究和审查。所以传销窝中各色人等都有，包括见过世面的老江湖，甚至还有专门报导传销的新闻记者。人类心理的弱点之强大可见一斑。更匪夷所思的是，他们竟能把外面铺垫盖地的反传销宣传解释为是“国家宏观调控”，目的是为了不让人涌到这个行业中来，保证行业中的人的利益。“要不然，他们突击抓完人之后怎么给你买张票送你上火车之后就不管了呢——做做样子嘛”，他们相信这所谓“宏观调控”么？

相信，而且无比相信。只要一种解释是对自己有利的，我们便不想去推敲和反驳，再漏洞百出的事情看上去也不无可能，而且只要一种解释是有可能的，我们就认定其一定是的，强大的情绪大脑会阻止理性大脑去往深入了想。而对于对自己不利的解释，我们或者忽略，或者则会异常仔细去推敲，抓住一个漏洞则相信已完全推翻了该解释。

尤其是当人们为一件事情付出了金钱，社会关系，很多很多之后（这在宗教末世论组织和传销组织中何其相似），这些既有付出便会对他们的思维产生越来越强大的影响（经济学中的“沉没成本”便是如此），我们的思想被迫对自己的行为作出合理的解释（这就是著名的“认知失调”——这个心理学词汇已经如此有名，以至于出现在了呆伯特漫画中了），因为谁也不希望自己那么大的付出是愚蠢而错误的，为了让自己不是愚蠢且错误的，理性大脑不再是客观的代名词，而是一个唯唯诺诺为了维护自己情感的下属系统——“因为我们的信念感动了上帝，所以毁灭没有发生，这是唯一的解释”，至于那个

另一种解释，因为会不可避免地涉及到“我很愚蠢”这个结论，被人们的情绪大脑无情地驳回了。

一件事情总是有两个解释：一个平凡的解释和一个疯狂的解释（《逃出你的肖申克（二）：从视觉错觉到偏见》。而从自我辩护的角度看，一件事情总是有两种解释：一种对自己有利的解释，和一种对自己不利的解释。只要选择前者，我们便能够自欺欺人地将自己蒙混过关。

刘慈欣在《三体 II》里面提到“思想钢印”和“钢印族”，其实何必去设想那样一种能够改变人类大脑中神经元连接方式的机器。我们每个人大脑中都有思想钢印。这道钢印由经验打造，用自尊来维护，牢不可破，比钻石的硬度还要高。

社会心理学研究发现，我们会对那些对自己有利的证据不加细查，而对那些对自己不利的证据则死抠烂打揪住一点小辫子就不放；同样，我们还会倾向于勤劳收集有利证据，并忽视不利证据。事实是，当我们内心的天平早已经倾斜了之后，看来荒谬无比的理由也变得光

辉灿烂，别人很有道理的反驳也能被抠出“致命”漏洞。

因此，当你觉得自己想的很有道理，无懈可击，客观公正的时候，你是否真正像你认为的那样客观呢？Artemus Ward 曾经说过 :并不是那些我们不知道的事情让我们陷入麻烦，而是那些我们认定自己知道，却实际上是错误的知识，让我们陷入麻烦。客观意味着承认存在未知信息的可能性，理性意味着能够从对立面的视角去看问题和思考。学会质疑自己的判断，假设自己是站在对立面的立场上帮他说话，往往能够发现很多意料之外的东西；即便别人是错的，自己是对的，试着去理解错误的一方为什么会错，为什么会有那样的看法和认识，也往往能够得到很多有益的东西，你也许会发现自己的正确其实常常也是碰巧站对了队伍，而不像自己所认为的那样，来自于严密的逻辑和不可辩驳的证据。最后，与其让别人指出自己的错误，不如自己试着去发现自己的错误。

难道没有办法克服人类心理的天生漏洞么？有。大

脑符合用进废退的原理，越经常使用的区域会越来越强大。如果你总是情绪用事，不假思索，那么这种思维习惯便会越来越强大；如果你总是理性思考，反省自己的判断，能换立场去看问题，那么这样的思维习惯逐渐也会越来越强大。习惯之所以难以改变，就是因为习惯是自我巩固的，越用越强，越强越用。要想从既有习惯中跳出来，必然要依赖于外界的力量——对于心理机制的知识。仅仅是知道一些常见的行为陷阱和心理弱点的存在（《别做正常的傻瓜》，《决策与判断》，《Predictably Irrational》，《How We Know What Isn’t So》等等）便已经可以帮我们避免很多的决策失误。而如果能够进一步理解这些陷阱和弱点的深层原因（《Kludge》，《进化心理学》，《追寻记忆的痕迹》，《Simple Heuristics that Makes Us Smart》，《欲望之源》，《自私的基因》等等），则更可能说服自己做正确的事情。能够改变既有的习惯，依靠的不是自制力，而是知识。单纯的自制是一件非常痛苦的事情，你理智上知道应该怎么做，但是你的情绪大脑却就是不买账，一些比较坚定的人能够不管

三七二十一就强迫自己去做正确的事情，这殊为不易，不是像我这样的一般人能够做到的。但是，无论任何人，都有一个共同的倾向，就是去做正确的事情，不去做错误的事情。很多时候我们无法自制是因为情绪大脑并不知道也并不承认这件事情是错误的。举个最稀松平常的例子，去学习还是去玩游戏（并不是提倡不玩游戏，这里只是说在你希望自己能够不玩游戏做点别的事情的那些时候，你能够成功地实现自己的愿望而不是纠结半天并败下阵来。），理智上我们倾向于认为学习是件“好”事情，游戏则常常是件“不好”的事情，然而情感上，我们认为学习是痛苦的，游戏是开心的。而开心的的确确就是一件好事情，痛苦就是一件不好的事情。两个大脑模块的声音完全相反。如果你无法说服你的情绪大脑，那么所谓的自制就是强迫和纠结，天人交战，正如前文所说，最后败下阵来的也往往是理性大脑。然而如果你意识到对于游戏的热爱其实是完全正常的，我们玩游戏是为了获得群体认同感和成就感，对它们的追求早在几十万年以前就刻在了我们的基因上，获得群体认同和成

就是非常重要的优势。然而，由于这部分动机来源于我们相对原始的大脑，而后者的进化年代早在几十万年之前，在那个时候还没有网络，电脑，虚拟世界，虚拟货币这些东西，尽管我们的理性大脑能够认识到虚拟世界中的成就往往并不能转化为现实世界中的成就（电子竞技是一个反例），然而我们的原始大脑却无法区分虚拟和现实。意识到这一点之后，至少你就理解了为什么我们会受到这样那样的诱惑（我们对于高脂肪和高热量的无穷热爱也是如此——在先祖生存的贫瘠环境中，脂肪和热量是稀缺的，因而“尽量吃了存起来”几乎总是正确的），而当你进一步意识到自己无法自制的原因是因为你大脑原始的那部分仍然天真地认为你还处在石器时代的时候，你就会觉得任其驱使自己是愚蠢的事情，而我们的情绪大脑自然不希望自己是愚蠢的；而另一方面，认识到以上这些知识，认识到大脑的局限性，并最终摆脱它的错误驱使，则让人情绪上感到聪明和愉悦。于是我们就以彼之矛攻彼之盾，利用情绪大脑本身的动力来推动了它本身。

另一个类似的例子则来自于一项著名的心理学实验，该实验被称为“棉花糖实验”，其目的是建立儿童在延迟满足方面的自我约束力与日后取得个人成就之间的联系，但我想说的是实验当中那些成功地抵制住了棉花糖诱惑的儿童，这里有意思的地方在于为什么他们最终成功了，成功的原因并不在于棉花糖对他们的诱惑较小，对于这部分儿童而言，棉花糖的诱惑同样巨大，他们在抵制诱惑的时候显得异常痛苦，但他们的能耐在于他们通过各种各样的方法和技巧来分散自己的注意力，不让自己盯着棉花糖，让自己忙于干其他事情，通过这样的技巧，他们成功地将强大的刺激源从原始大脑面前移开，并且通过让自己忙于干一些其他事情来让大脑处于“忙碌”“被占用”的状态，阻止原始大脑老去往棉花糖上想。但是这跟学习心理学的好处又有什么关系呢？难道这些小孩在实验之前系统学习了进化心理学不成？他们显然没有。但他们所使用的方法恰恰是能够克服这些缺陷的方法。殊途同归的是，即便我们并非像一些有天分的人那样一开始就知道怎么对付自己内心的另一个声音，通

过学习一些基本的心理学知识，我们也能够后天地获得这些方法，而通过这些方法，我们便更可能成功地绕过甚至克服我们大脑天生的缺陷。这就是为什么我相信人人都该学点心理学的原因。

## 书写是为了更好的思考

我经常在走路和睡前总结所学过的内容，思考遗留的问题，一段时间的阅读和思考之后，一个总体的知识框架就会逐渐浮现在脑海中。然后我会将它书写下来，然而，我往往非常惊讶地发现，当我书写的时候，新的内容仍然源源不断的冒出来，就像我的键盘自己也会思考一样。

大半年前的时候，我曾在一篇文章《跟波利亚学解题》中写到将问题求解的思维过程记录下来的好处，现在再次回忆起来，当时列出的几点其实不仅对于问题求解是大有好处，对于平时的思考也是同样的道理。

书写的好处有以下几点：

•书写是对思维的备忘：人在思考一个问题的时候，就像是在黑暗中打着电筒往前走（事实上，我们的工作记忆资源是有限的，有研究证明我们只能在工作记忆里面持有 7 加减 2 个项目；此外认知负荷也是有极限的），每一步推导都将我们往前挪一小步，然而电筒的光亮能照到的范围是有限的，我们走了几步发现后面又黑了，想到后面就忘了前面的，想到某个分支上去就忘了另一个分支，我们常常想着想着就想岔了，想岔了也就罢了，问题是一旦想岔了太远，就很难回到当初岔开的地方了。有时候，我们是如此努力地试图一下就走出很远，同时又老是怕忘记目前已经取得的进展和重要结论，结果意识的微光就在一个很小的范围内打转，始终无法往前走出很远。而将思维过程记录下来，则给了我们完全的回溯自己的思维轨迹的可能。举个具体的例子，平时面对一个问题我们常常首先会想出几个主要的、关键的思考方向，但是这个时候如果没有笔记，就只能一个一个展开思考，结果展开思考了一个，却忘掉了第二点是什么了。如果记笔记，我就会先一二三的罗列出思考的关键

方向，然后逐一展开。思考任何一个分支的过程中有新的发现，但一时间没有剩余的思维去细想的话，就先用关键字记在一旁，一会回头再仔细思考。某种程度上这里笔记起到了备忘的作用。

•书写是对思维的缓存：正因为我们的工作记忆有限，所以我们在头脑中思考问题的时候就往往只能将几个最重要的核心概念保持在工作记忆中，导致想来想去在一个有限的范围内打转，思维总是走不太远。这方面我就有强烈的感觉，平时在走路的时候虽然也思考问题，但总是觉得思维的广度很有限。我们不妨设想数学家如果没有纸和笔的话，数学的发展会遭受到多大的阻碍，也许爱因斯坦能够在大脑中构思一个证明的最关键环节，但是你是否能够设想不用纸笔来“缓存”思维的中间步骤，而完全在大脑中证明费马大定理呢？有时候我甚至觉得能够用纸笔缓存思考的中间结果正是人类的理性之光能够走得如此之远的最重要条件。上一篇文章其实我原本的简记只有一半，另一半（更重要的那部分）却是在写成文章的时候自己冒出来的。

•书写是与自己的对话：在书写的时候，你不断地观察自笔端流出的信息，一行文字被你写下来之后，你就不用再将其费力地保持在大脑的临时记忆中，因为这行黑底白字会不断主动地通过视觉刺激来提醒你它的存在，于是你就可以将空出来的思维精力用于反思你自己的观点。不信你可以自己观察一下，如果不用纸笔，仅用大脑，是否很难在思考一个问题的同时对自己的思考进行反思呢？

•书写是与别人的交流：每个人的思维都有一些盲点，盲点之所以为盲点就是自己很难觉察得到，虽然我用了很长的时间来训练思维的客观和清晰，但总是不断发现自己的思维仍然还是时不时不自觉地陷入某个盲区，当我对人类思维的特点了解的越多，我就越是从心底里谦卑地认识到与人讨论是多么重要的一件事情，每个人的盲点不一样，你的盲点可以在别人那里得到补充，别人的盲点也可以被你纠正。三个臭皮匠顶一个诸葛亮的含义便在于此。写下来，与别人交流，最重要的价值就在于此。除了盲点之外，我们对于自己的知识体系中

的缺口一般是很难觉知的，如果自己的思考因为对某个重要知识的无知，犯了严重的错误，一般自己是难以反省出来的——如果你不知道一个东西，很大的可能性是你也不知道你自己不知道它。而把自己的思考写出来让别人发现漏洞，则是对自己知识体系的善莫大焉。

•有时候，语言自己也会思考：在没有付诸笔端的时候，思想在脑海中的存在形式往往较为模糊、抽象，有时甚至是图像的形式，然而，如果需要写出来，甚至写出来给别人看和别人交流的话，就必须使用文字符号，文字符号其实有自己的一套系统，计算语言学上称为语义网络，同一个概念，在大脑中模糊的感觉，和明确地表达成某个特定的词语，是不一样的。你会因为用了某个特定的词语从而想到另一个词语，你写着写着就会发现一些词语就像本身有灵性一样，将其他的词语都带出来了。有时候，这种效应会导致书写变成一场文字游戏，但好的一面是有些时候也是有益于拓宽或启发思维的。

在开始书写你的想法之前，我知道很多人不书写的

原因是因为觉得没有什么可写的，其实这是一个怪圈，你越是不开始书写，总是拿有限的思维缓存去默想一个问题，就越是没有内容可以写，如果你逼着自己将一些不成熟的想法写下来，看着自己写的内容，试着进一步拓展它们，就有可能在理性的道路上走得很远，很远。

祝大家书写快乐！

## 为什么你应该（从现在开始就）写博客

（一）为什么你应该（从现在开始就）写博客

用一句话来说就是，写一个博客有很多好处，却没有任何明显的坏处。（阿灵顿的情况属于例外，而非常态，就像不能拿抽烟活到一百岁的英国老太太的个例来反驳抽烟对健康的极大损伤一样）

让我说得更明确一点：用博客的形式来记录下你有价值的思考，会带来很多好处，却没有任何明显的坏处。Note：碎碎念不算思考、心情琐记不算思考、唠唠叨叨也不算思考、没话找话也不算思考，请以此类推。

下面是我个人认为写一个长期的价值博客的最大的几点好处：

1)能够交到很多志同道合的朋友。我自己既写博客，也读别人的博客，在这个时代，对于生活中的绝大多数人来说，拓宽朋友圈子的途径几乎只有一个，通过网络，而如何在网络中寻找到气味相投的朋友，如何判断别人和自己是否有共同语言？显然，通过天天在 SNS 上碎碎念的那些日记是难以做到的。我佩服的一些朋友几乎全都是长期用博客记录想法的人，因此，和他们即便不打照面，也是心照不宣。即便素未谋面也能坐下来就聊得热火朝天。

为什么博客在结交志同道合的朋友方面的潜力要远胜于原始的交谈方式？很简单，第一，博客无地域限制，整个互联网上从 A 到 B 只有一个点击的距离，而传统的建立朋友圈子的方法则受到地域限制。第二，也是更重要的一点，即如果按照以前结交朋友的方式，需要互相聊天，交流观点，然后才逐渐熟悉起来，这需要一个较

长的过程，而且更糟糕的是，当你遇到另一个陌生人，又要把整个过程重复一次，表达你已经对老友表达过的那番想法。可博客却做到了“一次表达，无数次阅读”，当我看到一个写了好几年的博客，看完了之后我仿佛和这个人交谈了很久，用程序员们喜欢听的话来说就是，“博客极大地增强了话语的复用性”。

我曾在 CSDN 上写了近六年的博客，在一年半前建立了一个 Google Groups（TopLanguage），由于我的博客的长期阅读者都是互相有共同语言的，因此这个 Group 一开始就热火朝天，而高质量的技术讨论则进一步吸引了更多的牛人的参与，雪球滚起来之后，就很难停下来了，将近一年半下来，从这个 Group 的讨论中我获益良多[1]。而对于非程序员朋友，科学松鼠会则是一个很好的例子。

2)书写是为了更好的思考。我在《书写是为了更好的思考》里面详细总结了书写的好处，这里就不拷贝粘贴了。有些想法如果不写下来，也就忘掉了，有一个广

为流传的《数学牛人们的轶事》（荣耀属于 ukim）里面讲了希尔伯特的一个故事：一次在 Hilbert 的讨论班上，一个年轻人报告，其中用了一个很漂亮的定理，Hilbert 说“这真是一个妙不可言（wunderbaschon）的定理呀，是谁发现的？” 那个年轻人茫然的站了很久，对 Hilbert 说：“是你……”。

3) “教” 是最好的 “学”。如果一件事情你不能讲清楚，十有八九你还没有完全理解。绝大多数人应该都知道在程序员行业面试官经常要求你讲解一个东西给他听，他会说他不懂这个东西（他如果真的不懂的话效果其实是最好的），而你的任务则是说到让他理解为止。

为了让一个不明白的人做到明白，你必须要知道从明白到不明白他究竟需要掌握哪些概念，这就迫使我们对我们大脑中整个的知识体系来个寻根究底，把藏在水面之下的那些东西统统挖出来，把大脑中的那些我们知道、但不知道自己知道的潜在概念或假设（assumptions）都挖出来，把它们从内隐记忆拉扯到外显记忆中。因

为只有完全知道、并知道自己知道一切来龙去脉的人，才能真正把一件事情讲得通通透透。

但是，你可能会怀疑，那除了能够讲清楚之外，弄清自己到底知道哪些东西还有其他什么好处吗？如果没有其他好处，那我又何必费这个劲呢？我又不当老师。

TopLanguage 上的一位朋友 sagasw 曾经讲了这样一个小故事：据说在某个著名软件公司里，开发组的桌上会放着一只小熊，大家互相问问题之前，先对着小熊把问题说一遍，看能不能把问题描述的清晰，基本上说的比较有条理以后，答案也就随之而来了。当然，你不一定要对小熊说，你可以在大脑中虚构一个听众，一个不懂行的听众，然后你说给他听。这是可行的，我经常在路上用。不过如果你能坐下来，我建议你还是说给实际的听众听——即写下你的思考，因为书写是更好的思考。

我们的绝大多数知识在绝大多数时候都隐藏在潜意识中，其实我们意识的窗口很小，我们的工作记忆只能

容纳寥寥数个条目（记得那个“看你能够记住屏幕上同时闪现的多少个数字”的 flash 小游戏吗？），我们平时所作的推理过程很大部分都是自动的，发生在潜意识中，而我们只能感知到一些中间结论。不信你回忆一下你在和别人讨论问题的时候有多少次觉得“反正就是这样，我感觉得到它是对的，但是你问我，我也说不清到底怎么回事”，对此你不觉得很奇怪吗？如果你都不能从逻辑上支持你的结论，你怎么就能确信它是对的呢？仅仅因为你的直觉强烈地告诉你它是对的？那如果旁边有另一个人，他和你持相反的观念，而他的直觉也强烈地告诉他他是对的。这时候你又怎么想？“他的直觉错了，我的直觉是对的”？难道你这么自信你的直觉是世界上最可靠的？

我自己则是非常珍惜类似这样的机会，即当“我强烈地觉得它是对的，但我却说不出所以然来”，这时候往往是到大脑中翻箱倒柜的时候，弄清来龙去脉的时候，深入反思的时候，纠正一直以来错误的潜在前提假设的时候。另一方面，“我强烈地觉得这个说法有问题，但我

却说不清它为什么有问题，到底哪有问题”，这也是一个极有意义的瞬间，它几乎总是意味着你对一个问题的认识有潜在的偏差，肯定是在你自己都没有觉知到的地方引入了一个潜在的假设、偷换了一个重要的概念，等等。而这种时候就是深入反思的时候，当你终于潜到问题的底层，触摸到问题的实质，把水面之下的冰山整体看清了的时候你会有一种通体舒泰的感觉。

为什么说以上这些？因为刚才说的是你必须等待这样的反思机会，但如果你选择经常总结自己的知识体系，并说出来给你的读者听，你就会发现你自己创造了这样的机会。如果我们平时不反思，我们觉得很多事情都是当然的，但结果如果要你一开口说给别人听，常常会发现事情就开始变得不那么明显了，你说着说着，就开始莫名其妙地发现自己需要用到“反正”这个词了。

于是，反思的机会就来了。

一旦你把自己潜意识里面的东西从幕后拉出来，你就有了面对并反思它们的可能，而不是任它们在幕后阴

险地左右你的思维。很多时候我们的思路出了问题并不是我们不会反思，而是不知道自己的思维中有那些隐含的假设（assumptions），如果你只感觉到答案，却不知道你大脑得到这个答案之前做了哪些推理，你又怎么知道哪一环可能出了问题呢？另一方面，一旦你弄清了自己到底是怎么想的，离意识到问题就不远了，很简单的道理——如果别人和你争辩的时候总是只摆立场，你就很难和他辩，但如果他把自己的推理过程原原本本暴露给你，批判起来总是容易得多的。（也正因为这个原因有很多人总是把逻辑藏在背后，不敢暴露出来）

绝大多数时候其实我们都会不假思索地得出一些结论，就像上了发条的自动机，但其实我们并不知道这些结论到底怎么来的，在思维的背后到底发生了哪些事情，故而当我们发现我们的结论错了的时候，一头雾水，没法着手寻找到底在哪错了。如果你注意一下很多人的发言（论坛、博客等等），如果你把他们的发言分为“前提”、“假设”、“逻辑”、“结论”这四个部分，你会发现一大堆人只会不停地下结论，摆立场，却见不到这些结论或

离场的前提、假设和个中逻辑，倒也不是他们不愿意写出逻辑，而是因为反思自己的思维过程实在是一件困难非常的事情，我们的推理过程很大一部分发生在意识的水面之下，只有当有了重要结论的时候这条逻辑链才会浮出来冒一个泡，让我们的意识捕捉到。更何况绝大多数时候我们用的其实并不是完整严密的逻辑思维，而是思维捷径。

去教一个完全不懂的人，则是一种最最强大和彻底的反思途径——因为他没有任何预备的知识，所以要让他弄懂你所知道的，你就必须彻底反思你的知识体系，弄清这座大厦的根基在什么地方，弄清它的骨架在什么地方，一砖一瓦到底是怎么垒起来的，你不能自己站在 11 层上，然后假设你的读者站在第 10 层，指望着只要告诉他第 11 层有那些内容就让他明白。你的读者站在第一层，你必须知道你脚下踩着的另外 10 层到底是怎么构造的。这就迫使你对你所掌握的、或之前认为正确的那些东西作彻彻底底的、深刻的反思，你的受众越是不懂，你需要反思得就越深刻。

4)讨论是绝佳的反思。另一方面，很多时候我们并不是有机会说给完全不懂的人听，更大的可能性是说给同领域有一定基础的人听，这个时候并不代表就不能促使反思了，实际上，你会发现，如果你公开你的想法，几乎总能看到与你持不同意见的人，然后你通过比较你和他的观念之间的差别，会发现你们在一开始的思路上就存在差异，差异从哪里来的？在进一步讨论中你们就会不断地迫使对方拿出更深层次的理由，这同样也是一种非常有效地促使自己反思的方法，在讨论的过程中双方的理由自然会变得越来越深入，越来越接近问题的本质，一些平时难以注意到的深层面的差异性就会逐渐浮现出来，你也就多了一次难得的机会去审视自己的思维中到底存放了哪些错误的信息。

5)激励你去持续学习和思考。如果你没有持续学习和思考的习惯，你的博客很快就会没有内容可写，就只能整点碎碎念或者转载，然后你就会失去读者，然后你就会关掉博客，然后一旦关掉博客之后你也就死了写博客的心，然后就少了一条激励你去思考和总结的途径，

然后你变得更不高兴总结和思考，然后...

为了打破这个死循环，不要永久停止更新你的博客，就算你两个月，三个月都不写，只要你每篇都是写自己思考的产物，写有价值的东西，在互联网上，金子的确总是会发光的，因为有无数的信息聚合平台在期待这些有价值的内容，有搜索引擎为你的内容提供海量的潜在读者，有海量的人肉在手动挖掘和转载那些有价值的东西。我们所能做的最差的一个决策莫过于停止做一件没有任何坏处，却有一大堆好处的事情。

为了让你的博客有价值，你必须不断总结自己学习的结果，你必须不断思考，给出比别人深刻、独到的见解。这看起来有点本末倒置，但很快本和末就会正过来。

6)学会持之以恒地做一件事情。很多人在生活中容易觉得迷失，不知道想要做什么，是因为没有一件能够持续地做的事情，用俗话来说就是没有主心骨。用积极心理学的话来说就是没有一件能够创造流体验的事情，而书写自己的思想则是一件容易产生流体验的事情，在

书写的时候，特别是理性地书写的时候，大脑逐渐进入推理分析模块，一切不愉快的情绪，烦躁感都会逐渐消隐下去。不过前提是你得开始，并且坚持过一开始的困难期，以后的一切便成了习惯成自然。

7)一个长期的价值博客是一份很好的简历。这里的“简历”并非是狭义上的求职简历，毕竟现在还没有到价值博客的时代，很多人写博客都是到处转载或者干脆碎碎念，正因此面试官未必拿个人博客当成了解一个人的更可靠窗口。这里的“简历”是指一个让别人了解自己的窗口，虽然我们未必做得到像罗永浩、Keso 这样的博客，个人的影响力已经足以支撑出一份事业( 牛博和 5gme )，但至少你会因此而结识更多的人，你的博客价值越高，你结识的人就越牛，跟牛人交流又会让你的眼界得到极大的开阔，打开一扇又一扇你原本不知道的门，于是你就变得更牛...这是一个良性循环。

（二）怎么做到长期写一个价值博客

注意到我并没有说“怎么做到长期坚持写一个价值

博客”，因为当思考和总结成为习惯之后，诉诸文字以及借助书写来进一步思考就变成了一件自然而然的事情，就变成了一件“因为你在思考和总结从而必须书写下来”的事情，博客就变成了副产品。

一开始的时候你是因为要写博客而去使劲地思考和总结，指望给出令人眼睛一亮的东西，到了后来，就变成了因为你习惯了思考和总结，因为你意识到书写是更好的思考，你就必须使你的想法成为文字。至此本和末就会各归原位，不再颠倒。

怎样做到长期写一个价值博客？也许有人会给出很多有趣有用的小技巧来提供动机和激励，譬如如何做 SEO，如何鼓励读者留言等等，但是这些我都不想说，我只想说最最重要的，那就是：

让你自己成为一个持续学习和思考的人，并只写你真正思考和总结之后的产物，其他一切就会随之而来。

就像那句经常被人传阅的话：只做你最感兴趣的事情，钱会随之而来[2]。

这方面的具体例子大家可以留意一下，随处可见，就不一一举了。我想再重复一下的是，千万不要碎碎念，我能理解每个人都想偶尔发发牢骚的冲动，但是现在已经有了一个很好的窗口：twitter，所以立即停止在你的博客上碎碎念，阅读博客的人希望得到信息而非噪音。如果实在忍不住想碎碎念的话不妨换一下位置，这么来告诉自己：如果你看到别人博客来上这么一段，你会有兴趣看吗？

（三）可能出现的问题以及怎样应付

即便上文给出了 N 条写博客的理由，但有时候只要一条不写的理由就会让人停止做一件事情。所以我特别加上一节“可能出现的问题以及怎样应付”，《影响力 2》[3]第五章雄辩地证明，“Much of Will is Skill”，意志力很大程度上来源于有正确的方法，而非天生。

1)担心别人认为没有价值。事实是，你面临过的问题总会有人面临过，你独立思考了，别人没有，你的文章对他们就会有价值。当然，肯定会对某些人没有价值，

他们早就知道了，但就算你再厉害，也总是有人比你厉害的，不能说因为这些原因就不记录你自己的想法了，你自己思考了之后理解得最深刻，就算有别人想过了，总有人没有想到的。况且，思考成了习惯，你的思考能力也会越来越强，你的文章也会越来越有价值。重复，无论你面临什么困惑，总会有很多人同样面临过，于是你苦苦思索之后的结果，肯定会对很多人有意义。

或者，你想通了之后觉得其实也很简单于是不愿意或者不好意思写了，但要知道，问题在想通了之后总是简单的，问题的困难程度不在于想通了之后还觉得有多难，而在于从你觉得它难到你觉得它简单需要耗费多少思维体力，你耗费的时间越长，说明有越多的人最终还是没有想明白（路越长走到底的人越少）。

最后，虽然我现在看一年前的文章觉得挺不成熟，但是如果没有那些不成熟的思考，也不会有现在更成熟的思考，我几年后来看现在写的东西，还是会觉得不成熟。

2)担心想法太幼稚或有漏洞等等被别人笑话。人非圣贤。正是因为单个人的想法总是有漏洞，才值得拿出来交流（《书写是更好的思考》，讨论是绝佳的反思），被别人指出问题正是改进的空间，藏着掖着的想法永远不可能变得更成熟。

Much of intelligence is knowledge，有这么一个非常发人深省的经典心理学实验[4]：

将孩子们分成两组，通过给他们不同的阅读材料让一组相信智力是天生的，不可在后天改变的，另一组则让他们相信智力其实只是知识和技能的代名词，完全是后天习得的。接下来让他们做一组任务，那些被相信智力天生说的孩子，倾向于回避困难的任务，选择较容易的任务，这里的逻辑想必是这样的：如果做困难的任务，就增大了失败的几率，就在降低了自己在别人和自己心目中的智力的值。为了保护这个智力的值不被降低，应该避免那些有失败风险的项目。而另一组孩子则对于有挑战性的事情跃跃欲试，并且在失败的时候明显没有前

者沮丧，因为失败也是学得新的东西，不管怎样都是“智力”的提高。

况且，只会批判乃至嘲笑别人的人是最不知道怎么建设的人，忽略他们。

3)得不到激励。这其实是个最无聊的问题了，只有写碎碎念的博客才会面对“激励”的问题。如果写自己的总结，写自己独立的思考，那么书写下来、理解通透，本身就是一个极大的激励。就算放在自己的私密笔记本里面也一样有成就感。况且，如果你真做到了书写价值博客，那么绝对不用担心你的观点得不到传播，也许一开始会耗时长一点，但是这在任何事情上都是必要的初始阶段，Gmail 小组的核心人物、FriendFeed 创始人 Paul Buchheit，和编程界名博 Coding Horror 的博主 Jeff Atwood 都曾经感叹过：Overnight success takes a long time ( (1)，(2) )，不过对于价值博客来说，现在网络上的聚合类服务这么多，机器的、人肉的、半人肉的都有，情况又要好得多了，而且我相信情况还会越

来越好。

4)写不出来。这个问题也比较无聊，思考本不是一件急于求成的事情。长期订阅我的博客的朋友知道我一般发文频率在一个月三五篇，实际上有不少次我个把月也不发布文章，原因很简单，要么是有手头的事情要处理思考的时间被压缩了，要么是遇到比较大或者比较困难的问题需要长时间的思考和积淀，没有关系，如果没有想清楚就再想想，爱思考的人和不爱思考的人有一个本质的区别，前者在生活中总是挂着几个问题在大脑中，它们时常都会冒出来骚扰你一下，让你琢磨琢磨，不爱思考的则是没事不主动想问题，遇到问题还要先想想是否能找捷径（找人帮忙）解决。

无论如何，不用急于求成，在一个主题上深入下去思考，总能挖到别人挖不到的角落。你能让一个问题在大脑中停留的时间越长，就越是能够发现新的东西，一般来说，我认为有价值的问题我会让他在意识或潜意识中待短则一个星期，长则一个月（视问题大小而定），利

用走路吃饭的时间琢磨（我发现很多我佩服的人也都有这个习惯），有时即便已经想通了写下来了发出去了，大脑仍然还是会在回味问题，还没有把它撤出潜意识，然后看到某篇文章或某本书的时候忽然又有所新的感悟。

能够把问题长时间停靠在潜意识中是一种技能，能够带来很大的好处，停留得越长你越琢磨得透彻，比别人看到的就越多。我们必须要带着问题的眼镜看待事物才能发现新的视角，否则就会出现视而不见效应，别的不说，广为人知的例子是阿基米德的“尤里卡！”，如果不是长时间琢磨着一个问题，一直把它放在思维中，是不会从洗澡领悟到“排水测体积”的，否则他洗了那么多年澡怎么不早发现呢？[5]

所以，如果你习惯了思考问题，就总会有东西写，先有思考，然后有总结，然后在总结中进一步思考。

当然你也可以试试把不成熟的想法写下来，试图整理成条理清晰的文字，然后看看能否在整理的过程中走得更远。这往往是可行的。比如这篇文章在我的简记里

面原本其实只有三行字（包含大约十来个备忘关键词），而最初在我的大脑里面其实只有一个走路时冒出来的问题——为什么要写博客？

–

[1]你可以看一下我收藏的一些精彩主题。http://delicious.com/pongba/toplanguage

[2]尽管我并不完全同意这句话本身，但它这种解决问题链上更基本环节的问题的精神是我赞同的。

[3]《影响力 2》这个名字起得很聪明，其实它并不是《影响力》的作者写的。

[4]我忘了这则实验的出处了，但实验的精神是记忆犹新的，哪位同学记得原始出处的麻烦提醒我一下。

[5]对于阿基米德这个故事的真实性是有争议的，毕竟几千年久远的事情谁弄得清呢。但是故事的道理是很本质的，我们平时也经常有类似的体验，加上阿基米德的“尤里卡”实在太出名了，所以我相信用用无妨。

# 我不想与我不能

事情开始往往是这样的：你发现自己想做某事，但你同时又迅速发现，自己并不擅长做这件事或做不了这件事。

于是“我想做某事”这个念头被打败并暂时搁置起来——要不怎么办呢？你反正又不擅长这件事。

一段时间过后，我问你，你想做某事吗？你回答说想，但随后又加了一句，可是做不来。

就这样在“想做”与“不能做”之间痛苦徘徊了一阵子之后，我又问你，你想做某事吗？

你的回答变成了，不想。

你内心发生了什么变化？

首先，“想做”与“不能做”这两个冲突的念头是难以共存的，它们如果一起存在于你的脑子里的话就会不停地折磨你。当你被折磨了足够长的时间之后你的内心就会作出一个选择，是改变“想做”还是改变“不能做”。

改变“想做”很简单，只要改成“不想做”就行了。而改变“不能做”则难多了，需要你“做到”这件事情。

于是你作出了一个决定，放弃“想做”。改为“不想做”。

最终你还是没有做成那件事情，但奇怪的是，你觉得你最终没有做是因为“不想做”，而不是因为“不能做”。这下你的理由就充分了，你就舒坦了，因为因不想做而不做某事，这是一个天经地义的理由。你不会承认自己是因为不能做所以不去做的。

可惜，事实是你把自己给骗了，为什么呢？因为你“不想做”的原因正是因为发觉自己“不能做”。你“不想做”并不是因为真正的不想做或没兴趣做，而是对“不能做”的一个妥协。

心理学上把类似这样的过程叫做“自利归因”。术语挺能唬人，但其实很简单，比如你做一件事失败了，你说“唉，都怪 XXX（这里的 XXX 可能代表人，也可能代表天气不好，情绪不好，路况不好这样的事儿）。”（归结

于客观原因，推卸自己的责任）。比如一件事情成功了，你说“还不都是我的功劳”（归结于自己，抬高自己）。简而言之自利归因就是把一件事情发生的原因归结为对自己有利的那种情况。用大白话说就是不能给自己难堪，不能让自己下不来台。功劳都给自己占，责任都给别人担。

其实，很多事情的发生都是多个原因共同造成的结果。比如你上班迟到，你的解释是“路况不好”，但除此之外还有另一个原因就是你比平时起晚了半小时，遇上了高峰塞车。这时，“起晚了”和“塞车”共同导致了“迟到”这个结果。无论取消“起晚了”还是“塞车”都不可能“迟到”。于是你就面临了一个选择，是把“迟到”这个事件的原因归结到“塞车”上，还是“起晚了”上呢？你选择前者，因为这样你就好过了，就不是你的错了。

果真就没有你的错了吗？别忘了，如果你起早半个小时的话，“迟到”这个结果同样也是不会发生的。

再比如，你今年考研没考上。你怎么好意思说自己不行呢？于是你说“唉，今年竞争激烈啊，报纸上说今年报考人数增加了 XXX，再说我们宿舍的那谁谁，不也没考上么？”。有人问你，那你事前复习得怎样呢？你怎么好意思说复习得不够刻苦呢？于是你就说“唉，复习怎样又有什么用呢？那谁谁，每天挑灯夜战，最后还不是挂了。”

但问题是，“没考上”是由两个因素共同造成的，一是考试本身的难度，二是你没有足够努力。无论这两个因素中的哪一个不成立都不可能造成那个结果。而你通过选择这两个原因中的那个客观原因作为解释，让自己好过了。

如果别人再问你：“那不是还有那谁谁，他不是考上了么？考试难大家不也一样难，人家考上了你咋没考上呢？”这时你貌似无路可退只能承认是自己技不如人了。可你居然又想出了一招金蝉脱壳，你说“他运气好啊。就算我复习得再好，考试这事情，总有个闪失的。复习

得再好，也可能挂掉啊”，你还理直气壮的加上一句“那谁谁，不就挂了么？”

但问题是，运气固然会造成考试失败，但是你没有去努力从而最大化成功的几率，同样也是会导致失败的。然而你却不会这么想，你怎么肯承认是自己不够别人努力的错呢？你会把这件事情归因于其中的那个与你无关的因素上，即是运气的原因。然而事实却是，你的确没有努力，这使得你本来可能享有的高成功率消失了。你通过否认这一点，缩进了自己造成的保护壳中。

而另一方面，一旦你说出“这件事不是我的原因”这样的话之后，除了心理好受了之外，你也开始相信这件事的失败真的与你无关，你自己这方面不需要作任何改变（因为你觉得不是你的问题），于是结果你下次同样也不会成功。

俗语有谋事在人，成事在天的说法。其实这话说的是，我们的努力是为了增大结果发生的几率，而不是为了那个确定的结果。不要奢望你努力了就绝对会成功，

你不会。更不要以这个理由来作为不去努力的原因，因为不去努力，那就永远不会成功。

增大成功的几率，本来就是我们付出时间的原因。

为什么这么简单的道理被人们一次一次忽视，并不是人们不懂。原因与这个道理本身无关，更与人们的智商无关。而是根植在我们内心的一个毛病：自利归因。

自利归因其实还有另一个表现形式。有一句很有名的话叫做“注意力等于事实”。这句话其实揭示了一个心理学上的深刻现象，用心理学领域的术语来说便是“观察者偏见所起的作用像一个过滤器”。用 M$的话来说就是“WYSIWYG”（所见即所得）。用唯心主义人择原理的话来说就是“世界是这个样子是因为我看到它是这个样子”。

我们每个人内心的观念都会对我们看待周围的事物起到一个滤镜的作用，过滤一切所见之物。扭曲它们以使它们符合我们内心的主观意识。心理学家们在进行心理学实验的时候，为了防止观察者的主观判断影响实验

的客观性发明了种种手法。

《20 世纪最伟大的心理学实验》里面提到这样一则发人深省的实验：当年的一个著名的心理学家，怀疑精神病院的医生们的主观判断会影响诊断的客观性，于是他和他的七个好友以身试法，他们花了一周时间，把自己各方面都装扮成精神病的样子，然后分头前往当地最大的七家精神病医院。由于他们装得很像，所以医生们都判断他们为精神病——这无可厚非。但接着令人吃惊的事实出现了，他们约好了，一旦住进了精神病院内，就开始立即表现出正常人的方方面面，也就是说，立即恢复他们原本的正常人状态。但结果，每次他们要求出院的时候，都被复查的医生阻止，医生们在他们的病历上写到“有明显妄想和心理强迫倾向；努力试图让别人认为他是正常人...”等各种带有主观倾向的病情判断。很显然，在医生们脑袋里已经形成的“他们是精神病人”的这样一个观念影响了他们作出客观判断，他们会将那个装成精神病人的心理学家的行为往符合他们既有观念的方向解释和扭曲。反而是精神病里面的真正的精神病

人看出他们不对劲，有一个家伙偷偷趴在他耳边说，“你不是精神病，你肯定是记者或教授”。这位心理学家出来后发表了一篇论文《精神病院里的正常人》，引起学界的轩然大波。

这事后来还有一个更好玩的续集——精神病专家们被他们的这次实验搞怒了，觉得尊严受到了挑战，于是悍然下出战书：接下来的 X 月内，你们尽管派人过来装丫挺的，我们一概用火眼金睛将汝等识破。但结果呢？X 月后 精神病专家们信心满满地说他们识别出了 XX 个假冒的，但实际上，那 X 个月中，对方一个人也没派过去。

话说回来，这事儿其实不新鲜，生活中一抓一大把。俗语云“公说公有理，婆说婆有理。”你敢于表达自己的观点，他说你喜欢炫耀。你韬光养晦呢，他又说你夹着尾巴做人；你为自己的成就骄傲一下，他说你得意忘形。你低调做人呢，他又说你怎么这么没种；你说彪悍的人生不需要解释，他说你自大狂。你说做人贵有自知之明呢，他又说你丫做人怎恁费劲捏？...

再比如，你人生不得意。大学毕业一两年，公司换了三四家，薪水挣了五六千，失恋闹了七八次。总之就是似乎天底下郁闷的事都栽你头上了。你想是不是得把老家那房子挪挪窝搞不好原来的风水不对头刚好盖在上帝他老人家的 WC 正下方了。你想老子怎么运气就他妈那么背馅饼都掉人家头上去了搁我这却给掉下一大坨圆锥体下来。但如果仅因为这些原因就一定会造成你遇到的结果的话，那些从困境中走出来的牛人也就不会存在了。

事实上，人生不得意往往由两个原因造成，一个是外因，即你抱怨的那些原因，另一个则是内因。无论哪一个因素不成立，都不会造成现在的结果。如果你运气好了，手头股票一星期飙升三倍，你或许一下飞黄腾达。但外界原因很大程度上是你不可控制的，你死了都不卖结果真死了也说不定。能控制的是另一个因素，就是你自己的主观能动性。无论外因如何，只要不是到了人力所不能抗的地步，发挥一下自己的自由意志和主观能动性，总能获得同样好的结果。GFW 纵有“中国网，封天

下”之能事，我们就不能用 tor 了吗？只要不是海底线缆断掉，而且即便海底线缆断了还有无线上网呢不是，慢点是慢点，但在客观条件限制之下已经是最好的选择了。另一方面，如果你想改变客观现实，那也是用自己的行动，而不是抱怨。

“兵来将挡，水来土掩”，正是人类自由意志和主观能动性的充分体现。如果你“兵来腿软，水来上船”，然后给自己找一大堆唧唧歪歪的理由。除了表明你没能耐之外不能表明其它的任何东西。所以你千万要记住，别给自己找理由，因为那等同于说“我不能__，我没有__”（填空自己填），等于否认自己的自由意志和主观能动性。凡事多往自身归因，你才能披荆斩棘。才能“漫漫路在何方，路在脚下。”

## 遇到问题为什么应该自己动手

1.遇到问题寻找捷径为什么是很聪明的做法

我们在生活中总是在不停地试图做最优经济决策，只不过很多时候我们为适应远古社会而进化的大脑未必

适用于现代工业社会(《Mean Genes》,《进化心理学》,《How We Decide》),所以很多时候我们可以在超市为选择哪一卷卫生纸斟酌半天(《Predictably Irrational》),却在面对生活中重大抉择的时候轻易就随波逐流(《Paradox Of Choice》)。

我们的很多决策依赖于情绪系统的输出(从进化时间上比较“旧”的大脑部分)(《How We Decide》,《Synaptic Self》),这部分大脑属于典型的经过了漫长进化时间所雕琢过的,决策机制严重适应远古社会的模块(《Mean Genes》),比如在物质贫乏的远古时期,不管什么时候遇到富含热量的食物是必吃无误的,所以我们的情绪大脑只要闻到美食是绝对不去克制诱惑的,长出脂肪又如何?有的是饥寒交迫的时候去燃烧这些脂肪。然而这条规则到了现代这个物质充裕的社会却成了灾难(去查一下美国的肥胖比例?),可谓成也萧何败萧何。这样的例子在《Mean Genes》中还有不少。

我们在学习新东西,遇到困难的时候,为什么会放

弃？因为我们下意识中会对所面临的困难以及成功后所得的收益作一个评估（经典的 cost/return 分析），这里特别重要的是对面临的困难的评估：我们都知道学习任何一门技能，一开始可能还兴趣浓厚，捋袖子上阵，过了一阵子便会遇到一个典型的分水岭，你会发现未知的东西比你想象得要多，困难重重，似乎一眼看过去没法确信什么时候才能掌握，甚至觉得有点 Mission Impossible，当觉知到的困难到一定程度之后，我们的大脑便会想：既然很大可能最终失败，甚至看不到成功的可能，为什么要白费力气去学一通呢？还不如省省呢。这是一个聪明的经济决策，去权衡性价比应该是每个经济个体的原则。然而，这个决策笨就笨在，它把困难评估得过高了，因此决策的前提就弄错了。为什么这么说呢？现代社会很多新东西是知识密集型的，而不像我们祖先生活的远古社会可能绝大部分是体力活。对体力活的评估我们很在行，大约能知道困难有多大，需要耗时多久，有没有可能完成。然而对学习新知识的困难程度的评估，我们却很不在行，因为大部分知识都是需要等你掌握了

之后才会“豁然开朗”、“柳暗花明的”，而在这之前你会觉得这东西太难了，完全没有头绪，摸不着门道，觉得山重水复疑无路，你会想“既然无路，就别去碰得满头是包了吧？何苦呢？”。

有一个很不错的概念叫做“Unknown Unknown”，大意是如果你不知道一个东西的话，你也不会知道你自己不知道它。很多时候新知识就有这个特性——掌握了之后觉得很明白，掌握之前却觉得“不可能啊”、“这简直没有解嘛”。在这样的认知之下，你自然会高估前方的困难、风险和不确定性，因为你不知道什么样的知识才能解决你的困惑。然而事实上呢？只要智商没有根本的差别，别人的大脑能够掌握的知识，你的大脑也能掌握，你所感觉到的巨大困难只不过是因为 Unknown Unknown，你所需要的只是耐心地踏遍这块知识版图，当你掌握了那些你该掌握的知识之后自然会柳暗花明。

2.遇到问题寻找捷径为什么只是小聪明

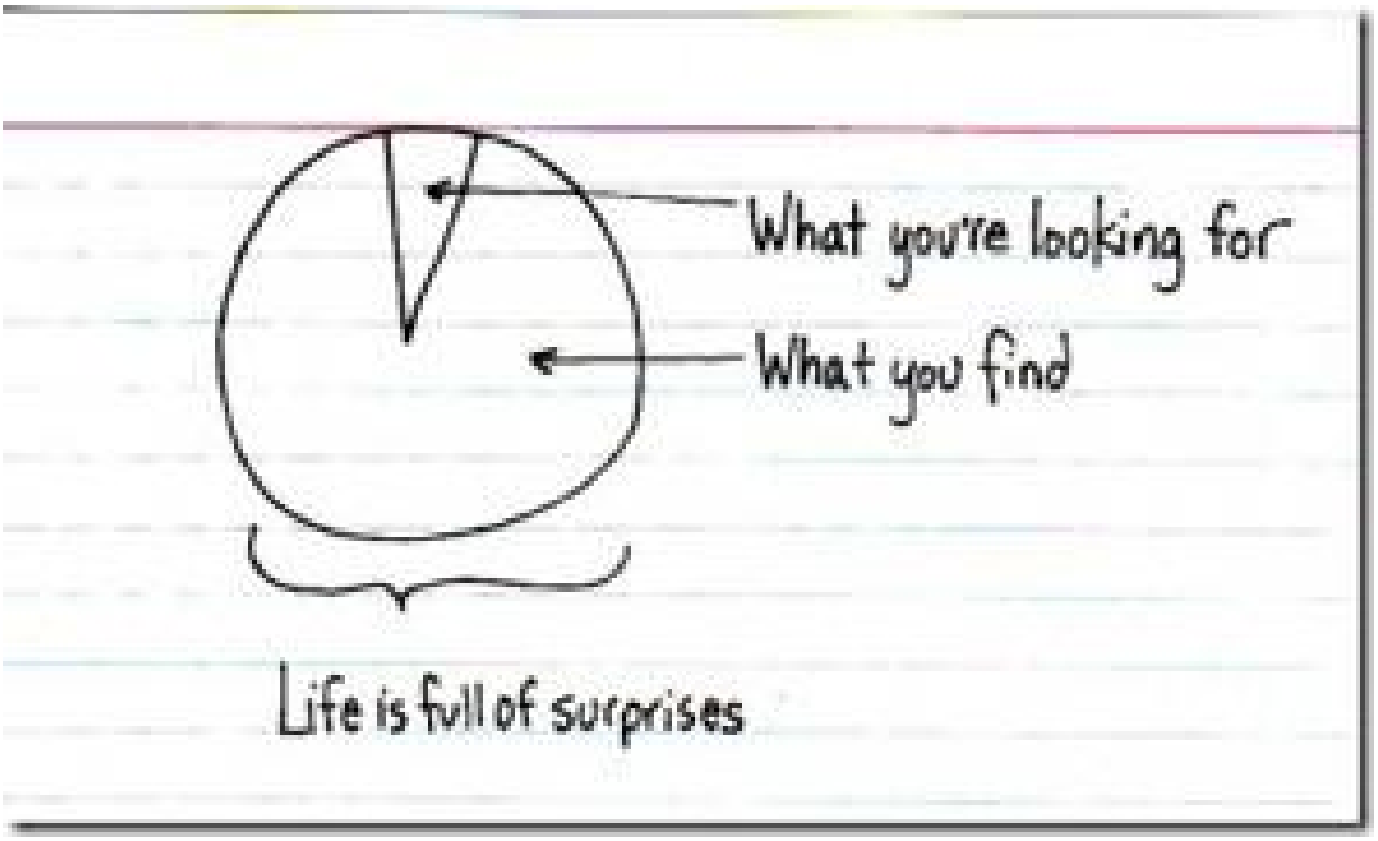
我们在遇到困难的时候会试图去寻找捷径，心里的

想法大概是：既然我自己解决可能需要耗费极大的精力，甚至连最终能否解决都无法判断，那么为什么要冒风险花费大量的时间去尝试呢？还不如想想其他法子。比如绕过问题，或者将问题外包给别人。

这很聪明，很经济：用最小的代价解决手头的问题。看上去是一个寻求经济上最优解的法子。

不过到底是局部最优还是全局最优呢？

“用最小的代价解决手头的问题”——这里的问题在于，难道我们计算收益的时候仅仅考虑是否解决了手头的问题吗？如果解决的过程中得到了其他的收益呢？




（图片注：荣耀属于 indexed）

为了解决一个技术问题，你踏遍互联网，翻了若干教程、网站、书籍，最终解决了这个问题的同时还知道了以后遇到类似的问题该到哪儿最快最有效地找到参考，你还知道了哪些网站是寻找这个领域最有价值信息的地方，你还知道了哪些书是领域内最经典的书，说不定你在到处乱撞的过程中还会遇到其他若干意想不到的收益。

为了解决一个内存泄漏的 bug，你学习了一堆底层

知识、了解了一堆调试工具、学习了若干 wikipedia 页面，表面上看来，仅仅为了解决这一个小 bug 你的时间花销未免太大了点，然而关键就在于，它的收益远远不止于解决了这一个小 bug，下次你遇到任何类似的 bug 的时候就能够哐当两下就解决之了。

生活或工作中，很大程度上你遇到的每个问题都不是孤立的，既然你遇到了某问题，那么很大的可能性你以后还会遇到类似的问题。当然，这个说法的另一面是，也有一些问题是一锤子买卖，即以后不会遇到类似的问题，因此只求速解决。不过按照我的经验这样的问题实在太少了，此外，你觉得你真的能够分辨你面对的问题是否属于这类问题吗？底线是，就算是这样的问题，你自己动手解决也能培养学习能力和思考能力。如果你判断它是一锤子问题，外包给别人解决，那么你就永远没机会发现这个问题背后蕴藏着哪些知识，这就成了一个自我实现的预言。

如果选择总是问别人的话，下次你还得继续问别人，

每次直接问到问题的答案的同时意味着你永远都要靠别人的大脑来获得答案。

困难的路越走越容易，容易的路越走越难。

## 什么才是你的不可替代性和核心竞争力

我虽不是经济学专业，但是翻开任何一本经济学的教材，或者直接翻开 wikipedia 的 economics 条目，都会看到物以稀为贵这条铁律。人才作为资源的一种，也是同样的道理。而稀缺性，换种说法也可以叫做不可替代性。一种资源越是稀缺，不可替代性就越强。再加上如果这种资源是一种具有实实在在使用价值的东西（而不是荷兰的郁金香泡沫），那么其价格就会越高。

问题是，如何构筑你的个人知识体系，使得你的知识技能集尽可能成为不可替代的呢？

CSDN 的孟岩先生前段时间发表了一篇博客“技术路线的选择重要但不具有决定性”，用有说服力的数据阐述了技术路线的选择对于个人知识体系的不可替代性并

非一个关键因素，文中也提到了这样一段话：

那么核心竞争力是什么？我观察圈子里很多成功和不成功的技术人，提出一个观点，那就是个人的核心竞争力是是他独特的个性知识经验组合。这个行业里拥挤着上百万聪明人，彼此之间真正的不同在哪里？不在于你学的是什么技术，学得多深，IQ 多少，而在于你身上有别人没有的独特的个性、背景、知识和经验的组合。如果这种组合，1，绝无仅有；2，在实践中有价值，3，具有可持续发展性，那你就具备核心竞争力。因此，当设计自己的发展路线时，应当最大限度地加强和发挥自己独特的组合，而不是寻求单项的超越。而构建自己独特组合的方式，主要是通过实践，其次是要有意识地构造。关于这个观点，话题太大，我不打算赘述。

孟岩先生在文中没有对这个问题展开叙述。但我一直也在寻思这个问题，后来在 TopLanguage 上一次讨论的时候，把一些想法整理成形。

长话短说，我相信以下的知识技能组合是具有相当

程度的不可替代性的：

专业领域技能：成为一个专业领域的专家，你的专业技能越强，在这个领域的不可替代性就越高。这个自是不用多说的。

跨领域的技能：解决问题的能力，创新思维，判断与决策能力 ,Critical-Thinking ,表达沟通能力 ,Open Mind 等等。

学习能力：严格来说学习能力也属于跨领域的技能，但由于实在太重要，并且跨任何领域，所以独立出来。如何培养学习能力，到目前为止我所知道的最有效的办法就是持续学习和思考新知识。

性格要素：严格来说这也属于跨领域技能，理由同上。一些我相信很重要的性格要素包括：专注、持之以恒、自省（意识到自己的问题所在的能力，这是改进自身的大前提）、好奇心、自信、谦卑（自信和谦卑是不悖的，前者是相信别人能够做到的自己也能够做到，后者是不要总认为自己确信正确的就一定是正确的，Keep a

n open mind）等等。

关于如何培养这些方面的能力，呃..需要学习的东西太多，对于第 2 项中列出的一些子项，可以参考我上次列的一些资料（《如何清晰地思考》），我自己也在学习之中。另外我在《一直以来伴随我的一些学习习惯》中也提到了一些相关的方法。

注：

以上将个人的核心竞争力分为 4 个部分，其中每个部分的罗列并不一定详尽，也有可能我忽略了重要的东西或罗列了不重要的东西，所以欢迎补充和纠正。

以上只是我个人所认为的具有相当程度不可替代性的知识技能集，至于是否有更具不可替代性的“装备”，不妨思考。

## 第三篇 跟波利亚学解题

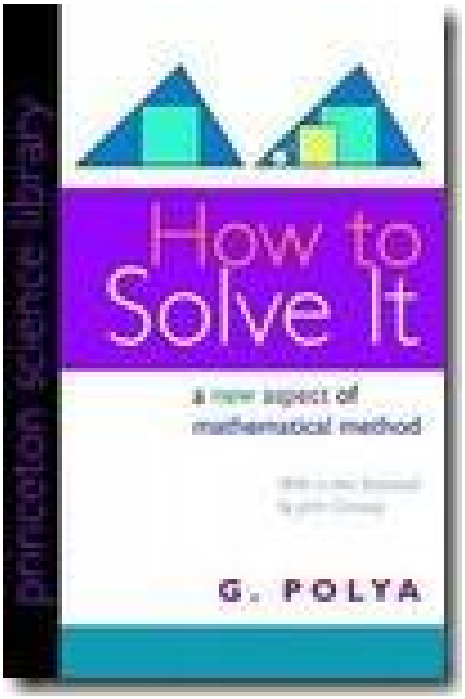
## 跟波利亚学解题

一些故事

波利亚在他著名的《How To Solve It》中讲了这么一个有趣的心理学实验：

用一个缺了一条边的正方形围栏围住一只动物（狗、黑猩猩、母鸡、人类婴儿），在围栏的另一侧放上一个被试很想要的物体（对动物来说是食物，对人类婴儿来说是有趣的玩具），然后观察他们各自的行为。发现，狗在扒着围栏吠了几声发现无法通过的时候，不久便学会了从围栏的缺口的那一边绕出去，母鸡则朝着围栏一个劲的扑腾，不会想到绕弯子。此外，人类婴儿很快就学会了绕过障碍；而黑猩猩也学得很快（黑猩猩是和人类最近的灵长类亲属）。这个实验有力的证明了，动物解决问题的能力是进化而来的、天生的、硬编码在大脑的神经元网络里面的。

事实上，不仅解决问题方面是如此，人类整个认知系统中绝大部分功能从本质上都是硬编码的，能在后天习得的只是“程度”的不同，而不是“本质”的不同。《动机心理学》中有一个令人印象深刻的一个例子：




先给小鼠喝某种甜味水（称为“可口水”），然后用 X 射线促使其产生反胃感，能使小鼠形成对这种味道的水的厌恶和回避（经典条件反射）。但如果不是在水里面加味道，而是在它喝水的时候伴随强光刺激（即让它喝“光噪水”），然后同样刺激其反胃，却无法使它养成对“光噪水”的厌恶。另一方面，如果不是促使其反胃（身体不适），而是用电击惩罚，则它无法形成对“可口水”的厌恶，而是形成对“光噪水“的厌恶。显然，小鼠对事件之间的关联的归因也具有着某种硬编码好了的倾向。在这个例子中，老鼠的大脑里面硬编码了“将身体不适（内部事件）归因于食物而不是闪光”、” 将电击（外部

事件）归因于闪光而非食物”这种逻辑。

而人类也有类似的归因倾向。金出武雄在《像外行一样思考，像专家一样实践》中也提到，他认为人类的直觉实际上也是计算，捷径式的计算，只不过由于我们目前还不了解人类大脑内神经元的全部结构( 或者说“感性”的物质基础）这才把“感性”当成人类所特有的；金出武雄的这种观点跟心理学中的认知捷径不谋而合。实际上，越是高等的动物，大脑中用于处理特定问题的硬编码神经元回路就越是多和复杂。例如，达尔文早在《人类和动物的情绪表达》中就先知先觉的提出了动物情绪的适应价值；《Mean Genes》列出了用于解决生存繁衍问题的特定认知倾向；《决策与判断》里面则列出了人类在解决更具一般性的决策问题中的一些系统性的、可预测的认知偏差；而《Predictably Irrational》更是把这个认识提高到方法论的层面，主张人类的非理性实际上是完全可预知的。事实上，所有这些观点都建立在一个基本事实的基础上，即人类大脑中的千亿神经元是由在漫长的进化过程中被塑造出来的分工明确的、ad h

oc 的一组子系统构成的。

越是高等的动物，解题能力越高，猩猩能够进行某种顿悟，在脑子里就构想出通过堆放墙角的箱子来帮助获取高高吊着的香蕉；而出于进化之树顶端的人类则具有非比寻常的大脑，在人类整个进化的过程中，解决问题的能力一直在进化，所以说人脑中的神经元最重要的部分是为了解题而存在的也不为过。不同的人只是在解题能力程度上不同，并没有本质上能与不能的差异。

波利亚在《How To Solve It》中另外还举了下面这个例子：

一个原始人站在一条小溪前，他想要越过这条小溪，但溪水经过昨天一夜，已经涨了上来；因此他面临一个问题：如何越过这条小溪。他联想起以前曾经从一棵倒下并横在河上的树木上走过去，于是他的问题变成了如何找到这样一颗倒下并横在溪流上的树木。他环顾四周，发现溪流上没有这样的横着的树木，但他发现周围倒是有不少生长着的树木；于是问题再次变成了：如何使这些树木躺到溪流上。


在这个想像的故事中我们看到了一个问题是如何被一步步归约的：首先，原始人通过对一个已知的类似问题的联想认识到一个重要的性质：如果有一棵树横在河

上，我就可以借助这棵树过河。这就将一个无法直接解决的问题转化为了一个新的、已知的、并容易解决的问题。值得注意的是这里“联想”是极其重要的一个环节，联想可以将手上的问题与已知的类似问题联系起来，并从后者中吸取能够利用的方法。联想也能够将与问题有关的定理或性质从大脑的知识系统中提取出来。基本上，如果一个联想能够得到某个性质，而这个性质能够或者将问题往上归约一层，或者将条件往下推导一层，这个联想就是有用的。事实上，如果你仔细注意以下解题的过程，你也许会发现，所有的启发式思维方法（heuristics）实质上都是为了联想服务的，而联想则是为了从我们大脑的知识系统中提取出有价值的性质或定理，从而补上从条件到结论、从已知到未知之间缺失的链环。

一段历史

实际上，人类自从进入理性文明以来，不仅在不断的解题，还在不断的对自身的解题方法进行反省和总结。在这条路上，有一个真正光荣与辉煌的梦想，那就是发

现人类解题的所有一般性法则，并借此建造出一台能够解决人类能够解决的所有问题的一般解题机。与物理中的建造永动机不一样，这个梦想并非遥不可及的，自从古希腊哲学家对人类心智的反省思考以来，许多著名的数学和哲学家为此建造了阶梯，Pappus，亚历山大学派最后一位伟大的几何学家，就曾在他恢弘的八卷本《数学汇编》中描述了其中的一种法则，他将它称为“分析与综合”，大意如下：

首先我们把需要求解的问题本身当成条件，从它推导出结论，再从这个结论推导出更多的结论，直到某一个点上我们发现已经出现了真正已知的条件。这个过程称为分析。有了这条路径，我们便可以从已知条件出发，一路推导到问题的解。

波利亚在他的三卷本中把这种做法叫做 Working Backwards（倒过来解）。

笛卡尔也曾经试图将人类思维的规则总结为 36 条（最终完成了 21 条）。莱布尼兹，现代计算机实质上的

发明者，也说到：

在我看来，没有什么能比探索发明的源头还要重要，它远比发明本身更重要。

再后来，捷克数学家波尔查诺也试图总结人类思维的本质规律，他在他的著作《科学的理论》中写道：

我根本不奢望自己能够提供任何超于其他天才所使用过的科学探索方法之外的新方法，从这个意义上，你别指望能在书中看到什么新的东西。但是，我会尽我的全力去总结所有伟大的思想者们共有的、思维的原则和方法，我认为即便是他们自己在思考的时候也未必全都意识到自己在使用什么方法。

再后来，就到了近代，随着科学技术的进步，心理学最活跃的子学科——认知科学——开始辉煌起来，人类开始向思维乃至自我意识的物质基础发起进攻。两位多才多艺的计算机科学家兼认知科学家，Herbert Simon（另外还是经济学家）和 Allen Newell 写出了世界上第一个一般性解题机的程序（GPS），虽然 GPS 只能解

决很狭窄的一类问题，但这是第一个将“问题解决策略”和“知识”分离开来的程序。显然，在知识之外，人类的思维是有着一些一般性的指导规则的。事实上，波利亚在《数学与猜想》中写道，欧拉是最重数学思维的教学的，欧拉认为如果不能把解决数学问题背后的思维过程教给学生的话，数学教学就是没有意义的。

一些方法


这些一般性的思维方法，就是波利亚用了整整三本书，五卷本（《How To Solve It》、《数学的发现》、《数学与猜想》）来试图阐明的。波利亚的书是独特的，从小到大，我们看过的数学书几乎无一不是欧几里德式的：

从定义到定理，再到推论。是属于“顺流而下”式的。这样的书完全而彻底的扭曲了数学发现的真实过程。举个例子，《证明与反驳：数学发现的逻辑》在附录一中讲了一个非常有趣的例子：柯西当年试图将函数的连续性从单个函数推广到无穷级数上面去，即证明由无穷多个连续函数构成的收敛级数本身也是一个连续的函数，柯西给出了一个巧妙的证明，似乎漂亮地解决了这个问题。然而傅立叶却给出了一个噩梦般的三角函数的收敛级数，它的和却并不是连续的。这令柯西大为头疼，以至于延迟了他的数学分析教程的出版好些年。后来，赛德尔解决了这个问题：原来柯西在他看似无懈可击的证明中非常隐蔽（他自己也不知觉的情况下）引入了一个潜在的假设，这个假设就是后来被称为的“一致收敛”条件。当时我看到这里就去翻我们的数学分析书，发现“一致收敛”这个概念第一次出现的时候是这样写的：定义：一致收敛...

所以说，从这个意义上，《数学，确定性的丧失》从历史的角度再现了真实的数学发展过程，是一本极其难

得的好书。而事实上，从真实的数学历史发展的角度去讲授数学，也是数学教学法的最佳方法。不过，《数学，确定性的丧失》的弱点是并没有从思维的角度去再现数学发现的思维过程，而这正是波利亚所做的。

总结波利亚在书中提到的思维方法，尤其是《How To Solve It》中的启发式思考方法，有这样一些：


•时刻不忘未知量（即时刻别忘记你到底想要求什么，问题是什么。）莱布尼兹曾经将人的解题思考过程比喻成晃筛子，把脑袋里面的东西都给抖落出来，然后正在搜索的注意力会抓住一切细微的、与问题有关的东西。事实上，要做到能够令注意力抓住这些有关的东西，就

必须时刻将问题放在注意力层面，否则即使关键的东西抖落出来了也可能没注意到。

•用特例启发思考。一个泛化的问题往往给人一种无法把握、无从下手、或无法抓住里面任何东西的感觉，因为条件太泛，所以看起来哪个条件都没法入手。一个泛化的问题往往有一种“不确定性”（譬如元素的个数不确定，某个变量不确定等等），这种不确定性会成为思维的障碍，通过考虑一个合适的特例，我们不仅使得问题的条件确定下来从而便于通过试错这样的手法去助探问题的内部结构，同时很有可能我们的特例中实质上隐藏了一般性问题的本质结构，于是我们便能够通过对特例的考察寻找一般问题的解。

•反过来推导。反过来推导是一种极其重要的启发法，正如前面提到的，Pappus 在他的宏篇巨著中将这种手法总结为解题的最重要手法。实际上，反向解题隐含了解题中至为深刻的思想：归约。归约是一种极为重要的手法，一个著名的关于归约的笑话这样说：有一位数学家失业

了，去当消防员。经过了一些培训之后，正式上任之前，训练的人考他：如果房子失火了怎么办？数学家答出了所有的正确步骤。训练人又问他：如果房子没失火呢？数学家答：那我就把房子点燃，这样我就把它归约为了一个已知问题。人类思维本质上善于“顺着”推导，从一组条件出发，运用必然的逻辑关系，得出推论。然而，如果要求的未知量与已知量看上去相隔甚远，这个时候顺着推实际上就是运用另一个启发式方法——试错——了。虽然试错是最常用，又是也是最有效的启发法，然而试错却并不是最高效的。对于许多题目而言，其要求的结论本身就隐藏了推论，不管这个推论是充分的还是必要的，都很可能对解题有帮助。如果从结论能够推导出一个充要推论，那么实际上我们就将问题进行了一次“双向”归约，如果原问题不容易解决，那么归约后的问题也许就容易解决了，通过一层层的归约，让逻辑的枝蔓从结论上一节节的生长，我们往往会发现，离已知量越来越近。此外，即便是从结论推导出的必要非充分推论（“单向”归约），对问题也是有帮助的——任何

不满足这个推论的方案都不是问题的解：譬如通过驻点来求函数的最值，我们通过考察函数的最值（除了函数边界点外），发现它必然有一个性质，即在这个点上函数的一阶导数为 0，虽然一阶导数为 0 的点未必是最值点，但我们可以肯定的是，任何一阶导数不为 0 的点都可以排除，这就将解空间缩小到了有穷多个点，剩下的只要做做简单的排除法，答案就出现了。再譬如线性规划中经典的单纯形算法（又见《Algorithms》），也是通过对结论的考察揭示出只需遍历有限个顶点便必然可以到达最值的。此外很多我们熟知的经典题目也都是这种思路的典范，譬如《How To Solve It》上面举的例子：通过一个 9 升水的桶和一个 4 升水的桶在河里取 6 升水。这个题目通过正向试错，很快也能发现答案，然而通过反向归约，则能够不偏不倚的命中答案。另一些我们耳熟能详的题目也是如此，譬如：100 根火柴，两个人轮流取，每个人每次只能取 1~7 根，谁拿到最后一根火柴谁赢；问有必胜策略吗，有的话是先手还是后手必胜？这个问题通过试错就不是那么容易发现答案了。同样，这

个问题的推广被收录在《编程之美》里面：两堆橘子，各为 m 和 n 个，两人轮流拿，拿的时候你只能选择某一堆在里面拿（即不能跨堆拿），你可以拿 1~这堆里面所有剩下的个橘子，谁拿到最后一个橘子谁赢；问题同上。算法上面很多聪明的算法也都是通过考察所求结论隐藏的性质来减小复杂度的，譬如刚才提到的单纯形问题，譬如经典面试题“名人问题”、“和最小（大）的连续子序列”等等。倒推法之所以是一种极为深刻的思维方法，本质上是因为它充分利用了题目中一个最不易被觉察到的信息——结论。结论往往蕴含着丰富的条件，譬如对什么样的解才是满足题意的解的约束。一般来说，借助结论中蕴含的知识，我们便可以更为“智能地”搜索解空间。举一个直白的例子，有人要你在地球上寻找一栋满足如下条件的建筑：__层高（填空自己填），__风格，__年代始建，...（省略若干约束条件）。对于这样一个问题，最平凡的解法是穷举地球上每一栋建筑，直到遇到一个满足条件的为止。而更“智能”的（或者说更“启发”的）方法则是充分利用题目里面的约束信息，譬如假若

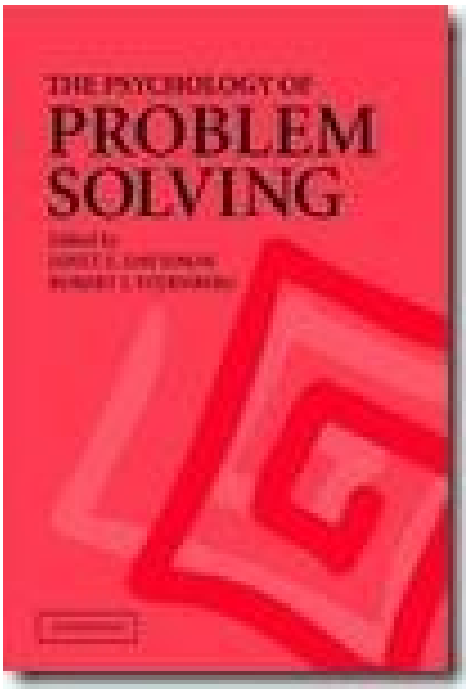
条件里面说要 60 层楼房，你就不会去非洲找，如果要拜占庭风格的，你估计也不会到中国来找，如果要始建于很早的年代的，你也不会去非常新建的城市里面去找，等等。倒推法是如此的重要，以至于笛卡尔当时认为可以把一切问题归结为求解代数方程组，笛卡尔的万能解题法就是首先将问题转化为代数问题，然后设出未知数，列出方程，最后解这组（个）方程。其中设未知数本质上就是一种倒推：通过设出一个假想的结论 x，来将题目对 x 的需求表达出来，然后顺势而下推导出 x。仔细想想设未知数这种手法所蕴含的深刻思想，也就难怪笛卡尔会认为它是那个解决所有问题的一般性钥匙了。

•试错。试错估计是世界上被运用最广泛的启发法，你拿到一个题目，里面有一些条件，你需要求解一个未知量。于是你对题目这里捅捅那里捣捣，你用上所有的已知量，或使用所有你想到的操作手法，尝试着看看能不能得到有用的结论，能不能离答案近一步。事实上，如果一个问题的状态空间是有限的话，往往可以通过穷举所有可能性来找到那个关键的性质。譬如这样一个问

题：有一个囚犯，国王打算处决他，但仁慈的国王给了他一个生还的机会。现在摆在他面前有两个瓶子，一个里面装了 50 个白球，一个装了 50 个黑球，这个囚犯有一个机会可以随便怎样重新分配这些球到两个瓶子中（当然，要保证不空），分配完了之后囚犯被蒙上眼睛，国王随机取一个瓶子给他，他在里面摸出一个球（因为蒙着眼睛，所以也是随机抽取），如果白球，则活，否则挂掉。问，这个囚犯如何分配，才能最大化生还几率。结合特例和试错法，这个题目的答案是很容易发现的。这样的题目还有很多。实际上，历史上很多有名的发现也都是无意间发现的（可以看作是试错的一种）。

•调整题目的条件（如，删除、增加、改变条件）。有时候，通过调整题目的条件，我们往往迅速能够发现条件和结论之间是如何联系的。通过扭曲问题的内部结构，我们能发现原本结构里面重要的东西。譬如这样一个题目（感谢 alai 同学提供）：A 国由 1000000 个岛组成，岛与岛之间只能用船作为交通工具，有些岛之间有船来往，从任意一个岛都可以去到另外任一个岛，当然

其中可能要换船。现在有一个警察要追捕一个逃犯，开始时他们在不同的岛上，警察和逃犯都是每天最多乘一次船，但这个逃犯还有点迷信，每个月的 13 日不乘船，警察则不迷信。警察每天乘船前都知道逃犯昨天在哪个岛上，但不知道他今天会去哪个岛。请证明，警察一定可以抓到逃犯（即到达同一个岛）。通过拿掉题目中一个关键的条件，观察区别，然后再放上那个条件，我们就能“感觉”到题目的内在结构上的某种约束，进而得到答案。




•求解一个类似的题目。类似的题目也许有类似的结构，类似的性质，类似的解方案。通过考察或回忆一个

类似的题目是如何解决的，也许就能够借用一些重要的点子。然而如何在大脑中提取出真正类似的题目是一个问题。所谓真正类似的题目，是指那些抽象结构一样的题目。很多问题表面看是类似的，然而抽象结构却不是类似的；另一些题目表面看根本不像，然而抽象层面却是一致的。表面一致抽象不一致会导致错误的、无效的类比；而表面不一致（抽象一致）则会阻碍真正有用的类比。《Psychology of Problem Solving》里面对此有详细的介绍。后面也会提到，为了便于脑中的知识结构真正能够“迁移”，在记忆掌握和分析问题的时候都应该尽量抽象的去看待，这样才能够建立知识的本质联系，才能够最大化联想空间。

•列出所有可能跟问题有关的定理或性质。这个不用说，我们在最初学习解题的时候就是这么做的了。

•考察反面，考察其他所有情况。很多时候，我们在解题时容易陷入一种特定的手法，比如为什么一定要是构造式的来解这个题目呢？为什么不能是逼近式的？为

什么一定要一步到位算出答案？为什么不能从一个错误的答案调整到正确答案？为什么这个东西一定成立？不成立又如何？等等。经典例子：100 个人比赛，要决出冠军至少需要赛多少场。

•将问题泛化，并求解这个泛化后的问题。刚才不是说过，应该通过特例启发思考吗？为什么现在又反倒要泛化呢？实际上，有少数题目，泛化之后更容易解决。即，解决一类问题，比解决这类问题里面某个特定的问题还要容易。波利亚称之为“发明者悖论”，关于“发明者悖论”，《数学与猜想》第一卷的开头有一个绝妙的例子，可惜这里空间太小，我就不摘抄了-_-|||

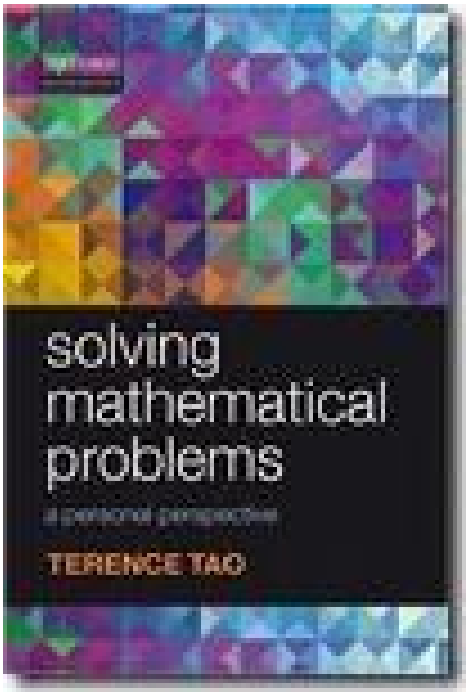
以上是我认为最重要的，也是最具一般性的、放之四海都可用的思维法则。一些更为“问题特定”的，或更为现代的启发法，可以参见《如何解题：现代启发式方法》以及所有的算法书。不过，在结束这一节之前，还有两个有趣的启发法值得一提：


•下意识孵化法。这个方法有点像老母鸡孵小鸡的过程：我们先把问题的吃透，放在脑子里，然后等着我们的下意识把它解出来。不过，不宜将这个方法的条件拉伸过远，实际上，除非能够一直保持一种思索的状态（金出武雄所谓“思维体力”），或者问题很简单，否则一转头去做别的事情之后，你的下意识很容易就把问题丢开了。据说庞加莱有一次在街上，踏上一辆马车的那一瞬间，想出了一个重要问题的解。其他人也像仿效，结果没一个人成功。实际上，非但马车与问题无关，更重要的是，庞加莱实际上在做任何事的时候除了投入有限的注意力之外，其他思维空间都让给了那个问题了。同样，

阿基米德从浴缸里面跳出来也是如此；如若不是经过了极其痛苦和长时间的思索，也不会如此兴奋。如果你也曾经花过几天的时间思考一个问题，肯定也是会有类似的经历的。

•烫手山芋法。说白了，就是把问题扔给别人解决。事实上，在这个网络时代，这个方法有着无可比拟的优越性。几乎任何知识性的问题，都可以迅速搜索或请教到答案。不过，如何在已知知识之外发掘出未知知识，如何解决未知问题，那就还是要看个人的能力了。数学界流传一个与此有关的笑话：如果你有一个未解决问题，你有两个办法，一，自己解决它。二，让陶哲轩对它感兴趣。

除了波利亚的书之外，陶哲轩的《Solving Mathematical Problems》也对解题的启发式思路作了极有意义的介绍，他在书的第一章遵循波利亚的思路从一个具体的题目出发，介绍了如何运用波利亚在书中提到的各种启发式方法来对解题进行尝试。




总而言之，充分挖掘题目中蕴含的知识，是解题的最关键步骤。本质上，所有启发式方法某种意义上都是为此服务的。这些知识，有些时候以联想的方式被挖掘出来，此时启发式方法充当的便是辅助联想的手段。有时候则以演绎和归纳的手法被挖掘出来，此时启发式方法则充当助探（辅助探索）工具。

一点思考

1.联想的法则

人类的大脑是一个复杂而精妙的器官，然而某种程度上，人类的大脑也是一个愚蠢的器官。如果你总结过

你解过的一些有意义的好题目，你会发现它们有一个共同点：没有用到你不知道的知识，然而那个最关键的、攸关成败的知识点你就是想不到。所以你不禁要问，为什么明明这个知识在我脑子里（也就是说，明明我是“能够”解决这个问题的），但我就是没法想到它呢？“你是怎么想到的？”这是问题解决者最常问的一个问题。甚至对于熟练的解题者来说，这个问题的答案也并不总是很明确的，很可能他们自己也不清楚那个关键的想法是怎么“蹦”出来的。我们在思考一个问题的时候，自己能意识到的思维部分似乎是很少的，绝大多数时候我们能感知到的就是一个一个的转折点在意识层面显现，我们的意识就像一条不连续的线，在其上的每一段之间那个空档内发生了什么我们一无所知，往往我们发现被卡在一个地方，我们苦思冥想，然后一个知识（也许是一个性质，也许是一个定理）从脑子里冒了出来，或者说，被我们意识到，然后我们沿着这条路走一段，然后又卡住，然后又等待一个新的关键知识的出现。而至于这些知识是怎么冒出来的？我们可以对它们的“冒出来”提

供怎样的帮助？我们可以在意识层面做一些工作，帮助我们的下意识联想到更多重要的知识吗？那些灵光一现的瞬间，难道只能等待它们的出现？难道我们不能通过一些系统化的步骤去“捕获”或“生成”它们？又或者我们能不能至少做些什么工作以使得它们更容易发生呢？

正如金出武雄在《像外行一样思考，像专家一样实践》中所说的，人类的灵感一定是有规律的，认知科学目前至少已经确认了人类思维的整个物质基础——神经元。而既然它们是物质，自然要遵循物质的运行规律。只不过我们目前还没有窥破它们，但至少我们可以确信的是，它们在那里。事实上，不需要借助于认知科学，单单是通过对我们自己思维过程的自我观察，也许就已经能够总结出一些重要的规律了，也许，对自身思维过程的反观真的是人有别于其它动物的本质区别。

《专注力》当中有这样一个例子：一天夜里，你被外面的吵闹声叫醒了，你出去一看，发现有一群人，其

中有一个人开着很名贵的轿车，他跟你说他们正在玩一个叫“拾荒者”的游戏，由于一些原因，他必须要赢这个游戏，现在他需要一块 1.5m*1m 的木板，如果你能帮忙的话，愿以一万美元酬报。你怎么办？被测试的大多数人都没有想到，只要把门拆给他就可以了（如果你想到了，祝贺你:-)），也许你会说现在的门都是钢的，没关系，那你有没有想到床板、立柜的门、大桌子的桌面之类的？这个问题测试的就是心理学上所谓的“范畴陷阱”，“木板”这个名词在你脑子里的概念中如果是指“那些没有加工的，也许放在木材厂门口的，作为原材料的木板”的话，那么“木板”就会迅速在你的下意识里面建立起一个搜索范畴，你也会迅速的反应到“这深更半夜叫我上哪去找木板呢？”如果你一下就想到了，那么很大的可能性是“木板”这个概念在你脑子里的范畴更大，更抽象，也许包含了所有“木质的、板状的东西”。

这就是联想的法则。

我们的大脑无时无刻不在对事物进行归类，实际上，

不仅是事物，一切知识，都在被自动的归类。在有关对世界的认知方面，被称为认知图式，我们根据既有的知识结构来理解这个世界，会带来很大的优势。实际上，模块化是一个重要的降低复杂性的手段。然而，知识是一把双刃剑，一方面，它们提供给了我们解决问题的无以伦比的捷径优势，“砖头是砌墙的”，于是我们遇到砌墙这个问题的时候就可以迅速利用砖头。然而另一方面，知识却也是思维的桎梏。思维定势就是指下意识遵循既有知识框架思考的过程。上面的那个木板的例子也是思维定势的例子。每一个知识都是一个优势，同时又是一个束缚。著名的科幻作家阿瑟·克拉克有一句名言：如果一位德高望重的老科学家说某个事情是不可能的，那么他很可能是错的。所以，如何在获取知识优势的同时，防止被知识束缚住，是一门技术。

掌握这门技术的钥匙，就是抽象。在吸收知识的时候进行抽象，同时在面对需要用到知识的新问题时也要对问题进行抽象。就以大家都知道的“砖头”有多少种用途为例，据说这道题目是用于测试人的发散思维的，

能联想到的用途越多，思维定势就越小。实际上，借助于抽象这个利器，这类题目（乃至更广的一类问题）是可以系统性的进行求解的，我们只需对砖头从各个属性维度进行抽象。譬如，砖头是——长方形的（长方形的东西有什么用途？还有哪些东西也是长方形的，它们都有什么用途？）、有棱角的（问题同上）、坚硬的、固体、有一定大小的体积的、红色的、边界线条平直的、有一定重量的...对于每一个抽象，我们不妨联想还有其他什么物体也是具有同样抽象性质的，它们具有同样的用途吗？当然，除了抽象之外，还有“修改”，我们可以在各个维度上对砖头的属性进行调整，以期得到新的属性：譬如大小可以调整、固体可以调整为碎末、棱角可以打磨、重量也可以调整、形状也可以调整...然后看看新的属性可以如何联想开去。

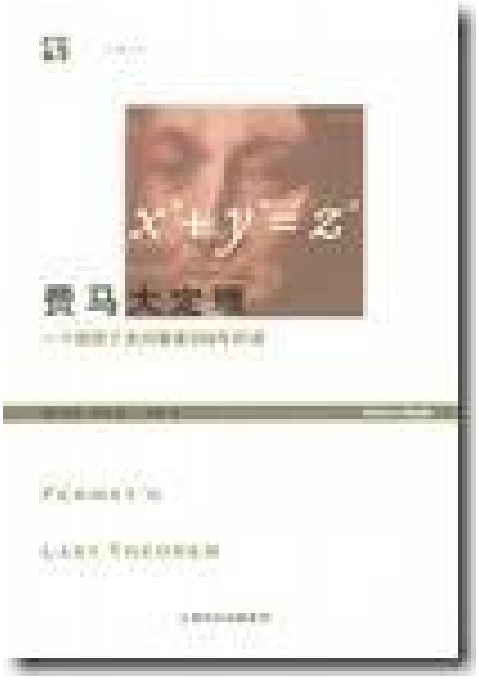
除了这个简单的例子之外，我们也不妨看一看一些算法上的例子，同样一个算法，不同的人来理解，也许你脑子里记得的是某个特定的巧妙技巧（也许这个技巧在题目的某步关键的地方出现，从而带来了最令人意外

的转折点），然而另一人个记得得也许是“递归”这种手法，还有另外一个人记得的也许是“分治”这种更一般化的解题思路。从不同的抽象层面去掌握这道题目的知识信息，以后遇到类似的问题，你能够想起这道题所提供的知识的可能性是有极大的差异的。《Psychology of Problem Solving》的第 11 章举了这样一个例子：先让被试（皆为大学生）阅读一段军事材料，这个材料是说一小撮军队如何通过同时从几个不同方向小规模攻击来击溃一个防守严实的军事堡垒的。事实上这个例子的本质是对一个点的同时的弱攻击能够集聚成强大的力量。然后被试被要求解决一个问题：一个医生想要用 X 射线杀死一个恶性肿瘤 这个肿瘤只可以通过高强度的X射线杀死，然而那样的话就会伤及周围的良好组织。医生应该怎么办呢？在没有给出先前的军队的例子的被试中只有 10%想到答案，这是控制基线。然后，在先前学习了军队例子的被试中 这个比例也仅仅只增加到 30%，也就是说只有额外 20%的人“自动”地将知识进行了转移。最后一组是在提醒之下做的，达到了 75%，即比“自

动”转移组增加了 45%之多。这个例子说明，知识的表象细节会迷惑我们的眼睛，阻碍我们对知识的运用，在这个例子中是阻碍问题之间的类比。

而抽象，则正是对非本质细节去枝减叶的过程，抽象是我们在掌握知识和解决问题时候的一把有力的奥卡姆剃刀。所以，无论是在解题还是在学习的过程中，问自己一个问题“我是不是已经掌握了这个知识最深刻最本质的东西”是非常有益的。

2.知识，知识




如果你是一个熟练的解题者，你也许会发现，除了

一些非常一般性的、本质的思维法则之外，将不同“能力”的解题者区分开来的，实际上还是知识。知识是解题过程中的罗塞塔碑石。一道几何题为什么欧几里德能够做出来我们不能，是因为欧几里德比我们所有人都更了解几何图形有哪些性质，借助于一个性质，他很容易就能抵达问题的彼岸；反之，对于不知道某个性质的我们，倒过来试图“发现”需要这样的性质有时几乎是不可能的。有人说数学是在黑暗中摸索的学科，是有道理的。并不是所有的问题都能够通过演绎、归纳、类比等手法解出来的。这方面，费马大定理就是一个绝好的例子，《费马大定理：一个困惑了世间智者 358 年的谜》一书描述了费马大定理从诞生到被解决的整个过程，事实上，通过对费马大定理本身的考察，几乎是毫无希望解决这个问题的，我们根本不能推导出“好，这里我只需要这样一个性质，就可以解决它了”，也许大多数时候我们可以，但那或者是因为我们有已知的知识，或者这样的归约很显然。而对于一些致命的问题，譬如费马大定理，最重要的归约却是由别人在根本不是为了解决费马

大定理的过程中得出来的。运气好的话，我们在既有的知识系统中会有这样的定理可以用于归约，运气不好的话，就得去摸索了。

所幸的是，绝大多数问题并不像费马大定理这样难以解决。而且绝大多数问题需要用到的知识，在现有的知识系统里面都是存在的。我们只要掌握得足够好，就有希望联想起来，并用于解题。

当然，也有许多题目，求解它们的那个关键的知识可以通过考察题目本身蕴涵的条件来获得，这类题目就是测试思维本身的能力的好题目了。而如果这个性质根本无法通过对题目本身的考察得出来，那么这个题目测试的就是知识储备以及联想能力。

3.好题目、坏题目

在我看来，好题目即测试一个人思维的习惯的题目（因为知识性的东西是更容易弥补的，尤其是在这样一个年代；而好习惯不是一朝一夕养成的），它应有这样一些性质：

•不需要用到未知的知识，或者

•需要用到未知的知识，但一个敏锐的解题者可以通过对题目的分析自行发现这些所需的知识。

•考察解题的一般性思路，而不是特定（ad hoc）的解题技巧，尤其是当这个技巧几乎不可能在短时间内通过演绎和试错发现的时候。譬如题目需要用到某种性质，而这个性质对于不知道它的人来说几乎是无法从对题目的考察中得出来的。

•考察思维能力：联想能力、类比能力、抽象能力、演绎能力、归纳能力、观察能力、发散能力（思维不落巢臼的能力）。

•考察一般性的思维方法：通过特例启发思考、通过试错寻找规律、通过泛化试探更一般性命题、通过倒过来推导将问题进行归约、通过调整（分解、删除、增加等等）题目的条件来感知它们之间的联系以及和结论的联系、通过系统化的分类讨论来覆盖每种可能性。

•好题目举例：烙饼排序问题（考察特例启发法以及观察能力）、Nim 问题（还有简单版本的取火柴问题）（烙饼排序问题和 Nim 问题可参见《编程之美》）、9 公升 4 公升水桶倒 6 公升水的问题（考察倒过来思考问题的能力）、9 点连线问题、6 根火柴搭出 4 个面的问题、“木板”问题（考察思维定势，此外《心理学与生活》的第九章也有好几个经典的问题）、许多数论问题（观察能力、演绎能力、归纳能力）。此外，我们最近也在讨论好题目。

而坏题目呢：

•好题目各有各的好，坏题目都是相似的。

•坏题目基本上就是指那些所谓的 unfair questions，什么是 unfair，举个例子：一个人住在一栋非常高的楼上，每天早晨他乘电梯下到一楼，出门上班。但晚上回来之后却最多只能坐到一半高度的楼层，剩下一半只能走楼梯上去，除非是下雨天。问为什么。这个例子据说不少人小时候在脑筋急转弯里面做过，但我很怀疑基本上任何正常人是不是可能想出来。这个问题的问题在

于他需要用到千百个有可能与问题有关的性质中的一个，而且这个性质还根本无法通过对题目本身的考察得出来，只可能某天我们碰巧遇到类似的场景也许才能想到。知道答案的人也许会说答案很显然，但别忘了心理学上的事后偏见——一旦知道结果之后，所有指向结果的证据看上去都那么显然和充分，而同时所有反结果的证据看起来都那么不显然和不充分。譬如这题关键是要想到这人是矮子和雨天要带伞，也许你会说“只要考虑一下电梯的按钮面板就会发现了”，或者“看到下雨，那还不想到带伞么？”，然而这只是事后的合情推断。在不知道答案的情况下，这个故事中有数不清的因素可能会成为问题的解释，除非某天我们碰到类似的问题，否则大致也只能一个个穷举了去使劲往上凑，譬如除了身高之外还有：是不是瞎子、是不是聋子、是不是哑子、男人女人、什么牌子的电梯、大厦是哪种大厦？这些因素重要吗？不重要吗？最令人头疼的是，在不知道答案的时候，我们也根本不知道他们重不重要，一个出谜语的人可能从任何一个微小的地方引申出某个谜语来；更头

疼的是，我们不知道我们不知道的那些因素是不是也可能与题目的解有关，譬如这样一个问题：一个人走进酒吧，问酒保要一杯水，酒保掏出一只枪，拉上扳机；这人说声“谢谢”，走了出去。这些题目固然有趣，但几乎没有价值。

•值得注意的是，这样的问题跟著名的 9 点连线问题和 6 根火柴搭出 4 个面的问题（还有《如何解题：现代启发式方法》里面那个经典的“小球在盒内碰撞何时回到原轨迹”的问题）不同，后者的条件都在眼前，并且解的搜索空间无论如何很小，就看思维能不能突破某一个框框。而上面这些问题则是要人进行根本不可能的联想。9 点问题实际上是可以系统化思考解决的，但 unfair question 则像许多谜语一样，随便哪个人都可以出一个另一个人根本无法想出来的谜语，因为从谜语隐含的信息加上人可能从谜语中联想出来的信息，加起来也不足以构成解题的充分条件；这种情况下除非你遇到出题人在出题时的心理或所处情况，否则是无法解的。

•最后，发散性思维其实是可以系统化的，参见前文“联想的规则”。

出题的误区：

最大的误区就是把知识性的题目误当成能力型的题目。如果题目中需要用到某个重要的定理或性质，而对于一个原本不知道这个定理或性质的人来说是无法通过题目本身到达这个性质的，那这就属于知识性的题目。

虽然几乎所有题目归根到底都是知识性的，但有些题目更为知识性，尤其是当解题中需要用到的定理或性质并不那么 trivial 的时候。

一个最好的题目就是问题明明白白，而且最终的解也没有用到什么神秘的定理，但要想获知到解，取决于你会不会思考一个问题（参见“好问题”）。譬如烙饼问题和 Nim 问题，还有许许多多问题简洁明确但很锻炼思考的算法问题。

4.一个好习惯

在解题的过程中，除了必要条件——知识储备——之外，对于一些并不涉及什么你不知道的定理的题，很大程度上就要看思维能力或者习惯了。而在思考一个问题的时候，最容易犯的一类错误就是忘了考虑某种可能性，不管这种可能性是另一种做法（譬如只顾着构造一个能一步得出结果的算法，没记得还可以从错误情况逼近。譬如只顾着正着推导，却忘了可以反过来推。只顾着反过来推，居然忘了可以考察简单特例。试了各种手法，却发现忘了考虑题目的某个条件。觉得试遍了所有可能性，已经走不下去了，然后其实在思维的早些时候就已经落入了思维陷阱。等等）事实上，即便是一个熟练的解题者也容易犯顾此失彼的问题，因为我们一旦意识到一个看似能够得到结论的解法，整个注意力就容易被吸引过去，而由于推导的路径是很长的，所以很容易在一条路上走到黑，试图再往下走一步就得出解。却忘了回过头来看看再更高的层面上还有没有其它手法，思路上有没有其他可能性。

而对于像我这样目前尚不谙熟所有思维方法的人来

说，则更容易犯这样的错误。为了避免这样的错误，一个有效的办法就是将自己的思考过程（中的重要环节）清晰的写在纸上（称为“看得见的思考”），这有如下几个好处：

•人在思考一个问题的时候，就像是在黑暗中打着电筒往前走（事实上，我们的工作记忆资源是有限的，有研究证明我们只能在工作记忆里面持有7加减2个项目；此外认知负荷也是有极限的），每一步推导，每一步逻辑或猜测都将我们往前挪一步，然而电筒的光亮能找到的范围是有限的，我们走了几步发现后面又黑了。有时候，我们是如此努力地试图一下就走出很远，同时又老是怕忘记目前已经取得的进展和重要结论，结果意识的微光就在一个很小的范围内打转，始终无法往前走出很远。而将思维过程记录下来，则给了我们完全的回顾机会。如果你是经常做笔记的人，你肯定会发现，有时候一个在脑子里觉得两句话就能说完不需要记下来的东西，一旦开始往纸上写下来，你就自然而然能得出更多的结论和东西，越写越多，最终关于你的问题的所有方面都被

推导出来展现在你面前。

•思考问题时的注意力是自上而下控制大脑的神经处理过程的，当我们集中注意力在某一个过程上时，其它的过程就会受到抑制。我们平常都遇到过一些时候，由于集中注意力从而忽略了周围发生的事情的时候（处理环境输入的神经回路受到抑制）。所以，当我们竭尽全力将一些非常重要的因素控制在工作记忆里面的时候，实际上很大程度上抑制了其余的思考——可以想见如果科比在跳投的瞬间集中注意力思考跳投的各种技术要素的话会发生怎样的灾难。此外，这么做还占用了宝贵的工作记忆空间，从这个意义上，借助于纸笔，将思考的东西写下来实际上就是扩充了我们的工作记忆，增大了思维的缓存。注意，这倒不是说思考问题不需要集中注意力，而是说由于将项目维持在工作记忆中需要很大的认知精力，使得我们的注意力无法暂时移开去思考其它相关的子问题，而写到纸上的话我们就减轻了工作记忆的负担，可以转移注意力去集中思考某个子问题；同时我们又可以随时回过头来，重新将以前想过的结论装入

记忆（内存），完全不用费心去阻止它们被我们的工作记忆遗忘。

•一句话从嘴里说出来，或者写到纸上，被视觉或听觉模块接收，再认知；跟在心里默念所产生的神经兴奋程度是不一样的。我们都有过这样的经历：一句令人不愉快的话，我们心里清楚，但就是不愿意自己也不愿意别人从嘴里说出来。同样，将思考的过程写到纸上，能够激起潜在的更多的联想。为什么会这样的另一个可能的原因是我们大脑中思维过程的呈现形式和纸上的表现形式是不一样的，既没有那么严格、详细也没有那么多的符号（如数学符号）——再一次，工作记忆资源是很有限的——而后者，作为视觉线索，可能激起更多对既有知识的回忆。

•我们在思考问题的过程中容易落入思维定势，不知不觉就走上来某条“绝大部分时候是如此”的思维捷径，对于一些问题而言这固然能够让我们快速得到解，但对于另一些问题而言却是致命的。我们容易在逻辑的路径

上引入想当然的假设，从而排除某种不该排除的可能性或做法。通过将思路过程写到纸上，我们便能够回头细细考察自己的思考过程，觉察到什么地方犯了想当然的毛病。

•我们在思维过程中的每一个关键的一步也许都有另一种可能性，一个问题越复杂，需要推导的步骤就越多，我们就越容易忽视过程中的其它可能性，容易一条路走到黑。而将思维过程写下来，在走不下去的时候可以回过头看看，也许会发现另一种可能性，另一条“少有人走的路”。

•最后，通过将思维过程写下来，我们就能够在解题完毕之后完整的回顾自己的整个思维过程，并从中再次体悟那些关键的想法背后所发生的心理活动过程，总结思考中的重要的一般原则，分析思维薄弱的环节，等等。就算是最终发现并没有到达结果的无效思路，也未必就没有意义，因为不是因为错误的思路，也不会知道正确的思路，况且对一道题目用不上的思路，对其它题目未

必用不上。通过对自己思维过程的彻底反思，就能从每次解题中获得最多的收获。

5.练习，练习

本质上，练习并不产生新能力。然而练习最重要的一个作用就是将外显记忆转化为内隐记忆。用大白话来说就是将平时需要用脑子去想（参与）的东西转化为内在的习惯。譬如我们一开始学骑自行车的时候需要不断提醒自己注意平衡，但随着不断的联系，这种技能就内化成了所谓的程序式记忆（内隐记忆的一种），从而就算你一边骑车一边进行解题这样需要消耗大量脑力的活动，也无需担心失去平衡（不过撞树是完全可能的，但那是另一回事）。

同样，对于解题中的思维方法来说，不断练习这些思维方法就能做到无意识间就能运用自如，大大降低了意识的负担和加快了解题速度。

不过，并非所有的练习方法都是等效的，有些练习方法肯定要比另一些更有效率。譬如就解题来说，解题

是一项涉及到人类最高级思维机制的活动，其中尤其是推理（归纳和演绎）和联想。而后者中又尤数联想是最麻烦的，前面提到，绝大多数时候启发式方法实质上都是在为联想服务——为了能像晃筛子那样把你脑袋里那个关键的相关知识抖落出来。并且，为了方便以后能够联想，在当初吸收知识的时候就需要做最恰当的加工才行，譬如前面提到的“抽象”加工，除此之外还有将知识与既有的知识框架整合，建立最多的思维连接点（或者说“钩子”）。对于知识的深浅加工所带来的影响，《找寻逝去的自我》里面有精彩的介绍（里面也提到了提取线索对回忆的影响——从该意义上来说运用启发式思维方法来辅助联想，其实就是进行策略性记忆提取的过程）。最后，人类的无意识思维天生有着各种各样的坏习惯，譬如前面提到的范畴陷阱就是创新思维的杀手，譬如根据表面相似性进行类比也是知识转移的一大障碍。更遑论各种各样的思维捷径了（我们平常进行的绝大多数思考和决策，都是通过认知捷径来进行的）。所以说，如果任由我们天生的思维方式发展，也许永远都避不开

无意识中的那些陷阱，好在我们除了无意识之外还多出了一层监督机制——意识。通过不断反省思维本身，时时纠正不正确的思考方式，我们就能够对其进行淬炼，最终养成良好的思维习惯。反之被动的练习虽然也能熟能生巧，但势必花的时间更多，而且对于涉及复杂的思维机制的解题活动来说，远远不是通过钱眼往油壶里面倒油这样简单的活动所能类比的，倒油不像思维活动那样有形形色色的陷阱，倒油不需要联想和推理，倒油甚至几乎完全不需要意识的辅助性参与，除了集中注意力（而解题活动就算对于极其熟练的人来说也不断需要大量的意识参与）。所以对于前者，良好的思维习惯至关重要，而反省加上运用正确的思维方法则是最终养成良好思维习惯的途径。

练习还有另外一个很重要的作用，就是增加领域知识（关于知识在问题解决中的作用，前面已经提到过）。我们看到很多人，拿到一道题目立即脑子里就反应出解法，这个反应快到他自己都不能意识到背后有什么逻辑。这是因为既有的知识（我们常说的“无他，实在是题做

得太多了”）起到了极大的作用，通过对题目中几个关键元素或结构的感知，大脑中的相关知识迅速被自动提取出来。而对于知道但不熟悉相应知识（譬如很早我们就知道归纳法，但是很久以后我们才真正能够做到面对任何一道可能用归纳法的题目就立即能够想到运用归纳法），或者干脆就不知道该知识的人来说，就需要通过启发法来辅助联想或探索了。后者可以一定程度上代偿对知识的不够熟悉，但在一些时候知识的缺失则是致命的（参见上面第 2 点）。不过要注意的是，那种看到题目直接反应出答案的或许也不是纯粹的好事，因为这样的解题过程严重依赖于既有知识，尤其是做过的类似的题目，其思维过程绝大部分运用的是联想或类比，而非演绎或归纳。更重要的是，联想也分两种，被动联想和策略性联想（参考《找寻逝去的自我》），这里用的却是被动联想。所以，能直接反应出答案并不代表遇到真正新颖的题目的时候的解决能力，后者由于不依赖于既有领域知识，就真正需要看一个人的思维能力和习惯究竟如何了。

6.启发法的局限性

首先肯定的是，启发法一定（也许很大）程度上是可以代偿知识的不足的（这里的知识主要是指大脑中的“联系”，下面还会提到另一种知识，即 hard knowledge）。譬如，一道题目，别人直接就能通过类比联想到某道解过的题目，并直接使用了其中的一个关键的性质把题目给解出来了。你并没有做过那道题目，这导致两种可能的结果：一，你就是不知道那个性质。二，你虽然“知道”那个性质，但并没有在以前的解题经历中将那个性质跟你手头的这个问题中的“线索”联系起来，所以你还是“想不到”。后一种可以称为 soft knowledge，即你“知道”，但就是联想（联系）不起来。所谓不能活学活用，某些时候就是这种情况，即书本上提供什么样的知识联系，脑子里也记住什么，而没有事后更广泛地去探索知识之间的本质联系（总结的作用）。前一种则可以称为 hard knowledge，即你就是不知道，它不在你的脑子里。

而启发式方法在两个层面上起作用：

•辅助联想起 soft knowledge：譬如，特例法是一种启发式思考方法，它通过引入一个简单的特例，特例中往往蕴含有更多的“线索”，通过这些线索，有可能就会激发起对既有的知识的联想。另外一种强大的辅助联想办法就是对题目进行变形，变形之后就产生了新的视觉和语意线索，比如式子的对称性、从直角坐标到极坐标从而引发对后者的知识的联想等等。大量的启发式方法实际上的作用就是辅助联想，通过对题目中的线索的发掘，激起大脑中已知相关知识的浮现。在这个意义上，相对于那些能够直接联想到某个性质的人，那些不知道但可以通过启发式思维联想到的，启发式思维就提供了一种“曲径通幽”的策略性联想。还是以经典的例子来说：砖头的用途。有人立即能够直接联想到“敲人”。有人也许不能。然而启发式联想策略“抽象”就能够帮助后者也能够联想到“敲人”，因为“抽象”策略启发人去考虑砖头的各个性质维度，如“质地”，“形状”，当你考察到“质地坚硬”，“棱角”，离“敲人”的功能还会远么？本质上，能够直接联想到“敲人”功能的人是因为大脑

中从砖头到敲人这两个概念之间的神经通路被走过了很多遍（譬如由于经常拿砖头敲人），神经元之间的联系相当“粗”（形象的说法，严格的事实请参考《追寻记忆的痕迹》），而不经常拿砖头敲人的人呢，这个联系就非常的弱，乃至于根本激不起一次神经冲动。那么为什么通过启发式方法又能联想到呢？因为启发式方法相当于带入了一种新的神经调控回路，首先它增加你联系到砖头的属性维度上的可能性，使得“质地坚硬”、“棱角”这两个语意概念被激活起来（注意，如果没有启发式方法的参与，这是不会发生的），一旦后者被激活起来，从后者到“敲人”的联系就被激活起来了。从本质上，解题中的启发联想方法做的也就是这个工作。而越是一般性的启发式方法就越是能对广泛的问题有帮助（譬如《How to Solve It》中介绍的那些，譬如分类讨论、分治、乃至我认为很重要的一个——写下自己的思维过程，详细分解各个环节，考察思维路径中有无其它可能性（我们很容易拿到一道题目便被一种冲动带入到某一条特定的思路当中，并且遵循着“最可能的”推导路径往下走，

往往不自觉的忽略其它可能性，于是那些可能性上的联想就被我们的注意力“抑制”了。))。

•辅助探索出 hard knowledge：倒推法是一种启发式思考方法，它将你的注意力集中到问题的结论中蕴含的知识上，一旦你开始关注可能从结论中演绎出来的知识，你就可能得到 hard knowledge，即并不是早先就存在你脑子里，但是可以通过演绎获得的。上文中的最小和子序列中的倒推方法就是一个例子。

而启发式方法的局限性也存在于这两个方面：

•有些联系是不管怎样“启发”也想不起来的。譬如“当布被刺破了，干草堆就重要了”，你怎么解释这句话？如果有人提示一下“降落伞”,每个人都会恍然大悟。这是因为从“布”到“降落伞”之间的单向联系是近乎不存在的。而且就算运用启发法，譬如，考虑所有布做的东西，也基本绝无可能想到降落伞，因为同样，从“布做的东西”到“降落伞”之间的关联也是极其微弱的。我们脑子里只能保留那些最最重要的联系。( 如果一提到

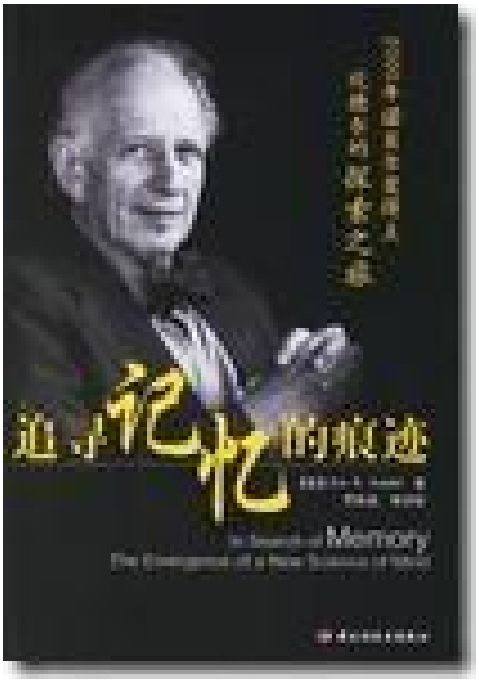
布，“降落伞”和“衣服”、“被单”、“窗帘”等日常物品以同等重要级别闪现，就乱套了。）那为什么从降落伞我们能想到布呢？我们实际上不能，我们为什么有些时候能，是因为譬如有人叫你“考虑降落伞的材料”，后者就激发了“降落伞之材料”这个语意，后者又指导了我们去考察降落伞的材料构成，于是我们想到是布。否则“布”是不会直接被激发起来的。那为什么在我们的这个问题中，一旦有人提到降落伞，我们就能建立从布到降落伞的关联呢？这是因为“降落伞”和“布”这两个语意单元的同时兴奋增大了它们之间关联的可能性，就好比是加大另一端的电压从而发生了“击穿”一样。从本质上，解数学题也是如此，费马大定理的求解过程是一个很好的例子，谷山志村猜想，就相当于那个“降落伞”的提示。我们还听到很多这样的故事（或者自己经历）：苦思冥想一个问题不得要领，某一天在路上走，看到某个东西或听到某句话，然后忽然，一道闪电划破长空，那个问题解开了（阿基米德是因为躺在浴缸里从而想到浮力原理的吗？）。我敢保证，如果一个人早就把那个问题从

脑海里扔到九霄云外去了（不再处于兴奋状态了），那么就算线索出现，也是不可能发生顿悟的。我们都知道，带着一个问题（使其在大脑中处于兴奋状态）去寻找答案更可能找到，即便不是有意去寻找，只要问题还在脑子里，任何周围的有可能与它相关的线索都不会被大脑漏掉，因为“问题”和“周围的其他线索”同时的兴奋增大了关联的可能性。如果问题早就被从大脑（意识或者潜意识）中撤下了，即便周围出现提示也不会被捕捉到。

•许多 hard knowledge 是不能被启发探索出来的。至少是不能被“直接命中目标”地探索出来的。一个问题有可能跟三角函数有关，也许你只能带着问题去探索三角函数的所有性质，从而最终发现那个关键的性质。费马大定理与椭圆方程有关，也许只能去探索椭圆方程的所有性质，这个过程一定程度上是盲目的，试错的，遍历的。而不是直接面向目标的。再聪明的人也无法从费马大定理直接反推到谷山志村猜想。在这些时候，启发式方法最多只能提供一个探索的大致方向：譬如，探

索三角函数的性质，并随时注意其中哪个可能对我这个问题有帮助。譬如，探索模运算的性质，看看哪些性质可能会有用。譬如，探索椭圆曲线的性质...等等。启发式方法并不能使我们的探索精准地命中目标。而只能划定一个大致的范围。也难怪有人说数学是盲目的。

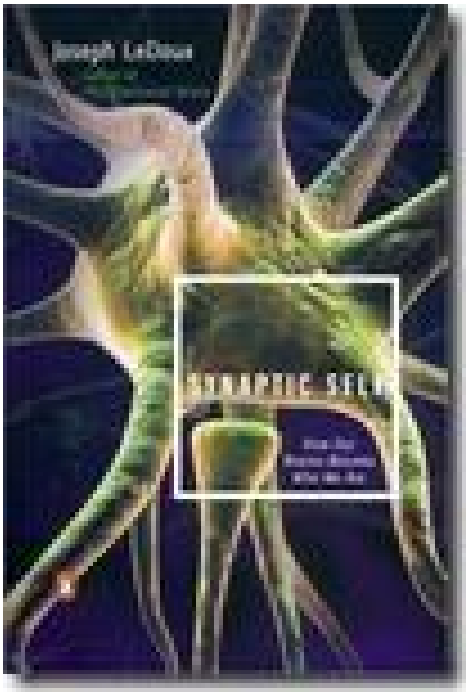
但话说回来，启发式方法的局限性并不能否认在大量场合启发式方法的巨大帮助，许多时候，单靠启发式方法就能带来突破。而且，一旦知识性的东西掌握的是一样多的，能否运用更优秀的思维方法就决定了能力的高下。有很多介绍思维方法的书。




7.总结的意义

解题练习的最重要目的不是将特定的题目解出来，而是在于反思解题过程中的一般性的，跨问题的思维法则。简单的将题目解出来（或者解不出来看答案，然后“恍然大悟”），只能得到最少的东西，解出来固然能够强化导致解出来的那个思维过程和方法，但缺少反思的话便不能抽取出一般性的东西供更多的题目所用。而解不出来，看答案然后“哦”的一声更是等同于没有收获，因为“理解”和“运用”相差何止十万八千里。每个人都有过这样的经历：一道题目苦思冥想不得要领，经某个人一指点其中的关键一步，顿时恍然大悟——这是理解。但这个理解是因为别人已经将新的知识（那个关键的一步）放到你脑子里了，故而你才能理解。而要运用的话，则需要自己去想出那关键的一步。因此，去揣测和总结别人的思维是如何触及那关键的一步，而你自己的思维又为什么触及不到它，有一些一般性的原则可以指导你下次也能想到那个“关键的一步”吗，是很有意义的。我们很多时候会发现，一道题目，解不出来，最终在提示下面解出来之后，发现其中并没有用到任何自

己不知道的知识，那么不仅就要问，既然那个知识是在脑子里的，为什么我们当时愣是提取不出来呢？而为什么别人又能够提取出来呢？我怎么才能像别人那样也提取出相应的知识呢？实际上这涉及到关于记忆的最深刻的原理，实际上文中已经提到了一些。（有兴趣的建议参考以下几本书：《追寻记忆的痕迹》，《找寻逝去的自我》，《Synaptic Self》，《Psychology of Problem Solving》）一般性的思维法则除了对于辅助联想（起关键的知识）之外，另一个作用就是辅助演绎/归纳（助探），一开始学解题的时候，我们基本上是先读懂题目条件，做可能的一些显然的演绎。如果还没推到答案的话，基本就只能愣在那里等着那个关键的步骤从脑子里冒出来了。




而所谓的启发式思维方法，就是在这个时候可以运用一些一般性的，所有题目都适用的探索手法，进一步去探索问题中蕴含的知识，从而增大成功解题的可能性。启发式的思维方法有很多，从一般到特殊，最具一般性的，在波利亚的《How to Solve It》中已经基本全部都介绍了。一些更为特殊性的（譬如“如果全局搜索空间没有递归结构，那么考虑分割搜索空间”，譬如那些“看到 XX，要想到 YY”的联系），则需要自己在练习中不断抽象总结。

一句结尾

“我想我就在这里结束”——如果你知道我在说什

么的话:-)

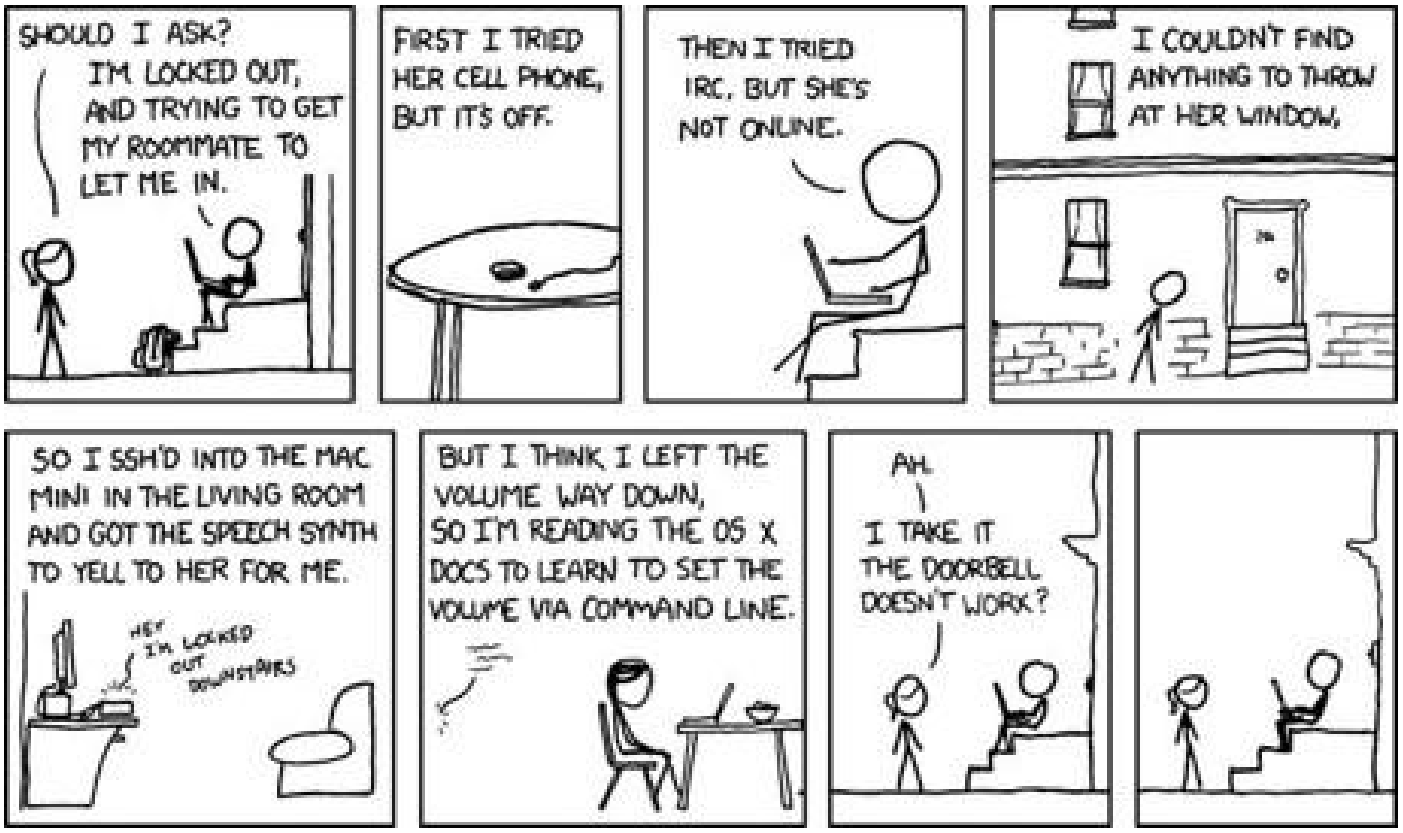
## 锤子和钉子

（一）

有这么一句古老的箴言：

如果你手里有一把锤子，所有东西看上去都像钉子。

其实这句话已经是老调中的老调重弹了，我们程序员有很多锤子：OO、设计模式、语言（C,C++,Java,Python,Ruby,etc.）、各种各样的架构 tricks&workarounds，以及一堆软件过程方法论（Agile,XP,Scrum,etc.）等等。




几则故事：

1.阿朱的（《走出软件作坊》）：

我过去领导过架构组。架构组的人在 2002 年的时候，疯狂迷上了 UML 和设计模式，人手一本《COM 本质论》和《设计模式》。我手下有一个新手，就处处是类，处处是抽象，处处是封装，处处是分离，尽量使代码高内聚低耦合。但是这样的的代码太麻烦，他花费了大量的时间，他看自己的代码赏心悦目，别人看他的代码云里雾里，不阅读懂《设计模式》就按照常规理解业务的

思路去阅读他的代码根本阅读不懂，不知道他为什么这样写代码，怪异的很。本来，这位想达到可维护性，可阅读性，却真正的失去了可维护性、可阅读性。这和我前几天看我的朋友周爱民写的《大道至简》中写到：有人希望拿 UML 去统一用户和软件设计者。殊不知 UML 有多难理解，而 UML 设计者却认为 UML 可以描述一切。就这个道理，要理解你的代码还要去读懂《设计模式》，这要求太高了吧。

所幸这位新手自己都每次写的累，慢慢的也就懒了，觉得确实需要分离的时候就分离，觉得没什么必要的就懒得做了。用他自嘲的话说就是：被磨平了。其实，依我看，他现在这个代码状态才是刚刚好，即照顾了设计扩展，又照顾了实用。真正的纯 OO，纯设计模式，可能只存在于教学和科学，而不在于我们的商业软件开发。我们作为商业开发，强调的是叫座的基础上叫好，所以折中方案是必须的，客户和我们自己两相宜就 OK，是否符合正宗，就不在我们的商业开发管理范畴了。

这位新手还写了大量的注释。在每个源代码文件头都写上几月几号，XX 创建的，这个原代码文件主要是干什么的，还画蛇添足的写上版权所有，公司名称。好像这个代码要开源，或者可能会被其他公司窃取了好表明公司版权。甚至每个函数都写了注释，每个参数是什么意思，每个参数可能出现的值代表什么意思，都写的一清二楚。久而久之，也懒的维护了。代码改动了，参数扩展了，参数状态值有了变化，注释说明却没有跟着改动，让后来看代码的人老误解，还不如不写这些注释。

我告诉他：做事不能走极端。要么全写注释，要么不写注释，都是不对的。我只在我认为要小心的地方，或者我自己都觉得很难理解懂的地方我才写注释。否则，我自己都可能会过段时间理解错了。如果某段代码我看看就能看懂，我就不写注释了。咱们做企业管理软件，深入技术又没有，只要代码能把复杂的业务处理描述的逻辑思路清晰就 OK。虽然说理解能力不同，我能快速理解了的未必有新手能够理解，但是你看看我的代码你就明白了。（摘自《走出软件作坊：代码那些事儿》）

再来一段：

有一部分所谓的架构师，技术超深厚，框架堪比 Spring 之类，但自己一个人闷头写框架不断优化，力竭使用最先进的技术思想，希望把最豪华的设计模式融进去，希望把 OSGi 融进去，希望把 AOP 融进去，全无视那些想利用框架减轻自己工作量提高自己工作效率的应用功能开发同事。这是在用公司工资玩技术呢，还是在满足个人技术幻想呢，还是在实验呢？到底在干吗？价值在哪里？

还有的人不会推广自己的框架。不善言辞，就幻想着技术总监能够通过行政命令让大家必须用框架，能不自己写代码就不自己写代码，能交给框架做的就交给框架做。但技术总监号召完了，大家仍然我行我素，各自开发为政，让框架开发者很孤单。

还有的人也不会推广自己的框架，沉迷在自己的理想世界。好不容易技术总监召集大家让大家来听听框架如何应用，但自说自话，满口自己最得意的词汇，听得

业务功能开发人云山雾罩。大家问些问题，如这样的业务开发难题，框架怎么解决？于是，框架开发员就和业务开发员争论了起来。框架开发员觉得这根本就不能答应客户这种变态的需求，而业务开发员说这就是现状。框架开发员说你可以这样这样，业务开发员说这样太麻烦，框架开发员立刻还口这还麻烦？于是双方各执一词，框架也没推广成功。

我手底下有个框架开发员。他的技术渴望很强烈，为了技术难题攻克，可以不吃不睡。并且技术敏感度很强，学习也快。所以当时我感觉他是个程序员的料，就把他拉到我的手下。

但是有个问题，他写出的框架代码，在平时开发业务功能的时候挺麻烦。大家可能需要的是一把铁锹，但是他却给大家 N 根不同长度不同粗细不同材质的木棍，N 个不同形状不同用途的铁锹头。大家会有 N 种组合。不仅导致他写代码老超任务期，而且也让使用人感觉没多大帮助。使用起来复杂，而且还得配置这个配置哪个，

需要注意的地方太多。业务开发组的同事就不愿意用，还不如把代码自己直接写死了得了。超期还会影响业务功能开发组的使用。本来人家是为了想加快自己的工作效率。你答应好这个星期给业务开发组提供一个功能，但你没有拿出来。就耽误人家进度。你多次拿不出来，人家业务开发组还不如自己开发一个呢，求人不如求己。

我最后警告他：如果你认为自己技术够牛，那么你必须证明你能很快做出来。如果你认为自己技术够牛，最好能牛到，只提供一个函数就解决了他们的问题。别这个代理类，那个聚合类，这个唯一实例类。最好连参数也没有，大家调用一下写一句代码就 OK。甚至你做的好，大家都不用调用你的代码，你可以包含在基础框架中，你自己去判断什么时候什么应用需要执行这个动作。如果你认为自己技术够牛，那么在业务功能需求发生变化的时候，你能够保证接口不变的情况下还能适合变化，这才你够牛。别让业务开发组的人跟着你也得改他们自己的代码，那样的设计就很烂了。

小伙听了我的话。进度保证，代码接口简洁。（摘自《走出软件作坊：走钢索的人》）

2.坊间流传的大家耳熟能详的小故事：

话说联合利华新换了一批自动香皂包装机以后，经常出现香皂盒子是空的没有香皂的情况，而在装配线一头用人工检查因为效率问题不太可能而且不保险。这不，一个由自动化，机械，机电一体化等专业的博士组成的 Solution 队伍来解决这个问题，没多久他们在装配线的头上开发了全自动的 X 光透射检查线，透射检查所有的装配线尽头等待装箱的香皂盒，如果有空的就用机械臂取走。

不巧，中国一乡镇企业生产香皂也遇到类似问题，老板吩咐线上小工务必想出对策决之，小工拿了一个电风扇放在装配线的头上，对着最后的成品吹之，空盒子被吹走，问题解决之。（摘自 TopLanguage 上的讨论。）

因此，

•心中有锤，就容易为其奴役：在遇到问题的时候不是具体问题具体分析，而是屁股决定脑袋，不管三七二十一先上黄金大锤再说，而且往往还颇有成就感，却将自己真正原本要解决的问题抛在脑后了。始终莫要忘记提醒自己，“问题是什么？”

•但毫无疑问，没有锤子是万万不行的，没有谁会傻到徒手钉钉。重点是选择合适你的工具。这又要求在学习工具的时候始终别忘记它的适用范围。

正确的态度应该是：

手中有锤，心中无锤。

容我具体解释一下这句话：任何工具都有其适用范畴和前提。然而，我们在学习工具的时候由于投入很多的时间，往往在情绪上面对工具产生了太强的感情，我们既投入了时间，当然内心希望能够用上这些工具，所以就容易忘掉其适用前提，欣欣然地不管三七二十一就把黄金大锤亮出来，以显示自己的厉害。但如果我们换一个态度，仅仅将它看作我们工具箱中的又一件工具，

就可以客观地评估它，视具体情况而使用了——始终别忘记自己要解决的问题是什么。Why 永远在 How 之前。

（二）

与上面对应的还有另一句话（实际上这是我杜撰的:D）:

如果你想钉一个钉子，所有东西看上去都像是锤子。

用大白话来说就是：如果你心中专注于你想要解决的问题，那么你所看到的东西就会呈现出以往你没有看到的一面。

例子：

1.阿基米德洗了一辈子的澡，然而，只有那一次，当他想要解决皇冠密度问题的时候，想到可以利用排水体积来测量不规则物体体积。


2.如果你也喜欢看《Monk》，就会体会到把问题装在心中，甚至把自己变成问题，问题即自己的作用——当 Monk 的潜意识里面始终在寻思 How,Why,Who did it 这几个问题时，周遭环境中的一切信息都会显出另一番面目，一个平常情况下根本不会注意到的细节也能成为破案的关键，看似不相干的信息也能带来出乎意料的启发。

3.BentObjects

如果你也像我一样，习惯于经常把疑问装在大脑中酝酿好几天，你肯定会有 eureka 的体验。

或者，如果你喜欢在一段时间之内关注某个主题，你在阅读书籍资料的时候就会带着问题的眼镜，看到平常看不到的东西，作出与平常不一样的思考。

把自己变成钉子，这就是 eureka 的奥秘。

**鱼是最后一个看到水的**

《你的灯亮着吗？》的最后一页画着一副大大的彩插：

鱼总是最后一个看到水的。

实际上，这句话有很多引申说法，其中最著名的一句是：

如果你有的是一把锤子，那么所有东西看起来都像是钉子。

不过后一句内涵文实在有误导嫌疑，因为这句话的表达方式很容易让人触摸不到问题的本质：即之所以所有东西看起来都像钉子，是因为人倾向于在既有框架下去解决问题；更重要的是，在这个过程中很难觉察到框架约束的存在，正如鱼觉察不到水的存在一样。而这一切背后的本质原因则是：

人是有很强的适应性的。

这个谚语应该作为问题解决者的座右铭之一：忽视既有框架的约束很容易导致 sub-optimal 的解决方案。

普通人遵守规则，牛人无视规则，伟人创造规则。

而要做到无视规则乃至创造规则，首先就要知道规则的存在才行。最近的“征途”事件就很好的说明了这一点：在既有框架下解决问题很多人都会，难就难在意识到框架的存在并突破它。

体现在程序员这个行业里面，也有很多相应的有趣现象：

#1 设计模式

《设计模式》被许多初学者奉为圭臬，认为那些看上去精巧的东西才是真正牛 13 的，值得学习的。而且，更聪明一点的人甚至会唯恐学的东西还不够复杂，因为越是复杂的东西搞出来越是有成就感。然而事实是，把简单的事情搞复杂的人比比皆是，把复杂的事情搞简单的人凤毛麟角。

实际上，《设计模式》里面曾经提到，其中大部分的模式都是建立在 smalltalk/C++的前提之下的。这个前

提（框架）很可惜的被所有人扔到了风中，并将那些模式泛化为放之四海而皆准的准则。Peter Norvig 老大有一个著名的 ppt，里面提到在《设计模式》的 23 个模式里面有 16 个模式放到动态语言（尤其是 LISP）下面就根本不是什么模式，而是显而易见无需费力就能完成的任务。c2 上的老大们则将 Peter Norvig 的言论进一步泛化，提出“设计模式象征着语言所缺乏内建支持的特性”的说法，并罗列了一个“设计模式<=>语言特性”的对照表。CodingHorror 的 Jeff 也跟着掺和，换汤不换药的跟贴说“设计模式其实是语言进化的路标”（设计模式的大量重复使用便意味着应该集成到语言内建支持中去了）。

不管怎样，有一点是确定的，就是 first-class 的内建支持（简洁）永远优于补丁式的解决方案（绕来绕去）。新版 Javascript 加入对 multi-method（VisitorPattern）的直接支持——Generic Function——就是一个极好的例子。当然，Rubyers 肯定也不会忘了跟着 Lispers 叫唤一把 Closure 是如何一个 each 方法搞定所有的迭

代过程，全面打败笨拙的 IteratorPattem 的，并顺便奚落一下石器时代的 Java。

总而言之，大众对设计模式的定性存在严重的问题，许多人把设计模式当成是精巧的利器，就如同解数学题时神来之笔的技巧一样。然而实际上远非如此，设计模式是补丁，其出现往往意味着语言不够强大，其使用意味着大量的、与所要达到的编程目的无关的样板式代码。

著名的 Head First 系列的《Head First Design Pattern》在书的靠近结尾的地方也郑重提醒（Jeff 同学觉得这个提醒应该用大号字体放在最前面）：

不要觉得不用设计模式就不够好不够强大，以尽可能简单的方式完成任务才是王道。

（不过值得注意的是，简单并不意味着邋遢。）所谓无码胜有码，设计模式，也是如此。

#2 语言之争

语言之争是程序员群落永远的话题。Flame war 是

个一触即发的东西。语言之争的原因之一就是人们容易在自己熟悉的语言框架下思考，并形成严重的偏见，只看到自己语言的好处，甚至于将并非好处的地方也觉知为好处。

#3 语言的使用

一个程序员越是熟悉一门语言，越是容易为这门语言所累。因为这门语言的特性对他来说就是鱼的水、木工的锤子。一遇到问题首先脑子里就会闪出若干语言特性，既有方案。当然，从统计意义上来说这并不是什么坏事，也许大多数时候是有助于问题的迅速解决的。但那 20%的时候这种思路带来的害处也许就带来了 80%的头大。

比如熟悉 Java/C#/C++的程序员可能会觉得“迭代器”的概念是非常直观的，甚至是非常漂亮的；而介绍这个概念的书当然也不会上来就给自己一瓢冷水说迭代器的出现实在是没办法的补丁。所以大家也就觉得这玩意好使，优美了，因为很多时候我们是用“足够”来定

义好坏的；只要还算能用，我们就可以觉得“不赖”。然而，事实上，正如前文所说，翻一翻《设计模式》的 Iterator Pattern 一节，再翻翻镐头书中介绍迭代的章节，就会发现更简单的迭代方案是根本无需定义多余的迭代器类和继承体系。

另一个思维被语言束缚的好例子是这篇“API 考古之：C 风格的 Java API”，里面提到一个家伙设计了这样一个 Java API：

int enumerate(Thread[]list);//list all the threads in the current ThreadGroup

避免思维被一门语言束缚的最好办法就是“学习其它语言”。

#4 C++

之所以加上 C++有两个原因。一，这个 Blog 以往主要写 C++。二，C++在所有语言里面有特殊性。在 C++里面，存在无数的惯用法和技巧，好听一点叫技术。

无数本书用无数比特的唾沫描述这些技术并将它们吹得精妙无比。然而，最近由 TopLanguage 的兄弟们群策群力汇集问题对 Bjarne 老大的一次“多对一”访谈（采访稿 en 版）中，我问了两个问题：

1.学习 C++的第一原则是什么？

2.使用 C++的第一原则是什么？

猜答案是什么？

1.学习 C++的第一原则是什么？

关注基本的（fundamental）概念和技术，而并非特定的语言特性，尤其不是C++中细枝末节的语言细节。

有人肯定会问，那还叫学习 C++吗？学习一门语言自然要从它的语言特性开始不是？否则还从何学起呢？这个问题的关键在于要意识到，我们当然是需要掌握语言特性的，但重点在于要从语言特性看到特性背后蕴藏的支持设计和编程的概念（concept）。比如类支持的概念，继承支持的概念，虚函数支持的概念，模板支持的

概念...这个论点泛化之后的结论就是 :学习编程重在学习基本的概念和素养，这些是长期稳定不变的东西。投入精力学习特定的细节，学得越是细，就越是容易淘汰。Bjarne 的说法就是：一旦掌握了基本的概念，要用到细节的时候，细节自然会 fall into places。

2.使用 C++的第一原则是什么？

将你的（pongba 按：与语言无关的）设计理念（概念）直接映射为 C++中的类或模板。

也即是我前一阵子文章中建议的“脱离语言思考，使用语言实现”。脱离语言思考的好处是显而易见的：可以避免受到语言细节作为既有框架的干扰，避免过早被实现细节缠住，于是便容易找到最直观的解决方案，即便后来发现语言成了绊脚石，也可以选择换语言或者明确地知道自己做了什么折衷。

结论

Think out of the box.

## 知其所以然

其实下文的绝大部分内容对所有学习都是同理的。只不过最近在正儿巴经地学算法，而后者又不是好啃的骨头，所以平时思考总结得就自然要比学其它东西要多一些。

问题：目前几乎所有的算法书的讲解方式都是欧几里德式的、瀑布式的、自上而下的、每一个推导步骤都是精准制导直接面向目标的。由因到果，定义、引理、定理、证明一样不少，井井有条一丝不乱毫无赘肉。而实际上，这完全把人类大脑创造发明的步骤给反过来了。看起来是阳关大道，实际上车马不通。

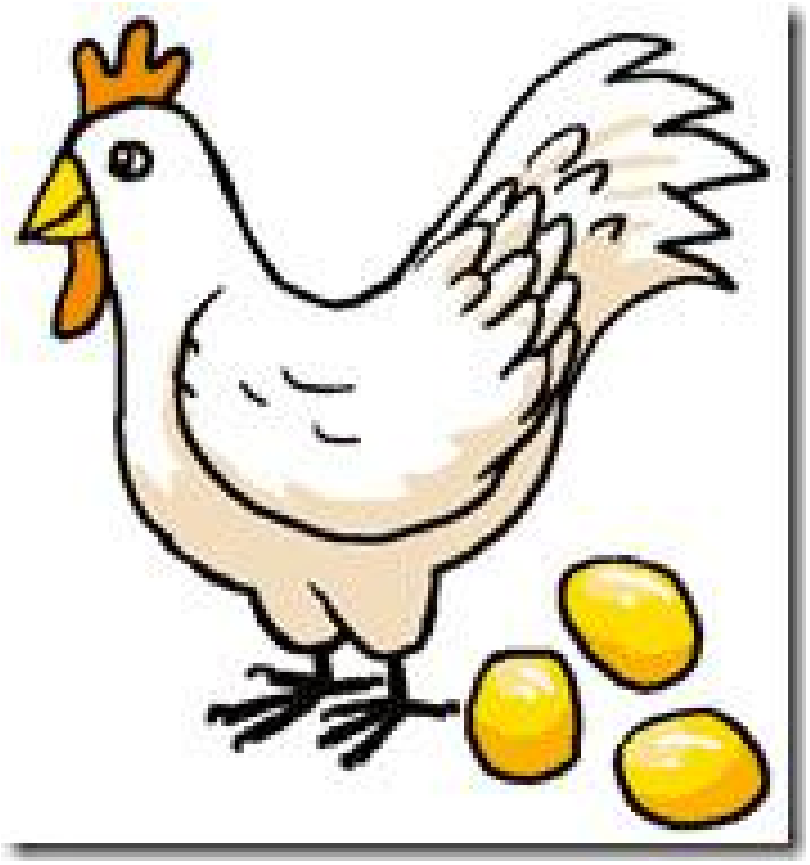

而对读者来说，这就等于直接告诉你答案&做法了，然后让你去验证这个答案&做法是可行&成立的。而关于答案&做法到底是怎么来的，从问题到答案之间经历了怎样的思维过程。却鲜有书能够很好的阐释。就我有限的阅（算法）书经验，除了波利亚的《怎样解题》还算合格之外（也并非最理想），其它的（包括有名的《算法导论》、《如何解题：现代启发式方法》、《Algorithms》、《编程珠玑》，甚至 TAOCP——公平地说由于高老大对算法领域历史了解得非常通透，所以许多地方能够从原始脉络来讲述一个问题，譬如令人印象深刻的从竞赛树到堆的讲解就寥寥一页纸道出了堆这个数据结构的本质

来，而像刚才列的几本有名的书却都没有做到），在思维的讲述上都算不上合格（当然不是说这些书没有价值，作为知识性的参考书籍，它们将知识整理出系统结构，极大的便利了知识的掌握，就像《什么是数学》所做的工作一样），为什么我这么说呢，因为我发现每每需要寻找对一个算法的解释的时候，翻开这些书，总是直接就看到关于算法逻辑的描述，却看不到整个算法的诞生过程背后的思想。

我们要的不是相对论，而是诞生相对论的那个大脑。我们要的不是金蛋，而是下金蛋的那只鸡。

Update(2008-7-24):收到不少同学的批评，想来这个开头对一些著作的语气过重了，实际上，注意，我完全不否认这些著作的价值，我自己也在通过阅读它们来学习算法，并且有很多收获。这篇文章更多的只是建议除了阅读这些著作之外还需要做的功课。此外，对于这类知识讲述（欧几里德）方式的批判西方（尤其是在数学领域）早就有了，早在欧拉和庞加莱的时候，他们俩

就极其强调思维的传授，欧拉认为如果不能传授思维，那数学教学是没意义的。而庞加莱本人则更是对数学思维有极大的兴趣和研究（我前阵子在讨论组上还转载了一篇庞加莱的著名演讲，就是说这个的，参见这里）。我只是在说目前的算法书没有做到思维讲述的层面，因此建议阅读这些书之余应该寻找算法的原始出处，应该寻根究底，多做一些功课，知道算法到底是怎么诞生的，并且我说明了为什么应该知其所以然，有哪些好处（见下文），我还给了几个例子譬如红黑树作者讲红黑树的，g9 讲后缀树的，以及 Knuth 讲 heap 的。唉，其实挺正统的观点，授人以渔，不管是东方西方都有类似的古老谚语。而我只是从认知科学的角度加了点解释，windstorm 称之为“解释文”。而已。可惜被开头的语气搞砸了，算了，既发了也就不改了。



为什么会这样，其实是有原因的。

我们在思考一个问题的过程中有两种思维形式：

联想：这种思维某种程度上可以说是“混乱”的（虽然从一个更根本的层面上说是有规则的），所谓混乱是指很多时候并不确定联想到的做法最终是否可行，这些联想也许只是基于题目中的某个词语、语法结构、问题的某个切片、一些零星局部的信息。这个过程是试探性的。最后也许有很大一部分被证明是不可行的。很多时候我们解决问题用的都是这种思维，简言之就是首先枚举你

关于这个问题能够想到的所有你学过的知识，然后——往上套看看能否解决手头的问题。这种思维方式受限于人脑联想能力本身的局限性。我在《跟波利亚学解题》中就提到了几个例子。联想本身需要记忆提取的线索，所以受到记忆提取线索的制约，如果线索不足，那怎么也联想不起来。而提取线索的建立又取决于当初保存记忆的时候的加工方法（《找寻逝去的自我》里面有阐述），同时，面对一个问题，你能够从中抽取出来的联想线索又取决于你对问题的认识层度/抽象深度，表浅的线索很可能是无关的，导致无效的联想&试错（《Psychology of Problem Solving》里面有阐述）。总之，联想这个过程充满了错误的可能。

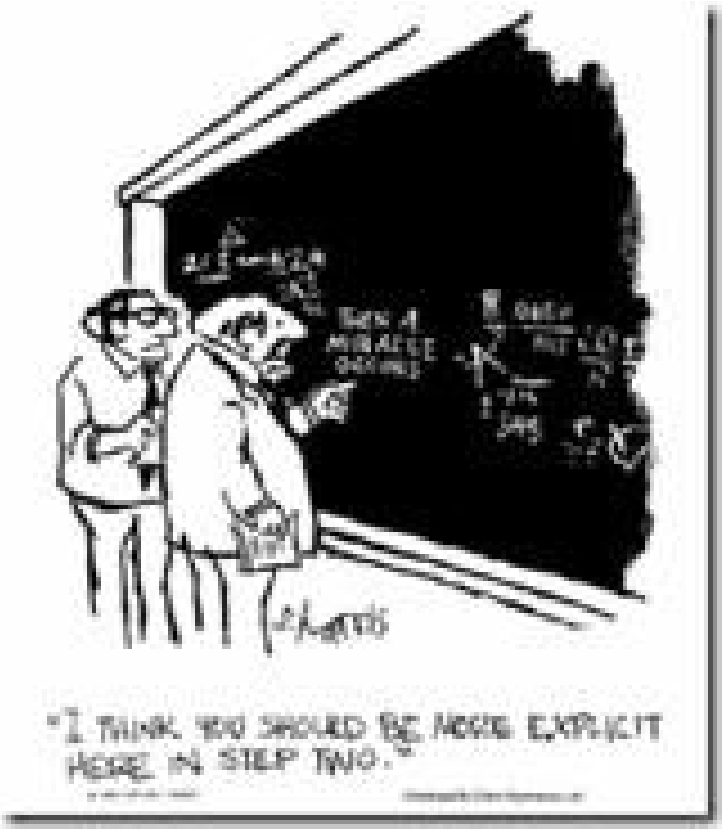
演绎&归纳：演绎&归纳是另一种思维形式。它们远比联想有根据。其中演绎是严格的，必然的。归纳也是有一定根据的。在面对一个问题的时候，我们有意无意的对问题中的各个条件进行着演绎；譬如福尔摩斯著名的“狗叫”推理——狗+生人=>吠叫&昨晚狗没有叫=>那个人是熟人。就是一个典型的对问题的各个条件进行

演绎的推理过程。还有就是通过对一些特殊形式的观察来进行归纳，试图总结问题中的规律。然而，不幸的是，面对复杂的问题，演绎&归纳也并不总是“直奔”问题的解决方案的。人的思维毕竟只能一下子看到有限的几步逻辑结论，一条逻辑演绎路径是否直奔答案，不走到最后往往是不知道的，只要答案还未出现，我们大脑中的逻辑演绎之树的末端就始终隐藏在黑暗之中。而当最终答案出现了之后，我们会发现，这棵演绎之树的很多分支实际上都并不通往答案。所以，虽然演绎&归纳是一种“必然”的推理，然而却并不“必然”引向问题的结论，它也是试错的，只不过比联想要更为靠谱一些。

既然认识到，人类解决问题的两大思维方式实际上都是有很大的试错成分的（好听一点叫“探索”），那么就不难意识到，对一个问题的思考过程实际上是相当错综复杂的，而且充满了无效分支——在思考的过程中我们也会不断的对分支进行评估，做适当的剪枝——因此当我们找到问题的解之后，一来思维的漫长繁杂的过程已经在大脑里面淡化得差不多了，只有那些引向最终结

论的过程会被加“高亮”——我们在思考的过程中本就会不断的抛弃无效的思路，只留下最有希望的思路。简而言之就是最后证明没用或者早先我们就不抱希望的一些想法就被从工作记忆中扔掉了。二来，思考过程是我们的空气和水，而“鱼是最后一个感觉到水的”，我们感觉不到思维法则本身的存在，我们只是不知不觉运用它。三来，由于我们的目标是问题的解，解才是我们为之兴奋和狂喜的东西，而不是求解的过程，过程只是过程，目的才是目的。这就像一个寻宝者，在漫长曲折的寻宝历程之后，在找到宝藏的时候，他会对宝藏感到狂喜（记得阿基米德的“找到了！”吗？）而迫不及待地要展示出来，而漫长的思考本身却成了注脚。我们是有目的的动物，目的达到了，其它的就相对不那么重要了。最后，对于传授知识的人，也许还有其四：感到介绍思维过程是不相干的，毕竟思维过程并不是算法问题的解，算法问题的解才是算法问题的解。然而不幸的是，忽视到达解的那个过程实际上却变成了舍本逐末。我们看到的是寥寥数行精妙绝伦的算法，然后仰天长叹自己想不出来

啊想不出来。为什么想不出来，因为你不知道那短短数行算法背后经历的事怎样漫长的思考过程，如果问题求解是一部侦探小说，那么算法只是结局而已，而思考过程才是情节。




既然如此，也就难怪古往今来算法牛人们算法牛，但却没有几个能真正在讲述的时候还原自己的思维过程的（那个“渔”字），手把手的教学生走一遍推理的思路，就可以让学生获得思维过程的训练。金出武雄在《像外行一样思考，像专家一样实践》中说写论文应该写得像侦探小说一样，我很赞同。欧几里德式的介绍，除了提

供枯燥的知识之外，并没有提供帮助人获得知识的东西——思维（关于对数学书籍的欧几里德式写法的批评其实也是由来已久了，并且有人呼吁了好几种其它的教学方法）。从这方面，我们所尊敬的一些“圣经”级书籍在传道授业上还不如侦探小说，前者是罗列一大堆知识，后者则是阐述获得知识的过程——推理&联想。

然而，我们都是人，人类该有的思维形式，我们难道不是都有吗。既然如此，思维本身又有什么需要一遍遍教的呢？

并非如此。

讲述思维过程而非结果有几个极其重要的价值：

•内隐化：思维法则其实也是知识（只不过它是元知识——是帮助我们获得新知识的知识）；是内隐的记忆。我们在思考的过程中觉察不到思维法则的作用，它们却在幕后实实在在的左右着我们的思维轨迹。要将思维方法内隐化，需要不断练习，就像需要不断练习才能无意识状态下就能骑自行车一样。

•跨情境运用：思维法则也是知识记忆，是问题解决策略。既然是记忆，就受到提取线索的制约，这就是为什么当波利亚告诉你要“注意未知数”之后你还是不能真正在所有需要你“注意未知数”的地方都能提醒自己“注意未知数”。很多时候未知数是很隐蔽的，未知数并不会总是头顶一个大帽子上面写着“我是未知数”。所以很多时候缺乏对这个策略的“提醒”线索，这也是为什么你学会了在解决数学问题的时候“注意未知数”却不一定能在解决现实生活中的问题中时刻都能“注意你的未知数”（《你的灯亮着吗？》整本书的价值便在于此），因为解数学题和解决生活中问题的场景不一样，不同的环境线索，在你大脑中激发的记忆也不一样。就连问题求解中，不同的问题之间的细小差别也可能导致思维轨迹很大的不同，有时你的注意力会被一个无关线索激发的联想吸引开去，忘记如“注意你的未知数”这样的重要法则。而一本从思维角度来讲问题求解的书则可以一遍遍将你置于不同的问题场景下然后在该提醒你的时候提醒你，让你醒悟到“哦，原来这个时候也应该想到这

个啊。”，做多了这样的思维演习你就会逐渐从中领悟到某种共性，并将一些思维习惯得到强化，于是终于能够在需要运用某策略的时候能适时的想起来了。

•对问题解的更多记忆提取线索：我们平时学习算法时几乎仅止于“理解”，别人把一个方案放在你面前，你去验证一下，心说“哦，不错，这个的确可以工作”。然后就没了。稍微简单一点的算法还好，复杂一点的对于记忆的负担是很大的，这就是为什么有时候我们看到一个绝妙的解法，这个解法看上去不知道从哪里来的，但经过我们的理解，却发现是对的，我们感叹，真巧妙，结果一些天之后，别人问起这个问题，我们说：“唉，那是个多么巧妙的算法啊，但是我只记得它巧妙，却不记得它到底是怎样的了。”为什么？因为在不知其所以然的情况下，算法只是一堆离散的机械步骤，缺少背后的思想的支撑，这些步骤之间就没有一个本质层面上的关联（先知亚里士多德早就指出：学习即联接）。所以就跟背历史书也没多大区别。然而，知道了算法是怎样一步步被推导出来的，我们就一下拥有了大量的记忆提取线索：

对算法发现过程中的任何一个关键步骤（尤其是本质）的回忆都可能使我们能够自己动手推导出剩余的内容。譬如你知道堆(heap)是怎样由朴素的决策树演化而来的，它又是为了解决什么问题的，你即便忘记了具体的细节，也可以自己推导出来。譬如你知道 KMP 算法的本质在于消除回溯，至于如何消除回溯却并不是那么难以推导的，所以即便忘了也可以借助于大脑的逻辑演绎能力再现出来。譬如你知道 Tarjan 算法其实只是从后序遍历经过两个优化调整而来的（其中并查集的使用其实只是优化手段——为了能够迅速判断祖先节点是谁——而非算法本质——当然，算法设计的主要任务本来就是通过问题条件中蕴含的知识来“消除冗余计算”和“避免不必要计算”，所以你也可以说并查集的使用是关乎本质的，只不过，知道了为什么需要引入并查集，就会强烈地感觉到一切是顺理成章的了），那这个出了名的绕人的算法也就不那么难以理解和记忆了。譬如你知道排序的本质，就能够对什么是最优排序，为什么它是最优排序有深刻的认识。四两拨千斤。

•包含了多得多的知识：记一个算法，就只有一个算法。一个萝卜一个坑。就好比背 99 乘法表只能解决乘法问题一样。而记背后的思想，却有助于解决一类问题。思想所处的抽象层面往往比到处都是实现细节的算法本身要低，越是低的抽象层次，越是本质，涵盖范围越是广泛。数学的发展本身就体现了这个过程，抽象代数就是非常好的例子。算法诞生过程中的思路往往包含了比实际算法更本质得多的知识，实际算法乃至算法的某个特定语言的实现包含了太多表面的不相干知识，它们会阻碍对本质的理解。

•重在分析推理，而不是联想：学了一大通算法和数据结构之后的一个副作用就是，看到一个问题之后，脑袋里立即不管三七二十一冒出一堆可能相干的数据结构和算法来。联想是强大的思维捷径，在任何时候都会抢占大脑的工作记忆，由不得你控制——比如我问你“如何寻找区间的最大值”，首先进入你的意识的肯定就是学过的那个算法，甚至算法的实现细节都一一跳了出来，也许最先跳出来的还是算法实现中某个最容易弄错的边

界细节，或是某个比较 tricky 的实现技巧！然而这些其实根本不反映一个算法的本质，结果想来想去总是停留在问题的表层。而另一方面，重在思维的传授则可以让人养成从问题本质入手，逐步分析推理的习惯，而不是直接生搬硬套。当然，完全不可否认，联想本身也是极其重要的思维方法，甚至可以说是人类思维最重要的特征。很多时候我们并不知道问题的本质是什么，就需要靠联想、类比来领路探索。只不过，养成优先从问题的本质入手进行考察的好习惯绝对是有更大的好处的。

那到底什么样的才算是授人以渔的呢？波利亚的《如何解题》绝对算是一本，他的《数学的发现》也值得一看。具体到算法书，那就不是光看 text book 就足够的了，为了深入理解一个算法的来龙去脉前因后果，从一个算法中领悟尽量深刻的东西，则需要做到三件事情：

•寻找该算法的原始出处：TAOCP 作为一个资料库是绝对优秀的，基础的算法只要你能想到的，几乎都可

以在上面找到原始出处。查到原始出处之后( 譬如一篇 paper ), 就可以去网上搜来看了。因为最初的作者往往对一个方案的诞生过程最为了解。比如经典数据结构中的红黑树是出了名的令人费解的结构之一，但它的作者 Sedgewick 一张 PPT，给你讲得通通透透，比算法导论上的讲法强上数倍。

•原始的出处其实也未必就都推心置腹地和你讲得那么到位：前面说过，算法设计出来了之后人们几乎是不会去回顾整个的思维过程细节的，只把直指目标的那些东西写出来。结果就又是一篇欧几里德式的文章了。于是你就迷失在一大堆“定义”、“引理”、“定理”之中了。这种文章看上去整个写得井井有条，其实是把发明的过程整个给颠倒过来了，我一直就想，如果作者们能够将整个的思路过程写出来，哪怕文字多上十倍，我也绝对会比看那一堆定义定理要容易理解得多。话说回来，怎么办？可以再去网上找找，牛人讲得未必比经典教材上的差。那倘若实在找不出好的介绍呢，就只能自己揣摩了。揣摩的重要性，是怎么说都不为过的。揣摩的一

些指导性的问题有：为什么要这样（为什么这是好的）？为什么不是那样（有其它做法吗？有更好的做法吗？）？这样做是最好的吗？（为什么？能证明吗？）这个做法跟其它的什么做法有本质联系吗？这个跟这个的区别是什么？问题的本质是什么？这个做法的本质又是什么？到底本质上是什么东西导致了这个做法如此..？与这个问题类似的还有其它问题吗？（同样或类似的做法也适用吗？）等等。

•不仅学习别人的思路，整理自己的思路也是极其重要的：详见《跟波利亚学解题》的“4.一个好习惯”和“7.总结的意义”。

## 为什么有必要知其所以然

查了一下，上篇知其所以然（以学习算法为例）是 08 年 7 月写的，现在已经是 10 年 11 月，过去了两年零 4 个月，这说明了三件事情：1，一个问题其实你可以一直放在脑子里面，利用暗时间对其软泡硬磨，时间足够久你总会有一点新的感悟，问题其实就像那句老话说的

那样，不怕贼偷就怕贼惦记，聚精会神的思考一天，也许比不上惦记一个星期（据说数学家庞加莱就特别会惦记问题）。2，事实上，当你感觉懂了的时候，你至少得反问自己一句，真的懂了吗？当你确信自己真的懂了的时候，你至少得讲给别人听，别人听懂了吗？考察你自己是否真懂了的一个很好的依据是，你是否有一种“哦，原来是这样啊，这下再也不可能忘记了”的感觉。3，我其实没有忘记这个博客。如我之前说的，记录只是学习和思考的副作用，只要还在学习和思考，就必然会有新的记录。

我有一个习惯，看定理必看证明。一个你不明白其证明的定理在我看来比不知道这个定理还要糟糕，因它给你造成一种懂了的错觉。在没有明白背后的证明之前，任何一个定理对你来说都是等价的——等价于背乘法口诀（只不过有的长一点有的短一点）。一个原本美妙的定理，把其证明扔掉就是真正的买椟还珠，暴殄天物。

从现实意义来说，去理解一个定理的证明会带来巨

大的好处，首当其冲的好处就是你很难再忘掉它。这一点其实很容易解释——在理解一个定理的证明之前，定理对你而言是一堆没有内在联系的词句，而在理解了证明之后，定理就归约为证明它所需的条件加上逻辑，“逻辑”本来就存在于你的大脑里面，而证明的过程中除了公理和用到的常见定理（往往没几条）之外，宽泛地说，需要你去记的，一般来说也只有一个或两个关键的 insights，也就是我们常说的证明中的神来之笔，比如几何证明里面的某条看上去莫名其妙的辅助线，一旦你知道了这条辅助线，那么整个证明就毫无难处，那么该定理的信息量便直接缩减为一条辅助线的信息量；虽然看上去这一步信息并没有缩减多少，但是如果你考虑到类似的辅助线不仅会用在这个特定的定理上，往往会在很多地方用到。很多关键的证明手法是通用的。那么其实你就是把所有以这个辅助线为关键证明手法的定理的集合的信息量归约为了这条辅助线。如果你进而甚至能够理解了作这条辅助线的思想精髓，那就更牛逼了，因为解决问题的思路更具有一般性，理解了寻找正确的辅助线

的思路，你就根本不需要去记得某条特定辅助线的作法，你就把所有以作一条或几条辅助线为证明核心的定理的集合的信息量归约为了这个“寻找辅助线的思路”。

这是一个树状的知识结构，越往上层走，需要记忆的节点就越少。所谓触类旁通者，其实便是因为他擅长去理解解法背后的更具一般性的东西。所以我还有一个习惯，就是看到美妙的证明和解法总是会去一遍又一遍的去反复揣摩，试图理解想出这个证明的人到底是怎么想出来的，有没有什么一般性的方法可循，很多时候，在这样揣摩的过程中，你会理解到更深刻的东西，对问题性质更深刻的认识，对解决问题的思路更深刻的认识，这些认识不仅对于你理解当前这个定理或问题有极大的帮助，同时也有助于你解决以后会遇到的表面不同但本质一样的问题。

与看定理必看证明类似，看一个问题的解法，必然要看解法所诞生的过程，背后是否隐藏着更具一般性的解决问题的思路和原则。否则一个解法就只是一个问题

的解法，跟背口诀一样。即便记住了也无法推广，即便当时记住了也容易遗忘。

举个经典的例子：每本算法书都会讲动态规划，每本讲动态规划的书都会讲背包问题，每次讲背包问题都会讲可重复背包和 01 背包，我们就拿《Algorithms》这本还算不错的算法书对背包问题的讲解来说吧，重复背包问题的递归公式是这样的：

K(W)=max{K(W-Wi)+Vi:Wi<=W}

这个公式的理解倒是很简单：为了把问题降阶，我们在最终的最优解里面去掉一个元素，对这个元素的可能性进行讨论，它必然是任何 Vi 之一（前提是 Wi<=W，否则就装不下），而在去掉这个元素之后，剩下的元素肯定构成问题 K(W-Wi)的最优解，于是递归关系出现了。

此外也可以这样来理解：要拿一组最优元素，那么总得开始一个个拿吧，对第一个拿的元素进行讨论，而问题的最优解等于讨论的各个分支的最优解中的最优者；如果拿掉 Vi 之后，剩下来要怎么拿才能最优呢？这

就是一个 K(W-Wi)的问题了。

01 背包问题就大不一样了——每个物品都只有一件，拿掉之后就不能再拿了。我们不妨看看重复背包问题的解法是不是能用到 01 背包上呢？还是讨论第一个拿的元素，设被拿掉的是第 i 个元素，问题就归结为把剩下的物品（注意，可拿的物品少了一件）最优地装入容量为 W-Wi 的包里，所以，问题的参数便变成了两个，一个是背包剩余容量 W-Wi，另一个是剩余可拿的物品集合 S\'7bi}（表示去掉 i 之后的子集），显而易见第二个参数是物品集合的各种可能的子集，那么其可能性个数就是 2^n，这就导致子问题的个数是 2^n，由于要依次计算每个子问题，那么算法复杂度显然也是 2^n，是不可接受的。

那么，《Algorithms》上又是怎么来讲解 01 背包问题的解法的呢？以下是原文：

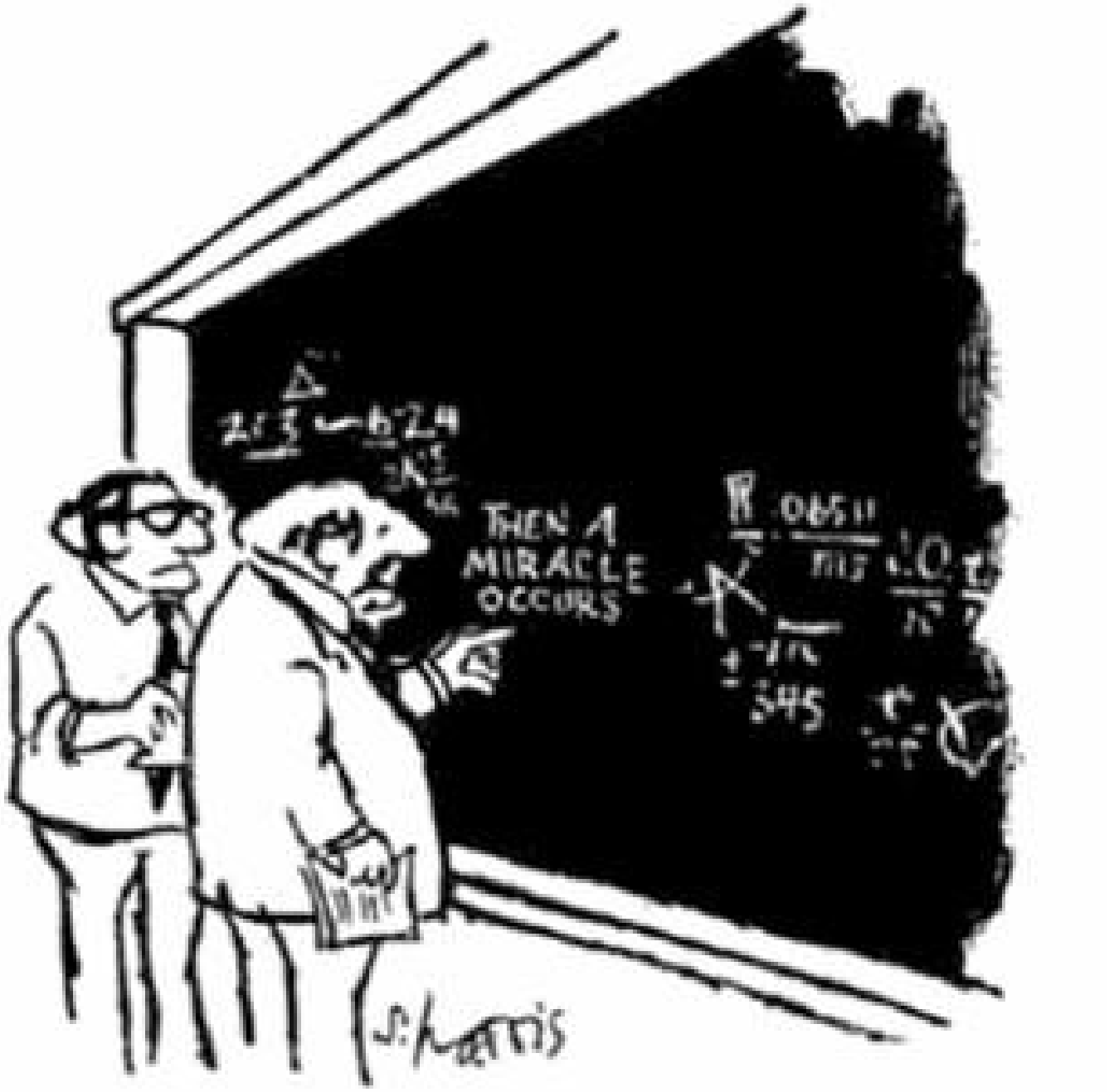
Our earlier subproblems now become completely useless.We must therefore refine our concep

t of a subproblem to carry additional information about the items being used.We add a second parameter,0<=j<=n:K(W,j)=maximum value achievable using a knapsack of capacity w and items 1..j:The answer we seek is K(W,n).

首先作者说了，之前重复背包问题的解法在这里完全废掉了，所以我们必须重新定义子问题，并且子问题的条件必须要包含目前拿剩下的物品。以上这些都还不错，关键是接下来就让人吐血了。作者接着说道，我们给子问题加上一个新的参数 j...

凭什么啊？

还是让我们回顾一下这样一幅经典的漫画吧：




"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

"我们给子问题加上一个参数 j"，这就像你在看数学证明时看到无比邪恶的"我们考虑..."一样，一看到这样的句子，你就知道，这个问题的证明远远不像看上去

那么简单，之所以你一路看下去理解上全无困难，那完全是因为作者直接把最重要的一个 insight 告诉你了，举个很简单的例子，证明素数无最大，谁都会第一时间想到去反证：假设存在一个最大的素数 P，那么找到比 P 大的素数就是证明中最关键的一步，怎么找的？一般书上是不会说的，你会看到书上这样说：假设 P 是最大的素数，那么我们考虑 P’=小于等于 P 的所有素数的乘积+1。那么 P’ 一来显然大于 P，二来不能被小于它的所有素数整除，那么 P’ 就成了大于 P 的素数。

如果你经常注意反证法，你会发现一个有趣的现象，反证法里面经常会有这样一句“我们考虑”，而“我们考虑”后面几乎肯定接着一个天外飞仙一般的 insight。素数无最大这个古老的证明里面的“我们考虑”尚算是比较有迹可循的（我们想要构造一个更大的素数，而素数的等价定义就是“不能被小于它的所有素数整除，为了达到这个目的，构造的方法就较明显了）。但是有非常非常多的证明，其中关键的一步就跟嗑药磕出来做梦做出来走路跌跟头跌出来的一样（不信去翻一翻《Proofs fr

om THE Book》)，让你完全不知道他怎么想到的。

话说回来，虽然有很多数学证明的关键步骤是很难逆向工程的（因为很多时候想出那个关键步骤的本人其实也是尝试了各种方法，撞了无数堵墙，在寻求证法的尝试空间中作了 N 次回溯才“妙手偶得”，与其说是妙手偶得，不如说是绞尽脑汁），但并非全无章法可循，否则陶哲轩也不会写出《Solving Mathematical Problems》这样的著作来，而求解问题也就成了真正的 Black Art 了。

算法的解法则比精妙的数学证明稍加更容易逆向工程一点。只要你有耐心仔细地去琢磨算法的关键步骤和本质，总能从中窥探到一些更 general 的思想和思路来。

此外，很多经典问题，算法书上的讲法虽然时时令我们失望，但如果去网上一搜，则通常会发现更优秀的解释来。比如背包问题就是如此。

简单地说，如果你对于每个问题都能真正弄清以下这几个问题的答案，那么可以肯定的是，你的理解，记

忆，以及学习的效率都会得到质的提高：

为什么这种解法是对的？

为什么那种解法是错的？

为什么这种解法不是最优的？

证明为什么没有更优的解法。

回到人民群众喜闻乐见的经典例子：背包问题。为什么 01 背包问题的正确（高效）算法是正确（高效）的。表面的解释是，因为 01 背包问题的子问题定义是 K(W, j)，其两个维度相乘的可能性一共有 nW 种，也就是说一共要计算 nW 个子问题，而计算每个子问题的复杂度是 O(1)的。

但是如果仅仅满足于这样的解释，可以说是隔靴搔痒，并没有触及到本质。算法本质上可以看做是在一个解空间当中的搜索问题，所以要分析一个算法的好坏，首先弄清它的解空间的结构，然后分析它是怎么来探索这个解空间的。

弄清解空间的是第一步，例如排序算法，其解空间可以看做是所有可能的下标排列组合，其中有且仅有一个排列是正确的排序排列（简单起见假设元素各不相同）。那么一个算法在探索这个解空间方面的行为就决定了它的效率高低，最简单的，如果一个算法每次只能检查解空间中的一个点，那么这个算法的复杂度就是解空间的大小。对排序算法而言也就是 n!。从这个角度来看，我们就会很容易的发现，所有基于比较的排序算法，其复杂度为什么是以 O(nlogn)为下界的，因为一次比较操作最多有两个结果，a>b 或 a<b，既然只有两种结果，那么最多只能将解空间进行 2 分，如果每次都能完美的 2 分，那么找到那个唯一点最终需要的步骤就是 log(n!)=O(nlogn)。如此就不难理解什么基于比较的排序算法的复杂度最好不过如此了。

回到 01 背包问题，01 背包问题的解空间其实也是类似的。一次选取就是一个 01 数组，其中每个元素代表其所对应的物品要不要选取。很显然，这个解空间的大小是 2^n。在 01 背包的算法里面，每当我们解出 K(W,

j)（需要 O(W)次计算）之后，解空间就会被折半（排除掉 1/2 的可能性），一共如此做 n 次，就能得到最终解。由于每次折半的代价是 O(W) ,便不难理解为什么算法复杂度是 O(nW)了。

那么，为什么每次计算出 K(W,j)就能使解空间折半呢？那就需要来看看这个算法是如何探索解空间的，算法探索解空间的方式在其递归公式里面：

K(W,j)=max{K(W,j-1),K(W-Wj,j–1)+Vj}

也就是说，首先看你要不要选取第一个物品，有两种可能性（两个分支），每个分支都是一个更低阶的子问题，即在其中的任意一个分支下都要决定要不要选取第二个物品（又是两个分支），如此下递归去，可以构建出一棵有 2^n 方个叶子节点的树，每条从根结点到叶子节点的路径“01..101”就对应一个解，其中每个分叉代表“选”或“不选”当前的物品。

建立在对这个解空间的理解上 ,我们再来看为什么 0 1 背包问题的正确解法能做到 O(nW)。（首先你最好将这

棵树画在纸上，其中每个节点都是一个子问题 K(W,j)，每条分叉都是 0 或 1。）当我们计算出所有的 K(W,1)（需要 O(W)次操作）之后，我们容易注意到，所有离叶子节点的距离为 1 的内部节点 K(W,2)到叶子节点的两个分支都必然只能取其一了，也就是说，有一半的叶子节点被排除掉了（对解空间折半）。当我们进而计算出 K(W,2)之后，同样的道理，我们容易看到，到叶子节点距离为 2 的内部节点的两个分支也只能取其一了，这就进而再次将解空间折半。由于每次折半需要 O(W)的复杂度，所以就不难理解算法的总复杂度为 O(nW)了。另一种理解的方法是，当我们计算出 K(W,j)的时候，从内部节点 K(W,j)到根节点的唯一路径便确定了。经过 O(nW)次计算，从根节点到那个唯一解（叶子节点）的路径便完全确定了。

知道怎么做是从正确（高效）解法得到的，而知道为什么必须得那样做则往往是从错误（低效）的解法当中得到的。

然而遗憾的是，绝大多数算法书或教程都只顾一上来就告诉你正确的做法是什么，对于一些常见的错误解法，或者常见的低效解法，却根本不加分析。经验告诉我们，理解错误的做法为什么错误同样甚至更为重要，往往是在理解了错误的解法为什么错误之后，我们才能深刻的体会到为什么正确的解法是如此正确。

还是拿经典的背包问题来作例子，你几乎看不到哪本书会告诉你一个典型的低效解法为什么低效的深刻原因。我们都知道动态规划的核心在于子问题的划分，同样的问题，不同的划分办法得到的复杂度完全不一样。前面已经提到了，重复背包问题的思路在 01 背包问题上会带来指数级的复杂度，但是为什么呢？如果你满足于说：因为如果拿重复背包问题的思路来解 01 背包问题，那么子问题定义的第二个维度（物品的子集）（见前文）是指数级的，那么要计算所有子问题，当然是指数级的。那么你只是看到这个问题的表象。

如果从对解空间的探索方式来说，可以容易看出这

个现象的本质 ,我们回顾一下 01 背包问题的正确( 高效 )算法：

K(W,j)=max{K(W,j-1),K(W-Wj,j–1)+Vj}

这个算法讨论的是两种情况，“要”或者“不要”选取第 j 个物品，这两种情况所对应的解空间是完全不交的，这就有效地将解空间划分为了不重复的两个部分。

而再来看利用重复背包问题思路的解法：

K(W,S)=max{K(W-Wi,S\'7bi})+Vi:Wi<=W}

这里讨论的是首先拿掉哪一个物品，还是那句话，讨论的每一个分支都对应了算法对解空间的一个切分，我们容易看出，在“先拿物品 i”和”先拿物品 j“这两个分支里面，存在大量的重复，因为先拿物品 i 再拿 j，和先拿物品 j 再拿 i 对应的是完全一样的一组选取。事实上，如果你将这个递归公式画成树状结构，会发现有 n!个叶子节点。n!是什么概念？01 背包问题的解空间大小本质上就只有 2^n 次方 ,穷举也不过 O(2^n)的复杂度，

结果这样一切分却变成了 n!，可见这种对解空间的切分方法的冗余度是多么高了。你不妨看看 ,每一次计算 K(W, S)子问题能对解空间排查多少呢？是否能像前面正确的算法那样，每次都能有效排查一半情况？理解了这一点之后，我们便注意到在划分解空间，也就是定义子问题的时候的一个原则，就是在建立递归公式的时候，尽量将解空间进行不交的切分。同时我们便有了趁手的工具去分析一个动态规划的解法的效率。

最后再举一个例子：算法书上几乎必讲的霍夫曼树。你所看的算法书在讲霍夫曼树的时候给了证明吗？讲过霍夫曼树的历史八卦吗？也许你看了霍夫曼树的构造方法之后觉得："哦，这样啊，显然"。但是你可曾想到，在最优编码这个问题上，连香农本人之前给出的解法都只是 suboptimal 的，而且霍夫曼本人在得到这个算法之前也是绞尽脑汁几近放弃。如果你 10 分钟就"理解"了，那么百分之百只是背了课文而已。

**康托尔、哥德尔、图灵——永恒的金色对角线**

我看到了它，却不敢相信它[1]。

——康托尔

计算机是数学家一次失败思考的产物。

——无名氏

哥德尔的不完备性定理震撼了 20 世纪数学界的天空，其数学意义颠覆了希尔伯特的形式化数学的宏伟计划 ,其哲学意义直到 21 世纪的今天仍然不断被延伸到各个自然学科，深刻影响着人们的思维。图灵为了解决希尔伯特著名的第十问题而提出有效计算模型，进而作出了可计算理论和现代计算机的奠基性工作，著名的停机问题给出了机械计算模型的能力极限，其深刻的意义和漂亮的证明使它成为可计算理论中的标志性定理之一。丘齐，跟图灵同时代的天才，则从另一个抽象角度提出了 lambda 算子的思想，与图灵机抽象的倾向于硬件性不同，丘齐的 lambda 算子理论是从数学的角度进行抽象，不关心运算的机械过程而只关心运算的抽象性质，只用最简洁的几条公理便建立起了与图灵机完全等价的

计算模型，其体现出来的数学抽象美开出了函数式编程语言这朵奇葩，Lisp、Scheme、Haskell...这些以抽象性和简洁美为特点的语言至今仍然活跃在计算机科学界，虽然由于其本质上源于 lambda 算子理论的抽象方式不符合人的思维习惯从而注定无法成为主流的编程语言[2]，然而这仍然无法妨碍它们成为编程理论乃至计算机学科的最佳教本。而诞生于函数式编程语言的神奇的 Y combinator 至今仍然让人们陷入深沉的震撼和反思当中...

然而，这一切的一切，看似不很相关却又有点相关，认真思考其关系却又有点一头雾水的背后，其实隐隐藏着一条线，这条线把它们从本质上串到了一起，而顺着时光的河流逆流而上，我们将会看到，这条线的尽头，不是别人,正是只手拨开被不严密性问题困扰的 19 世纪数学界阴沉天空的天才数学家康托尔，康托尔创造性地将一一对应和对角线方法运用到无穷集合理论的建立当中，这个被希尔伯特称为“谁也无法将我们从康托尔为我们创造的乐园中驱逐出去”、被罗素称为 “19 世纪最

伟大的智者之一”的人，他在集合论方面的工作终于驱散了不严密性问题带来的阴霾，仿佛一道金色的阳光刺破乌云，19 世纪的数学终于看到了真正严格化的曙光，数学终于得以站在了前所未有的坚固的基础之上；集合论至今仍是数学里最基础和最重要的理论之一。而康托尔当初在研究无穷集合时最具天才的方法之一——对角线方法——则带来了极其深远的影响，其纯粹而直指事物本质的思想如洪钟大吕般响彻数学和哲学的每一个角落[3]。随着本文的展开，你将会看到，刚才提到的一切，歌德尔的不完备性定理，图灵的停机问题，lambda 算子理论中神奇的 Y combinator、乃至著名的罗素悖论、理发师悖论等等，其实都源自这个简洁、纯粹而同时又是最优美的数学方法，反过来说，从康托尔的对角线方法出发，我们可以轻而易举地推导出哥德尔的不完备性定理，而由后者又可以轻易导出停机问题和 Y combinator，实际上，我们将会看到，后两者也可以直接由康托尔的对角线方法导出。尤其是 Y combinator，这个形式上绕来绕去，本质上捉摸不透，看上去神秘莫测的算子，

其实只是一个非常自然而然的推论，如果从哥德尔的不完备性定理出发，它甚至比停机问题还要来得直接简单。总之，你将会看到这些看似深奥的理论是如何由一个至为简单而又至为深刻的数学方法得出的，你将会看到最纯粹的数学美。

图灵的停机问题(The Halting Problem)

了解停机问题的可以直接跳过这一节，到下一节“Y Combinator”，了解后者的再跳到下一节“哥德尔的不完备性定理”

我们还是从图灵著名的停机问题说起，一来它相对来说是我们要说的几个定理当中最简单的，二来它也最贴近程序员。实际上，我以前曾写过一篇关于图灵机的文章，有兴趣的读者可以从那篇开始，那篇主要是从理论上阐述，所以这里我们打算避开抽象的理论，换一种符合程序员思维习惯的直观方式来加以解释。

停机问题

不存在这样一个程序（算法），它能够计算任何程序（算法）在给定输入上是否会结束（停机）。

那么，如何来证明这个停机问题呢？反证。假设我们某一天真做出了这么一个极度聪明的万能算法（就叫 God_algo 吧），你只要给它一段程序（二进制描述），再给它这段程序的输入，它就能告诉你这段程序在这个输入上会不会结束（停机），我们来编写一下我们的这个算法吧：

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

这里我们假设 if 的判断语句里面是你天才思考的结

晶，它能够像上帝一样洞察一切程序的宿命。现在，我们从这个 God_algo 出发导出一个新的算法：

```
bool Satan_algo(char*program)
{
if(God_algo(program,program)){
while(1);//loop forever!
return false;//can never get here!
}
else
return true;
}
```

正如它的名字所暗示的那样，这个算法便是一切邪恶的根源了。当我们把这个算法运用到它自身身上时，会发生什么呢？

Satan_algo(Satan_algo);

我们来分析一下这行简单的调用：

显然，Satan_algo(Satan_algo)这个调用要么能够运行结束返回（停机），要么不能返回（loop forever）。

如果它能够结束，那么 Santa_algo 算法里面的那个 if 判断就会成立（因为 God_algo(Santa_algo,Santa_algo)将会返回 true），从而程序便进入那个包含一个无穷循环 while(1);的 if 分支，于是这个 Satan_algo(Satan_algo)调用便永远不会返回（结束）了。

而如果 Satan_algo(Satan_algo)不能结束（停机）呢，则 if 判断就会失败，从而选择另一个 if 分支并返回 true，即 Satan_algo(Satan_algo)又能够返回（停机）。

总之，我们有：

Satan_algo(Satan_algo)能够停机=>它不能停机

Satan_algo(Satan_algo)不能停机=>它能够停机

所以它停也不是，不停也不是。左右矛盾。

于是，我们的假设，即 God_algo 算法的存在性，便不成立了。正如拉格朗日所说：“陛下，我们不需要（上帝）这个假设”[4]。

这个证明相信每个程序员都能够容易的看懂。然而，这个看似不可捉摸的技巧背后其实隐藏着深刻的数学原理（甚至是哲学原理）。在没有认识到这一数学原理之前，至少我当时是对于图灵如何想出这一绝妙证明感到无法理解。但后面，在介绍完了与图灵的停机问题“同构”的 Y combinator 之后，我们会深入哥德尔的不完备性定理，在理解了哥德尔不完备性定理之后，我们从这一同样绝妙的定理出发，就会突然发现，离停机问题和神奇的 Y combinator 只是咫尺之遥而已。当然，最后我们会回溯到一切的尽头，康托尔那里，看看停机问题、Y combinator、以及不完备性定理是如何自然而然地由康托尔的对角线方法推导出来的，我们将会看到这些看似神奇的构造性证明的背后，其实是一个简洁优美的数学

方法在起作用。

Y Combinator

了解 Y combinator 的请直接跳过这一节，到下一节“哥德尔的不完备性定理”。

让我们暂且搁下但记住绕人的图灵停机问题，走进函数式编程语言的世界，走进由跟图灵机理论等价的 lambda 算子发展出来的另一个平行的语言世界。让我们来看一看被人们一代一代吟唱着的神奇的 Y Combinator...

关于 Y Combinator 的文章可谓数不胜数，这个由师从希尔伯特的著名逻辑学家 Haskell B.Curry（Haskell 语言就是以他命名的，而函数式编程语言里面的 Curry 手法也是以他命名）“发明”出来的组合算子（Haskell 是研究组合逻辑(combinatory logic)的）仿佛有种神奇的魔力，它能够算出给定 lambda 表达式（函数）的不动点。从而使得递归成为可能。事实上，我们待会就会看到，Y Combinator 在神奇的表面之下，其实隐藏

着深刻的意义，其背后体现的意义，曾经开出过历史上最灿烂的数学之花，所以 MIT 的计算机科学系将它做成系徽也就不足为奇了[5]。

当然，要了解这个神奇的算子，我们需要一点点 lambda 算子理论的基础知识，不过别担心，lambda 算子理论是我目前见过的最简洁的公理系统，这个系统仅仅由三条非常简单的公理构成，而这三条公理里面我们又只需要关注前两条。

以下小节——lambda calculus——纯粹是为了没有接触过 lambda 算子理论的读者准备的，并不属于本文重点讨论的东西，然而要讨论 Y combinator 就必须先了解一下 lambda（当然，以编程语言来了解也行，但是你会看到，丘齐最初提出的 lambda 算子理论才是最最简洁和漂亮的，学起来也最省事。）所以我单独准备了一个小节来介绍它。如果你已经知道，可以跳过这一小节。不知道的读者也可以跳过这一小节去 wikipedia 上面看，这里的介绍使用了 wikipedia 上的方式

lambda calculus

先来看一下 lambda 表达式的基本语法(BNF)：

<expr>::=<identifier>

<expr>::=lambda<identifier-list>.<expr>

<expr>::=(<expr><expr>)

前两条语法用于生成 lambda 表达式（lambda 函数），如：

lambda x y.x+y

haskell 里面为了简洁起见用“\u8221?来代替希腊字母 lambda，它们形状比较相似。故而上面的定义也可以写成：

\~x y.x+y

这是一个匿名的加法函数，它接受两个参数，返回两值相加的结果。当然，这里我们为了方便起见赋予了 lambda 函数直观的计算意义，而实际上 lambda calcul

us 里面一切都只不过是文本替换，有点像 C 语言的宏。并且这里的“+”我们假设已经是一个具有原子语义的运算符[6]，此外，为了方便我们使用了中缀表达（按照 lambda calculus 系统的语法实际上应该写成“(+x y)”才对——参考第三条语法）。

那么，函数定义出来了，怎么使用呢？最后一条规则就是用来调用一个 lambda 函数的：

((lambda x y.x+y)2 3)

以上这一行就是把刚才定义的加法函数运用到 2 和 3 上（这个调用语法形式跟命令式语言(imperative language)惯用的调用形式有点区别，后者是“f(x,y)”，而这里是“(f x y)”，不过好在顺序没变:)）。为了表达简洁一点，我们可以给(lambda x y.x+y)起一个名字，像这样：

let Add=(lambda x y.x+y)

这样我们便可以使用 Add 来表示该 lambda 函数

了：

(Add 2 3)

不过还是为了方便起见，后面调用的时候一般用“Add(2,3)”，即我们熟悉的形式。

有了语法规则之后，我们便可以看一看这个语言系统的两条简单至极的公理了：

Alpha 转换公理：例如，“lambda x y.x+y”转换为“lambda a b.a+b”。换句话说，函数的参数起什么名字没有关系，可以随意替换，只要函数体里面对参数的使用的地方也同时注意相应替换掉就是了。

Beta 转换公理：例如，“(lambda x y.x+y)2 3”转换为“2+3”。这个就更简单了，也就是说，当把一个 lambda 函数用到参数身上时，只需用实际的参数来替换掉其函数体中的相应变量即可。

就这些。是不是感觉有点太简单了？但事实就是如此，lambda 算子系统从根本上其实就这些东西，然而

你却能够从这几个简单的规则中推演出神奇无比的 Y combinator 来。我们这就开始！

递归的迷思

敏锐的你可能会发现，就以上这两条公理，我们的 lambda 语言中无法表示递归函数，为什么呢？假设我们要计算经典的阶乘，递归描述肯定像这样：

f(n):

if n==0 return 1

return n*f(n-1)

当然，上面这个程序是假定 n 为正整数。这个程序显示了一个特点，f 在定义的过程中用到了它自身。那么如何在 lambda 算子系统中表达这一函数呢？理所当然的想法如下：

lambda n.If_Else n==0 1 n*<self>(n-1)

当然，上面的程序假定了 If_Else 是一个已经定义好

的三元操作符（你可以想象 C 的“?:”操作符，后面跟的三个参数分别是判断条件、成功后求值的表达式、失败后求值的表达式。那么很显然，这个定义里面有一个地方没法解决，那就是<self>那个地方我们应该填入什么呢？很显然，熟悉 C 这类命令式语言的人都知道应该填入这个函数本身的名字，然而 lambda 算子系统里面的 lambda 表达式（或称函数）是没有名字的。

怎么办？难道就没有办法实现递归了？或者说，丘齐做出的这个 lambda 算子系统里面根本没法实现递归从而在计算能力上面有重大的缺陷？显然不是。马上你就会看到 Y combinator 是如何把一个看上去非递归的 lambda 表达式像变魔术那样变成一个递归版本的。在成功之前我们再失败一次，注意下面的尝试：

let F=lambda n.IF_Else n==0 1 n*F(n-1)

看上去不错，是吗？可惜还是不行。因为 let F 只是起到一个语法糖的作用，在它所代表的 lambda 表达式还没有完全定义出来之前你是不可以使用 F 这个名字的。

更何况实际上丘齐当初的 lambda 算子系统里面也并没有这个语法元素，这只是刚才为了简化代码而引入的语法糖。当然，了解这个 let 语句还是有意义的，后面还会用到。

一次成功的尝试

在上面几次失败的尝试之后，我们是不是就一筹莫展了呢？别忘了软件工程里面的一条黄金定律："任何问题都可以通过增加一个间接层来解决"。不妨把它沿用到我们面临的递归问题上：没错，我们的确没办法在一个 lambda 函数的定义里面直接（按名字）来调用其自身。但是，可不可以间接调用呢？

我们回顾一下刚才不成功的定义：

lambda n.If_Else n==0 1 n*<self>(n-1)

现在<self>处不是缺少"这个函数自身"嘛，既然不能直接填入"这个函数自身",我们可以增加一个参数，也就是说，把<self>参数化：

lambda self n.If_Else n==0 1 n*self(n-1)

上面这个 lambda 算子总是合法定义了吧。现在，我们调用这个函数的时候，只要加传一个参数 self，这个参数不是别人，正是这个函数自身。还是为了简单起见，我们用 let 语句来给上面这个函数起个别名：

let P=lambda self n.If_Else n==0 1 n*self(n-1)

我们这样调用，比如说我们要计算 3 的阶乘：

P(P,3)

也就是说，把 P 自己作为 P 的第一个参数（注意，调用的时候 P 已经定义完毕了，所以我们当然可以使用它的名字了）。这样一来，P 里面的 self 处不就等于是 P 本身了吗？自身调用自身，递归！

可惜这只是个美好的设想，还差一点点。我们分析一下 P(P,3)这个调用。利用前面讲的 Beta 转换规则，这个函数调用展开其实就是（你可以完全把 P 当成一个宏来进行展开！）：

IF_Else n==0 1 n*P(n-1)

看出问题了吗？这里的 P(n-1)虽然调用到了 P，然而只给出了一个参数；而从 P 的定义来看，它是需要两个参数的（分别为 self 和 n）！也就是说，为了让 P(n-1)变成良好的调用，我们得加一个参数才行，所以我们得稍微修改一下 P 的定义：

let P=lambda self n.If_Else n==0 1 n*self(self,n-1)

请注意，我们在 P 的函数体内调用 self 的时候增加了一个参数。现在当我们调用 P(P,3)的时候，展开就变成了：

IF_Else 3==0 1 3*P(P,3-1)

而 P(P,3-1)是对 P 合法的递归调用。这次我们真的成功了！

不动点原理

然而，看看我们的 P 的定义，是不是很丑陋？“n*

self(self,n-1)”？什么玩意？为什么要多出一个多余的 self？DRY！怎么办呢？我们想起我们一开始定义的那个失败的 P，虽然行不通，但最初的努力往往是大脑最先想到的最直观的做法，我们来回顾一下：

let P=lambda self n.If_Else n==0 1 n*self(n-1)

这个 P 的函数体就非常清晰，没有冗余成分，虽然参数列表里面多出一个 self，但我们其实根本不用管它，看函数体就行了，self 这个名字已经可以说明一切了对不对？但很可惜这个函数不能用。我们再来回想一下为什么不能用呢？因为当你调用 P(P,n)的时候，里面的 self(n-1)会展开为 P(n-1)而 P 是需要两个参数的。唉，要是这里的 self 是一个“真正”的，只需要一个参数的递归阶乘函数，那该多好啊。为什么不呢？干脆我们假设出一个“真正”的递归阶乘函数：

power(n):

if(n==0)return 1;

return n*power(n-1);

但是，前面不是说过了，这个理想的版本无法在 lambda 算子系统中定义出来吗（由于 lambda 函数都是没名字的，无法自己内部调用自己）？不急，我们并不需要它被定义出来，我们只需要在头脑中“假设”它以“某种”方式被定义出来了，现在我们把这个真正完美的 power 传给 P，这样：

P(power,3)

注意它跟 P(P,3)的不同，P(P,3)我们传递的是一个有缺陷的 P 为参数。而 P(power,3)我们则是传递的一个真正的递归函数 power。我们试着展开 P(power,3):

IF_Else 3==0 1 3*power(3-1)

发生了什么？？power(3-1)将会计算出 2 的阶乘（别忘了，power 是我们设想的完美递归函数），所以这个式子将会忠实地计算出 3 的阶乘！

回想一下我们是怎么完成这项任务的：我们设想了

一个以某种方式构造出来的完美的能够内部自己调用自己的递归阶乘函数 power ,我们发现把这个 power 传给 P 的话，P(power,n)的展开式就是真正的递归计算 n 阶乘的代码了。

你可能要说：废话！都有了 power 了我们还要费那事把它传给 P 来个 P(power,n)干嘛？直接 power(n)不就得了？!别急，之所以设想出这个 power 只是为了引入不动点的概念 ,而不动点的概念将会带领我们发现 Y combinator。

什么是不动点？一点都不神秘。让我们考虑刚才的 power 与 P 之间的关系。一个是真正可递归的函数，一个呢，则是以一个额外的 self 参数来试图实现递归的伪递归函数 ,我们已经看到了把 power 交给 P 为参数发生了什么，对吧？不，似乎还没有，我们只是看到了，“把 power 加上一个 n 一起交给 P 为参数”能够实现真正的递归。现在我们想考虑 power 跟 P 之间的关系，直接把 power 交给 P 如何？

P(power)

这是什么？这叫函数的部分求值(partial evaluation)。换句话说，第一个参数是给出来了，但第二个参数还悬在那里，等待给出。那么，光给一个参数得到的是什么呢？是“还剩一个参数待给的一个新的函数”。其实也很简单，只要按照 Beta 转换规则做就是了，把 P 的函数体里面的 self 出现处皆替换为 power 就可以了。我们得到：

IF_Else n==0 1 n*power(n-1)

当然，这个式子里面还有一个变量没有绑定，那就是 n，所以这个式子还不能求值，你需要给它一个 n 才能具体求值，对吧。这么说，这可不就是一个以 n 为参数的函数么？实际上就是的。在 lambda 算子系统里面，如果给一个 lambda 函数的参数不足，则得到的就是一个新的 lambda 函数，这个新的 lambda 函数所接受的参数也就是你尚未给出的那些参数。换句话来说，调用一个 lambda 函数可以分若干步来进行，每次只给出一

部分参数，而只有等所有参数都给齐了，函数的求值结果才能出来，否则你得到的就是一个“中间函数”。

那么，这跟不动点定理有什么关系？关系大了，刚才不是说了，P(power)返回的是一个新的“中间函数”嘛？这个“中间函数”的函数体我们刚才已经看到了，就是简单地展开 P(power)而已，回顾一遍：

IF_Else n==0 1 n*power(n-1)

我们已经知道，这是个函数，参数 n 待定。因此我们不妨给它加上一个“lambda n”的帽子，这样好看一点：

lambda n.IF_Else n==0 1 n*power(n-1)

这是什么呢？这可不就是 power 本身的定义？（当然，如果我们能够定义 power 的话）。不信我们看看 power 如果能够定义出来像什么样子：

let power=lambda n.IF_Else n==0 1 n*power(n-1)

一模一样！也就是说，P(power)展开后跟 power 是一样的。即：

P(power)=power

以上就是所谓的不动点。即对于函数 P 来说 power 是这样一个“点”：当把 P 用到 power 身上的时候，得到的结果仍然还是 power，也就是说，power 这个“点”在 P 的作用下是“不动”的。

可惜的是，这一切居然都是建立在一个不存在的 power 的基础上的，又有什么用呢？可别过早提“不存在”这个词，你觉得一样东西不存在或许只是你没有找到使它存在的正确方法。我们已经看到 power 是跟 P 有着密切联系的。密切到什么程度呢？对于伪递归的 P，存在一个 power，满足 P(power)=power。注意，这里所说的“伪递归”的 P，是指这样的形式：

let P=lambda self n.If_Else n==0 1 n*self(n-1)//注意，不是 self(self,n-1)

一般化的描述就是，对任一伪递归 F（回想一下伪递归的 F 如何得到——是我们为了解决 lambda 函数不能引用自身的问题，于是给理想的 f 加一个 self 参数从而得到的），必存在一个理想 f（F 就是从这个理想 f 演变而来的），满足 F(f)=f。

那么，现在的问题就归结为如何针对 F 找到它的 f 了。根据 F 和 f 之间的密切联系（F 就比 f 多出一个 self 参数而已），我们可以从 F 得出 f 吗？假设我们可以（又是假设），也就是说假设我们找到了一根魔棒，把它朝任意一个伪递归的 F 一挥，眼前一花，它就变成了真正的 f 了。这根魔棒如果存在的话，它具有什么性质？我们假设这个神奇的函数叫做 Y，把 Y 用到任何伪递归的函数 F 上就能够得到真正的 f，也就是说：

Y(F)=f

结合上面的 F(f)=f，我们得到：

Y(F)=f=F(f)=F(Y(F))

也就是说，Y 具有性质：

Y(F)=F(Y(F))

性质倒是找出来了，怎么构造出这个 Y 却又成了难题。一个办法就是使用抽象法，这是从工程学的思想的角度，也就是通过不断迭代、重构，最终找到问题的解。然而对于这里的 Y combinator ,接近问题解的过程却显得复杂而费力，甚至过程中的有些点上的思维跳跃有点如羚羊挂角无迹可寻。然而，在这整个 Y combinator 介绍完了之后我们将会介绍著名的哥德尔不完备性定理，然后我们就会发现，通过哥德尔不完备性定理证明中的一个核心构造式，只需一步自然的推导就能得出我们的 Y combinator。而且，最美妙的是，还可以再往下归约，把一切都归约到康托尔当初提出的对角线方法，到那时我们就会发现原来同样如羚羊挂角般的哥德尔的证明其实是对角线方法的一个自然推论。数学竟是如此奇妙，我们由简单得无法再简单的 lambda calculus 系统的两条公理居然能够导出如此复杂如此令人目眩神迷

的 Y Combinator，而这些复杂性其实也只是荡漾在定理海洋中的涟漪，拨开复杂性的迷雾我们重又发现它们居然寓于极度的简洁之中。这就是数学之美。

让我们先来看一看 Y combinator 的费力而复杂的工程学构造法 ,我会尽量让这个过程显得自然而流畅[7]：

我们再次回顾一下那个伪递归的求阶乘函数：

let P=lambda self n.If_Else n==0 1 n*self(n-1)

我们的目标是找出 P 的不动点 power，根据不动点的性质，只要把 power 传给 P，即 P(power)，便能够得到真正的递归函数了。

现在，关键的地方到了，由于：

power=P(power)//不动点原理

这就意味着，power 作为一个函数（lambda calculus 里面一切都是函数）,它是自己调用了自己的。那么，我们如何实现这样一个能够自己调用自己的 power 呢？回顾我们当初成功的一次尝试，要实现递归，我们是通

过增加一个间接层来进行的：

let power_gen=lambda self.P(self(self))

还记得 self(self)这个形式吗？我们在成功实现出求阶乘递归函数的时候不就是这么做的？那么对于现在这个 power_gen，怎么递归调用？

power_gen(power_gen)

不明白的话可以回顾一下前面我们调用 P(P,n)的地方。这里 power_gen(power_gen)展开后得到的是什么呢？我们根据刚才 power_gen 的定义展开看一看，原来是：

P(power_gen(power_gen))

看到了吗？也就是说：

power_gen(power_gen)=>P(power_gen(power_gen))

现在，我们把 power_gen(power_gen)当成整体

看，不妨令为 power，就看得更清楚了：

power=>P(power)

这不正是我们要的答案么？

OK，我们总结一下：对于给定的 P，只要构造出一个相应的 power_gen 如下：

let power_gen=lambda self.P(self(self))

我们就会发现，power_gen(power_gen)这个调用展开后正是 P(power_gen(power_gen))。也就是说，我们的 power_gen(power_gen)就是我们苦苦寻找的不动点了！

铸造 Y Combinator

现在我们终于可以铸造我们的 Y Combinator 了，Y Combinator 只要生成一个形如 power_gen 的 lambda 函数然后把它应用到自身，就大功告成：

let Y=lambda F.

let f_gen=lambda self.F(self(self))

return f_gen(f_gen)

稍微解释一下，Y 是一个 lambda 函数，它接受一个伪递归 F，在内部生成一个 f_gen（还记得我们刚才看到的 power_gen 吧），然后把 f_gen 应用到它自身（记得 power_gen(power_gen)吧），得到的这个 f_gen(f_gen)也就是 F 的不动点了（因为 f_gen(f_gen)=F(f_gen(f_gen))），而根据不动点的性质，F 的不动点也就是那个对应于 F 的真正的递归函数！

如果你还觉得不相信，我们稍微展开一下看看，还是拿阶乘函数说事，首先我们定义阶乘函数的伪递归版本：

let Pwr=lambda self n.If_Else n==0 1 n*self(n-1)

让我们把这个 Pwr 交给 Y，看会发生什么（根据刚才 Y 的定义展开吧）：

Y(Pwr)=>

let f_gen=lambda self.Pwr(self(self))

return f_gen(f_gen)

Y(Pwr)的求值结果就是里面返回的那个 f_gen(f_gen)，我们再根据 f_gen 的定义展开 f_gen(f_gen)，得到：

Pwr(f_gen(f_gen))

也就是说：

Y(Pwr)=>f_gen(f_gen)=>Pwr(f_gen(f_gen))

我们来看看得到的这个 Pwr(f_gen(f_gen))到底是不是真有递归的魔力。我们展开它（注意，因为 Pwr 需要两个参数，而我们这里只给出了一个，所以 Pwr(f_gen(f_gen))得到的是一个单参（即 n）的函数）：

Pwr(f_gen(f_gen))=>If_Else n==0 1 n*f_gen(f_gen)(n-1)

而里面的那个 f_gen(f_gen)，根据 f_gen 的定义，

又会展开为 Pwr(f_gen(f_gen))，所以：

Pwr(f_gen(f_gen))=>If_Else n==0 1 n*Pwr(f_gen(f_gen))(n-1)

看到加粗的部分了吗？因为 Pwr(f_gen(f_gen))是一个接受 n 为参数的函数，所以不妨把它令成 f（f 的参数是 n），这样上面的式子就是：

f=>If_Else n==0 1 n*f(n-1)

完美的阶乘函数！

哥德尔的不完备性定理

了解哥德尔不完备性定理的可以跳到下一节，“大道至简——康托尔的天才”

然而，漫长的 Y Combinator 征途仍然并非本文的最终目的，对于 Y combinator 的构造和解释，只是给不了解lambda calculus或Y combinator的读者看的。关键是马上你会看到 Y combinator 可以由哥德尔不完备性定理证明的一个核心构造式一眼瞧出来！

让我们的思绪回到 1931 年，那个数学界风起云涌的年代，一个名不经传的 20 出头的学生，在他的博士论文中证明了一个惊天动地的结论。

在那个年代，希尔伯特的数学天才就像太阳的光芒一般夺目，在关于数学严格化的大纷争中希尔伯特带领的形式主义派系技压群雄，得到许多当时有名望的数学家的支持。希尔伯特希望借助于形式化的手段，抽掉数学证明中的意义，把数学证明抽象成一堆无意义的符号转换，就连我们人类赖以自豪的逻辑推导，也不过只是一堆堆符号转换而已（想起 lambda calculus 系统了吧：））。这样一来，一个我们日常所谓的，带有直观意义和解释的数学系统就变成了一个纯粹由无意义符号表达的、公理加上推导规则所构成的形式系统，而数学证明呢，只不过是在这个系统内玩的一个文字游戏。令人惊讶的是，这样一种做法，真的是可行的！数学的意义，似乎竟然真的可以被抽掉！另一方面，一个形式系统具有非常好的性质，平时人们证明一个定理所动用的推导，变成了纯粹机械的符号变换。希尔伯特希望能够证明，

在任一个无矛盾的形式系统中所能表达的所有陈述都要么能够证明要么能够证伪。这看起来是个非常直观的结论，因为一个结论要么是真要么是假，而它在它所处的领域/系统中当然应该能够证明或证伪了（只要我们能够揭示出该系统中足够多的真理）。

然而，哥德尔的证明无情的击碎了这一企图，哥德尔的证明揭示出，任何足够强到蕴含了皮亚诺算术系统（PA）的一致（即无矛盾）的系统都是不完备的，所谓不完备也就是说在系统内存在一个为真但无法在系统内推导出的命题。这在当时的数学界揭起了轩然大波，其证明不仅具有数学意义，而且蕴含了深刻的哲学意义。从那时起这一不完备性定理就被引申到自然科学乃至人文科学的各个角落...至今还没有任何一个数学定理居然能够产生这么广泛而深远的影响。

哥德尔的证明非常的长，达到了 200 多页纸，但其中很大的成分是用在了一些辅助性的工作上面，比如占据超过 1/3 纸张的是关于一个形式系统如何映射到自然

数，也就是说，如何把一个形式系统中的所有公式都表示为自然数，并可以从一自然数反过来得出相应的公式。这其实就是编码，在我们现在看来是很显然的，因为一个程序就可以被编码成二进制数，反过来也可以解码。但是在当时这是一个全新的思想，也是最关键的辅助性工作之一，另一方面，这正是“程序即数据”的最初想法。

现在我们知道，要证明哥德尔的不完备性定理，只需在假定的形式系统 T 内表达出一个为真但无法在 T 内推导出（证明）的命题。于是哥德尔构造了这样一个命题，用自然语言表达就是：命题 P 说的是“P 不可在系统 T 内证明”（这里的系统 T 当然就是我们的命题 P 所处的形式系统了），也就是说“我不可以被证明”，跟著名的说谎者悖论非常相似，只是把“说谎”改成了“不可以被证明”。我们注意到，一旦这个命题能够在 T 内表达出来，我们就可以得出“P 为真但无法在 T 内推导出来”的结论，从而证明 T 的不完备性。为什么呢？我们假设 T 可以证明出 P，而因为 P 说的就是 P 不可在系统 T 内

证明，于是我们又得到 T 无法证明出 P，矛盾产生，说明我们的假设“T 可以证明 P”是错误的，根据排中律，我们得到 T 不可以证明 P，而由于 P 说的正是“T 不可证明 P”，所以 P 就成了一个正确的命题，同时无法由 T 内证明！

如果你足够敏锐，你会发现上面这番推理本身不就是证明吗？其证明的结果不就是 P 是正确的？然而实际上这番证明是位于 T 系统之外的，它用到了一个关于 T 系统的假设“T 是一致（无矛盾）的”，这个假设并非 T 系统里面的内容，所以我们刚才其实是在 T 系统之外推导出了 P 是正确的，这跟 P 不能在 T 之内推导出来并不矛盾。所以别担心，一切都正常。

那么，剩下来最关键的问题就是如何用形式语言在 T 内表达出这个 P，上面的理论虽然漂亮，但若是 P 根本没法在 T 内表达出来，我们又如何能证明“T 内存在这个为真但无法被证明的 P”呢？那一切还不是白搭？

于是，就有了哥德尔证明里面最核心的构造，哥德

尔构造了这样一个公式：

N(n)is unprovable in T

这个公式由两部分构成，n 是这个公式的自由变量，它是一个自然数，一旦给定，那么这个公式就变成一个明确的命题。而 N 则是从 n 解码出的货真价实的（即我们常见的符号形式的）公式（记得哥德尔的证明第一部分就是把公式编码吗？）。”is unprovable in T”则是一个谓词，这里我们没有用形式语言而是用自然语言表达出来的，但哥德尔证明了它是可以用形式语言表达出来的，大致思路就是：一个形式系统中的符号数目是有限的，它们构成这个形式系统的符号表。于是，我们可以依次枚举出所有长度为 1 的串，长度为 2 的串，长度为 3 的串...此外根据形式系统给出的语法规则，我们可以检查每个串是否是良构的公式（well formed formula，简称 wff，其实也就是说，是否符合语法规则，前面我们在介绍 lambda calculus 的时候看到了，一个形式系统是需要语法规则的，比如逻辑语言形式化之后我们就会看

到 P->Q 是一个 wff，而->PQ 则不是），因而我们就可以枚举出所有的 wff 来。最关键的是，我们观察到形式系统中的证明也不过就是由一个个的 wff 构成的序列（想想推导的过程，不就是一个公式接一个公式嘛）。而 wff 构成的序列本身同样也是由符号表内的符号构成的串。所以我们只需枚举所有的串，对每一个串检查它是否是一个由 wff 构成的序列（证明），如果是，则记录下这个 wff 序列（证明）的最后一个 wff，也就是它的结论。这样我们便枚举出了所有的可由 T 推导出的定理。然后为了表达出"X is unprovable in T"，本质上我们只需说"不存在这样一个自然数 S，它所解码出来的 wff 序列以 X 为终结"！这也就是说，我们表达出了"is unprovable in T"这个谓词。

我们用 UnPr(X)来表达"X is unprovable in T"，于是哥德尔的公式变成了：

UnPr(N(n))

现在，到了最关键的部分，首先我们把这个公式简

记为 G(n)——别忘了 G 内有一个自由变量 n ,所以 G 现在还不是一个命题，而只是一个公式，所以谈不上真假：

G(n):UnPr(N(n))

又由于 G 也是个 wff，所以它也有自己的编码 g，当然 g 是一个自然数，现在我们把 g 作为 G 的参数，也就是说，把 G 里面的自由变量 n 替换为 g，我们于是得到一个真正的命题：

G(g):UnPr(G(g))

用自然语言来说，这个命题 G(g)说的就是“我是不可在 T 内证明的”。看 ,我们在形式系统 T 内表达出了“我是不可在 T 内证明的”这个命题。而我们一开始已经讲过了如何用这个命题来推断出 G(g)为真但无法在 T 内证明，于是这就证明了哥德尔的不完备性定理[8]。

哥德尔的不完备性定理被称为 20 世纪数学最重大的发现 ( 不知道有没有“之一”:) ) 现在我们知道为真但无法在系统内证明的命题不仅仅是这个诡异的“哥德尔

命题”，还有很多真正有意义的明确命题，其中最著名的就是连续统假设，此外哥德巴赫猜想也有可能是个没法在数论系统中证明的真命题。

从哥德尔公式到 Y Combinator

哥德尔的不完备性定理证明了数学是一个未完结的学科，永远有需要我们以人的头脑从系统之外去用我们独有的直觉发现的东西。罗杰·彭罗斯在《The Emperor's New Mind》中用它来证明人工智能的不可实现。当然，这个结论是很受质疑的。但哥德尔的不完备性定理的确还有很多很多的有趣推论，数学的和哲学上的。哥德尔的不完备性定理最深刻的地方就是它揭示了自指（或称自引用，递归调用自身等等）结构的普遍存在性，我们再来看一看哥德尔命题的绝妙构造：

G(n):UnPr(N(n))

我们注意到，这里的 UnPr 其实是一个形式化的谓词，它不一定要说“X 在 T 内可证明”，我们可以把它泛化为一个一般化的谓词，P：

G(n):P(N(n))

也就是说，对于任意一个单参的谓词 P，都存在上面这个哥德尔公式。然后我们算出这个哥德尔公式的自然数编码 g，然后把它扔给 G，就得到：

G(g):P(G(g))

是不是很熟悉这个结构？我们的 Y Combinator 的构造不就是这样一个形式？我们把 G 和 P 都看成一元函数，G(g)可不正是 P 这个函数的不动点么！于是，我们从哥德尔的证明里面直接看到了 Y Combinator！

至于如何从哥德尔的证明联系到停机问题，就留给你去解决吧:)因为更重要的还在后面，我们看到，哥德尔的证明虽然巧妙至极，然而其背后的思维过程仍然飘逸而不可捉摸，至少我当时看到 G(n)的时候，“乃大惊”“不知所从出”，他怎么想到的？难道是某一个瞬间“灵光一现”？一般我是不信这一说的，已经有越来越多的科学研究表明一瞬间的“灵感”往往是潜意识乃至表层意识长期思考的结果。哥德尔天才的证明也不例外，我们马

上就会看到，在这个神秘的构造背后，其实隐藏着某种更深的东西，这就是康托尔在 19 世纪 80 年代研究无穷集合和超限数时引入的对角线方法。这个方法仿佛有种神奇的力量，能够揭示出某种自指的结构来，而同时，这又是一个极度简单的手法，通过它我们能够得到数学里面一些非常奇妙的性质。无论是哥德尔的不完备性定理还是再后来丘齐建立的 lambda calculus，抑或我们非常熟悉的图灵机理论里的停机问题，其实都只是这个手法简单推演的结果！

大道至简——康托尔的天才

“大道至简”这个名词或许更多出现在文学和哲学里面，一般用在一些模模糊糊玄玄乎乎的哲学观点上。然而，用在这里，数学上，这个名词才终于适得其所。大道至简，看上去最复杂的理论其实建立在一个最简单最纯粹的道理之上。

康托尔在无穷集合和超限数方面的工作主要集中在两篇突破性的论文上，这也是我所见过的最纯粹最美妙

的数学论文，现代的数学理论充斥了太多复杂的符号和概念，很多时候让人看不到最本质的东西，当然，不否认这些东西很多也是有用的，然而，要领悟真正的数学美，像集合论和数论这种纯粹的东西，真的非常适合。不过这里就不过多谈论数学的细节了，只说康托尔引入对角线方法的动机和什么是对角线方法。

神奇的一一对应

康托尔在研究无穷集合的时候，富有洞察性地看到了对于无穷集合的大小问题，我们不能再使用直观的“所含元素的个数”来描述，于是他创造性地将一一对应引入进来，两个无穷集合“大小”一样当且仅当它们的元素之间能够构成一一对应。这是一个非常直观的概念，一一对应嘛，当然个数相等了，是不是呢？然而这同时就是它不直观的地方了。对于无穷集合，我们日常的所谓“个数”的概念不管用了，因为无穷集合里面的元素个数本就是无穷多个。不信我们来看一个小小的例子。我们说自然数集合能够跟偶数集合构成一一对应，从而

自然数集合跟偶数集合里面元素“个数”是一样多的。怎么可能？偶数集合是自然数集合的真子集，所有偶数都是自然数，但自然数里面还包含奇数呢，说起来应该是二倍的关系不是？不是！我们只要这样来构造一一对应：

1 2 3 4...

2 4 6 8...

用函数来描述就是 f(n)=2n。检验一下是不是一一对应的？不可思议对吗？还有更不可思议的，自然数集是跟有理数集——对应的！对应函数的构造就留给你解决吧，提示，按如下方式来挨个数所有的有理数：

1/1 1/2 2/1 1/3 2/2 3/1 1/4 2/3 3/2 4/1...

用这种一一对应的手法还可以得到很多惊人的结论，如一条直线上所有的点跟一个平面上所有的点构成一一对应（也就是说复数集合跟实数集合构成一一对应）。以致于连康托尔自己都不敢相信自己的眼睛了，这

也就是为什么他在给戴得金的信中会说“我看到了它，却不敢相信它”的原因。

然而，除了一一对应之外，还有没有不能构成一一对应的两个无穷集合呢？有。实数集合就比自然数集合要“大”，它们之间实际上无法构成一一对应。这就是康托尔的对角线方法要解决的问题。

实数集和自然数集无法构成一一对应？！

我们只需将实数的小数位展开，并且我们假设实数集能够与自然数集一一对应，也就是说假设实数集可列，所以我们把它们与自然数一一对应列出，如下：

1 a10.a11a12a13...

2 a20.a21a22a23...

3 a30.a31a32a33...

4...

5...

（注：aij 里面的 ij 是下标）

现在，我们构造一个新的实数，它的第 i 位小数不等于 aii。也就是说，它跟上面列出的每一个实数都至少有一个对应的小数位不等，也就是说它不等于我们上面列出的所有实数，这跟我们上面假设已经列出了所有实数的说法相矛盾。所以实数集只能是不可列的，即不可与自然数集一一对应！这是对角线方法的最简单应用。

对角线方法——停机问题的深刻含义

对角线方法有很多非常奇妙的结论。其中之一就是文章一开始提到的停机问题。我想绝大多数人刚接触停机问题的时候都有一个问题，图灵怎么能够想到这么诡异的证明，怎么能构造出那个诡异的“说停机又不停机，说不停机又停机”的悖论机器。马上我们就会看到，这其实只是对角线方法的一个直接结论。

还是从反证开始，我们假设存在这样一个图灵机，他能够判断任何程序在任何输入上是否停机。由于所有图灵机构成的集合是一个可列集（也就是说，我们可以

逐一列出所有的图灵机，严格证明见我以前的一篇文章《图灵机杂思》)，所以我们可以很自然地列出下表，它表示每个图灵机分别在每一个可能的输入（1,2,3,...）下的输出，N 表示无法停机，其余数值则表示停机后的输出：

1 2 3 4...

M1 N 1 N N...

M2 2 0 N 0...

M3 0 1 2 0...

M4 N 0 5 N...

...

M1，M2，M3...是逐一列出的图灵机，并且，注意，由于程序即数据，每个图灵机都有唯一编码，所以我们规定在枚举图灵机的时候 Mi 其实就代表编码为 i 的图灵机，当然这里很多图灵机将会是根本没用的玩意，但这不要紧。此外，最上面的一行 1 2 3 4...是输入数据，如，

矩阵的第一行代表 M1 分别在 1，2，3，...上面的输出，不停机的话就是 N。

我们刚才假设存在这样一个图灵机 H，它能够判断任何程序在任何输入上能否停机，换句话说，H(i,j)（i 是 Mi 的编码）能够给出“Mi(j)”是 N（不停）呢还是给出一个具体的结果（停）。

我们现在来运用康托尔的对角线方法，我们构造一个新的图灵机 P，P 在 1 上的输出行为跟 M1(1)“不一样”，在 2 上的输出行为跟 M2(2)“不一样”，...总之 P 在输入 i 上的输出跟 Mi(i)不一样。只需利用一下我们万能的 H，这个图灵机 P 就不难构造出来，如下：

P(i):

if(H(i,i)==1)then//Mi(i)halts

return 1+Mi(i)

else//if H(i,i)==0(Mi(i)doesn’t halt)

return 0

也就是说，如果 Mi(i)停机，那么 P(i)的输出就是 Mi(i)+1，如果 Mi(i)不停机的话，P(i)就停机且输出 0。这就保证了 P(i)的输出行为跟 Mi(i)反正不一样。现在，我们注意到 P 本身是一个图灵机，而我们上面已经列出了所有的图灵机，所以必然存在一个 k，使得 Mk=P。而两个图灵机相等当且仅当它们对于所有的输入都相等，也就是说对于任取的 n，有 Mk(n)=P(n)，现在令 n=k，得到 Mk(k)=P(k)，根据上面给出的 P 的定义，这实际上就是：

Mk(k)=P(k)=

1+Mk(k)if Mk(k)halts

0 if Mk(k)doesn’t halt

看到这个式子里蕴含的矛盾了吗？如果 Mk(k)停机，那么 Mk(k)=1+Mk(k)；如果 Mk(k)不停机，则 Mk(k)=0（给出结果 0 即意味着 Mk(k)停机）；不管哪种情况都是矛盾。于是我们得出，不存在那样的 H。

这个对角线方法实际上说明了，无论多聪明的 H，总存在一个图灵机的停机行为是它无法判断的。这跟哥德尔定理“无论多‘完备’的形式化公理系统，都存在一个‘哥德尔命题’是无法在系统内推导出来的”从本质上其实是一模一样的。只不过我们一般把图灵的停机问题称为“可判定问题”,而把数学的称为“可证明问题”。

等等！如果我们把那个无法判定是否停机的图灵机作为算法的特例纳入到我们的 H 当中呢？我们把得到的新的判定算法记为 H1。然而，可惜的是，在 H1 下，我们又可以相应地以同样的手法从 H1 构造出一个无法被它（H1）判定的图灵机来。你再加，我再构造，无论你加多少个特例进去，我都可以由同样的方式构造出来一个你无法够到的图灵机，以彼之矛，攻彼之盾。其实这也是哥德尔定理最深刻的结论之一，哥德尔定理其实就说明了无论你给出多少个公理，即无论你建立多么完备的公理体系，这个系统里面都有由你的那些公理出发所推导不到的地方，这些黑暗的角落，就是人类直觉之光才能照射到的地方！

本节我们从对角线方法证明了图灵的停机问题，我们看到，对角线方法能够揭示出某种自指结构，从而构造出一个“悖论图灵机”。实际上，对角线方法是一种有深远影响的方法，哥德尔的证明其实也是这个方法的一则应用。证明与上面的停机问题证明如出一辙，只不过把 Mi 换成了一个形式系统内的公式 fi ,具体的证明就留给聪明的你吧:)我们现在来简单的看一下这个奇妙方法的几个不那么明显的推论。

罗素悖论

学过逻辑的人大约肯定是知道著名的罗素悖论的，罗素悖论用数学的形式来描述就是：

R={X:X 不属于 X};

这个悖论最初是从康托尔的无穷集合论里面引申出来的。当初康托尔在思考无穷集合的时候发现可以称“一切集合的集合”，这样一个集合由于它本身也是一个集合，所以它就属于它自身。也就是说，我们现在可以称世界上存在一类属于自己的集合，除此之外当然就是不

属于自己的集合了。而我们把所有不属于自己的集合收集起来做成一个集合 R，这就是上面这个著名的罗素悖论了。

我们来看 R 是否属于 R，如果 R 属于 R，根据 R 的定义，R 就不应该属于 R。而如果 R 不属于 R，则再次根据 R 的定义，R 就应该属于 R。

这个悖论促使了集合论的公理化。后来策梅罗公理化的集合论里面就不允许 X 属于 X（不过可惜的是，尽管如此还是没法证明这样的集合论不可能产生出新的悖论。而且永远没法证明——这就是哥德尔第二不完备性定理的结论——一个包含了 PA 的形式化公理系统永远无法在内部证明其自身的一致（无矛盾）性。从而希尔伯特想从元数学推出所有数学系统的一致性的企图也就失败了，因为元数学的一致性又得由元元数学来证明，后者的一致性又得由元元元数学来证明…）。

这里我们只关心罗素是如何想出这个绝妙的悖论的。还是对角线方法！我们罗列出所有的集合，S1,S2,S

3…

S1 S2 S3…

S1 0 1 1…

S2 1 1 0…

S3 0 0 0…

……

右侧纵向列出所有集合，顶行横向列出所有集合。0/1 矩阵的(i,j)处的元素表示 Si 是否包含 Sj，记为 Si(j)。现在我们只需构造一个新的 0/1 序列 L ,它的第 i 位与矩阵的(i,i)处的值恰恰相反：L(i)=1-Si(i)。我们看到，这个新的序列其实对应了一个集合，不妨也记为 L，L(i)表示 L 是否包含 Si。根据 L 的定义，如果矩阵的(i,i)处值为 0（也就是说，如果 Si 不包含 Si），那么 L 这个集合就包含 Si,否则就不包含。我们注意到这个新的集合 L 肯定等于某个 Sk（因为我们已经列出了所有的集合），L=Sk。既然L与Sk是同一集合 那么它们肯定包含同样的元素，

从而对于任意 n，有 L(n)=Sk(n)。于是通过令 n=k，得到 L(k)=Sk(k)，而根据 L 的定义，L(k)=1-Sk(k)。这就有 Sk(k)=1-Sk(k)，矛盾。

通过抽象简化以上过程，我们看到，我们构造的 L 其实是“包含了所有不包含它自身的集合的集合”，用数学的描述正是罗素悖论！

敏锐的你可能会注意到所有集合的数目是不可数的从而根本不能 S1,S2...的一一列举出来。没错，但通过假设它们可以列举出来，我们发现了一个与可列性无关的悖论。所以这里的对角线方法其实可以说是一种启发式方法。

同样的手法也可以用到证明 P(A)（A 的所有子集构成的集合，也叫幂集）无法跟 A 构成一一对应上面。证明就留给聪明的你了:)

希尔伯特第十问题结出的硕果

希尔伯特是在 1900 年巴黎数学家大会上提出著名

的希尔伯特第十问题的，简言之就是是否存在一个算法，能够计算任意丢番图方程是否有整根。要解决这个问题，就得先严格定义“算法”这一概念。为此图灵和丘齐分别提出了图灵机和 lambda calculus 这两个概念，它们从不同的角度抽象出了“有效（机械）计算”的概念，著名的图灵——丘齐命题就是说所有可以有效计算出来的问题都可以由图灵机计算出来。实际上我们已经看到，丘齐的 lambda calculus 其实就是数学推理系统的一个形式化。而图灵机则是把这个数学概念物理化了。而也正因为图灵机的概念隐含了实际的物理实现，所以冯·诺依曼才据此提出了奠定现代计算机体系结构的冯·诺依曼体系结构，其遵循的，正是图灵机的概念。而“程序即数据”的理念，这个发端于数学家哥德尔的不完备性定理的证明之中的理念，则早就在黑暗中预示了可编程机器的必然问世。

对角线方法——回顾

我们看到了对角线方法是如何简洁而深刻地揭示出

自指或递归结构的。我们看到了著名的不完备性定理、停机问题、Y Combinator、罗素悖论等等等等如何通过这一简洁优美的方法推导出来。这一诞生于康托尔的天才的手法如同一条金色的丝线，把位于不同年代的伟大发现串联了起来，并且将一直延续下去…

P.S

1.lambda calculus 里面的“停机问题”

实际上 lambda calculus 里面也是有“停机问题”的等价版本的。其描述就是：不存在一个算法能够判定任意两个 lambda 函数是否等价。所谓等价当然是对于所有的 n,有 f(n)=g(n)了。这个问题的证明更加能够体现对角线方法的运用。仍然留给你吧。

2.负喧琐话(http://blog.csdn.net/g9yuayon)是个非常不错的 blog:)。g9 的文字轻松幽默，而且有很多名人八卦可以养眼，真的灰常…灰常…不错哦。此外 g9 老兄还是个理论功底非常扎实的牛。所以，anyway，看了他的 blog 就知道啦 !最初这篇文章的动机也正是看了上

面的一篇关于 Y Combinator 的铸造过程的介绍，于是想揭示一些更深的东西，于是便有了本文。

3.文章起名《康托尔、哥德尔、图灵——永恒的金色对角线》其实是为了纪念看过的一本好书 GEB，即《Godel、Escher、Bach-An Eternal Golden Braid》中文译名《哥德尔、埃舍尔、巴赫——集异璧之大成》——商务印书馆出版。对于一本定价 50 元居然能够在 douban 上卖到 100 元的二手旧书，我想无需多说。另，幸福的是，电子版可以找到:)

4.其实很久前想写的是一篇《从哥德尔到图灵》，但那篇写到 1/3 不到就搁下了，一是由于事务，二是总觉得少点什么。呵呵，如今把康托尔扯进来，也算是完成当时扔掉的那一篇吧。

5.这恐怕算是写得最曲折的一篇文章了。不仅自己被这些问题搞得有点晕头转向（还好总算走出来），更因为要把这些东西自然而然的串起来，也颇费周章。很多时候是利用吃饭睡觉前或走路的时间思考本质的问题以

及如何表达等等，然后到纸上一气呵成。不过同时也锻炼了不拿纸笔思考数学的能力，呵呵。

6.关于图灵的停机问题、Y Combinator、哥德尔的不完备性定理以及其它种种与康托尔的对角线之间的本质联系，几乎查不到完整系统的深入介绍，一些书甚至如《The Emperor’s New Mind》也只是介绍了与图灵停机问题之间的联系（已经非常的难得了），google 和 baidu 的结果也是基本没有头绪。很多地方都是一带而过让人干着急。所以看到很多地方介绍这些定理和构造的时候都是弄得人晕头转向的，绝大部分人在面对如 Y Combinator、不完备性定理、停机问题的时候都把注意力放在力图理解它是怎么运作的上面了，却使人看不到其本质上从何而来，于是人们便对这些东东大为惊叹。这使我感到很不痛快，如隔靴搔痒般。这也是写这篇文章的主要动机之一。

Reference

[1]《数学——确定性的丧失》

[2]也有观点认为函数式编程语言之所以没有广泛流行起来是因为一些实际的商业因素。

[3]Douglas R.Hofstadter 的著作《Godel,Escher, Bach:An Eternal Golden Braid》(《哥德尔、艾舍尔、巴赫——集异璧之大成》)就是围绕这一思想写出的一本奇书。非常建议一读。

[4]《数学——确定性的丧失》

[5]虽然我觉得那个系徽做得太复杂，要表达这一简洁优美的思想其实还能有更好的方式。

[6]关于如何在 lambda calculus 系统里实现“+”操作符以及自然数等等，可参见这里，这里，和这里。

[7]g9 的 blog（负暄琐话）http://blog.csdn.net/g9yuayon/上有一系列介绍 lambda calculus 的文章（当然，还有其它好文章:)），非常不错，强烈推荐。最近的两篇就是介绍 Y combinator 的。其中有一篇以 javaScript 语言描述了迭代式逐步抽象出 Y Combinator

的过程。

[8]实际上这只是第一不完备性定理，它还有一个推论，被称为第二不完备性定理，说的是任一个系统 T 内无法证明这个系统本身的一致性。这个定理的证明核心思想如下：我们前面证明第一不完备性定理的时候用的推断其实就表明 Con/T->G(g)（自然语言描述就是，由系统 T 的无矛盾，可以推出 G(g)成立），而这个“Con/T->G(g)”本身又是可以在 T 内表达且证明出来的（具体怎么表达就不再多说了）——只需要用排中律即可。于是我们立即得到，T 里面无法推出 Con/T，因为一旦推出 Con/T 就立即推出 G(g)从而推出 UnPr(G(g))，这就矛盾了。所以，Con/T 无法在 T 内推出（证明）。

## 数学之美番外篇：快排为什么那样快

目录

2.称球

3.排序

3.1 为什么堆排比快排慢

3.2 为什么快排其实也不是那么快

3.3 基排又为什么那么快呢

4.信息论！信息论？

5.小结

0.前言

知道这个理论是在 TopLanguage 上的一次讨论，先是 g9 转了 David MacKay 的一篇文章，然后引发了牛人们的一场关于信息论的讨论。Anyway，正如 g9 很久以前在 Blog 里面所说的：

有时无知是福。俺看到一点新鲜的科普也能觉得造化神奇。刚才读 Gerald Jay Sussman（SICP 作者）的文章，Building Robust Systems–an essay，竟然心如

小鹿乱撞，手心湿润，仿佛第一次握住初恋情人温柔的手。

而看到 MacKay 的这篇文章我也有这种感觉——以前模糊的东西忽然有了深刻的解释，一切顿时变得明白无比。原来看问题的角度或层面能够带来这么大的变化。再一次印证了越是深刻的原理往往越是简单和强大。所以说，土鳖也有土鳖的幸福:P

这篇文章相当于 MacKay 原文的白话文版。MacKay 在原文中用到了信息论的知识，后者在我看来并不是必须的，尽管计算的时候方便，但与本质无关。所以我用大白话解释了一通。

1.猜数字

我们先来玩一个猜数字游戏：我心里默念一个 1~64 之间的数，你来猜（你只能问答案是“是”或“否”的问题）。为了保证不论在什么情况下都能以尽量少的次数猜中，你应该采取什么策略呢？很显然，二分。先是猜是不是位于 1~32 之间，排除掉一半可能性，然后对

区间继续二分。这种策略能够保证无论数字怎么跟你捉迷藏，都能在 log_2{n}次以内猜中。用算法的术语来说就是它的下界是最好的。

我们再来回顾一下这个游戏所蕴含的本质：为什么这种策略具有最优下界？答案也很简单，这个策略是平衡的。反之如果策略不是平衡的，比如问是不是在 1~10 之间，那么一旦发现不是在 1~10 之间的话就会剩下比 N/2 更多的可能性需要去考察了。

徐宥在讨论中提到，这种策略的本质可以概括成“让未知世界无机可乘”。它是没有“弱点的”，答案的任何一个分支都是等概率的。反之，一旦某个分支蕴含的可能性更多，当情况落到那个分支上的时候你就郁闷了。比如猜数字游戏最糟糕的策略就是一个一个的猜：是 1 吗？是 2 吗？...因为这种猜法最差的情况下需要 64 次才能猜对，下界非常糟糕。二分搜索为什么好，就是因为它每次都将可能性排除一半并且无论如何都能排除一半（它是最糟情况下表现最好的）。

2.称球

12 个小球，其中有一个是坏球。有一架天平。需要你用最少的称次数来确定哪个小球是坏的并且它到底是轻还是重。

这个问题是一道流传已久的智力题。网络上也有很多讲解，还有泛化到 N 个球的情况下的严格证明。也有零星的一些地方提到从信息论的角度来看待最优解法。本来我一直认为这道题目除了试错之外没有其它高妙的思路了，只能一个个方法试，并尽量从结果中寻找信息，然后看看哪种方案最少。

然而，实际上它的确有其它的思路，一个更本质的思路，而且根本用不着信息论这么拗口的知识。

我们先回顾一下猜数字游戏。为了保证任何情况下以最少次数猜中，我们的策略是每次都排除恰好一半的可能性。类比到称球问题上：坏球可能是 12 个球中的任意一个，这就是 12 种可能性；而其中每种可能性下坏球可能轻也可能重。于是“坏球是哪个球，是轻是重”这

个问题的答案就有 12×2=24 种可能性。现在我们用天平来称球，就等同于对这 24 种可能性发问，由于天平的输出结果有三种“平衡、左倾、右倾”，这就相当于我们的问题有三个答案，即可以将所有的可能性切成三份，根据猜数字游戏的启发，我们应当尽量让这三个分支概率均等，即平均切分所有的可能性为三等份。如此一来的话一次称量就可以将答案的可能性缩减为原来的 1/3，三次就能缩减为 1/27。而总共才有 24 种可能性，所以理论上是完全可以 3 次称出来的。

如何称的指导原则有了，构造一个称的策略就不是什么太困难的事情了。首先不妨解释一下为什么最直观的称法不是最优的——6、6 称：在 6、6 称的时候，天平平衡的可能性是 0。刚才说了，最优策略应该使得天平三种状态的概率均等，这样才能三等分答案的所有可能性。

为了更清楚的看待这个问题，我们不妨假设有 6 个球，来考虑一下 3、3 称和 2、2 称的区别：

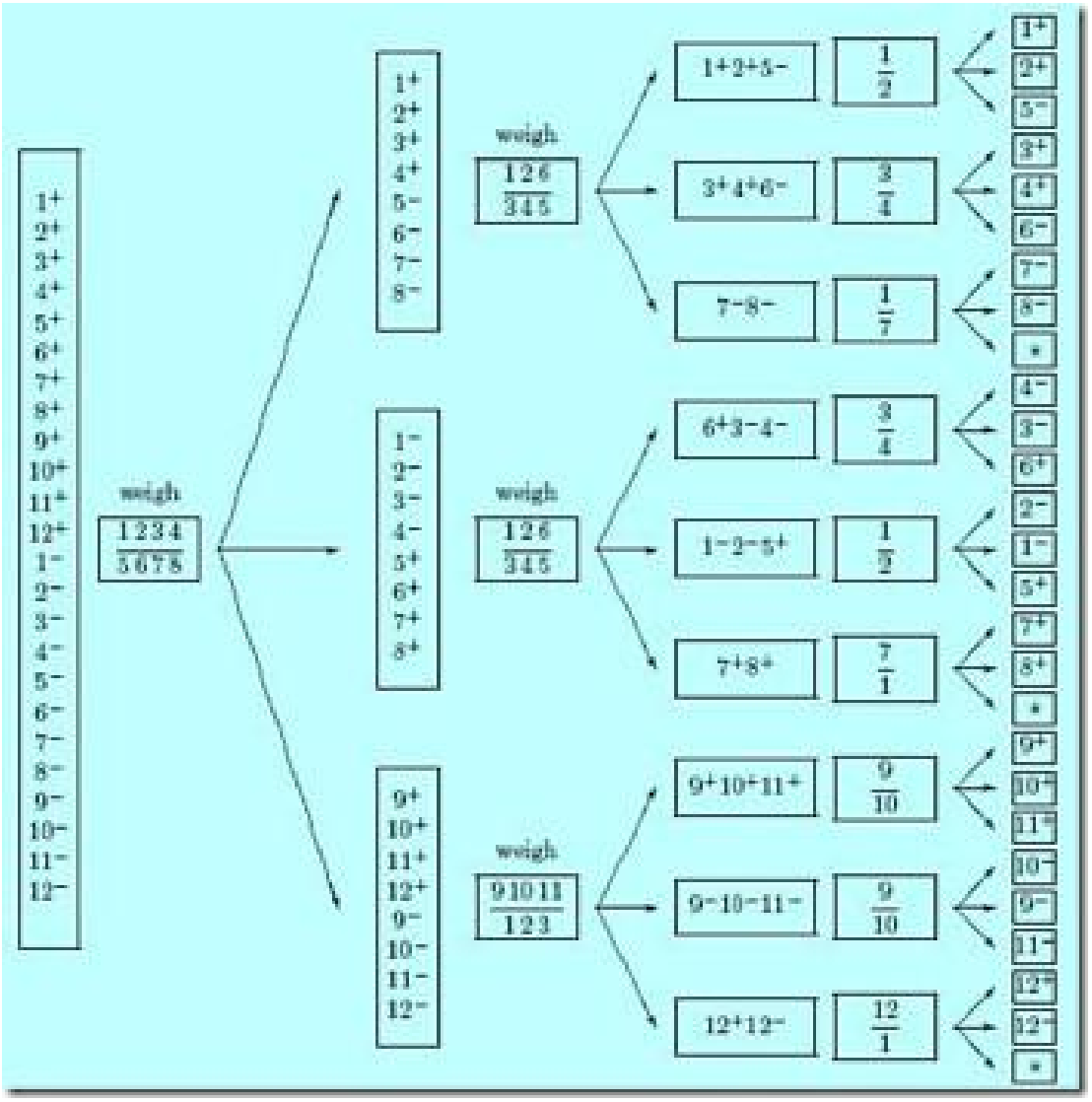
在未称之前，一共有 12 种可能性：1 轻、1 重、2 轻、2 重、…、6 轻、6 重。现在将 1、2、3 号放在左边，4、5、6 放在右边 3、3 称了之后，不失一般性假设天平左倾，那么小球的可能性就变成了原来的一半（6 种）：1 重、2 重、3 重、4 轻、5 轻、6 轻。即这种称法能排除一半可能性。

现在再来看 2、2 称法，即 1、2 放左边，3、4 放右边，剩下的 5、6 不称，放一边。假设结果是天平平衡，那么可能性剩下——4 种：5 重、5 轻、6 重、6 轻。假设天平左倾，可能性也剩下 4 种：1 重、2 重、3 轻、4 轻。右倾和左倾的情况类似。总之，这种称法，不管天平结果如何，情况都被我们缩小到了原来的三分之一！我们充分利用了“天平的结果状态可能有三种”这个条件来三等分所有可能性，而不是二等分。

说到这里，剩下的事情就实在很简单了：第二步称法，只要记着这样一个指导思想——你选择的称法必须使得当天平平衡的时候答案剩下的可能性和天平左倾

（右倾）的时候答案剩下的可能性一样多。实际上，这等同于你得选择一种称法，使得天平输出三种结果的概率是均等的，因为天平输出某个结果的概率就等同于所有支持这个结果（左倾、右倾、平衡）的答案可能性的和，并且答案的每个可能性都是等概率的。

MacKay 在他的书《Information Theory:Inference and Learning Algorithms》（作者开放免费电子书）里面 4.1 节专门讲了这个称球问题，还画了一张不错的图，我就照抄了：




图中“1+”是指“1 号小球为重”这一可能性。一开始一共有 24 种可能性。4、4 称了之后不管哪种情况（分支），剩下来的可能性总是 4 种。这是一个完美的三分。然后对每个分支构造第二次称法，这里你只要稍加演算就可以发现，分支 1 上的第二次称法，即“1、2、6 对 3、4、5”这种称法，天平输出三种结果的可能性是均等的（严格来说是几乎均等）。这就是为什么这个称

法能够在最坏的情况下也能表现最好的原因，没有哪个分支是它的弱点，它必然能将情况缩小到原来的 1/3。

3.排序

用前面的看问题视角，排序的本质可以这样来表述：一组未排序的 N 个数字，它们一共有 N!种重排，其中只有一种排列是满足题意的（譬如从大到小排列）。换句话说，排序问题的可能性一共有 N!种。任何基于比较的排序的基本操作单元都是“比较 a 和 b”，这就相当于猜数字游戏里面的一个问句，显然这个问句的答案只能是“是”或“否”，一个只有两种输出的问题最多只能将可能性空间切成两半，根据上面的思路，最佳切法就是切成 1/2 和 1/2。也就是说，我们希望在比较了 a 和 b 的大小关系之后，如果发现 a<b 的话剩下的排列可能性就变成 N!/2，如果发现 a>b 也是剩下 N!/2 种可能性。由于假设每种排列的概率是均等的，所以这也就意味着支持 a<b 的排列一共有 N!/2 个，支持 a>b 的也是 N!/2 个，换言之，a<b 的概率等于 a>b 的概率。

我们希望每次在比较 a 和 b 的时候，a<b 和 a>b 的概率是均等的，这样我们就能保证无论如何都能将可能性缩小为原来的一半了！最优下界。

一个直接的推论是，如果每次都像上面这样的完美比较，那么 N 个元素的 N!种可能排列只需要 log_2{N!}就排查玩了，而 log_2{N!}近似于 NlogN。这正是快排的复杂度。

3.1 为什么堆排比快排慢

回顾一下堆排的过程：

1.建立最大堆（堆顶的元素大于其两个儿子，两个儿子又分别大于它们各自下属的两个儿子...以此类推）

2.将堆顶的元素和最后一个元素对调（相当于将堆顶元素（最大值）拿走，然后将堆底的那个元素补上它的空缺），然后让那最后一个元素从顶上往下滑到恰当的位置（重新使堆最大化）。

3.重复第 2 步。

这里的关键问题就在于第 2 步，堆底的元素肯定很小，将它拿到堆顶和原本属于最大元素的两个子节点比较，它比它们大的可能性是微乎其微的。实际上它肯定小于其中的一个儿子。而大于另一个儿子的可能性非常小。于是，这一次比较的结果就是概率不均等的，根据前面的分析，概率不均等的比较是不明智的，因为它并不能保证在糟糕情况下也能将问题的可能性削减到原本的 1/2。可以想像一种极端情况，如果 a 肯定小于 b，那么比较 a 和 b 就会什么信息也得不到——原本剩下多少可能性还是剩下多少可能性。

在堆排里面有大量这种近乎无效的比较，因为被拿到堆顶的那个元素几乎肯定是很小的，而靠近堆顶的元素又几乎肯定是很大的，将一个很小的数和一个很大的数比较，结果几乎肯定是“小于”的，这就意味着问题的可能性只被排除掉了很小一部分。

这就是为什么堆排比较慢（堆排虽然和快排一样复杂度都是 O(NlogN)但堆排复杂度的常系数更大）。

MacKay 也提供了一个修改版的堆排：每次不是将堆底的元素拿到上面去，而是直接比较堆顶（最大）元素的两个儿子，即选出次大的元素。由于这两个儿子之间的大小关系是很不确定的，两者都很大，说不好哪个更大哪个更小，所以这次比较的两个结果就是概率均等的了。具体参考这里。

3.2 为什么快排其实也不是那么快

我们考虑快排的过程：随机选择一个元素做“轴元素”，将所有大于轴元素的移到左边，其余移到右边。根据这个过程，快排的第一次比较就是将一个元素和轴元素比较，这个时候显而易见的是，“大于”和“小于”的可能性各占一半。这是一次漂亮的比较。

然而，快排的第二次比较就不那么高明了：我们不妨令轴元素为 pivot，第一次比较结果是 a1<pivot，那么可以证明第二次比较 a2 也小于 pivot 的可能性是 2/3！这容易证明：如果 a2>pivot 的话，那么 a1，a2，pivot 这三个元素之间的关系就完全确定了——a1<pivo

t<a2，剩下来的元素排列的可能性我们不妨记为 P（不需要具体算出来）。而如果 a2<pivot 呢？那么 a1 和 a2 的关系就仍然是不确定的，也就是说，这个分支里面含有两种情况：a1<a2<pivot，以及 a2<a1<pivot。对于其中任一种情况，剩下的元素排列的可能性都是 P，于是这个分支里面剩下的排列可能性就是 2P。所以当 a2<pivot 的时候，还剩下 2/3 的可能性需要排查。

再进一步，如果第二步比较果真发现 a2<pivot 的话，第三步比较就更不妙了，模仿上面的推理，a3<pivot 的概率将会是 3/4！

这就是快排也不那么快的原因，因为它也没有做到每次比较都能将剩下的可能性砍掉一半。

3.3 鸡排为什么又那么快呢？

传统的解释是：基排不是基于比较的，所以不具有后者的局限性。话是没错，但其实还可以将它和基于比较的排序做一个类比。

基排的过程也许是源于我们理顺一副牌的过程：如果你有 N（N<=13）张牌，乱序，如何理顺呢？我们假象桌上有十三个位置，然后我们将手里的牌一张一张放出去，如果是 3，就放在位置 3 上，如果是 J，就放在位置 11 上，放完了之后从位置 1 到位置 13 收集所有的牌（没有牌的位置上不收集任何牌）。

我们可以这样来理解基排高效的本质原因：假设前 i 张牌都已经放到了它们对应的位置上，第 i+1 张牌放出去的时候，实际上就相当于“一下子”就确立了它和前 i 张牌的大小关系，用 O(1)的操作就将这张牌正确地插入到了前 i 张牌中的正确位置上，这个效果就相当于插入排序的第 i 轮原本需要比较 O(i)次的，现在只需要 O(1)了。

但是，为什么基排能够达到这个效果呢？上面只是解释了过程，解释了过程不代表解释了本质。

当 i 张牌放到位之后，放置第 i+1 张牌的时候有多少种可能性？大约 i+1 种，因为前 i 张牌将 13 个位置分割成了 i+1 个区间——第 i+1 张牌可以落在任意一个区

间。所以放置第 i+1 张牌就好比是询问这样一个问题：“这张牌落在哪个区间呢？”而这个问题的答案有 i+1 种可能性？所以它就将剩下来的可能性均分成了 i+1 份（换句话说，砍掉了 i/i+1 的可能性！）。再看看基于比较的排序吧：由于每次比较只有两种结果，所以最多只能将剩下的可能性砍掉一半。

这就是为什么基排要快得多。而所有基于比较的排序都逃脱不了 NlogN 的宿命。

4.信息论！信息论？

本来呢，MacKay 写那篇文章是想用信息论来解释为什么堆排慢，以及为什么快排也慢的。MacKay 在他的文章中的解释是，只有提出每种答案的概率都均等的问题，才能获得最大信息量。然而，仔细一想，其实这里信息论并不是因，而是果。这里不需要用信息论就完全能够解释，而且更明白。信息论只是对这个解释的一个形式化。当然，信息论在其它地方还是有应用的。但这里其实用不着信息论这么重量级的东西（也许具体计

算一些数据的时候是需要的），而是只需要一种看问题的本质视角：将排序问题看成和猜数字一样，是通过问问题来缩小/排除（narrow down）结果的可能性区间，这样一来，就会发现，“最好的问题”就是那些能够均分所有可能性的问题，因为那样的话不管问题的答案如何，都能排除掉 k-1/k（k 为问题的答案有多少种输出——猜数字里面是 2，称球里面是 3）种可能性，而不均衡的问题总会有一个或一些答案分支排除掉的可能性要小于 k-1/k。于是策略的下界就被拖累了。

5.小结

这的确是“小结”，因为两点：

1.这个问题可以有信息论的理论解释，而信息论则是一个相当大的领域了。

2.文中提到的这种看问题的视角除了用于排序、称球，还能够运用到哪些问题上（比如搜索）。

Update(06/13/2008):徐宥在讨论中继续提到：

另外，这几天我重新把 TAOCP 第三卷(第二版)翻出来看了看 Knuth 怎么说这个问题的,发现真是牛大了：

先说性能：

pp148,section 5.2.3 说：

When N=1000,the approximate average runiing time on MIX are

160000u for heapsort

130000u for shellsort

80000u for quicksort

这里,Knuth 同学发现一般情况下 heapsort 表现很不好.于是，在下文他就说，习题 18(pp156,难度 21)

(R.W.Floyd)During the selection phase of heapsort,the key K tends to

be quite small,so that nearly all the comparisons in step H6 find

K<K_j.Show how to modify the algorithm so that K is not compared with

K_j in the main loop of the computation,thereby nearly cutting the

average number of comparisons in half.

答案里面的方法和 DMK 的方法是一样的。(我觉得 DMK 是看了这个论文或者 TAoCP 的)这里说 by half，就正好和快排差不多了。

再说信息论分析：

在 5.3.1(pp181)高爷爷就说，“排序问题可以看成是一个树上的鸟儿排排站的问题.(还特地画了一棵树),下一段就说,其实这个也

有等价说法,就是信息论,我们从称球问题说起…”

然后后面一直讲信息论和最小比较排序…

高爷爷真不愧是姓高的，囧 rz..

# 数学之美番外篇：平凡而又神奇的贝叶斯方法

概率论只不过是把常识用数学公式表达了出来。

——拉普拉斯

记得读本科的时候，最喜欢到城里的计算机书店里面去闲逛，一逛就是好几个小时；有一次，在书店看到一本书，名叫贝叶斯方法。当时数学系的课程还没有学到概率统计。我心想，一个方法能够专门写出一本书来，肯定很牛逼。后来，我发现当初的那个朴素归纳推理成立了——这果然是个牛逼的方法。

——题记

目录

2.拼写纠正

3.模型比较与贝叶斯奥卡姆剃刀

3.1 再访拼写纠正

3.2 模型比较理论（Model Comparasion）与贝叶斯奥卡姆剃刀（Bayesian Occam’s Razor）

3.3 最小描述长度原则

3.4 最优贝叶斯推理

4.无处不在的贝叶斯

4.1 中文分词

4.2 统计机器翻译

4.3 贝叶斯图像识别，Analysis by Synthesis

4.4 EM 算法与基于模型的聚类

4.5 最大似然与最小二乘

5.朴素贝叶斯方法（又名“愚蠢者的贝叶斯（idiot’

s bayes )" )

5.1 垃圾邮件过滤器

5.2 为什么朴素贝叶斯方法令人诧异地好——一个理论解释

6.层级贝叶斯模型

6.1 隐马可夫模型（HMM）

7.贝叶斯网络

0.前言

这是一篇关于贝叶斯方法的科普文，我会尽量少用公式，多用平白的语言叙述，多举实际例子。更严格的公式和计算我会在相应的地方注明参考资料。贝叶斯方法被证明是非常 general 且强大的推理框架，文中你会看到很多有趣的应用。

1.历史

托马斯·贝叶斯（Thomas Bayes）同学的详细生平

在这里。以下摘一段 wikipedia 上的简介：

所谓的贝叶斯方法源于他生前为解决一个“逆概”问题写的一篇文章，而这篇文章是在他死后才由他的一位朋友发表出来的。在贝叶斯写这篇文章之前，人们已经能够计算“正向概率”，如“假设袋子里面有 N 个白球，M 个黑球，你伸手进去摸一把，摸出黑球的概率是多大”。而一个自然而然的问题是反过来：“如果我们事先并不知道袋子里面黑白球的比例，而是闭着眼睛摸出一个（或好几个）球，观察这些取出来的球的颜色之后，那么我们可以就此对袋子里面的黑白球的比例作出什么样的推测”。这个问题，就是所谓的逆概问题。

实际上，贝叶斯当时的论文只是对这个问题的一个直接的求解尝试，并不清楚他当时是不是已经意识到这里面包含着的深刻的思想。然而后来，贝叶斯方法席卷了概率论，并将应用延伸到各个问题领域，所有需要作出概率预测的地方都可以见到贝叶斯方法的影子，特别地，贝叶斯是机器学习的核心方法之一。这背后的深刻

原因在于，现实世界本身就是不确定的，人类的观察能力是有局限性的（否则有很大一部分科学就没有必要做了——设想我们能够直接观察到电子的运行，还需要对原子模型争吵不休吗？），我们日常所观察到的只是事物表面上的结果，沿用刚才那个袋子里面取球的比方，我们往往只能知道从里面取出来的球是什么颜色，而并不能直接看到袋子里面实际的情况。这个时候，我们就需要提供一个猜测（hypothesis，更为严格的说法是“假设”，这里用“猜测”更通俗易懂一点），所谓猜测，当然就是不确定的（很可能有好多种乃至无数种猜测都能满足目前的观测），但也绝对不是两眼一抹黑瞎蒙——具体地说，我们需要做两件事情：1.算出各种不同猜测的可能性大小。2.算出最靠谱的猜测是什么。第一个就是计算特定猜测的后验概率，对于连续的猜测空间则是计算猜测的概率密度函数。第二个则是所谓的模型比较，模型比较如果不考虑先验概率的话就是最大似然方法。

## 1.1 一个例子：自然语言的二义性

下面举一个自然语言的不确定性的例子。当你看到这句话：

The girl saw the boy with a telescope.

你对这句话的含义有什么猜测？平常人肯定会说：那个女孩拿望远镜看见了那个男孩（即你对这个句子背后的实际语法结构的猜测是：The girl saw-with-a-telescope the boy）。然而，仔细一想，你会发现这个句子完全可以解释成：那个女孩看见了那个拿着望远镜的男孩（即：The girl saw the-boy-with-a-telescope）。那为什么平常生活中我们每个人都能够迅速地对这种二义性进行消解呢？这背后到底隐藏着什么样的思维法则？我们留到后面解释。

1.2 贝叶斯公式

贝叶斯公式是怎么来的？

我们还是使用 wikipedia 上的一个例子：

一所学校里面有 60%的男生，40%的女生。男生总

是穿长裤，女生则一半穿长裤一半穿裙子。有了这些信息之后我们可以容易地计算“随机选取一个学生，他(她)穿长裤的概率和穿裙子的概率是多大”，这个就是前面说的“正向概率”的计算。然而，假设你走在校园中，迎面走来一个穿长裤的学生（很不幸的是你高度近似，你只看得见他（她）穿的是否长裤，而无法确定他（她）的性别），你能够推断出他(她)是男生的概率是多大吗？

一些认知科学的研究表明（《决策与判断》以及《Rationality for Mortals》第 12 章：小孩也可以解决贝叶斯问题），我们对形式化的贝叶斯问题不擅长，但对于以频率形式呈现的等价问题却很擅长。在这里，我们不妨把问题重新叙述成：你在校园里面随机游走，遇到了 N 个穿长裤的人（仍然假设你无法直接观察到他们的性别），问这 N 个人里面有多少个女生多少个男生。

你说，这还不简单：算出学校里面有多少穿长裤的，然后在这些人里面再算出有多少女生，不就行了？

我们来算一算：假设学校里面人的总数是 U 个。60%

的男生都穿长裤，于是我们得到了 U*P(Boy)*P(Pants|Boy)个穿长裤的（男生）（其中 P(Boy)是男生的概率=60%，这里可以简单的理解为男生的比例；P(Pants|Boy)是条件概率，即在 Boy 这个条件下穿长裤的概率是多大，这里是 100%，因为所有男生都穿长裤）。40%的女生里面又有一半（50%）是穿长裤的，于是我们又得到了 U*P(Girl)*P(Pants|Girl)个穿长裤的（女生）。加起来一共是 U*P(Boy)*P(Pants|Boy)+U*P(Girl)*P(Pants|Girl)个穿长裤的，其中有 U*P(Girl)*P(Pants|Girl)个女生。两者一比就是你要求的答案。

下面我们把这个答案形式化一下：我们要求的是 P(Girl|Pants)（穿长裤的人里面有多少女生），我们计算的结果是 U*P(Girl)*P(Pants|Girl)/[U*P(Boy)*P(Pants|Boy)+U*P(Girl)*P(Pants|Girl)]。容易发现这里校园内人的总数是无关的，可以消去。于是得到

P(Girl|Pants)=P(Girl)*P(Pants|Girl)/[P(Boy)*P(Pants|Boy)+P(Girl)*P(Pants|Girl)]

注意，如果把上式收缩起来，分母其实就是 P(Pants)，分子其实就是 P(Pants,Girl)。而这个比例很自然地就读作：在穿长裤的人（P(Pants)）里面有多少（穿长裤）的女孩（P(Pants,Girl)）。

上式中的 Pants 和 Boy/Girl 可以指代一切东西，所以其一般形式就是：

$$P(B|A)=P(A|B)*P(B)/[P(A|B)*P(B)+P(A|\sim B)*P(\sim B)]$$

收缩起来就是：

$$P(B|A)=P(AB)/P(A)$$

其实这个就等于：

$$P(B|A)*P(A)=P(AB)$$

难怪拉普拉斯说概率论只是把常识用数学公式表达了出来。

然而，后面我们会逐渐发现，看似这么平凡的贝叶

斯公式，背后却隐含着非常深刻的原理。

2.拼写纠正

经典著作《人工智能：现代方法》的作者之一 Peter Norvig 曾经写过一篇介绍如何写一个拼写检查/纠正器的文章（原文在这里，徐宥的翻译版在这里，这篇文章很深入浅出，强烈建议读一读），里面用到的就是贝叶斯方法，这里我们不打算复述他写的文章，而是简要地将其核心思想介绍一下。

首先，我们需要询问的是：“问题是什么？”

问题是我们看到用户输入了一个不在字典中的单词，我们需要去猜测：“这个家伙到底真正想输入的单词是什么呢？” 用刚才我们形式化的语言来叙述就是，我们需要求：

P(我们猜测他想输入的单词|他实际输入的单词)

这个概率。并找出那个使得这个概率最大的猜测单词。显然，我们的猜测未必是唯一的，就像前面举的那

个自然语言的歧义性的例子一样；这里，比如用户输入：thew，那么他到底是想输入 the，还是想输入 thaw？到底哪个猜测可能性更大呢？幸运的是我们可以用贝叶斯公式来直接出它们各自的概率，我们不妨将我们的多个猜测记为 h1 h2..（h 代表 hypothesis），它们都属于一个有限且离散的猜测空间 H（单词总共就那么多而已），将用户实际输入的单词记为 D（D 代表 Data，即观测数据），于是

P(我们的猜测 1|他实际输入的单词)

可以抽象地记为：

P(h1|D)

类似地，对于我们的猜测 2，则是 P(h2|D)。不妨统一记为：

P(h|D)

运用一次贝叶斯公式，我们得到：

P(h|D)=P(h)*P(D|h)/P(D)

对于不同的具体猜测 h1 h2 h3..，P(D)都是一样的，所以在比较P(h1|D)和P(h2|D)的时候我们可以忽略这个常数。即我们只需要知道：

P(h|D)∝P(h)*P(D|h)（注：那个符号的意思是“正比例于”，不是无穷大，注意符号右端是有一个小缺口的。）

这个式子的抽象含义是：对于给定观测数据，一个猜测是好是坏，取决于“这个猜测本身独立的可能性大小（先验概率，Prior）”和“这个猜测生成我们观测到的数据的可能性大小”（似然，Likelihood）的乘积。具体到我们的那个 thew 例子上，含义就是，用户实际是想输入 the 的可能性大小取决于 the 本身在词汇表中被使用的可能性（频繁程度）大小（先验概率）和想打 the 却打成 thew 的可能性大小（似然）的乘积。

下面的事情就很简单了，对于我们猜测为可能的每个单词计算一下 P(h)*P(D|h)这个值，然后取最大的，得到的就是最靠谱的猜测。

一点注记：Norvig 的拼写纠正器里面只提取了编辑距离为 2 以内的所有已知单词。这是为了避免去遍历字典中每个单词计算它们的 P(h)*P(D|h)，但这种做法为了节省时间带来了一些误差。但话说回来难道我们人类真的回去遍历每个可能的单词来计算他们的后验概率吗？不可能。实际上，根据认知神经科学的观点，我们首先根据错误的单词做一个 bottom-up 的关联提取，提取出有可能是实际单词的那些候选单词，这个提取过程就是所谓的基于内容的提取，可以根据错误单词的一些模式片段提取出有限的一组候选，非常快地缩小的搜索空间（比如我输入 explaination，单词里面就有充分的信息使得我们的大脑在常数时间内把可能性 narrow down 到 explanation 这个单词上，至于具体是根据哪些线索——如音节——来提取，又是如何在生物神经网络中实现这个提取机制的，目前还是一个没有弄清的领域）。然后，我们对这有限的几个猜测做一个 top-down 的预测，看看到底哪个对于观测数据（即错误单词）的预测效力最好，而如何衡量预测效率则就是用贝叶斯公式里面的

那个 P(h)*P(D|h)了——虽然我们很可能使用了一些启发法来简化计算。后面我们还会提到这样的 bottom-up 的关联提取。

3.模型比较与奥卡姆剃刀

3.1 再访拼写纠正

介绍了贝叶斯拼写纠正之后，接下来的一个自然而然的问题就来了：“为什么？”为什么要用贝叶斯公式？为什么贝叶斯公式在这里可以用？我们可以很容易地领会为什么贝叶斯公式用在前面介绍的那个男生女生长裤裙子的问题里是正确的。但为什么这里？

为了回答这个问题，一个常见的思路就是想想：非得这样吗？因为如果你想到了另一种做法并且证明了它也是靠谱的，那么将它与现在这个一比较，也许就能得出很有价值的信息。那么对于拼写纠错问题你能想到其他方案吗？

不管怎样，一个最常见的替代方案就是，选择离 th

ew 的编辑距离最近的。然而 the 和 thaw 离 thew 的编辑距离都是 1。这可咋办捏？你说，不慌，那还是好办。我们就看到底哪个更可能被错打为 thew 就是了。我们注意到字母 e 和字母 w 在键盘上离得很紧，无名指一抽筋就不小心多打出一个 w 来，the 就变成 thew 了。而另一方面 thaw 被错打成 thew 的可能性就相对小一点，因为 e 和 a 离得较远而且使用的指头相差一个指头（一个是中指一个是小指，不像 e 和 w 使用的指头靠在一块——神经科学的证据表明紧邻的身体设施之间容易串位）。OK，很好，因为你现在已经是在用最大似然方法了，或者直白一点，你就是在计算那个使得 P(D|h)最大的 h。

而贝叶斯方法计算的是什么？是 P(h)*P(D|h)。多出来了一个 P(h)。我们刚才说了，这个多出来的 P(h)是特定猜测的先验概率。为什么要掺和进一个先验概率？刚才说的那个最大似然不是挺好么？很雄辩地指出了 the 是更靠谱的猜测。有什么问题呢？既然这样，我们就从给最大似然找茬开始吧——我们假设两者的似然程度是

一样或非常相近，这样不就难以区分哪个猜测更靠谱了吗？比如用户输入 tlp，那到底是 top 还是 tip？（这个例子不怎么好，因为 top 和 tip 的词频可能仍然是接近的，但一时想不到好的英文单词的例子，我们不妨就假设 top 比 tip 常见许多吧，这个假设并不影响问题的本质。）这个时候，当最大似然不能作出决定性的判断时，先验概率就可以插手进来给出指示——“既然你无法决定，那么我告诉你，一般来说 top 出现的程度要高许多，所以更可能他想打的是 top”）。

以上只是最大似然的一个问题，即并不能提供决策的全部信息。

最大似然还有另一个问题：即便一个猜测与数据非常符合，也并不代表这个猜测就是更好的猜测，因为这个猜测本身的可能性也许就非常低。比如 MacKay 在《Information Theory:Inference and Learning Algorithms》里面就举了一个很好的例子：-1 3 7 11 你说是等差数列更有可能呢？还是-X^3/11+9/11*X^2+23/11

每项把前项作为 X 带入后计算得到的数列？此外曲线拟合也是，平面上 N 个点总是可以用 N-1 阶多项式来完全拟合，当 N 个点近似但不精确共线的时候，用 N-1 阶多项式来拟合能够精确通过每一个点，然而用直线来做拟合/线性回归的时候却会使得某些点不能位于直线上。你说到底哪个好呢？多项式？还是直线？一般地说肯定是越低阶的多项式越靠谱（当然前提是也不能忽视“似然”P(D|h)，明摆着一个多项式分布您愣是去拿直线拟合也是不靠谱的，这就是为什么要把它们两者乘起来考虑。），原因之一就是低阶多项式更常见，先验概率（P(h)）较大（原因之二则隐藏在 P(D|h)里面），这就是为什么我们要用样条来插值，而不是直接搞一个 N-1 阶多项式来通过任意 N 个点的原因。

以上分析当中隐含的哲学是，观测数据总是会有各种各样的误差，比如观测误差（比如你观测的时候一个 MM 经过你一不留神，手一抖就是一个误差出现了），所以如果过分去寻求能够完美解释观测数据的模型，就会落入所谓的数据过配（overfitting）的境地，一个过配

的模型试图连误差（噪音）都去解释（而实际上噪音又是不需要解释的），显然就过犹不及了。所以 P(D|h)大不代表你的 h（猜测）就是更好的 h。还要看 P(h)是怎样的。所谓奥卡姆剃刀精神就是说：如果两个理论具有相似的解释力度，那么优先选择那个更简单的（往往也正是更平凡的，更少繁复的，更常见的）。

过分匹配的另一个原因在于当观测的结果并不是因为误差而显得“不精确”而是因为真实世界中对数据的结果产生贡献的因素太多太多，跟噪音不同，这些偏差是一些另外的因素集体贡献的结果，不是你的模型所能解释的——噪音那是不需要解释——一个现实的模型往往只提取出几个与结果相关度很高，很重要的因素（cause）。这个时候观察数据会倾向于围绕你的有限模型的预测结果呈正态分布，于是你实际观察到的结果就是这个正态分布的随机取样，这个取样很可能受到其余因素的影响偏离你的模型所预测的中心，这个时候便不能贪心不足地试图通过改变模型来“完美”匹配数据，因为那些使结果偏离你的预测的贡献因素不是你这个有限模

型里面含有的因素所能概括的，硬要打肿脸充胖子只能导致不实际的模型，举个教科书例子：身高和体重的实际关系近似于一个二阶多项式的关系，但大家都知道并不是只有身高才会对体重产生影响，物理世界影响体重的因素太多太多了，有人身材高大却瘦得跟稻草，有人却是横长竖不长。但不可否认的是总体上来说，那些特殊情况越是特殊就越是稀少，呈围绕最普遍情况（胖瘦适中）的正态分布，这个分布就保证了我们的身高——体重相关模型能够在大多数情况下做出靠谱的预测。但是——刚才说了，特例是存在的，就算不是特例，人有胖瘦，密度也有大小，所以完美符合身高——体重的某个假想的二阶多项式关系的人是不存在的，我们又不是欧几里德几何世界当中的理想多面体，所以，当我们对人群随机抽取了 N 个样本（数据点）试图对这 N 个数据点拟合出一个多项式的话就得注意，它肯定得是二阶多项式，我们要做的只是去根据数据点计算出多项式各项的参数（一个典型的方法就是最小二乘）；它肯定不是直线（我们又不是稻草），也不是三阶多项式四阶多项式..

如果硬要完美拟合 N 个点，你可能会整出一个 N-1 阶多项式来——设想身高和体重的关系是 5 阶多项式看看？

3.2 模型比较理论（Model Comparasion）与贝叶斯奥卡姆剃刀（Bayesian Occam’s Razor）

实际上，模型比较就是去比较哪个模型（猜测）更可能隐藏在观察数据的背后。其基本思想前面已经用拼写纠正的例子来说明了。我们对用户实际想输入的单词的猜测就是模型，用户输错的单词就是观测数据。我们通过：

$$P(h|D) \propto P(h)*P(D|h)$$

来比较哪个模型最为靠谱。前面提到，光靠 P(D|h)（即“似然”）是不够的，有时候还需要引入 P(h)这个先验概率。奥卡姆剃刀就是说 P(h)较大的模型有较大的优势，而最大似然则是说最符合观测数据的（即 P(D|h)最大的）最有优势。整个模型比较就是这两方力量的拉锯。我们不妨再举一个简单的例子来说明这一精神：你随便找枚硬币，掷一下，观察一下结果。好，你观察到的结

果要么是“正”，要么是“反”（不，不是少林足球那枚硬币:P），不妨假设你观察到的是“正”。现在你要去根据这个观测数据推断这枚硬币掷出“正”的概率是多大。根据最大似然估计的精神，我们应该猜测这枚硬币掷出“正”的概率是 1，因为这个才是能最大化 P(D|h)的那个猜测。然而每个人都会大摇其头——很显然，你随机摸出一枚硬币这枚硬币居然没有反面的概率是“不存在的”,我们对一枚随机硬币是否一枚有偏硬币，偏了多少，是有着一个先验的认识的，这个认识就是绝大多数硬币都是基本公平的，偏得越多的硬币越少见（可以用一个 beta 分布来表达这一先验概率）。将这个先验正态分布 p(θ)（其中θ表示硬币掷出正面的比例，小写的 p 代表这是概率密度函数）结合到我们的问题中，我们便不是去最大化 P(D|h)，而是去最大化 P(D|θ)*p(θ)，显然θ=1 是不行的，因为 P(θ=1)为 0，导致整个乘积也为 0。实际上，只要对这个式子求一个导数就可以得到最值点。

以上说的是当我们知道先验概率 P(h)的时候，光用最大似然是不靠谱的，因为最大似然的猜测可能先验概

率非常小。然而，有些时候，我们对于先验概率一无所知，只能假设每种猜测的先验概率是均等的，这个时候就只有用最大似然了。实际上，统计学家和贝叶斯学家有一个有趣的争论，统计学家说：我们让数据自己说话。言下之意就是要摒弃先验概率。而贝叶斯支持者则说：数据会有各种各样的偏差，而一个靠谱的先验概率则可以对这些随机噪音做到健壮。事实证明贝叶斯派胜利了，胜利的关键在于所谓先验概率其实也是经验统计的结果，譬如为什么我们会认为绝大多数硬币是基本公平的？为什么我们认为大多数人的肥胖适中？为什么我们认为肤色是种族相关的，而体重则与种族无关？先验概率里面的“先验”并不是指先于一切经验，而是仅指先于我们“当前”给出的观测数据而已，在硬币的例子中先验指的只是先于我们知道投掷的结果这个经验，而并非“先天”。

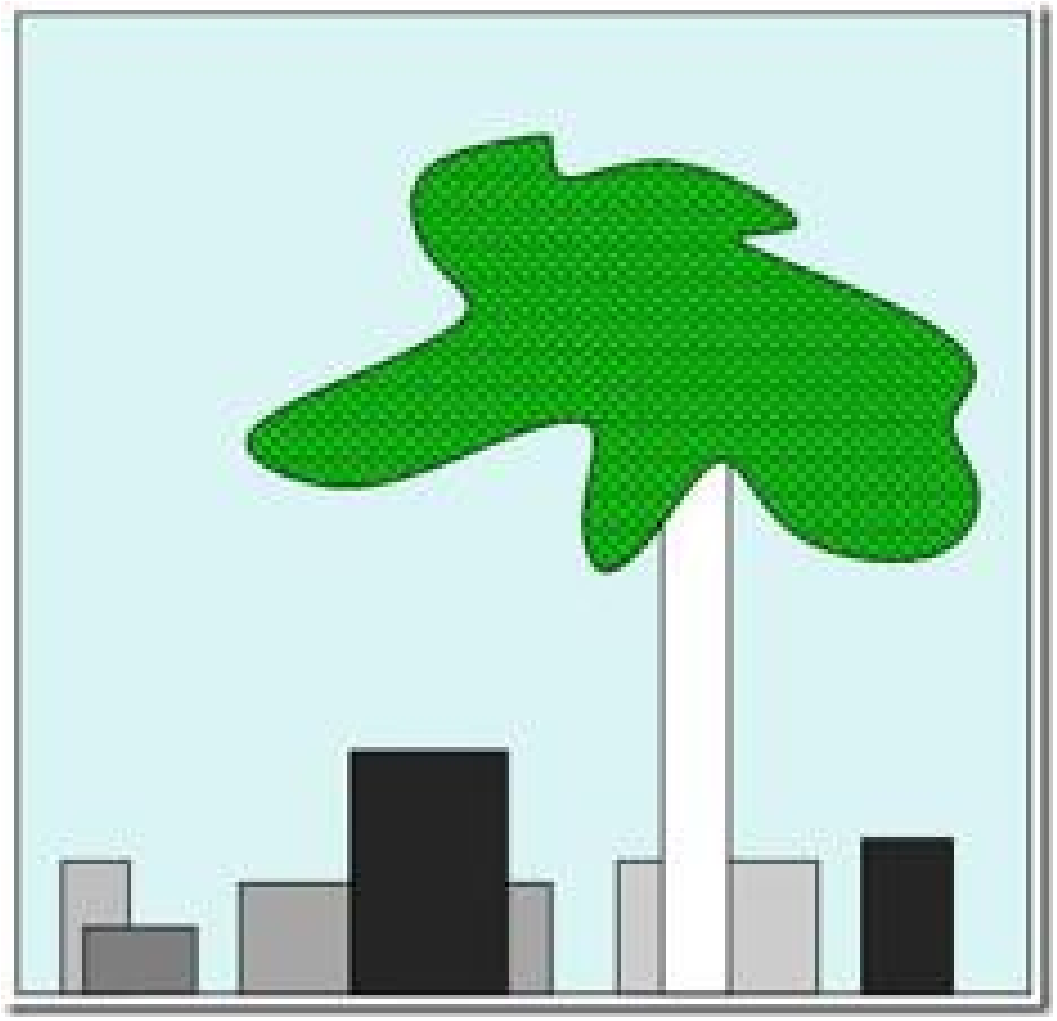
然而，话说回来，有时候我们必须得承认，就算是基于以往的经验，我们手头的“先验”概率还是均匀分布，这个时候就必须依赖用最大似然，我们用前面留下

的一个自然语言二义性问题来说明这一点：

The girl saw the boy with a telescope.

到底是 The girl saw-with-a-telescope the boy 这一语法结构，还是 The girl saw the-boy-with-a-telescope 呢？两种语法结构的常见程度都差不多( 你可能会觉得后一种语法结构的常见程度较低，这是事后偏见，你只需想想 The girl saw the boy with a book 就知道了。当然，实际上从大规模语料统计结果来看后一种语法结构的确稍稍不常见一丁点，但是绝对不足以解释我们对第一种结构的强烈倾向 )。那么到底为什么呢？

我们不妨先来看看 MacKay 在书中举的一个漂亮的例子：



图中有多少个箱子？特别地，那棵书后面是一个箱子？还是两个箱子？还是三个箱子？还是..你可能会觉得树后面肯定是一个箱子，但为什么不是两个呢？如下图：

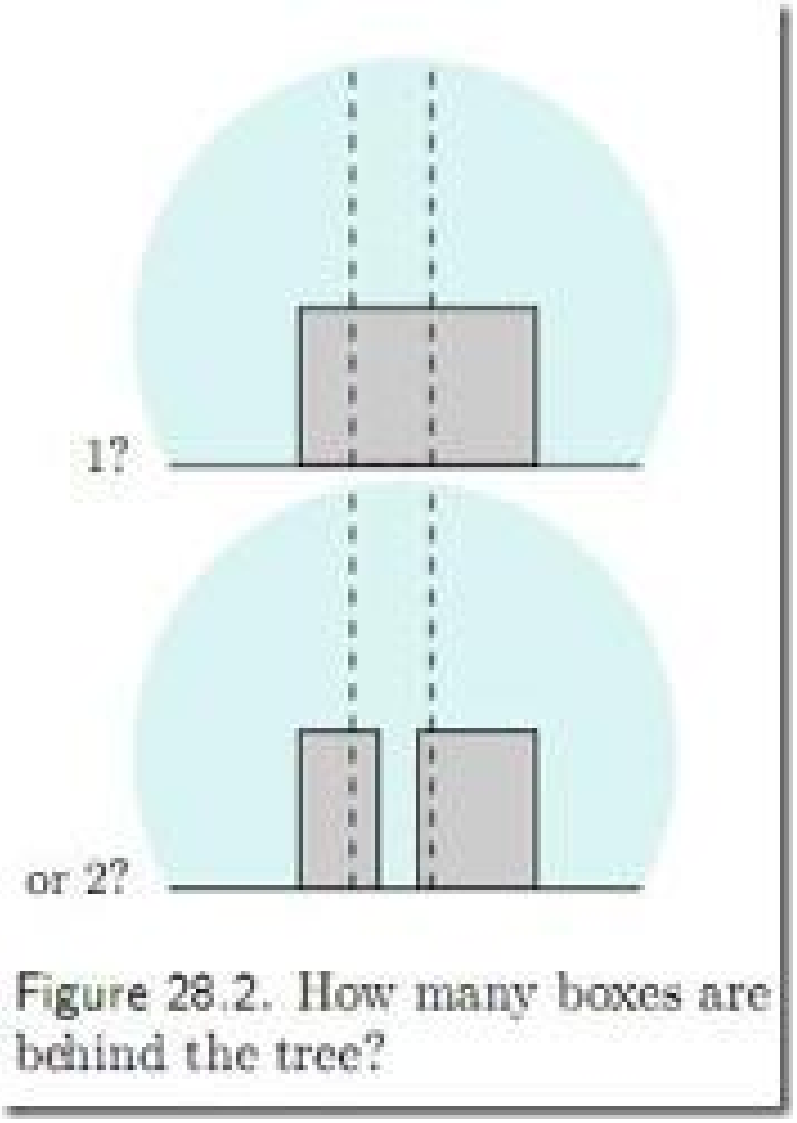



Figure 28.2. How many boxes are behind the tree?

很简单，你会说：要是真的有两个箱子那才怪了，怎么就那么巧这两个箱子刚刚好颜色相同，高度相同呢？

用概率论的语言来说，你刚才的话就翻译为：猜测 h 不成立，因为 P(D|h)太小（太巧合）了。我们的直觉是：巧合（小概率）事件不会发生。所以当一个猜测（假设）使得我们的观测结果成为小概率事件的时候，我们就说

“才怪呢，哪能那么巧捏？！”

现在我们可以回到那个自然语言二义性的例子，并给出一个完美的解释了 :如果语法结构是 The girl saw the-boy-with-a-telecope 的话 ,怎么那个男孩偏偏手里拿的就是望远镜——一个可以被用来 saw-with 的东东捏？这也忒小概率了吧。他咋就不会拿本书呢？拿什么都好。怎么偏偏就拿了望远镜？所以唯一的解释是，这个“巧合”背后肯定有它的必然性，这个必然性就是，如果我们将语法结构解释为 The girl saw-with-a-telescope the boy 的话 ,就跟数据完美吻合了——既然那个女孩是用某个东西去看这个男孩的，那么这个东西是一个望远镜就完全可以解释了（不再是小概率事件了）。

自然语言二义性很常见，譬如上文中的一句话：

参见《决策与判断》以及《Rationality for Mortals》第 12 章：小孩也可以解决贝叶斯问题

就有二义性 :到底是参见这两本书的第 12 章 ,还是仅仅是第二本书的第 12 章呢？如果是这两本书的第 12

章那就是咄咄怪事了，怎么恰好两本书都有第 12 章，都是讲同一个问题，更诡异的是，标题还相同呢？

注意，以上做的是似然估计（即只看 P(D|h)的大小），不含先验概率。通过这两个例子，尤其是那个树后面的箱子的例子我们可以看到，似然估计里面也蕴含着奥卡姆剃刀：树后面的箱子数目越多，这个模型就越复杂。单个箱子的模型是最简单的。似然估计选择了更简单的模型。

这个就是所谓的贝叶斯奥卡姆剃刀（Bayesian Occam’s Razor），因为这个剃刀工作在贝叶斯公式的似然（P(D|h)）上，而不是模型本身（P(h)）的先验概率上，后者是传统的奥卡姆剃刀。关于贝叶斯奥卡姆剃刀我们再来看一个前面说到的曲线拟合的例子：如果平面上有 N 个点，近似构成一条直线，但绝不精确地位于一条直线上。这时我们既可以用直线来拟合（模型 1），也可以用二阶多项式（模型 2）拟合，也可以用三阶多项式（模型 3），..，特别地，用 N-1 阶多项式便能够保证肯定能

完美通过 N 个数据点。那么，这些可能的模型之中到底哪个是最靠谱的呢？前面提到，一个衡量的依据是奥卡姆剃刀：越是高阶的多项式越是繁复和不常见。然而，我们其实并不需要依赖于这个先验的奥卡姆剃刀，因为有人可能会争辩说：你怎么就能说越高阶的多项式越不常见呢？我偏偏觉得所有阶多项式都是等可能的。好吧，既然如此那我们不妨就扔掉 P(h)项，看看 P(D|h)能告诉我们什么。我们注意到越是高阶的多项式，它的轨迹弯曲程度越是大，到了八九阶简直就是直上直下，于是我们不仅要问：一个比如说八阶多项式在平面上随机生成的一堆 N 个点偏偏恰好近似构成一条直线的概率（即 P(D|h)）有多大？太小太小了。反之，如果背后的模型是一条直线，那么根据该模型生成一堆近似构成直线的点的概率就大得多了。这就是贝叶斯奥卡姆剃刀。

这里只是提供一个关于贝叶斯奥卡姆剃刀的科普，强调直观解释，更多理论公式请参考 MacKay 的著作《Information Theory:Inference and Learning Algorithms》第 28 章。

3.3 最小描述长度原则

贝叶斯模型比较理论与信息论有一个有趣的关联：

$P(h|D) \propto P(h) * P(D|h)$

两边求对数，将右式的乘积变成相加：

$\ln P(h|D) \propto \ln P(h) + \ln P(D|h)$

显然，最大化 $P(h|D)$也就是最大化 $\ln P(h|D)$。而 $\ln P(h) + \ln P(D|h)$则可以解释为模型（或者称“假设”、“猜测”）h 的编码长度加上在该模型下数据 D 的编码长度。使这个和最小的模型就是最佳模型。

而究竟如何定义一个模型的编码长度，以及数据在模型下的编码长度则是一个问题。更多可参考 Mitchell 的《Machine Learning》的 6.6 节，或 Mackay 的 28.3 节）

3.4 最优贝叶斯推理

所谓的推理，分为两个过程，第一步是对观测数据

建立一个模型。第二步则是使用这个模型来推测未知现象发生的概率。我们前面都是讲的对于观测数据给出最靠谱的那个模型。然而很多时候，虽然某个模型是所有模型里面最靠谱的，但是别的模型也并不是一点机会都没有。譬如第一个模型在观测数据下的概率是 0.5。第二个模型是 0.4，第三个是 0.1。如果我们只想知道对于观测数据哪个模型最可能，那么只要取第一个就行了，故事到此结束。然而很多时候我们建立模型是为了推测未知的事情的发生概率，这个时候，三个模型对未知的事情发生的概率都会有自己的预测，仅仅因为某一个模型概率稍大一点就只听他一个人的就太不民主了。所谓的最优贝叶斯推理就是将三个模型对于未知数据的预测结论加权平均起来（权值就是模型相应的概率）。显然，这个推理是理论上的制高点，无法再优了，因为它已经把所有可能性都考虑进去了。

只不过实际上我们是基本不会使用这个框架的，因为计算模型可能非常费时间，二来模型空间可能是连续的，即有无穷多个模型（这个时候需要计算模型的概率

分布）。结果还是非常费时间。所以这个被看作是一个理论基准。

4.无处不在的贝叶斯

以下我们再举一些实际例子来说明贝叶斯方法被运用的普遍性，这里主要集中在机器学习方面，因为我不是学经济的，否则还可以找到一堆经济学的例子。

4.1 中文分词

贝叶斯是机器学习的核心方法之一。比如中文分词领域就用到了贝叶斯。Google 研究员吴军在《数学之美》系列中就有一篇是介绍中文分词的，这里只介绍一下核心的思想，不做赘述，详细请参考吴军的文章（这里）。

分词问题的描述为：给定一个句子（字串），如：

南京市长江大桥

如何对这个句子进行分词（词串）才是最靠谱的。例如：

1.南京市/长江大桥

2.南京/市长/江大桥

这两个分词，到底哪个更靠谱呢？

我们用贝叶斯公式来形式化地描述这个问题，令 X 为字串（句子），Y 为词串（一种特定的分词假设）。我们就是需要寻找使得 P(Y|X)最大的 Y，使用一次贝叶斯可得：

$$P(Y|X) \propto P(Y)*P(X|Y)$$

用自然语言来说就是这种分词方式（词串）的可能性乘以这个词串生成我们的句子的可能性。我们进一步容易看到：可以近似地将 P(X|Y)看作是恒等于 1 的，因为任意假想的一种分词方式之下生成我们的句子总是精准地生成的（只需把分词之间的分界符号扔掉即可）。于是，我们就变成了去最大化 P(Y)，也就是寻找一种分词使得这个词串（句子）的概率最大化。而如何计算一个词串：

W1,W2,W3,W4..

的可能性呢？我们知道，根据联合概率的公式展开：P(W1,W2,W3,W4..)=P(W1)*P(W2|W1)*P(W3|W2,W1)*P(W4|W1,W2,W3)*..于是我们可以通过一系列的条件概率（右式）的乘积来求整个联合概率。然而不幸的是随着条件数目的增加（P(Wn|Wn-1,Wn-2,..,W1)的条件有 n-1 个），数据稀疏问题也会越来越严重，即便语料库再大也无法统计出一个靠谱的 P(Wn|Wn-1,Wn-2,..,W1)来。为了缓解这个问题，计算机科学家们一如既往地使用了“天真”假设：我们假设句子中一个词的出现概率只依赖于它前面的有限的 k 个词（k 一般不超过 3，如果只依赖于前面的一个词，就是 2 元语言模型（2-gram），同理有 3-gram、4-gram 等），这个就是所谓的“有限地平线”假设。虽然这个假设很傻很天真，但结果却表明它的结果往往是很好很强大的，后面要提到的朴素贝叶斯方法使用的假设跟这个精神上是完全一致的，我们会解释为什么像这样一个天真的假设能够得到强大的结果。目前我们只要知道，有了这个假设，刚才

那个乘积就可以改写成：P(W1)*P(W2|W1)*P(W3|W2)*P(W4|W3)..（假设每个词只依赖于它前面的一个词）。而统计 P(W2|W1)就不再受到数据稀疏问题的困扰了。对于我们上面提到的例子“南京市长江大桥”，如果按照自左到右的贪婪方法分词的话，结果就成了“南京市长/江大桥”。但如果按照贝叶斯分词的话（假设使用 3-gram），由于“南京市长”和“江大桥”在语料库中一起出现的频率为 0，这个整句的概率便会被判定为 0。从而使得“南京市/长江大桥”这一分词方式胜出。

一点注记：有人可能会疑惑，难道我们人类也是基于这些天真的假设来进行推理的？不是的。事实上，统计机器学习方法所统计的东西往往处于相当表层（shallow）的层面，在这个层面机器学习只能看到一些非常表面的现象，有一点科学研究的理念的人都知道：越是往表层去，世界就越是繁复多变。从机器学习的角度来说，特征（feature）就越多，成百上千维度都是可能的。特征一多，好了，高维诅咒就产生了，数据就稀疏得要命，不够用了。而我们人类的观察水平显然比机器学习的观

察水平要更深入一些，为了避免数据稀疏我们不断地发明各种装置（最典型就是显微镜），来帮助我们直接深入到更深层的事物层面去观察更本质的联系，而不是在浅层对表面现象作统计归纳。举一个简单的例子，通过对大规模语料库的统计，机器学习可能会发现这样一个规律：所有的“他”都是不会穿 bra 的，所有的“她”则都是穿的。然而，作为一个男人，却完全无需进行任何统计学习，因为深层的规律就决定了我们根本不会去穿 bra。至于机器学习能不能完成后者（像人类那样的）这个推理，则是人工智能领域的经典问题。至少在那之前，声称统计学习方法能够终结科学研究（原文）的说法是纯粹外行人说的话。

4.2 统计机器翻译

统计机器翻译因为其简单，自动（无需手动添加规则），迅速成为了机器翻译的事实标准。而统计机器翻译的核心算法也是使用的贝叶斯方法。

问题是什么？统计机器翻译的问题可以描述为：给

定一个句子e,它的可能的外文翻译f中哪个是最靠谱的。即我们需要计算：P(f|e)。一旦出现条件概率贝叶斯总是挺身而出：

$$P(f|e) \propto P(f)*P(e|f)$$

这个式子的右端很容易解释：那些先验概率较高，并且更可能生成句子 e 的外文句子 f 将会胜出。我们只需简单统计（结合上面提到的 N-Gram 语言模型）就可以统计任意一个外文句子 f 的出现概率。然而 P(e|f)却不是那么好求的，给定一个候选的外文局子 f，它生成（或对应）句子 e 的概率是多大呢？我们需要定义什么叫“对应”，这里需要用到一个分词对齐的平行语料库，有兴趣的可以参考《Foundations of Statistical Natural Language Processing》第 13 章，这里摘选其中的一个例子：假设 e 为：John loves Mary。我们需要考察的首选 f 是：Jean aime Marie（法文）。我们需要求出 P(e|f)是多大，为此我们考虑 e 和 f 有多少种对齐的可能性，如：

John(Jean)loves(aime)Marie(Mary)

就是其中的一种（最靠谱的）对齐，为什么要对齐，是因为一旦对齐了之后，就可以容易地计算在这个对齐之下的 P(e|f)是多大，只需计算：

P(John|Jean)*P(loves|aime)*P(Marie|Mary)

即可。

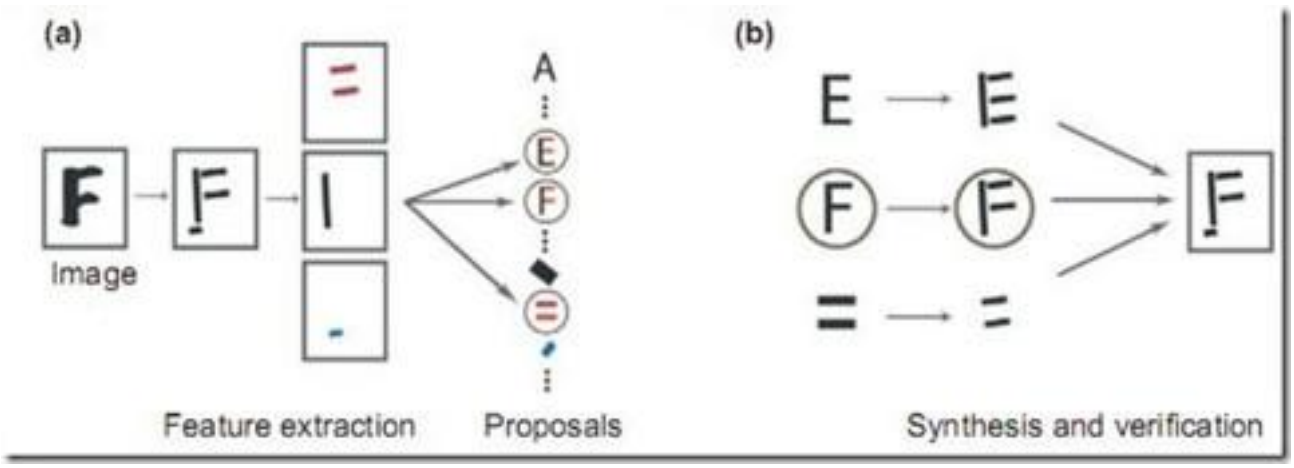
然后我们遍历所有的对齐方式，并将每种对齐方式之下的翻译概率∑求和。便可以获得整个的 P(e|f)是多大。

一点注记：还是那个问题：难道我们人类真的是用这种方式进行翻译的？highly unlikely。这种计算复杂性非常高的东西连三位数乘法都搞不定的我们才不会笨到去使用呢。根据认知神经科学的认识，很可能我们是先从句子到语义（一个逐层往上（bottom-up）抽象的 folding 过程），然后从语义根据另一门语言的语法展开为另一门语言（一个逐层往下（top-down）的具体化 unfolding 过程）。如何可计算地实现这个过程，目前仍然是个难题。（我们看到很多地方都有 bottom-up/top-d

own 这样一个对称的过程，实际上有人猜测这正是生物神经网络原则上的运作方式，对视觉神经系统的研究尤其证明了这一点，Hawkins 在《On Intelligence》里面提出了一种 HTM（Hierarchical Temporal Memory）模型正是使用了这个原则。）

4.3 贝叶斯图像识别，Analysis by Synthesis

贝叶斯方法是一个非常 general 的推理框架。其核心理念可以描述成：Analysis by Synthesis（通过合成来分析）。06 年的认知科学新进展上有一篇 paper 就是讲用贝叶斯推理来解释视觉识别的，一图胜千言，下图就是摘自这篇 paper：

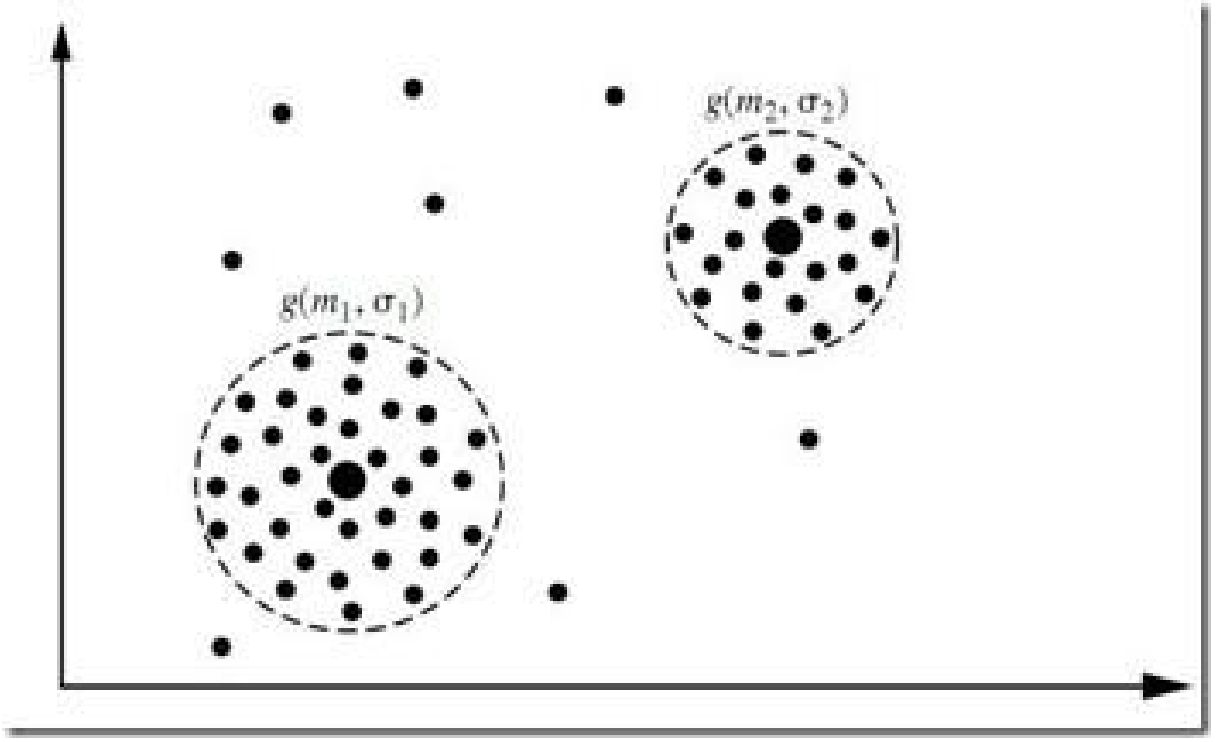



首先是视觉系统提取图形的边角特征，然后使用这

些特征自底向上地激活高层的抽象概念（比如是 E 还是 F 还是等号），然后使用一个自顶向下的验证来比较到底哪个概念最佳地解释了观察到的图像。

## 4.4 EM 算法与基于模型的聚类

聚类是一种无指导的机器学习问题，问题描述：给你一堆数据点，让你将它们最靠谱地分成一堆一堆的。聚类算法很多，不同的算法适应于不同的问题，这里仅介绍一个基于模型的聚类，该聚类算法对数据点的假设是，这些数据点分别是围绕 K 个核心的 K 个正态分布源所随机生成的，使用 Han JiaWei 的《Data Ming：Concepts and Techniques》中的图：

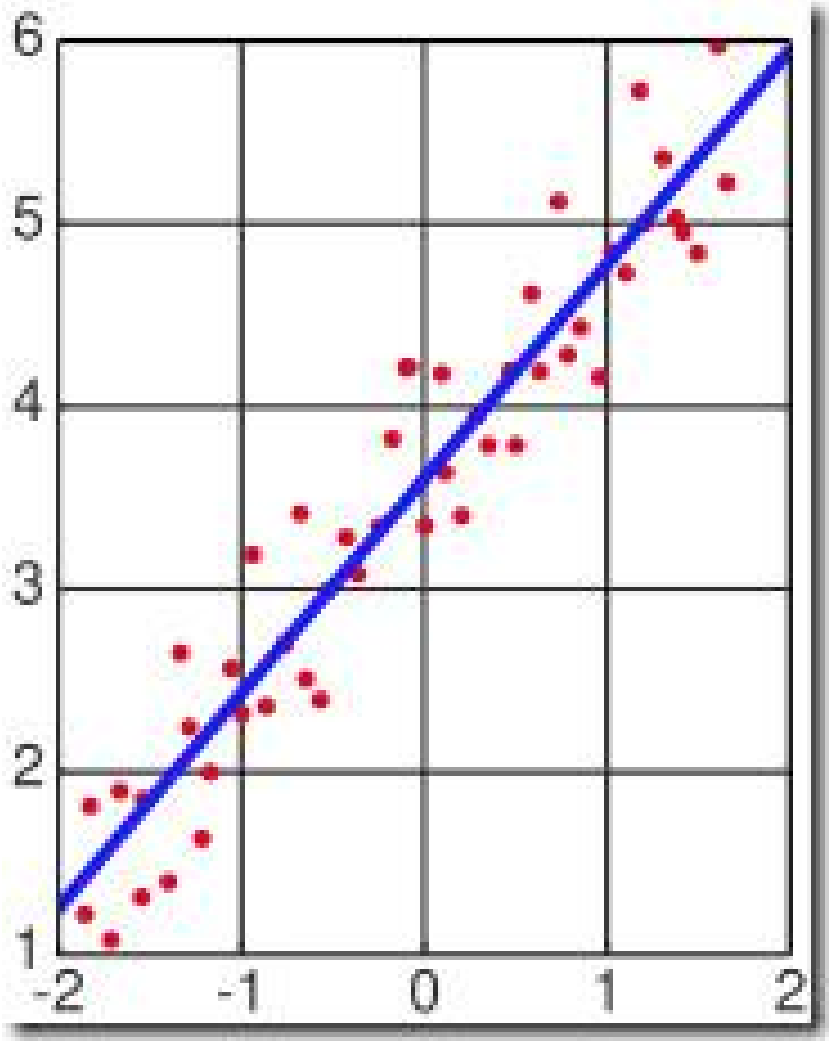



图中有两个正态分布核心，生成了大致两堆点。我们的聚类算法就是需要根据给出来的那些点，算出这两个正态分布的核心在什么位置，以及分布的参数是多少。这很明显又是一个贝叶斯问题，但这次不同的是，答案是连续的且有无穷多种可能性，更糟的是，只有当我们知道了哪些点属于同一个正态分布圈的时候才能够对这个分布的参数作出靠谱的预测，现在两堆点混在一块我们又不知道哪些点属于第一个正态分布，哪些属于第二个。反过来，只有当我们对分布的参数作出了靠谱的预测时候，才能知道到底哪些点属于第一个分布，那些点属于第二个分布。这就成了一个先有鸡还是先有蛋的问

题了。为了解决这个循环依赖，总有一方要先打破僵局，说，不管了，我先随便整一个值出来，看你怎么变，然后我再根据你的变化调整我的变化，然后如此迭代着不断互相推导，最终收敛到一个解。这就是 EM 算法。

EM 的意思是“Expectation-Maximazation”，在这个聚类问题里面，我们是先随便猜一下这两个正态分布的参数：如核心在什么地方，方差是多少。然后计算出每个数据点更可能属于第一个还是第二个正态分布圈，这个是属于 Expectation 一步。有了每个数据点的归属，我们就可以根据属于第一个分布的数据点来重新评估第一个分布的参数（从蛋再回到鸡），这个是 Maximazation。如此往复，直到参数基本不再发生变化为止。这个迭代收敛过程中的贝叶斯方法在第二步，根据数据点求分布的参数上面。

### 4.5 最大似然与最小二乘




学过线性代数的大概都知道经典的最小二乘方法来做线性回归。问题描述是：给定平面上 N 个点，(这里不妨假设我们想用一条直线来拟合这些点——回归可以看作是拟合的特例，即允许误差的拟合)，找出一条最佳描述了这些点的直线。

一个接踵而来的问题就是，我们如何定义最佳？我们设每个点的坐标为(Xi,Yi)。如果直线为 y=f(x)。那么(Xi,Yi)跟直线对这个点的“预测”:(Xi,f(Xi))就相差了一个Δ

Yi=|Yi–f(Xi)|。最小二乘就是说寻找直线使得(ΔY1)^2+(ΔY2)^2+..（即误差的平方和）最小，至于为什么是误差的平方和而不是误差的绝对值和，统计学上也没有什么好的解释。然而贝叶斯方法却能对此提供一个完美的解释。

我们假设直线对于坐标 Xi 给出的预测 f(Xi)是最靠谱的预测，所有纵坐标偏离 f(Xi)的那些数据点都含有噪音，是噪音使得它们偏离了完美的一条直线，一个合理的假设就是偏离路线越远的概率越小，具体小多少，可以用一个正态分布曲线来模拟，这个分布曲线以直线对 Xi 给出的预测 f(Xi)为中心，实际纵坐标为 Yi 的点(Xi,Yi)发生的概率就正比于 EXP[-(ΔYi)^2]。（EXP(..)代表以常数 e 为底的多少次方）。

现在我们回到问题的贝叶斯方面，我们要想最大化的后验概率是：

$$P(h|D) \propto P(h)*P(D|h)$$

又见贝叶斯！这里 h 就是指一条特定的直线，D 就

是指这 N 个数据点。我们需要寻找一条直线 h 使得 P(h)*P(D|h)最大。很显然，P(h)这个先验概率是均匀的，因为哪条直线也不比另一条更优越。所以我们只需要看 P(D|h)这一项，这一项是指这条直线生成这些数据点的概率，刚才说过了，生成数据点(Xi,Yi)的概率为 EXP[-(ΔYi)^2]乘以一个常数。而 P(D|h)=P(d1|h)*P(d2|h)*..即假设各个数据点是独立生成的，所以可以把每个概率乘起来。于是生成 N 个数据点的概率为 EXP[-(ΔY1)^2]*EXP[-(ΔY2)^2]*EXP[-(ΔY3)^2]*..=EXP{-[(ΔY1)^2+(ΔY2)^2+(ΔY3)^2+..]}最大化这个概率就是要最小化(ΔY1)^2+(ΔY2)^2+(ΔY3)^2+..。熟悉这个式子吗？

5.朴素贝叶斯方法

朴素贝叶斯方法是一个很特别的方法，所以值得介绍一下。我们用朴素贝叶斯在垃圾邮件过滤中的应用来举例说明。

5.1 贝叶斯垃圾邮件过滤器

问题是什么？问题是，给定一封邮件，判定它是否

属于垃圾邮件。按照先例，我们还是用 D 来表示这封邮件，注意 D 由 N 个单词组成。我们用 h+来表示垃圾邮件，h-表示正常邮件。问题可以形式化地描述为求：

P(h+|D)=P(h+)*P(D|h+)/P(D)

P(h-|D)=P(h-)*P(D|h-)/P(D)

其中 P(h+)和 P(h-)这两个先验概率都是很容易求出来的，只需要计算一个邮件库里面垃圾邮件和正常邮件的比例就行了。然而 P(D|h+)却不容易求，因为 D 里面含有 N 个单词 d1,d2,d3,..，所以 P(D|h+)=P(d1,d2,..,dn|h+)。我们又一次遇到了数据稀疏性，为什么这么说呢？P(d1,d2,..,dn|h+)就是说在垃圾邮件当中出现跟我们目前这封邮件一模一样的一封邮件的概率是多大！开玩笑，每封邮件都是不同的，世界上有无穷多封邮件。瞧，这就是数据稀疏性，因为可以肯定地说，你收集的训练数据库不管里面含了多少封邮件，也不可能找出一封跟目前这封一模一样的。结果呢？我们又该如何来计算 P(d1,d2,..,dn|h+)呢？

我们将 P(d1,d2,..,dn|h+)扩展为：P(d1|h+)*P(d2|d1,h+)*P(d3|d2,d1,h+)*...。熟悉这个式子吗？这里我们会使用一个更激进的假设，我们假设 di 与 di-1 是完全条件无关的，于是式子就简化为 P(d1|h+)*P(d2|h+)*P(d3|h+)*...。这个就是所谓的条件独立假设，也正是朴素贝叶斯方法的朴素之处。而计算 P(d1|h+)*P(d2|h+)*P(d3|h+)*..就太简单了，只要统计 di 这个单词在垃圾邮件中出现的频率即可。关于贝叶斯垃圾邮件过滤更多的内容可以参考这个条目，注意其中提到的其他资料。

一点注记：这里，为什么有这个数据稀疏问题，还是因为统计学习方法工作在浅层面，世界上的单词就算不再变多也是非常之多的，单词之间组成的句子也是变化多端，更不用说一篇文章了，文章数目则是无穷的，所以在这个层面作统计，肯定要被数据稀疏性困扰。我们要注意，虽然句子和文章的数目是无限的，然而就拿邮件来说，如果我们只关心邮件中句子的语义（进而更高抽象层面的“意图”（语义，意图如何可计算地定义出来是一个人工智能问题），在这个层面上可能性便大大缩

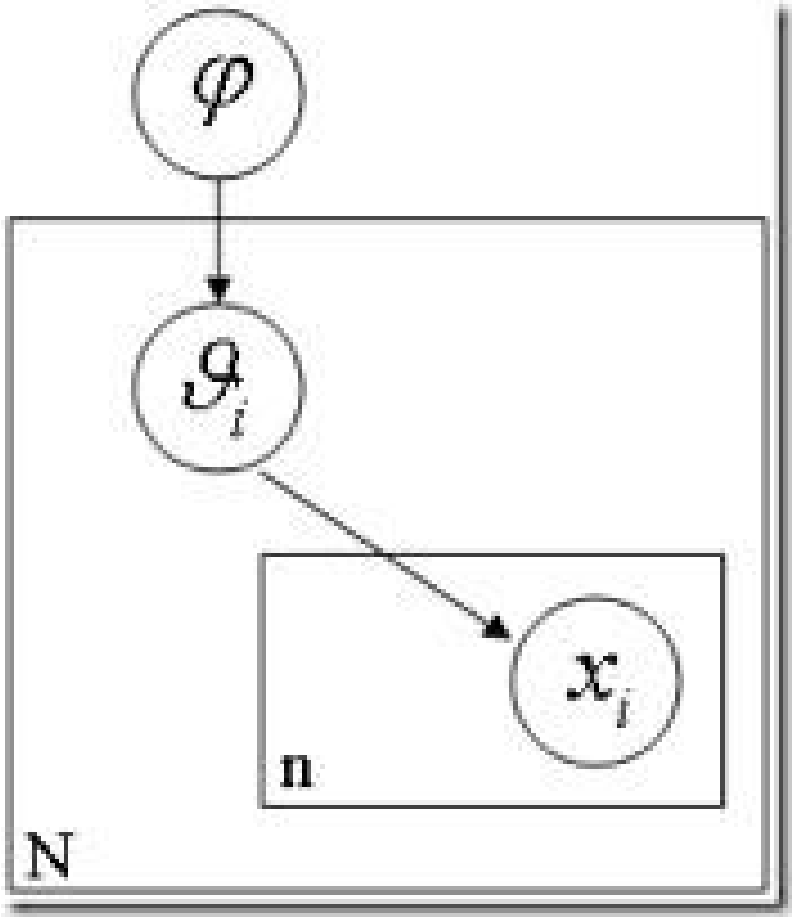
减了，我们关心的抽象层面越高，可能性越小。单词集合和句子的对应是多对一的，句子和语义的对应又是多对一的，语义和意图的对应还是多对一的，这是个层级体系。神经科学的发现也表明大脑的皮层大致有一种层级结构，对应着越来越抽象的各个层面，至于如何具体实现一个可放在计算机内的大脑皮层，仍然是一个未解决问题，以上只是一个原则（principle）上的认识，只有当 computational 的 cortex 模型被建立起来了之后才可能将其放入电脑。

5.2 为什么朴素贝叶斯方法令人诧异地好——一个理论解释

朴素贝叶斯方法的条件独立假设看上去很傻很天真，为什么结果却很好很强大呢？就拿一个句子来说，我们怎么能鲁莽地声称其中任意一个单词出现的概率只受到它前面的 3 个或 4 个单词的影响呢？别说 3 个，有时候一个单词的概率受到上一句话的影响都是绝对可能的。那么为什么这个假设在实际中的表现却不比决策树

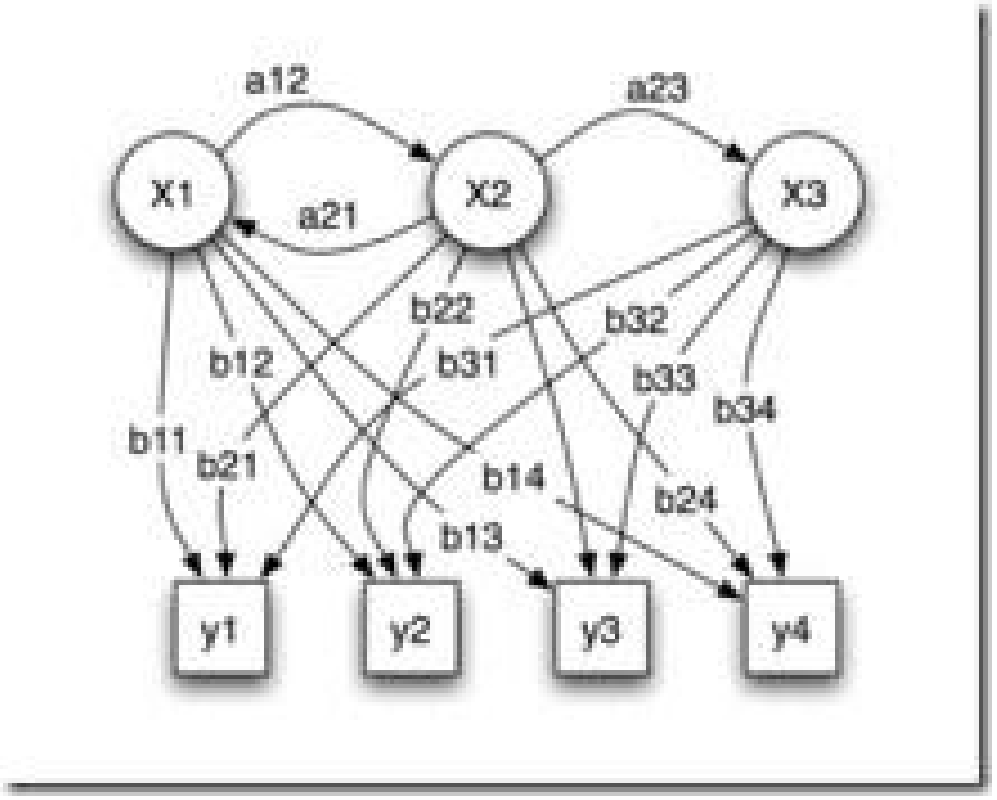
差呢？有人对此提出了一个理论解释，并且建立了什么时候朴素贝叶斯的效果能够等价于非朴素贝叶斯的充要条件，这个解释的核心就是：有些独立假设在各个分类之间的分布都是均匀的所以对于似然的相对大小不产生影响；即便不是如此，也有很大的可能性各个独立假设所产生的消极影响或积极影响互相抵消，最终导致结果受到的影响不大。具体的数学公式请参考这篇 paper。

6.层级贝叶斯模型



层级贝叶斯模型是现代贝叶斯方法的标志性建筑之一。前面讲的贝叶斯，都是在同一个事物层次上的各个因素之间进行统计推理，然而层次贝叶斯模型在哲学上更深入了一层，将这些因素背后的因素（原因的原因，原因的原因，以此类推）囊括进来。一个教科书例子是：如果你手头有 N 枚硬币，它们是同一个工厂铸出来的，你把每一枚硬币掷出一个结果，然后基于这 N 个结果对这 N 个硬币的θ（出现正面的比例）进行推理。如果根据最大似然，每个硬币的θ不是 1 就是 0（这个前面提到过的），然而我们又知道每个硬币的 p(θ)是有一个先验概率的，也许是一个 beta 分布。也就是说，每个硬币的实际投掷结果 Xi 服从以θ为中心的正态分布，而θ又服从另一个以Ψ为中心的 beta 分布。层层因果关系就体现出来了。进而Ψ还可能依赖于因果链上更上层的因素，以此类推。

6.1 隐马可夫模型（HMM）




吴军在数学之美系列里面介绍的隐马可夫模型（HMM）就是一个简单的层级贝叶斯模型：

那么怎么根据接收到的信息来推测说话者想表达的意思呢？我们可以利用叫做“隐含马尔可夫模型”（Hidden Markov Model）来解决这些问题。以语音识别为例，当我们观测到语音信号 o1,o2,o3 时，我们要根据这组信号推测出发送的句子 s1,s2,s3。显然，我们应该在所有可能的句子中找最有可能性的一个。用数学语言来描述，就是在已知 o1,o2,o3,…的情况下，求使得条件概率 P(s1,s2,s3,…|o1,o2,o3….)达到最大值的那个句子 s1,

s2,s3,...

吴军的文章中这里省掉没说的是，s1,s2,s3,..这个句子的生成概率同时又取决于一组参数 ,这组参数决定了 s1,s2,s3,..这个马可夫链的先验生成概率。如果我们将这组参数记为λ，我们实际上要求的是：P(S|O,λ)（其中 O 表示 o1,o2,o3,..，S 表示 s1,s2,s3,..）

当然，上面的概率不容易直接求出，于是我们可以间接地计算它。利用贝叶斯公式并且省掉一个常数项，可以把上述公式等价变换成

P(o1,o2,o3,...|s1,s2,s3....)*P(s1,s2,s3,...)

其中

P(o1,o2,o3,...|s1,s2,s3....)表示某句话 s1,s2,s3...被读成 o1,o2,o3,...的可能性,而 P(s1,s2,s3,...)表示字串 s1,s2,s3,...本身能够成为一个合乎情理的句子的可能性，所以这个公式的意义是用发送信号为 s1,s2,s3...这个数列的可能性乘以 s1,s2,s3..本身可以一个句子的可能性，得

出概率。

这里，s1,s2,s3...本身可以一个句子的可能性其实就取决于参数λ，也就是语言模型。所以简而言之就是发出的语音信号取决于背后实际想发出的句子，而背后实际想发出的句子本身的独立先验概率又取决于语言模型。

7.贝叶斯网络

吴军已经对贝叶斯网络作了科普，请直接跳转到这里。更详细的理论参考所有机器学习的书上都有。

参考资料

一堆机器学习，一堆概率统计，一堆 Google，和一堆 Wikipedia 条目，一堆 paper。

部分书籍参考《机器学习与人工智能资源导引》。http://blog.csdn.net/pongba/archive/2008/09/11/2915005.aspx